Srdjan Stanković
Irena Orović
Ervin Sejdić

Multimedia Signals and Systems

Springer

Multimedia
Signals and Systems

Srdjan Stanković • Irena Orović • Ervin Sejdić

Multimedia Signals and Systems

Springer

Srdjan Stanković
University of Montenegro
Džordža Vašingtona bb
Podgorica, Montenegro

Irena Orović
University of Montenegro
Džordža Vašingtona bb
Podgorica, Montenegro

Ervin Sejdić
University of Pittsburgh
Benedum Hall
Pittsburgh, PA, USA

ISBN 978-1-4614-4207-3 ISBN 978-1-4614-4208-0 (eBook)
DOI 10.1007/978-1-4614-4208-0
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012943967

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The book is composed as a combination of two intertwined areas: multimedia signal processing and multimedia systems. Note that multimedia signal processing is presented in a larger extent than in the standard books on fundamentals of multimedia systems.

Besides commonly used signal processing techniques, here we also consider the tools convenient for certain advanced applications that might inspire a reader for further improvement of the existing multimedia algorithms.

The book is divided into nine chapters. We should emphasize that the chapters on Mathematical Transforms Used for Multimedia Signal Processing (Chap. 1), on Compressive Sensing (Chap. 6), on Digital Watermarking (Chap. 7), and on Telemedicine (Chap. 8) contain a more extensive and detailed analysis than usual in the existing literature on multimedia systems. We especially note the chapter on Compressive Sensing and its application in multimedia is a completely new area that provides a new insight into the existing applications.

Chapters on Digital Audio (Chap. 2), on Digital Data Storage and Compression (Chap. 3), on Digital Image (Chap. 4), on Digital Video (Chap. 5), and on Multimedia Communications (Chap. 9) basically follow the classical pattern of the content in this type of literature. However, the authors have put considerable effort to enrich this material with a lot of comprehensive information, in order to facilitate the understanding of the presented text. These chapters also contain some new and, in our opinion, interesting recently published results.

Each chapter ends with a section with worked out examples that may be useful for additional mastering and clarification of the presented material and for taking into account certain interesting applications. Beside basic examples, strictly associated with the presented theory, the book also contains some advanced applications that could be considered as a complement to the presented theory. A considerable number of Matlab codes are included in the examples, so that the reader can easily reconstruct most of the particular presented techniques.

Thus, the book basically contains a necessary material for understanding the fundamentals of multimedia systems, and in that sense it may be used in the undergraduate courses. On the other hand, the parts related to the multimedia signal processing, together with the advanced techniques included in other chapters, may be used in the graduate courses as an appropriate literature related to the initial research.

Since this is the first edition, the authors are aware that, nevertheless all the efforts they have made to avoid the errors and ambiguities, they are practically unavoidable. Therefore, we will appreciate all the comments and suggestions to reduce these in the subsequent editions.

Finally, the authors gratefully acknowledge the useful and constructive suggestions of our colleagues during the preparation of the manuscript. We extend special gratitude to: Prof. Zdravko Usković, Prof. Ljubiša Stanković, Prof. Moeness Amin, and Prof. Victor Sucic. Also, we are thankful to Dr. Nikola Žarić, as well as to the Ph.D. students Branka Jokanović and Andjela Draganić.

Podgorica, Montenegro Srdjan Stanković
Podgorica, Montenegro Irena Orović
Pittsburgh, PA, USA Ervin Sejdić

Introduction

Nowadays, there is an intention to merge different types of data into a single vivid presentation. By combining text, audio, images, video, graphics, and animations we may achieve a more comprehensive description and better insight into areas, objects, and events. Formerly, each of the mentioned types of data were produced and presented by using a separate device. Consequently, making an integration of different data types was a demanding project by itself. The process of digitalization brings new perspectives and the possibility to make a universal data representation in binary (digital) format. Furthermore, this creates the possibility of computer-based multimedia data processing, and now we may observe computer as a multimedia device, which is a basis of modern multimedia systems.

Thus, *Multimedia* is one of the frequently used words during the last decade and it is mainly related to the representation and processing of combined data types/media into a single package by using the computer technologies. Nevertheless, one should make the difference between the term multimedia that is used within certain creative disciplines (assuming a combination of different data for the purpose of efficient presentation) and the engineering aspect of multimedia, where the focus is directed to optimizing the algorithms for merging, processing, and transmission of such complex data structures.

When considering the etymology, we may say that the term multimedia is derived from the Latin word *multus* meaning numerous (or several), and *medium* meaning middle or center.

The fundamentals of multimedia systems imply creating, processing, compression, storing, and transmission of multimedia data, and as such the multimedia systems are multidisciplinary (they include certain parts from different fields, especially digital signal processing, hardware design, telecommunications, and computer networking).

The fact that the multimedia data can be either time-dependent (audio, video, and animations) or space-dependent (image, text, and graphics) additionally complicates the attempts to provide unified algorithms, which would be used for all types of multimedia signals.

Most of the algorithms in multimedia systems have been derived from the general signal processing algorithms. Hence, significant attention should be paid to the signal processing theory and methods, which are the key issues in further enhancing multimedia applications. Finally, to keep up with the modern technologies, the multimedia systems should include advanced techniques related to digital data protection, compressive data acquiring, signal reconstruction, etc.

Since the multimedia systems are founded on the assumption of integrating the digital signals represented in the binary form, the process of digitalization and its reflection on the signal quality will be briefly reviewed in the sequel.

Analog to Digital Signal Conversion

The process of converting analog to digital signals is called the digitalization. It can be illustrated by using the following scheme:

$$\begin{tikzcd}
\boxed{\text{Analog signal}} \arrow[r] & \boxed{\text{Sampling}} \arrow[r] & \boxed{\text{Quantization}} \arrow[r] & \boxed{\text{Digitalization}}
\end{tikzcd}$$

The sampling of an analog signal is performed on the basis of the sampling theorem, which ensures the exact signal reconstruction from its digital samples. The Shannon-Nyquist sampling theorem defined the maximal sampling interval (the interval between successive samples) as follows:

$$T \le \frac{1}{2 f_{\max}},$$

where $f_{\max}$ represents the maximal signal frequency. According to the analog signal nature, the discrete signal samples may have any value from the set of real numbers. It means that, in order to represent the samples with high precision in the digital form, a large number of bits are required. Obviously, this is difficult to realize in practice, since limited number of bits are available for representation of signal samples. The number of bits per sample defines the number of quantization intervals, which further determines a set of possible values for digital samples. Hence, if the value of the sample is between two quantization levels, it is rounded to the closer quantization level. Therefore, the original values of samples are changed, which is modeled as a quantization noise. The signal, represented by n bits, will have 2^n quantization levels. As illustrations, let us observe the examples of 8-bit and 16-bit format; in the first case, the signal is represented by 256 quantization levels, while in the second case 65,536 levels are available.

Working with digital signals brings several advantages. For instance, due to the same digital format, different types of data can be stored in the same storage media, transmitted using the same communication channels, can be processed and displayed by using the same devices, which is inapplicable in the case of analog data format. Also, an important property is robustness to noise. Namely, the digital values "0" and "1" are associated to the low (e.g., 0 V) and high voltages (e.g., 5V). Usually, the threshold between the values 0 and 1 is set to the average between their corresponding voltage levels. During transmission, a digital signal can be corrupted by noise, but it does not affect the signal as long as the digital values are preserved, i.e., as long as the level of "1" does not become the level of "0" and vice versa.

However, the certain limitations and drawbacks of digital format should be mentioned as well, such as quantization noise and significant memory requirements, which further requires the development of sophisticated masking models and data compression algorithms.

In order to provide a better insight into the memory requirements of multimedia data, we can mention that text requires 1.28 Kb per line (80 characters per line, 2 bytes per character), stereo audio signal sampled at 44,100 Hz with 16 bits per sample requires 1.41 Mb, color image of size 1,024 $\times$ 768 requires 18.8 Mb (24 bits per pixel are used), while video signal with TV resolution requires 248.8 Mb (resolution 720 $\times$ 576, 24 bits per pixel, 25 frames per second).

Contents

1 Mathematical Transforms Used for Multimedia Signal Processing 1
- 1.1 Fourier Transform 1
  - 1.1.1 Discrete Fourier Transform 4
  - 1.1.2 Discrete Cosine Transform 5
- 1.2 Filtering in the Frequency Domain 5
- 1.3 Time-Frequency Signal Analysis 6
- 1.4 Ideal Time-Frequency Representation 8
- 1.5 Short-Time Fourier Transform 8
- 1.6 Wigner Distribution 11
- 1.7 Time-Varying Filtering 16
- 1.8 Wavelet Transform 17
  - 1.8.1 Continuous Wavelet Transform 17
  - 1.8.2 Wavelet Transform with Discrete Wavelet Functions 18
  - 1.8.3 Wavelet Families 19
  - 1.8.4 Haar Wavelet 20
  - 1.8.5 Multiresolution Analysis 23
  - 1.8.6 Filter Bank 25
  - 1.8.7 Daubechies Orthogonal Filters 26
  - 1.8.8 Two-Dimensional Signals 28
- 1.9 Signal Decomposition Using Hermite Functions 31
  - 1.9.1 One-Dimensional Signals and Hermite Functions 31
  - 1.9.2 Two-Dimensional Signals and Two-Dimensional Hermite Functions 33
- 1.10 Examples 36
- References 52

2 Digital Audio 55
- 2.1 The Nature of Sound 55
- 2.2 Development of Systems for Storing and Playback of Digital Audio 57

2.3 Effects of Sampling and Quantization on the Quality of Audio Signal 58
2.3.1 Nonlinear Quantization 60
2.3.2 Block Floating-Point Conversion 62
2.3.3 Differential Pulse Code Modulation 62
2.3.4 Super Bit Mapping 63
2.4 Speech Signals 63
2.4.1 Linear Model of Speech Production System 65
2.5 Voice Activity Analysis and Detectors 67
2.5.1 Word Endpoints Detector 71
2.6 Speech and Music Decomposition Algorithm 72
2.6.1 Principal Components Analysis Based on SVD 72
2.6.2 Components Extraction by Using the SVD and the S-Method 73
2.7 Psychoacoustic Effects 77
2.7.1 Audio Masking 77
2.8 Audio Compression 79
2.8.1 Lossless Compressions 79
2.8.2 Lossy Compressions 85
2.8.3 MPEG Compression 88
2.8.4 ATRAC Compression 94
2.9 Examples 95
References 107

3 Storing and Transmission of Digital Audio Signals 109
3.1 Compact Disc: CD 109
3.1.1 Encoding CD 111
3.2 Mini Disc 120
3.3 Super Audio CD (SACD) 122
3.4 DVD-Audio 123
3.5 Principles of Digital Audio Broadcasting: DAB 123
3.5.1 Orthogonal Frequency-Division Multiplexing (OFDM) 125
3.6 Examples 127
References 131

4 Digital Image 133
4.1 Fundamentals of Digital Image Processing 133
4.2 Elementary Algebraic Operations with Images 134
4.3 Basic Geometric Operations 136
4.4 The Characteristics of the Human Eye 137
4.5 Color Models 138
4.5.1 CMY, CMYK, YUV, and HSV Color 139
4.6 Filtering 142
4.6.1 Noise Probability Distributions 142
4.6.2 Filtering in the Spatial Domain 143

4.6.3 Filtering in the Frequency Domain 149
4.6.4 Image Sharpening 150
4.6.5 Wiener Filtering 150
4.7 Enhancing Image Details 151
4.8 Analysis of Image Content 152
4.8.1 The Distribution of Colors 152
4.8.2 Textures 153
4.8.3 Co-occurrence Matrix 154
4.8.4 Edge Detection 155
4.8.5 The Condition of the Global Edge (Edge-Based Representation: A Contour Image) 157
4.8.6 Dithering 157
4.9 Image Compression 158
4.9.1 JPEG Image Compression Algorithm 158
4.9.2 JPEG Lossless Compression 164
4.9.3 Progressive JPEG Compression 166
4.9.4 JPEG Compression of Color Images 166
4.9.5 JPEG2000 Compression 168
4.9.6 Fractal Compression 179
4.9.7 Image Reconstructions from Projections 181
4.10 Examples 183
References 197

5 Digital Video 199
5.1 Digital Video Standards 200
5.2 Motion Parameters Estimation in Video Sequences 201
5.3 Digital Video Compression 204
5.3.1 MPEG-1 Video Compression Algorithm 205
5.3.2 MPEG-2 Compression Algorithm 206
5.3.3 MPEG-4 Compression Algorithm 207
5.3.4 VCEG Algorithms 208
5.4 Data Rate and Distortion 216
5.5 Communications Protocols for Multimedia Data 219
5.6 H.323 Multimedia Conference 219
5.6.1 SIP Protocol 220
5.7 Audio Within a TV Signal 222
5.8 Video Signal Processor 223
5.9 Examples 224
References 230

6 Compressive Sensing 233
6.1 The Compressive Sensing Requirements 234
6.1.1 Sparsity Property 234
6.1.2 Incoherence 236
6.1.3 Restricted Isometry Property 237

6.1.4 Numerical Realizations 238
6.1.5 An Example of Using Compressive Sensing Principles 239
6.2 Applications of Compressive Sensing Approach 244
6.2.1 Multicomponent One-Dimensional Signal Reconstruction 244
6.2.2 Compressive Sensing Applied to Image Reconstruction 246
6.2.3 Compressive Sensing and Sparse Time-Frequency Analysis 249
References 253

7 Digital Watermarking 255
7.1 Classification of Digital Watermarking Techniques 255
7.1.1 Classification in Terms of the Embedding Domain 257
7.2 Common Requirements Considered in Watermarking 257
7.3 Watermark Embedding 259
7.4 Watermark Detection 260
7.4.1 Hypothesis Testing Approach 260
7.4.2 A Class of Locally Optimal Detectors 264
7.4.3 Correlation Coefficient and Similarity Measure 266
7.5 Examples of Watermarking Procedures 266
7.5.1 Audio Watermarking Techniques 266
7.5.2 Image Watermarking Techniques 268
7.5.3 The Procedure for Watermarking of Color Images 269
7.5.4 An Overview of Some Time-Frequency-Based Watermarking Techniques 270
7.6 Examples 274
References 283

8 Multimedia Signals and Systems in Telemedicine 285
8.1 General Health Care 285
8.1.1 Telenursing 285
8.1.2 Telepharmacy 286
8.1.3 Telerehabilitation 286
8.2 Specialist Health Care 287
8.2.1 Telecardiology 287
8.2.2 Teleradiology 289
8.2.3 Telesurgery 289
References 291

9 Multimedia Communications 293
9.1 An Overview of Different Networks Types 293
9.1.1 Telephone Networks 293
9.1.2 Data Network 294
9.1.3 Broadcast TV Networks 295

9.1.4 Integrated Services Digital Network (ISDN) 295
9.1.5 Multiservice Broadband Networks 297
9.2 Multimedia Applications 297
9.2.1 Communications 297
9.2.2 Interactive Internet Applications 299
9.2.3 Entertainment Applications 299
9.3 Multiplexing 299
9.4 Quality of Service: QoS 300
9.5 Internet 301
9.6 IP Address 301
9.6.1 Format of IPv4 Address 302
9.6.2 Classless Addressing 304
9.6.3 IPv6 Address Format 305
9.7 A Protocol Set for Data Transmission over the Internet (TCP/IP Environment) 307
9.7.1 IP Protocol and IP Datagram 308
9.7.2 TCP Protocol and Connection-Oriented Service 309
9.7.3 UDP and Connectionless Services 312
9.8 Higher-Order Protocols 313
9.8.1 HTTP Protocol 313
9.8.2 FTP Protocol 316
9.8.3 Other Higher-Order Protocols 318
9.9 HTML 318
9.9.1 Text Formatting 319
9.9.2 Background Color and Text Color 320
9.9.3 Adding an Image to HTML Page 321
9.9.4 Unordered and Ordered Lists 321
9.9.5 Links to Other Websites 323
9.9.6 Setting an Anchor Within a Web Page 324
9.9.7 Adding Video Content to a Web Page 327
9.9.8 Creating and Formatting Tables 328
9.9.9 Forms for Data Entry 331
9.10 Email 339
9.10.1 MIME: Multipurpose Internet Mail Extensions 340
9.11 Examples 341
References 344

Index 345

List of Figures

Fig. 1.1 Signal representations in: (a) Time domain; (b) Frequency domain 2
Fig. 1.2 Representations of a multicomponent signal: (a) Time domain representation; (b) Frequency domain representation 2
Fig. 1.3 Signal representation: (a) Time domain; (b) Frequency domain 2
Fig. 1.4 Finite duration discrete signal 4
Fig. 1.5 (a) Low-pass filter, (b) High-pass filter, (c) Band-pass filter 6
Fig. 1.6 (a) Sum of two sinusoids with equal duration, (b) Composition of two sinusoids appearing at different time instants, (c) Fourier transform for the first signal, (d) Fourier transform for the second signal 7
Fig. 1.7 (a) The ideal time-frequency representation of the sum of sinusoids from Fig. 1.6a, (b) The ideal time-frequency representation of the time-shifted sinusoids from Fig. 1.6b 7
Fig. 1.8 An illustration of the STFT calculations 9
Fig. 1.9 Ideal time-frequency representation of signal $f(t)$ 10
Fig. 1.10 Illustration of the uncertainty principle 10
Fig. 1.11 Spectrograms of multicomponent signals: (a) Two sinusoids and chirp, (b) Three sinusoids and chirp 11
Fig. 1.12 Calculation of marginal conditions 12
Fig. 1.13 (a) Wigner distribution of a multicomponent signal, (b) Auto-terms of the Wigner distribution 14
Fig. 1.14 Time-frequency representations of a multicomponent signal obtained by using SM for different values of window width (defined as $2L_d + 1$) 15
Fig. 1.15 Spectrogram and SM of speech signal 16
Fig. 1.16 Wavelet functions 21
Fig. 1.17 Approximation (scaling) and wavelet spaces 23

Fig. 1.18 Time-frequency representation of wavelet 24
Fig. 1.19 Analysis filters 25
Fig. 1.20 Wavelet decomposition and reconstruction of signals using a filter bank 26
Fig. 1.21 Standard wavelet decomposition 29
Fig. 1.22 Quincunx decomposition 29
Fig. 1.23 Pyramidal wavelet decomposition 30
Fig. 1.24 Uniform wavelet decomposition 30
Fig. 1.25 Illustration of the first few Hermite functions 31
Fig. 1.26 (a) Original signal of length 126 samples, (b) Reconstructed signal with 126 functions, (c) Reconstructed signal with 90 functions, (d) Reconstructed signal with 70 functions 34
Fig. 1.27 Examples of two-dimensional Hermite functions: (a) $\Psi_{00}(x,y)$, (b) $\Psi_{24}(x,y)$, (c) $\Psi_{44}(x,y)$ 35
Fig. 1.28 (a) Original image, reconstructed image by using Hermite projection method along the rows: (b) with 100 Hermite functions, (c) 80 Hermite functions, (d) 60 Hermite functions 36
Fig. 1.29 Illustration of the filter function 39
Fig. 1.30 Fourier transform and the ideal time-frequency distribution for signal: (a) y_a, (b) y_b 40
Fig. 1.31 Ideal time-frequency representation of signal $f(t)$ 40
Fig. 1.32 Spectrograms for different window widths 41
Fig. 1.33 Spectrogram and S-method 42
Fig. 1.34 Signal $f(0)$ before decomposition 48
Fig. 1.35 Signal decomposition 49

Fig. 2.1 An illustration of sound propagation within a closed room 56
Fig. 2.2 The Fletcher curve 56
Fig. 2.3 Block diagram of a DAB system 57
Fig. 2.4 Aliasing effects 58
Fig. 2.5 An example of antialiasing filter with a steep transition 58
Fig. 2.6 A circuit for signal sampling 59
Fig. 2.7 The probability density function of the quantization error 59
Fig. 2.8 Nonlinear quantization 61
Fig. 2.9 Floating-point conversion 61
Fig. 2.10 S/N ratio for a considered floating-point converter (8-bit mantissa and 3-bit exponent) 62
Fig. 2.11 Single-bit A/D converter 63
Fig. 2.12 Illustration of the speech generation system 64
Fig. 2.13 Time-frequency representation of speech formants 64
Fig. 2.14 A model of the speech production system 65
Fig. 2.15 Excitation signals $g(t)$, $g'(t)$, $g''(t)$ 67
Fig. 2.16 An illustration of different speech parts 68

Fig. 2.17 The outputs of the voice activity indicators based on the magnitudes (*MA*), zero-crossing rate (*ZC*), and prediction gain (*PG*) 70
Fig. 2.18 (a) Speech signal, (b) Energy-entropy feature for speech frames 71
Fig. 2.19 The formants isolated by using the eigenvalues decomposition method 74
Fig. 2.20 Illustrations of different components combinations selected by a few time-frequency masks 75
Fig. 2.21 The time-frequency representation of the violin signal obtained by using the S-method 75
Fig. 2.22 Separated components of violin signal 75
Fig. 2.23 Left and right vectors of speech (*left column*) and swallowing sound (*right column*) 76
Fig. 2.24 Illustration of human auditory system 77
Fig. 2.25 Masking noise 78
Fig. 2.26 An illustration of audio masking 78
Fig. 2.27 Lossless and lossy compressions 80
Fig. 2.28 Quality of a compressed signal depends on the complexity of the compression system and the compression ratio 80
Fig. 2.29 The system for measuring the noise/masking ratio 85
Fig. 2.30 Dividing the spectrum into critical bands by using a filter bank 87
Fig. 2.31 Illustration of the critical bands 87
Fig. 2.32 Illustration of the effects of masking tones: (a) Masking in frequency, (b) Masking of the bandwidth range 89
Fig. 2.33 Block diagram of MUSICAM-based coder 90
Fig. 2.34 A scheme for converting from five channels into two channels 90
Fig. 2.35 A block diagram of MPEG layer I compression 91
Fig. 2.36 A part of the bits packing structure in MPEG layer I 92
Fig. 2.37 Dividing the frequency range in MPEG layer II 93
Fig. 2.38 Windows used by MP3 algorithm: (a) Wide window, (b) Narrow window, (c) and (d) Transition windows, (e) Shifting from wide to narrow window and vice versa 93
Fig. 2.39 A block diagram of the system for MP3 compression 94
Fig. 2.40 A block scheme of ATRAC compression system 94
Fig. 2.41 Division of the time-frequency plane in the ATRAC algorithm 95
Fig. 2.42 Filter function 97
Fig. 2.43 (a) Fourier transform of the original signal, (b) Fourier transform of the filtered signal 97
Fig. 2.44 Parameters of band-pass filter function 98
Fig. 2.45 Filter transfer function 98

Fig. 2.46 Subband samples and masking level 102
Fig. 2.47 An example of Huffman coding 103
Fig. 2.48 Illustration of MPEG layer I sequence part 107

Fig. 3.1 The structure of CD 110
Fig. 3.2 Reflections from a CD 111
Fig. 3.3 An example of interleaving 113
Fig. 3.4 Interleaving based on the delay lines 113
Fig. 3.5 Interleaving used in CD coding 114
Fig. 3.6 The reconstruction principle of the damaged signal part 115
Fig. 3.7 Illustration of timing diagrams for *P* and *Q* channels 118
Fig. 3.8 An example of EFM encoding 119
Fig. 3.9 The structure of MD 120
Fig. 3.10 Recording the data on the MD 121
Fig. 3.11 Reflections of the laser beam in the case of MD 122
Fig. 3.12 A block diagram of the MD system 122
Fig. 3.13 Direct and reflected FM signals with equal strength are nulled when 180° phase difference occur 124
Fig. 3.14 A block scheme of DAB system 124
Fig. 3.15 An illustration of channel interleaving 124
Fig. 3.16 Diagram and table for QPSK modulation 125
Fig. 3.17 An OFDM spectrum: (a) One subcarrier, (b) Five subcarriers 126
Fig. 3.18 A simplified block diagram of OFDM system 126
Fig. 3.19 Example of interleaving 127

Fig. 4.1 Histogram of "Lena" image 134
Fig. 4.2 (a) Grayscale image "Baboon", (b) grayscale image "Lena" 135
Fig. 4.3 The resulting image obtained by adding 30% of "Baboon" to "Lena" 135
Fig. 4.4 Negative of "Lena" image 136
Fig. 4.5 Clipped "Lena" image 136
Fig. 4.6 "Lena" image translated for 31 columns and 31 rows 137
Fig. 4.7 "Lena" is rotated by 45° 137
Fig. 4.8 Color cube 138
Fig. 4.9 The HSV color model 141
Fig. 4.10 "Lena" affected by an impulse noise with density 0.05 142
Fig. 4.11 "Lena" affected by zero-mean white Gaussian noise whose variance is equal to 0.02 143
Fig. 4.12 Illustration of blurring after applying mean filter on the image edge (the mask size used is 5 × 5) 144
Fig. 4.13 (a) "Lena" affected by Gaussian noise with zero mean and variance 0.02, (b) filtered image obtained by using mean filter of size 3 × 3, (c) filtered image obtained by using mean filter of size 5 × 5 145

Fig. 4.14 (a) Original image, (b) noisy image (Gaussian noise with 0.05 mean and variance 0.025), (c) image filtered by using arithmetic mean of size 3 $\times$ 3, (d) image filtered by using geometric mean of size 3 $\times$ 3 146
Fig. 4.15 (a) Original image, (b) noisy image (Gaussian noise with 0.05 mean and variance 0.025), (c) image filtered by using arithmetic mean of size 3 $\times$ 3, (d) image filtered by using geometric mean of size 3 $\times$ 3 147
Fig. 4.16 (a) "Lena" affected by impulse noise with density 0.05, (b) denoised image (median filter of size 3 $\times$ 3 is used), (c) denoised image (5 $\times$ 5 median filter) 148
Fig. 4.17 Illustration of edge detection: (a) Original image, (b) L_v, (c) L_h, (d) L 156
Fig. 4.18 JPEG encoder block diagram 158
Fig. 4.19 (a) Original image "Lena" (b) image based on the first 128 $\times$ 128 DCT coefficients, (c) image based on the first 64 $\times$ 64 DCT coefficients, (d) image based on the first 25 $\times$ 25 DCT coefficients 159
Fig. 4.20 Quantization matrix obtained for $quality = 2$ 160
Fig. 4.21 Coefficients of the quantization matrix Q_{50} 161
Fig. 4.22 Zigzag reordering 161
Fig. 4.23 (a) 8 $\times$ 8 image block, (b) values of pixels after shifting to the range [−128, 127], (c) DCT coefficients (rounded to integers) for the given block, (d) quantization matrix QF = 70, (e) DCT coefficients after quantization 163
Fig. 4.24 JPEG decoder 164
Fig. 4.25 (a) Original "Lena" image, (b) "Lena" image after applying JPEG compression with QF = 70%, (c) "Lena" image after JPEG compression with QF = 25%, (d) "Lena" image after JPEG compression with QF = 5% 165
Fig. 4.26 An example of using the combined progressive JPEG algorithm 167
Fig. 4.27 Y, Cr, and Cb component 167
Fig. 4.28 An interleaving procedure for JPEG color image compression 168
Fig. 4.29 Expansion of a sequence of pixels 169
Fig. 4.30 Calculating the wavelet coefficients 171
Fig. 4.31 Illustration of applying the wavelet transform to two-dimensional signals 171
Fig. 4.32 Organization of subbands for JPEG2000 172
Fig. 4.33 Face is coded as ROI 173
Fig. 4.34 ROI mask mapped into subbands 173
Fig. 4.35 (a) Quantized wavelet coefficients before scaling, (b) quantized wavelet coefficients after scaling 174
Fig. 4.36 (a) The low-frequency subband is codded together with ROI regions in other subbands, (b) arbitrary ROI mask 175

Fig. 4.37 An example of subbands, precincts, and code-blocks partition 176
Fig. 4.38 The bit-planes scan method 177
Fig. 4.39 Part of bit planes that contain four considered coefficients 178
Fig. 4.40 An example of arithmetic coding 179
Fig. 4.41 Significance indicators for eight neighboring coefficients 180
Fig. 4.42 "Lena" image divided into different fractals 181
Fig. 4.43 (a) Original image cameraman, (b) negative of image cameraman 184
Fig. 4.44 Image lightening and darkening 185
Fig. 4.45 Binary image cameraman 186
Fig. 4.46 (a) Original image, (b) noisy image, (c) filtered image 187
Fig. 4.47 (a) Noisy image, (b) filtered image 187
Fig. 4.48 (a) Noisy image (Gaussian noise), (b) filtered image 188

Fig. 5.1 An illustration of video signal sampling 200
Fig. 5.2 Sampling schemes (a) 4:4:4, (b) 4:2:2, (c) 4:2:0 200
Fig. 5.3 Motion vector estimation for a 3×3 block 202
Fig. 5.4 Illustration of full search 203
Fig. 5.5 Illustration of three-step search 203
Fig. 5.6 Illustration of the logarithmic search 204
Fig. 5.7 Structure of frames in MPEG-1 algorithm 205
Fig. 5.8 The order of I, B and P frames during transmission 206
Fig. 5.9 Illustration of the bit rate (bit rate profile) 209
Fig. 5.10 Switching from one to another bit rate by using I frames 210
Fig. 5.11 Switching between different bit rates by using SP frames 211
Fig. 5.12 Illustration of the fast-forward procedure using the SP frames 212
Fig. 5.13 (a) Intra 4×4 prediction of block a–p based on the pixels A–Q, (b) eight prediction directions for Intracoding 212
Fig. 5.14 Five (from nine) 4×4 intraprediction modes 213
Fig. 5.15 Macroblocks and subblocks 213
Fig. 5.16 Interpolation method for ¼ pixel precision (luminance component) 214
Fig. 5.17 Multiple reference frames 215
Fig. 5.18 Determining an optimal point for distortion-data rate compromise 217
Fig. 5.19 Rate control for video signal 218
Fig. 5.20 Illustration of changes in quality due to reduced rate 218
Fig. 5.21 Error reduction by using the older reference frame 219
Fig. 5.22 H.323 terminal 220
Fig. 5.23 SIP architecture 222
Fig. 5.24 Transport stream multiplexing and demultiplexing 223
Fig. 5.25 Video processor 223

Fig. 6.1 Signal sampling: classical approach and compressive sensing alternative 234

Fig. 6.2 Illustration of the compressive sensing procedure 235
Fig. 6.3 (a) Signal f_x, (b) Fourier transform F_x 240
Fig. 6.4 (a) Original signal and randomly selected samples denoted by red points, (b) Reconstructed signal, (c) N point Fourier transform F_y, (**d**) Fourier transform of reconstructed signal 242
Fig. 6.5 (a) Original signal, (b) reconstructed signal, (c) reconstruction error 245
Fig. 6.6 (a) Fourier transform of the original signal, (b) Fourier transform of the reconstructed signal 246
Fig. 6.7 Maximal and mean absolute errors of signal reconstruction 246
Fig. 6.8 (a) Original (*left*) and reconstructed (*right*) flute signal, (b) zoomed segment of the original (*left*) and reconstructed (*right*) signal, (c) Fourier transform of original (*left*) and reconstructed (*right*) signal 247
Fig. 6.9 L_1 minimization based reconstruction: (a) Original "Lena" image, (b) reconstructed "Lena" image, (c) original "Baboon" image, (d) reconstructed "Baboon" image 248
Fig. 6.10 TV reconstruction: (a) Original "Lena" image, (b) reconstructed "Lena" image, (c) Original "Baboon" image, (d) reconstructed "Baboon" image 249
Fig. 6.11 Illustration of the ambiguity domain mask 251
Fig. 6.12 (a) Wigner distributions, (b) ambiguity function, (c) resulting sparse time-frequency representation 252
Fig. 6.13 (a) Wigner distribution, (b) ambiguity function, (c) resulting sparse time-frequency representation 252

Fig. 7.1 A block scheme of a watermarking procedure 256
Fig. 7.2 Classification of digital watermarking techniques 256
Fig. 7.3 (a) DFT of image "Baboon", (b) DCT of image "Baboon" 259
Fig. 7.4 (a) Original image "Lena", (b) watermarked image "Lena" 260
Fig. 7.5 Illustration of the errors that may occur in watermark detection 264
Fig. 7.6 Distribution of coefficients: Gaussian (*green line*) and Laplace distribution (*purple line*) 265
Fig. 7.7 A region of middle frequency DCT coefficients within 8×8 block (shaded in *gray*) 269
Fig. 7.8 Color image and the separated color channels 270
Fig. 7.9 Results of watermark detection 276
Fig. 7.10 Distribution of coefficients after omitting low-frequency components 279
Fig. 7.11 (a) Speech region selected for watermarking, (b) mask function, (c) time-frequency representation of modeled watermark 281

Fig. 7.12 (a) Busy region and its spectrogram,
(b) flat region and its spectrogram 282

Fig. 8.1 A typical telerehabilitation system 287
Fig. 8.2 An example of multimedia systems in telecardiology 288
Fig. 8.3 A teleradiology system 290
Fig. 8.4 A typical use of a telesurgery system 290

Fig. 9.1 An illustration of PSTN as a system of Central Offices
that provide access to subscribers and Inter
eXchange Carriers providing long-distance services 294
Fig. 9.2 An illustration of broadcast cable network 295
Fig. 9.3 Illustration of ISDN communication concepts 296
Fig. 9.4 Services provided by an ISDN 296
Fig. 9.5 A scheme of an VoIP system 298
Fig. 9.6 (a) Switch connects a pair of users,
(b) each user is connected by a separate link,
thus forming a radial network,
(c) time-division multiplexing 300
Fig. 9.7 Network as a system of routers, servers, and workstations 302
Fig. 9.8 Communication between server and client 302
Fig. 9.9 Information about the IP address using the
Command Prompt 303
Fig. 9.10 Format of IP datagram 308
Fig. 9.11 Format of TCP segment 310
Fig. 9.12 TCP pseudoheader and TCP segment 311
Fig. 9.13 Illustration of procedure for establishing TCP connection 312
Fig. 9.14 Format of UDP segment 313
Fig. 9.15 UDP pseudoheader 313
Fig. 9.16 HTTP communication 314
Fig. 9.17 System components for FTP file transfers 317
Fig. 9.18 Web page after setting the background color and text color 320
Fig. 9.19 A web page with image included 322
Fig. 9.20 Unordered list is added to the web page 323
Fig. 9.21 Web page after adding anchors to text paragraphs 325
Fig. 9.22 The layout of a created web page 328
Fig. 9.23 Message window in Microsoft Outlook 329
Fig. 9.24 The window with the code to embed video 329
Fig. 9.25 Embedded video on the Web page 330
Fig. 9.26 An example of the table 331
Fig. 9.27 A form for data entry 334
Fig. 9.28 Final Web page design 336
Fig. 9.29 Client–server model for sending/receiving email 340
Fig. 9.30 Networks X, Y and Z 342

List of Tables

Table 1.1 Coefficients of Daubechies D4 low-pass analysis filter *h* 28
Table 1.2 Hermite polynomials of orders from 1 to 10 and the corresponding zeros 32

Table 2.1 Critical bands 86
Table 2.2 An example of amplitude levels in different frequency bands 89

Table 3.1 Basic features of CD 110
Table 3.2 Lengths of pits 110
Table 3.3 Some of the polynomials used for CRC coding 113
Table 3.4 The Galois field $GF(2^3)$ 116
Table 3.5 Examples of extending 8-bit words to 14-bit words 119

Table 4.1 Encoding of the coefficients amplitudes 162
Table 4.2 Code words for the symbols obtained in the example 163
Table 4.3 Nine contexts for the first pass 180
Table 4.4 Symbols and corresponding code words for AC luminance components 195

Table 7.1 Common attacks in audio and image watermarking procedures 258

Table 9.1 Examples of long and short formats of IPv6 addresses 306

Chapter 1
Mathematical Transforms Used for Multimedia Signal Processing

Various mathematical transformations are used for multimedia signal processing due to the diverse nature of these signals. Specifically, multimedia signals can be time-dependent, i.e., the content changes over time (audio, video) or time-independent media (text, images). In addition to the Fourier analysis, the time-frequency and wavelet transforms are often used. In some cases, other advanced methods (e.g., the Hermite projection method) may be of interest as well. In this chapter, we will consider the basic principles of the commonly used signal transformations.

1.1 Fourier Transform

Fourier transform is one of the basic mathematical transformations used for multimedia signal processing, and many other mathematical transformations are based on the Fourier transform.

To understand Fourier transform, let us consider a simple example involving a sinusoidal signal, $f(t) = \cos(\omega_1 t)$, as shown in Fig. 1.1a.

The signal is completely defined by its frequency, initial phase, and amplitude. These three parameters can be obtained by the Fourier transform as depicted in Fig. 1.1b. Also, we may observe from Fig. 1.1b that a sinusoid is represented by two peaks in the frequency domain. This occurs due to the nature of the Fourier transform, namely, symmetrical components at negative frequencies appear for real signals. Hence, the signal is often transformed into its analytical form before processing.

Consider the signal in Fig. 1.2a. It is more beneficial to represent the signal in the frequency domain, since the signal consists of two sine waves of different frequencies and amplitudes (Fig. 1.2b).

The time domain representation can be especially difficult to interpret if the signal is corrupted by noise (e.g., white Gaussian noise, as shown in Fig. 1.3a). If the frequency domain representation is considered, it is easier to interpret the

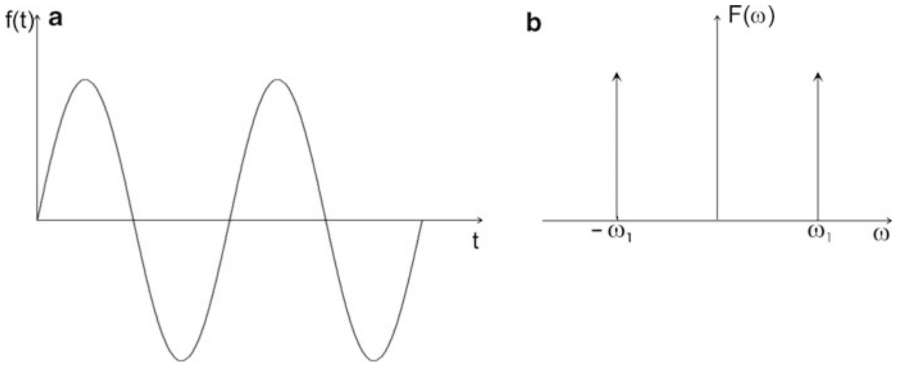


Fig. 1.1 Signal representations in: (**a**) Time domain; (**b**) Frequency domain

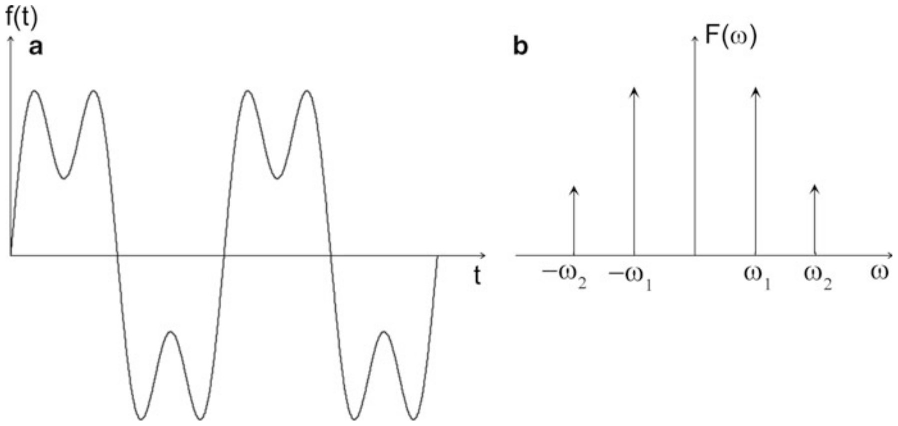


Fig. 1.2 Representations of a multicomponent signal: (**a**) Time domain representation; (**b**) Frequency domain representation

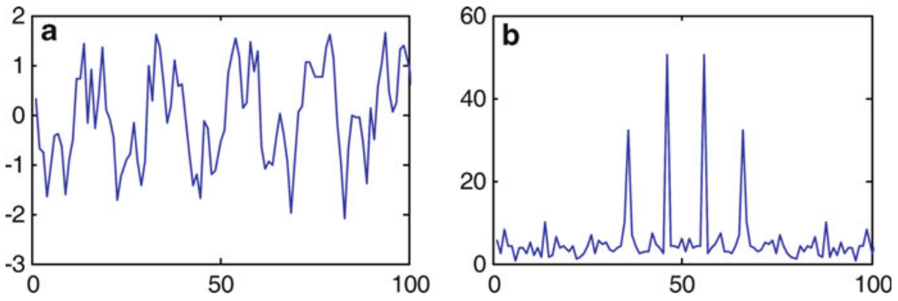


Fig. 1.3 Signal representation: (**a**) Time domain; (**b**) Frequency domain

signal parameters as shown in Fig. 1.3b. Specifically, the energy of noise is scattered across all frequencies, while the signal is concentrated at the frequencies of the sinusoidal components.

Let us introduce the mathematical definition of the Fourier transform for a signal *f*(*t*):

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\mathrm{d}t. \tag{1.1}$$

The inverse Fourier transform is used to obtain the time domain representation of the signal:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\mathrm{d}\omega. \tag{1.2}$$

Next, we briefly review some of the Fourier transform properties.

Linearity: The Fourier transform of a linear combination of signals is equal to the linear combination of their Fourier transforms:

$$\begin{aligned}\int_{-\infty}^{\infty}(\alpha f(t)+\beta g(t))e^{-j\omega t}\mathrm{d}t &= \alpha\int_{-\infty}^{\infty} f(t)e^{-j\omega t}\mathrm{d}t + \beta\int_{-\infty}^{\infty} g(t)e^{-j\omega t}\mathrm{d}t \\ &= \alpha F(\omega)+\beta G(\omega).\end{aligned} \tag{1.3}$$

In other words, $FT\{\alpha f(t)+\beta g(t)\} = \alpha FT\{f(t)\} + \beta FT\{g(t)\}$, where FT denotes the Fourier transform.

Time shift: Shifting the signal *f*(*t*) by t_0 in the time domain results in multiplying the Fourier transform with a phase factor:

$$\int_{-\infty}^{\infty} f(t-t_0)e^{-j\omega t}\mathrm{d}t = e^{-j\omega t_0}F(\omega). \tag{1.4}$$

Frequency shift: Modulating the signal with a complex exponential function shifts the Fourier transform *F*(ω) along the frequency axis:

$$\int_{-\infty}^{\infty}\left(e^{j\omega_0 t}f(t)\right)e^{-j\omega t}\mathrm{d}t = F(\omega-\omega_0). \tag{1.5}$$

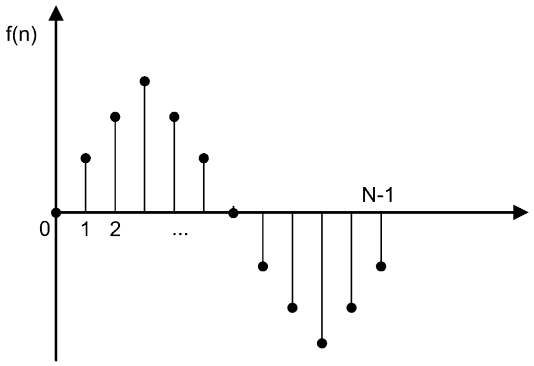


Fig. 1.4 Finite duration discrete signal

Convolution: The Fourier transform of convolution of two functions $f(t)$ and $g(t)$ is equal to the product of the Fourier transforms of the individual signals:

$$\mathrm{FT}\{f(t)_*g(t)\} = \mathrm{FT}\left\{ \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\mathrm{d}\tau \right\} = F(\omega)G(\omega). \tag{1.6}$$

On the other hand, the Fourier transform of the product of two signals equals to convolution of their Fourier transforms:

$$\mathrm{FT}\{f(t)\cdot g(t)\} = F(\omega)*_\omega G(\omega), \tag{1.7}$$

where $^*_\omega$ denotes the convolution in frequency domain.

1.1.1 Discrete Fourier Transform

Given that discrete signals are mainly used in applications, it is necessary to introduce the Fourier transform in its discrete form. Specifically, for a discrete signal (Fig. 1.4) of limited duration, the discrete Fourier transform is given by:

$$\mathrm{DFT}(k) = \sum_{n=0}^{N-1} f(n)e^{-j\frac{2\pi}{N}nk}. \tag{1.8}$$

The inverse discrete Fourier transform is defined as:

$$f(n) = \frac{1}{N}\sum_{k=0}^{N-1} \mathrm{DFT}(k)e^{j\frac{2\pi}{N}nk}. \tag{1.9}$$

It should be mentioned that computationally efficient algorithms have been derived based on the Fast Fourier Transform (FFT) algorithm to obtain discrete Fourier transform and its inverse.

To gain confidence in understanding the Fourier transform, we recommend working with problems provided at the end of this chapter.

1.1.2 Discrete Cosine Transform

Beside the Fourier transform, in many applications with real signals the discrete cosine transform (DCT) is used. The DCT is real-valued transform and represents the positive part of the spectrum. It is defined as:

$$\mathrm{DCT}(k) = c(k)\sum_{n=0}^{N-1} f(n)\cos\frac{(2n+1)k\pi}{2N}, \quad k = 0,...,N-1, \tag{1.10}$$

where the normalization coefficient $c(k)$ is:

$$c(k) = \begin{cases} \sqrt{1/N}, & k = 0 \\ \sqrt{2/N}, & k = 1,\ldots,N-1 \end{cases}.$$

The inverse DCT is given by:

$$f(n) = \sum_{k=0}^{N-1} c(k)\mathrm{DCT}(k)\cos\frac{(2n+1)k\pi}{2N}, \quad n = 0,...,N-1. \tag{1.11}$$

1.2 Filtering in the Frequency Domain

The frequency domain representation of signals is suitable for signal filtering, which can be done by using low-pass, high-pass, and/or band-pass filters. The ideal forms of these filters are defined as follows (Fig. 1.5):

Low-pass filter:

$$H(\omega) = \begin{cases} 1, & \text{for} \quad |\omega|<\omega_L, \\ 0, & \text{otherwise.} \end{cases} \tag{1.12}$$

High-pass filter:

$$H(\omega) = \begin{cases} 1, & \text{for} \quad |\omega|>\omega_H, \\ 0, & \text{otherwise.} \end{cases} \tag{1.13}$$

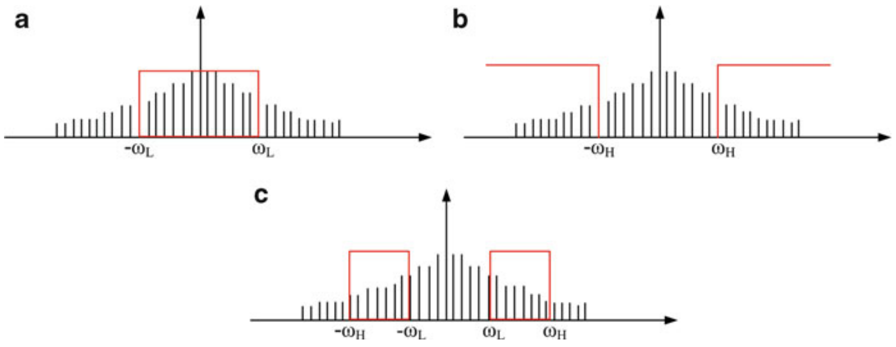


Fig. 1.5 (**a**) Low-pass filter, (**b**) High-pass filter, (**c**) Band-pass filter

Band-pass filter:

$$H(\omega) = \begin{cases} 1, & \text{for} \quad \omega_L < |\omega| < \omega_H, \\ 0, & \text{otherwise.} \end{cases} \tag{1.14}$$

Filtering in the frequency domain is simply performed by multiplying the Fourier transform of the signal with the filter transfer function. Then, the time domain representation of the filtered signal ($g(t)$) can be obtained by the inverse Fourier transform of their product:

$$\begin{gathered} G(\omega) = F(\omega)H(\omega), \\ g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{j\omega t}\mathrm{d}\omega\,. \end{gathered} \tag{1.15}$$

1.3 Time-Frequency Signal Analysis

Time-frequency analysis is used to represent signals with time-varying spectral content, since the Fourier transform does not provide sufficient information about these signals. Specifically, Fourier transform provides information about the frequency content of the signal, but there is no information about the time instants when spectral components appear. For example, using the Fourier transform to analyze a speech signal, we obtain the spectral content of spoken words, but not their timing.

Using a simple example, let us illustrate the advantages of using the time-frequency analysis in comparison to the Fourier transform of the signal.

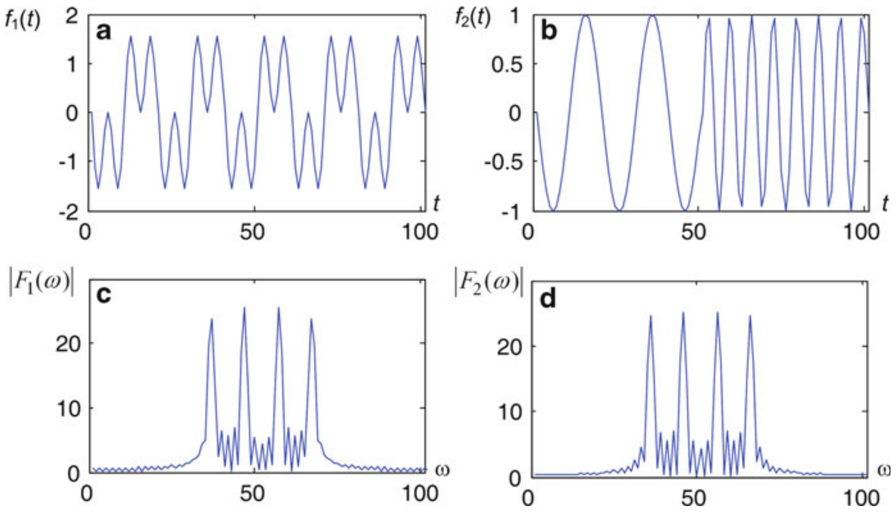


Fig. 1.6 (**a**) Sum of two sinusoids with equal duration, (**b**) Composition of two sinusoids appearing at different time instants, (**c**) Fourier transform for the first signal, (**d**) Fourier transform for the second signal

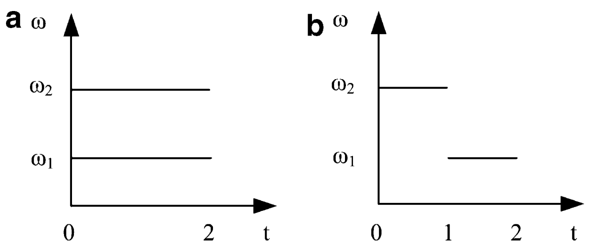


Fig. 1.7 (**a**) The ideal time-frequency representation of the sum of sinusoids from Fig. 1.6a, (**b**) The ideal time-frequency representation of the time-shifted sinusoids from Fig. 1.6b

For example, Fig. 1.6 depicts that the amplitude spectra (the amplitude of the Fourier transforms) of two different signals can be almost the same.

Hence, to obtain more information about these signals, it is necessary to use a representation from which one can follow temporal changes of the spectrum. Such a representation can be obtained by using the time-frequency analysis, as illustrated in Fig. 1.7.

Time-frequency distributions also provide information about the energy distribution around the instantaneous frequency. It is important to note that there is no single time-frequency distribution that is optimal for all nonstationary signals. In other words, different time-frequency distributions are used, depending on the application and on the signal type. The most commonly used distributions are the spectrogram and the Wigner distribution. The spectrogram is the squared module of the short-time Fourier transform, while the Wigner distribution is a quadratic distribution and exhibits significant drawbacks when applied to multicomponent signals that are often found in practical applications.

1.4 Ideal Time-Frequency Representation

Before we start considering various time-frequency representations, let us introduce an ideal time-frequency representation. Consider a signal defined as:

$$f(t) = Ae^{j\phi(t)}, \tag{1.16}$$

where A is the amplitude, and $\phi(t)$ is the phase of the signal. Note that the first phase derivative has the physical meaning and represents the instantaneous frequency, i.e., $\omega = \phi'(t)$. Therefore, the ideal time-frequency representation should concentrate energy along the instantaneous frequency of the signal and is defined as:

$$\mathrm{ITF}(t,\omega) = 2\pi A^2 \delta(\omega - \phi'(t)). \tag{1.17}$$

1.5 Short-Time Fourier Transform

The short-time Fourier transform (STFT) of a signal $f(t)$ is defined as:

$$\mathrm{STFT}(t,\omega) = \int_{-\infty}^{\infty} w(\tau) f(t+\tau) e^{-j\omega\tau} \mathrm{d}\tau, \tag{1.18}$$

where $w(t)$ is a window function. It provides the time-frequency representation by sliding the window and calculating the local spectrum for each windowed part of the signal, as illustrated in Fig. 1.8.

The STFT is a linear transform. In other words, the STFT of a multicomponent signal: $f(t) = \sum_{m=1}^{M} f_m(t)$, is equal to the sum of the STFTs of the individual components:

$$\mathrm{STFT}(t,\omega) = \sum_{m=1}^{M} \mathrm{STFT}_{f_m}(t,\omega). \tag{1.19}$$

This is an important feature of STFT, since many practical signals are the multicomponent ones.

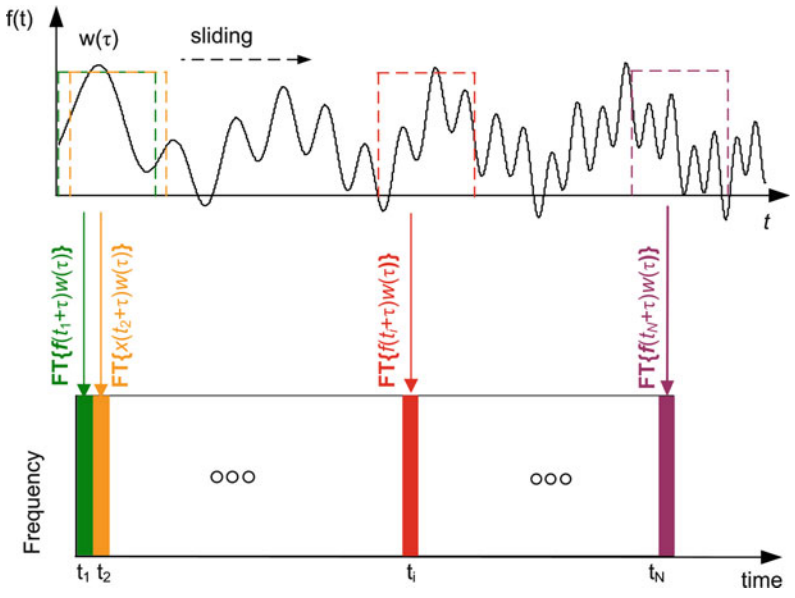


Fig. 1.8 An illustration of the STFT calculations

As previously mentioned, the spectrogram is the squared module of the STFT:

$$\mathrm{SPEC}(t,\omega) = |\mathrm{STFT}(t,\omega)|^2. \tag{1.20}$$

Unlike the STFT, the spectrogram is a real-value function. The main drawback of STFT (and the spectrogram) is the fact that the time-frequency resolution depends on the window width. Specifically, we obtain good time resolution (and poor frequency resolution) using a narrow window. On the other hand, a wider window enhances the frequency resolution, but decreases the time resolution. To illustrate this trade-off between time and frequency resolutions, let us consider the following example:

$$f(t) = \delta(t-t_1) + \delta(t-t_2) + e^{j\omega_1 t} + e^{j\omega_2 t}. \tag{1.21}$$

The ideal time-frequency representation of $f(t)$ is illustrated in Fig. 1.9.

Using the definition of STFT, we obtain the following time-frequency representation:

$$\begin{aligned}\mathrm{STFT}(t,\omega) =& w(t_1-t)e^{-j\omega(t_1-t)} + w(t_2-t)e^{-j\omega(t_2-t)} \\ &+ W(\omega-\omega_1)e^{j\omega_1 t} + W(\omega-\omega_2)e^{j\omega_2 t},\end{aligned} \tag{1.22}$$

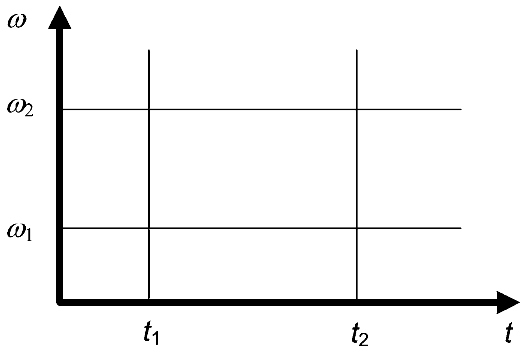


Fig. 1.9 Ideal time-frequency representation of signal *f*(*t*)

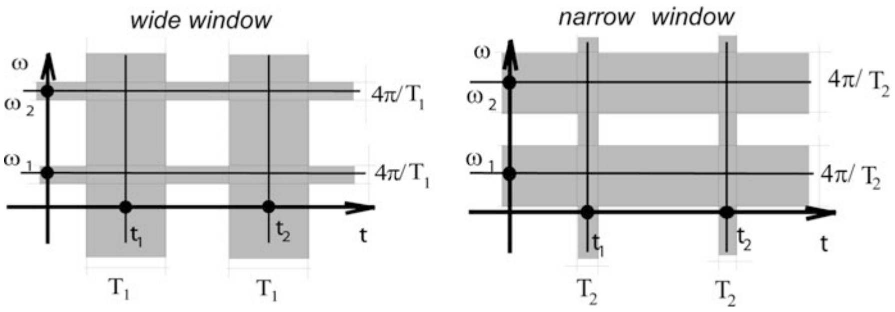


Fig. 1.10 Illustration of the uncertainty principle

where $W(\omega)$ is the Fourier transform of the window function. Figure 1.10 clearly shows the dependence of time-frequency representation on the window function. When using the rectangular window, the product of time and frequency resolutions for the considered example is $D{\cdot}d = 4\pi$, where d is the window width in the time domain, while D is the window width in the frequency domain. Hence, increasing the resolution in one domain decreases the resolution in other domain.

Generally, the uncertainty principle states that the product of measures of duration in time and frequency is:

$$M_T M_W \geq \frac{1}{2}, \tag{1.23}$$

where:

$$M_T = \frac{\int_{-\infty}^{\infty} \tau^2 |w(\tau)|^2 \mathrm{d}\tau}{\int_{-\infty}^{\infty} |w(\tau)|^2 \mathrm{d}\tau}, \quad M_w = \frac{\int_{-\infty}^{\infty} \omega^2 |W(\omega)|^2 \mathrm{d}\omega}{\int_{-\infty}^{\infty} |W(\omega)|^2 \mathrm{d}\omega}.$$

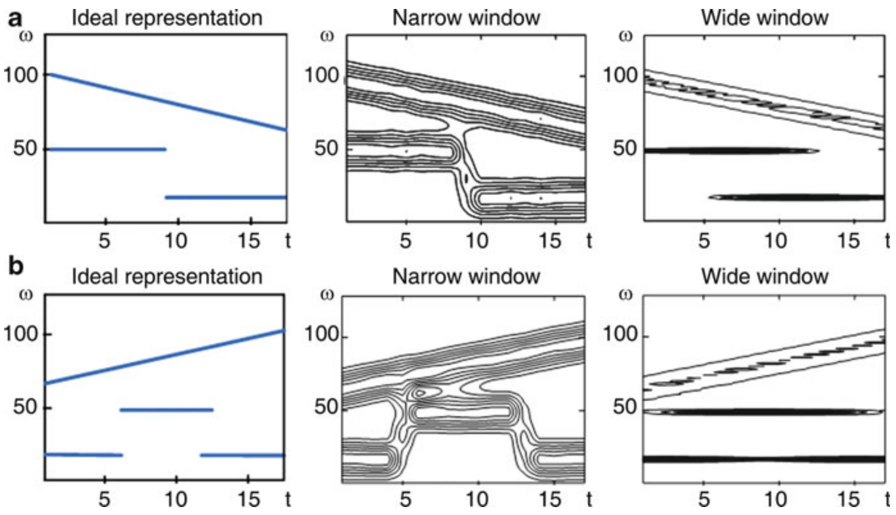


Fig. 1.11 Spectrograms of multicomponent signals: (**a**) Two sinusoids and chirp, (**b**) Three sinusoids and chirp

The signal should satisfy $w(t)\sqrt{t} \to 0 \; as \; t \to \pm\infty$. The lowest product is obtained for the Gaussian window: $M_T M_W = \frac{1}{2}$. In addition, Fig. 1.11 demonstrates time-frequency representations using wide and narrow windows of two multicomponent signals: the first one consists of two sinusoids and a linear-frequency modulated signal, i.e., chirp (Fig. 1.11a), while the second consists of three sinusoids and a chirp (Fig. 1.11b). The ideal time-frequency representations are presented as well.

Using a narrow window, we achieve good time resolution for sinusoidal signal components, as shown in the second column of Fig. 1.11. Using a wide window, good frequency resolution of these components is achieved, but the time resolution is significantly decreased. Notice that the time-frequency resolution of the chirp component is poor in both cases.

1.6 Wigner Distribution

In order to improve time-frequency representation, a number of quadratic distributions are introduced. A common requirement is that they meet the marginal conditions, which will be discussed in the sequel.

Given a time-frequency representation $P(t,\omega)$, the signal energy within the region $[(t, t + \Delta t), (\omega, \omega + \Delta\omega)]$ is equal to:

$$P(t, \omega)\frac{\Delta\omega}{2\pi}\Delta t. \tag{1.24}$$

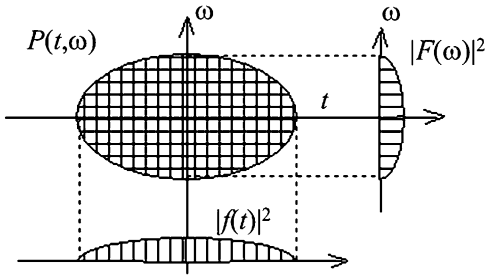


Fig. 1.12 Calculation of marginal conditions

Projections of the distribution on time and frequency axes provide spectral energy density and instantaneous power of the signal, respectively:

$$\int_{-\infty}^{\infty} P(t,\omega)\mathrm{d}t = |F(\omega)|^2,$$
$$\frac{1}{2\pi}\int_{-\infty}^{\infty} P(t,\omega)\mathrm{d}\omega = |f(t)|^2. \tag{1.25}$$

These conditions are known as marginal conditions (Fig. 1.12).

The signal energy can be obtained as:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(t,\omega)\mathrm{d}\omega\mathrm{d}t = \int_{-\infty}^{\infty} |f(t)|^2\mathrm{d}t = E_x. \tag{1.26}$$

One of the distributions that satisfy marginal conditions is the Wigner distribution, and it originated from the quantum mechanics. The distribution is defined as:

$$WD(t,\omega) = \int_{-\infty}^{\infty} R(t,\tau)e^{-j\omega\tau}\mathrm{d}\tau = \int_{-\infty}^{\infty} f\left(t+\frac{\tau}{2}\right)f^*\left(t-\frac{\tau}{2}\right)e^{-j\omega\tau}\mathrm{d}\tau, \tag{1.27}$$

where:

$$R(t,\tau) = f\left(t+\frac{\tau}{2}\right)f^*\left(t-\frac{\tau}{2}\right),$$

is the local autocorrelation function. In real applications, we use a windowed version of the Wigner distribution:

$$\mathrm{PWD}(t,\omega) = \int_{-\infty}^{\infty} w\left(\frac{\tau}{2}\right)w^*\left(-\frac{\tau}{2}\right)f\left(t+\frac{\tau}{2}\right)f^*\left(t-\frac{\tau}{2}\right)e^{-j\omega\tau}\mathrm{d}\tau, \tag{1.28}$$

which is often referred as the pseudo Wigner distribution (PWD). A window function does not play a significant role for PWD as for the STFT (or spectrogram). For example, it is possible to use a wider window and to keep good time resolution.

The Wigner distribution is a real-valued function and satisfies the marginal conditions, which makes it suitable for a wide range of applications. Let us consider the Wigner distribution of delta pulse, a sinusoidal signal, and a linear frequency modulated signal. For the delta pulse:

$$f(t) = A\delta(t - t_1), \tag{1.29}$$

the Wigner distribution equals to:

$$\mathrm{WD}(t,\omega) = 2\pi A^2 \delta(t - t_1). \tag{1.30}$$

For sinusoidal and chirp signals, we may still obtain the ideal representation by using the Wigner distribution:

$$\begin{array}{ll} f(t) = Ae^{j\omega_1 t} & f(t) = Ae^{jat^2/2} \\ \mathrm{WD}(t,\omega) = 2\pi A^2\delta(\omega - \omega_1) & \mathrm{WD}(t,\omega) = 2\pi A^2\delta(\omega - at) \end{array}$$

However, for a multicomponent signal $f(t) = \sum_{m=1}^{M} f_m(t)$, the Wigner distribution is:

$$\mathrm{WD}(t,\omega) = \sum_{m=1}^{M} \mathrm{WD}_{f_m f_m}(t,\omega) + \sum_{m=1}^{M} \sum_{\substack{n=1 \\ m \neq n}}^{M} \mathrm{WD}_{f_m f_n}(t,\omega). \tag{1.31}$$

Therefore, the Wigner distribution of the multicomponent signal equals to the sum of Wigner distributions of all signal components (auto-terms) and the Wigner distributions of quadratic terms obtained by multiplication of different signal components (f_m and f_n, $m \neq n$) called cross-terms. Hence, the Wigner distribution can be useless for time-frequency representations of multicomponent signals, since it can yield the time-frequency components that do not exist in the analyzed signal. For example, the Wigner distribution of a multicomponent signal, whose spectrogram is shown in Fig. 1.11a, is shown in Fig. 1.13a. The presence of strong cross-terms is obvious and they diminish the accuracy of the representation. However, when the cross-terms are removed, a concentrated representation is obtained as shown in Fig. 1.13b.

In order to reduce or completely eliminate the cross-terms, many distributions have been defined over the years. One such distribution is the S-method (SM),

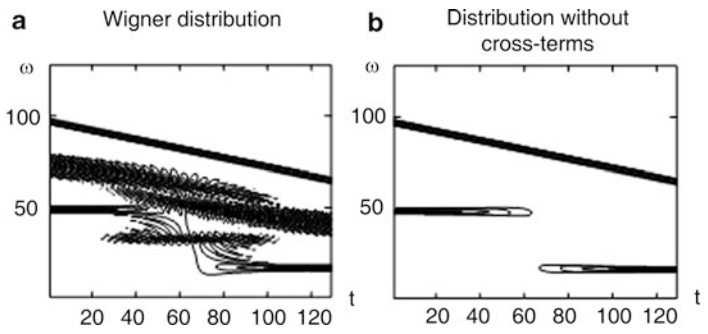


Fig. 1.13 (**a**) Wigner distribution of a multicomponent signal, (**b**) Auto-terms of the Wigner distribution

which combines good properties of the spectrogram and of the Wigner distribution. The SM is defined as:

$$\mathrm{SM}(t,\omega)=\frac{1}{\pi}\int_{-\infty}^{\infty}P(\theta)\mathrm{STFT}(t,\omega+\theta)\mathrm{STFT}^{*}(t,\omega-\theta)\mathrm{d}\theta, \tag{1.32}$$

and its discrete version is:

$$\begin{aligned}\mathrm{SM}(n,k)&=\sum_{i=-L_\mathrm{d}}^{L_\mathrm{d}}P(i)\mathrm{STFT}(n,k+i)\mathrm{STFT}^{*}(n,k-i)\\&=|\mathrm{STFT}(n,k)|^{2}+2\cdot\mathrm{Re}\left\{\sum_{i=1}^{L_\mathrm{d}}\mathrm{STFT}(n,k+i)\mathrm{STFT}^{*}(n,k-i)\right\},\end{aligned} \tag{1.33}$$

where the parameter L_d determines the frequency window length. In order to avoid the presence of cross-terms, the value of L_d should be less than half of the distance between two auto-terms. Note that the SM is suitable for hardware implementation.

Let us consider the previous example and its time-frequency representations obtained with the SM for various values of L_d (Fig. 1.14).

In many real applications, $L_\mathrm{d} = 3$ provides satisfactory results, since it eliminates the cross-terms and provides good concentration of the auto-terms, which almost equals the concentration achieved by the Wigner distribution. In addition, we consider the time-frequency representation of a speech signal obtained with the spectrogram and the SM, as shown in Fig. 1.15.

The time-frequency representation obtained with SM provides better temporal and frequency resolution, which allows us to obtain more accurate description and analysis of speech components.

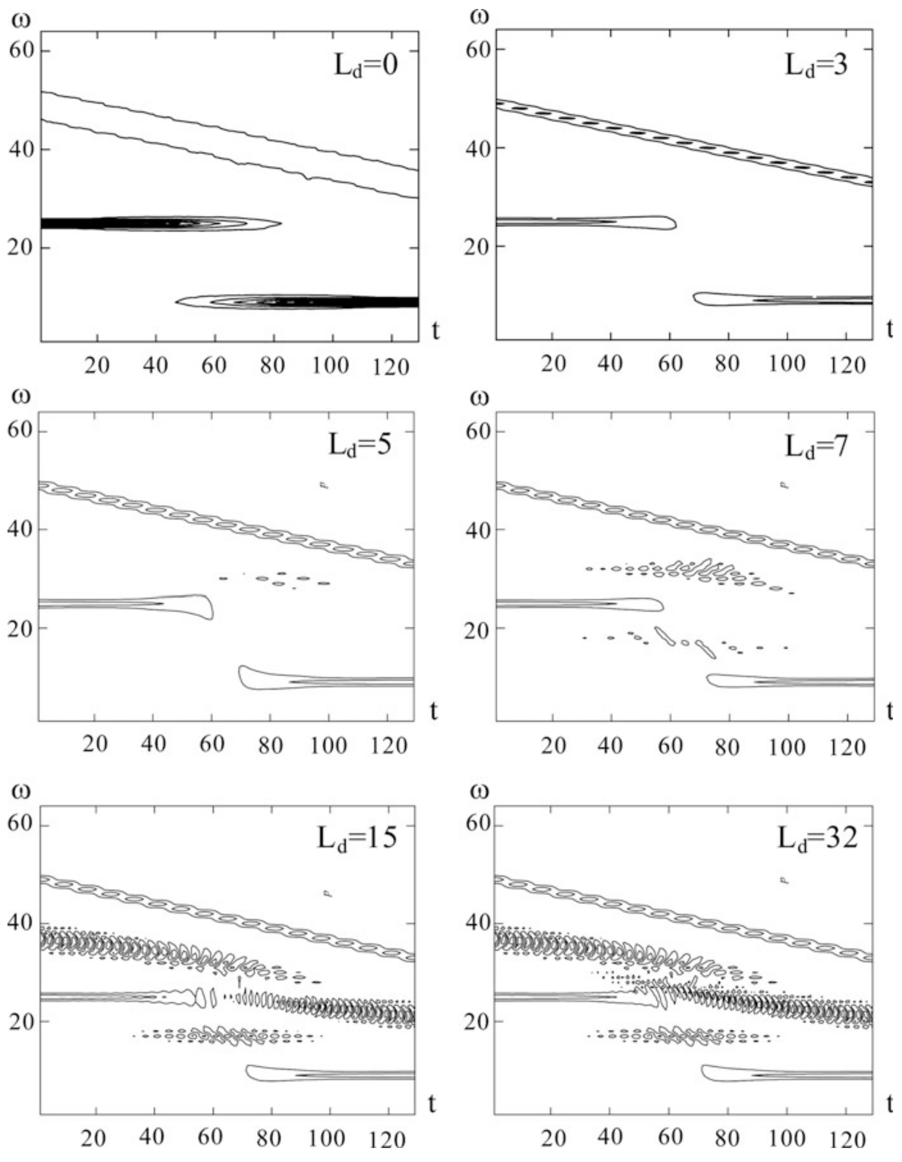


Fig. 1.14 Time-frequency representations of a multicomponent signal obtained by using SM for different values of window width (defined as $2L_d + 1$)

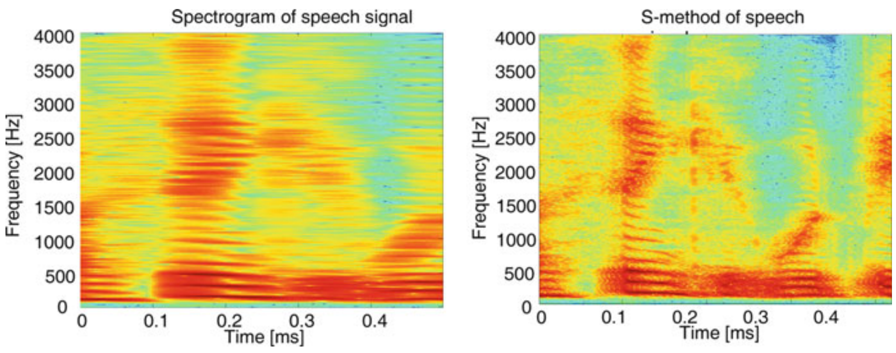


Fig. 1.15 Spectrogram and SM of speech signal

1.7 Time-Varying Filtering

For nonstationary signals, the time-varying filtering provides better results compared to the techniques performed in either the time of frequency domain separately. The time-varying filtering has been defined as:

$$(Hx)(t) = \int_{-\infty}^{\infty} h(t, t-\tau)x(\tau)\mathrm{d}\tau, \tag{1.34}$$

where $h(t,\tau)$ is the impulse response of the time-varying system H.

The optimal system form can be obtained by minimizing the mean squared error:

$$H_{\mathrm{opt}} = \arg\min_{H} E\left\{|f(t) - H[x(t)]|^2\right\}, \tag{1.35}$$

where $f(t)$ is the signal, while $x(t) = f(t) + v(t)$. Time-varying transfer function in the Wigner distribution framework has been defined as the Weyl symbol mapping of the impulse response into the time-frequency plane:

$$L_H(t,\omega) = \int_{-\infty}^{\infty} h\left(t+\frac{\tau}{2}, t-\frac{\tau}{2}\right)e^{-j\omega\tau}\mathrm{d}\tau. \tag{1.36}$$

Assuming that the signal and noise are uncorrelated, the optimal filter in the time-frequency domain is defined by:

$$L_H(t,\omega) = \frac{\overline{\mathrm{WD}}_{ff}(t,\omega)}{\overline{\mathrm{WD}}_{ff}(t,\omega) + \overline{\mathrm{WD}}_{vv}(t,\omega)}. \tag{1.37}$$

Here, $\overline{\mathrm{WD}}$ represents the mean value of the Wigner distributions for different realizations of the signal and noise. As a consequence of averaging, the cross-terms are significantly reduced as well as the noise. However, in practice, the time-varying filtering should be often performed on a single noisy realization. It means that for multicomponent signals, instead of the Wigner distribution, a cross-terms free distribution should be used (e.g., the S-method). The approximate filter, based on the cross-terms free and single distribution realization, provides satisfying results in many real applications.

Assuming that the Wigner distribution of the observed signal lies inside a region R, while the noise is outside this region, the support function is defined as:

$$L_H(t,\omega) = \begin{cases} 1, & \text{for} \quad (t,\omega) \in R, \\ 0, & \text{for} \quad (t,\omega) \notin R. \end{cases} \tag{1.38}$$

By using the Parseval's theorem, a form of time-varying filtering, appropriate for real-time applications, is obtained:

$$\begin{aligned}(Hx)(t) &= \int_{-\infty}^{\infty} h\left(t+\frac{\tau}{2}, t-\frac{\tau}{2}\right) w(\tau) x(t+\tau)\mathrm{d}\tau \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} L_H(t,\omega)\mathrm{STFT}(t,\omega) d\omega.\end{aligned} \tag{1.39}$$

1.8 Wavelet Transform

1.8.1 Continuous Wavelet Transform

Wavelets are mathematical functions formed by scaling and translation of basis functions $\psi(t)$ in the time domain. $\psi(t)$ is also called the mother wavelet and satisfies the following conditions:

1. The total area under the curve $\psi(t)$ is equal to zero:

$$\int_{-\infty}^{\infty} \psi(t)\mathrm{d}t = 0. \tag{1.40}$$

2. The function has a finite energy, i.e., it is square-integrable:

$$\int_{-\infty}^{\infty} |\psi(t)|^2 \mathrm{d}t < \infty. \tag{1.41}$$

The wavelet is defined by the formula:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right), \tag{1.42}$$

where a and b are two arbitrary real numbers used as scaling and translation parameters, respectively. The factor $a^{-1/2}$ also represents a normalization factor, which allows the energy of the wavelet function to remain independent of parameter a. For the values $0 < a < 1$, the basis function shrinks in time, while for $a > 1$, it spreads in time.

The wavelet transform of the signal $f(t)$ is mathematically described by the expression:

$$W(a,b) = \int_{-\infty}^{\infty} \psi_{a,b}(t)f(t)\mathrm{d}t. \tag{1.43}$$

$W(a,b)$ is called the continuous wavelet transform (CWT), where a and b are continuous variables and $f(t)$ is a continuous function. The inverse wavelet transform is obtained as:

$$f(t) = \frac{1}{C}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \psi_{a,b}(t)W(a,b)\mathrm{d}a\mathrm{d}b, \tag{1.44}$$

where:

$$C = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{\omega}\mathrm{d}\omega, \tag{1.45}$$

and $\Psi(\omega)$ is the Fourier transform of $\psi(t)$. The inverse continuous wavelet transform exists if the parameter C is positive and finite.

1.8.2 Wavelet Transform with Discrete Wavelet Functions

In practical applications, the parameters a and b are discretized (i.e., scaling and translation are performed in discrete steps). For discretization of parameter a, we choose powers of fixed dilation parameter $a_0 > 1$:

$$a = a_0^m, \quad \text{where } m \in \mathbf{Z}$$

while for b, we use:

$$b = nb_0 a_0^m, \quad n \in \mathbf{Z},\ b_0 > 0.$$

By using the discrete parameters a and b, we obtain the discretized family of wavelets:

$$\psi_{m,n}(t) = a_0^{-m/2} \psi\left(a_0^{-m} t - nb_0\right). \tag{1.46}$$

The corresponding wavelet transform is then given by:

$$W^d_{m,n} = a_0^{-m/2} \int_{-\infty}^{\infty} f(t)\psi\left(a_0^{-m} t - nb_0\right) \mathrm{d}t. \tag{1.47}$$

If $a_0 = 2$ and $b_0 = 1$ are used, we achieve the dyadic sampling. The corresponding signal decomposition is called the dyadic decomposition. In such a case, the discrete wavelet functions with the given parameters form a set of orthonormal basis functions:

$$\psi_{m,n}(t) = 2^{-m/2} \psi(2^{-m} t - n). \tag{1.48}$$

Therefore, the dyadic wavelet transform is calculated as:

$$W^d_{m,n} = 2^{-m/2} \int_{-\infty}^{\infty} f(t)\psi(2^{-m} t - n) \mathrm{d}t. \tag{1.49}$$

1.8.3 Wavelet Families

Wavelets are widely applied in image processing, biomedical signal processing, audio signal processing, just to name a few. Let us mention some of the most commonly used wavelets:

- Haar wavelet is the oldest and the simplest wavelet.
- Daubechies wavelets represent a set of orthonormal wavelets of limited duration. For example, Daubechies D4 wavelet has four coefficients; D8 has eight coefficients, and so on. Note that the Haar wavelet is actually the Daubechies wavelet of the first order.
- Biorthogonal wavelets are widely used in image compression. For example, JPEG2000 compression algorithm is based on biorthogonal Le Gall (5,3) and Cohen-Daubechies-Feauveau (CDF) (9,7) wavelets.

- Mexican Hat wavelet is a wavelet function that equals to the second derivative of the Gaussian function.
- Symlets are symmetric wavelets, created as a modification of the Daubechies wavelet.
- Morlet wavelet is based on a modulated Gaussian function.

1.8.4 Haar Wavelet

The Haar wavelet function is defined as:

$$\psi(t) = \begin{cases} 1, & 0 \le t < 0.5 \\ -1, & 0.5 \le t < 1. \end{cases} \tag{1.50}$$

Let us consider its scaled and shifted version, in the form:

$$\begin{gathered} \psi_{j,k}(t) = 2^{\frac{j}{2}}\psi(2^j t - k), \\ j = 0, \pm 1, \pm 2, \ldots, \quad k = 0, \pm 1, \ldots, 2^j - 1 \end{gathered} \tag{1.51}$$

where the scaling parameter is $a = 2^{-j}$, while the translation parameter is $b = 2^{-j}k$. The parameter j represents the scale. Greater j values shrink the basis function in time. In addition, for each scale, the basis function translates in time by k, as depicted in Fig. 1.16.

Note that the Haar wavelets are orthogonal functions:

$$<\psi_{j,k}, \psi_{j',k'}> = \int \psi_{j,k}(t)\psi_{j',k'}(t)\mathrm{d}t = \begin{cases} 1, & \text{for } j = j', k = k' \\ 0, & \text{otherwise.} \end{cases} \tag{1.52}$$

The Haar wavelets can be used to represent the function $f(t)$ as follows:

$$f(t) = \sum_{j,k} <f, \psi_{j,k}>\psi_{j,k}(t), \tag{1.53}$$

where $d_{j,k} = <f, \psi_{j,k}>$ denotes the Haar wavelet coefficients. In order to use the discrete values of f, the samples on the scale j can be taken as the mean values on the interval of length 2^{-j}, and we have that:

$$s_{j,k} = 2^{j/2} \int_0^{2^{-j}} f\left(t + 2^{-j}k\right)\mathrm{d}t = <f, \varphi_{j,k}>. \tag{1.54}$$

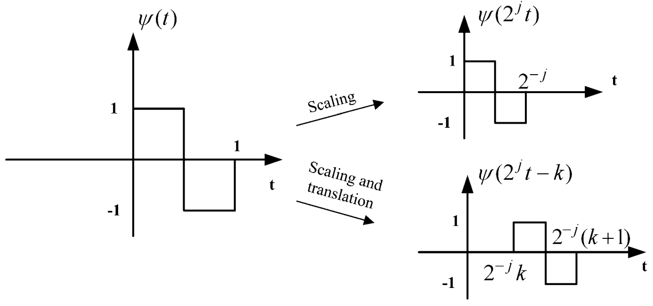


Fig. 1.16 Wavelet functions

Therefore, the $2^{j/2}s_{j,k}$ is the mean value of f within the kth time interval $[2^{-j}k,\ 2^{-j}(k+1)]$. The function φ in the previous relation is the basic scaling function:

$$\varphi(t)=\begin{cases}1,\ 0\le t<1\\ 0,\ \text{otherwise.}\end{cases} \tag{1.55}$$

while the $\varphi_{j,k}(t)=2^{\frac{j}{2}}\varphi(2^j t-k)$. Consider now the difference between two adjacent samples:

$$s_{j,2k}-s_{j,2k+1}=\int f(t)2^{j/2}\left(\varphi\left(2^j t-2k\right)-\varphi\left(2^j t-(2k+1)\right)\right)\mathrm{d}t. \tag{1.56}$$

The difference between the scaling functions is:

$$\varphi\left(2^j t-2k\right)-\varphi\left(2^j t-(2k+1)\right)=\begin{cases}1-0=1,(2^j t-2k)\in(0,1]\\ 0-1=-1,(2^j t-(2k+1))\in(0,1]\end{cases}=$$
$$=\begin{cases}1, & t\in\left(\frac{2k}{2^j},\frac{2k+1}{2^j}\right]\\ -1, & t\in\left(\frac{2k+1}{2^j},\frac{2k+2}{2^j}\right]\end{cases}=\begin{cases}1, & t\in\left(2^{-(j-1)}k,\frac{2k+1}{2^j}\right]\\ -1, & t\in\left(\frac{2k+1}{2^j},2^{-(j-1)}(k+1)\right)\end{cases}$$

Therefore, we have:

$$\varphi\left(2^j t-2k\right)-\varphi\left(2^j t-(2k+1)\right)=\psi\left(2^{j-1}t-k\right), \tag{1.57}$$

and the difference between the mean values can be expressed as:

$$\begin{aligned}s_{j,2k}-s_{j,2k+1}&=\int f(t)2^{j/2}\psi\left(2^{j-1}t-2k\right)\mathrm{d}t\\ &=\sqrt{2}\int f(t)\left[2^{(j-1)/2}\psi\left(2^{j-1}t-2k\right)\right]\mathrm{d}t,\end{aligned} \tag{1.58}$$

or,

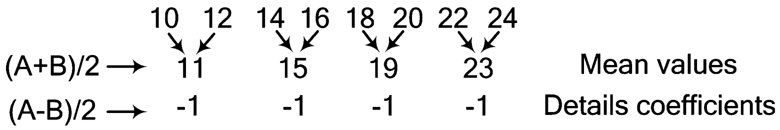
$$\frac{1}{\sqrt{2}}\left(s_{j,2k} - s_{j,2k+1}\right) = d_{j-1,k}. \tag{1.59}$$

To simplify the calculation of the Haar wavelet transform, we use an approach based on the calculation of mean values and their differences. In other words, samples at a certain level of decomposition are obtained as the mean of samples from the previous level, and the differences among samples can be interpreted as details (wavelet coefficients).

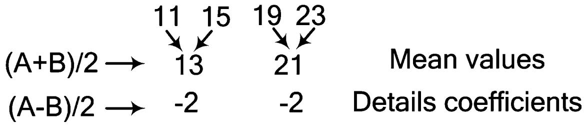
Let us illustrate how this simplified algorithm can be used to decompose a row of image pixels:

$$\{10, 12, 14, 16, 18, 20, 22, 24\}.$$

First, we find the mean value of pairs. As it is impossible to reconstruct the original sequence from the obtained mean values, we also calculate pixel differences representing the coefficients of details.



Clearly, the newly created pixel vector {11, 15, 19, 23, −1, −1, −1, −1} can be used to completely reconstruct the image row. In the next level, we use four mean values and obtain two new mean and two new detail coefficients.



The new vector has the values {13, 21, −2, −2, −1, −1, −1, −1}.

Then carry out the decomposition of the remaining two mean values, from which we get a vector {17,−4, −2, −2, −1, −1, −1, −1}. The last step is the wavelet transform normalization by using the parameter $2^{-j/2}$:

$$\left\{\frac{17}{\sqrt{2^0}}, \frac{-4}{\sqrt{2^0}}, \frac{-2}{\sqrt{2^1}}, \frac{-2}{\sqrt{2^1}}, \frac{-1}{\sqrt{2^2}}, \frac{-1}{\sqrt{2^2}}, \frac{-1}{\sqrt{2^2}}, \frac{-1}{\sqrt{2^2}}\right\}.$$

The same procedure can be applied to every image row, and thus the whole image can be decomposed. The mean values produce a lower resolution image from which the details are removed.

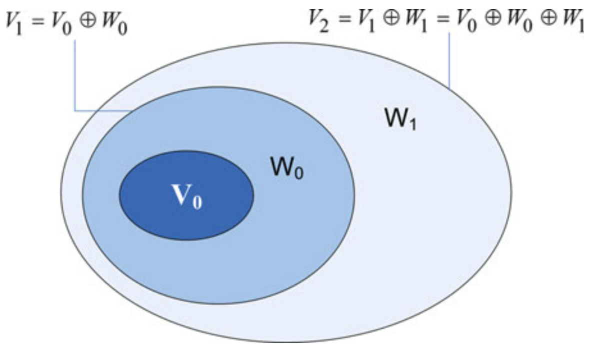


Fig. 1.17 Approximation (scaling) and wavelet spaces

1.8.5 Multiresolution Analysis

The previous considerations can be generalized by using the concepts of multiresolution analysis described in the sequel.

The Hilbert space can be represented as a decomposition of approximation spaces V_j, $j \in \mathbf{Z}$, where each approximation space represents the scaled version of the basic space V_0. Note that the approximation V_{j-1} contains more details compared to V_j, which are modeled by the wavelet space $W_j : V_{j-1} = V_j \oplus W_j$ (Fig. 1.17), where $\oplus$ denotes the orthogonal summation.

Therefore, the expansion functions from a certain space can be derived using the expansion functions from double-resolution space, which can be written in terms of dilation equation as follows:

$$\varphi(t) = \sum_k s(k)\sqrt{2}\varphi(2t-k), \tag{1.60}$$

while the wavelet equation is:

$$\psi(t) = \sum_k d(k)\sqrt{2}\varphi(2t-k). \tag{1.61}$$

Starting from the coefficients $s(k)$ (which satisfy certain conditions), we solve the dilation equation to obtain the basis function $\varphi_{j,k}(t)$ of space V_j. Then the coefficients $d(k)$ should be determined (usually depending on the choice of $s(k)$), and thus, the basis $\psi_{j,\,k}(t)$ of wavelet space W_j is determined.

Now, observe the approximation or projection of the function $f(t)$ in the space V_j given by:

$$f_j(t) = \sum_k \left\langle f_j, \varphi_{j,k} \right\rangle \varphi_{j,k}, \tag{1.62}$$

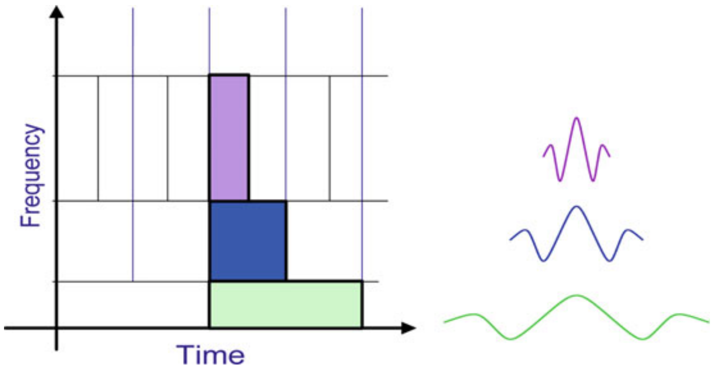


Fig. 1.18 Time-frequency representation of wavelet

The details that remain after the approximation in V_j are modeled by:

$$\Delta f_j(t) = \sum_k \left\langle f_j, \psi_{j,k} \right\rangle \psi_{j,k}, \tag{1.63}$$

such that:

$$f_{j-1}(t) = f_j(t) + \Delta f_j(t). \tag{1.64}$$

Therefore, we may observe that $f_j(t) \to f(t)\ for\ j \to \infty$, or in other words we have:

$$f(t) = f_J(t) + \sum_{j=-\infty}^{J} \Delta f_j(t) \quad or \quad f(t) = \sum_{j=-\infty}^{-\infty} \Delta f_j(t), \tag{1.65}$$

where J denotes the lowest resolution scale. Thus, the Hilbert space can be observed as a composition of approximation space V_J and the infinite set of wavelet spaces W_j, $(j = J + 1, \ldots, \infty)$.

The wavelet decomposition of a signal can be described by using a set of coefficients, each providing the information about time and frequency localization of the signal. However, the uncertainty principle prevents us from a precise localization in both time and frequency. For example, the Haar wavelet is well-localized in time, but supports a wide frequency band. The Mexican wavelet is well-localized in the frequency domain, but not in the time domain.

We can conclude that the multiresolution analysis allows the discrete wavelet transform to decompose the signal into different subbands. The subbands at lower frequencies have better frequency resolution and poor time resolution, while the subbands at higher frequencies have better time resolution and poor frequency resolution, as illustrated in Fig. 1.18.

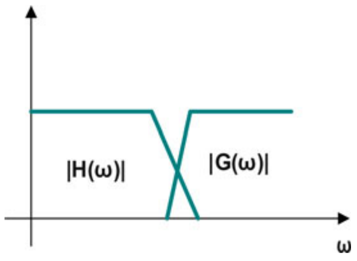


Fig. 1.19 Analysis filters

1.8.6 Filter Bank

Using the multiresolution analysis, the signal can be decomposed into two parts: one representing the approximation of the original signal and the other containing information about the details. Thus, the signal can be represented by the formula:

$$f_m(t) = \sum_n \alpha_{m+1,n}\,\varphi_{m+1,n} + \sum_n \beta_{m+1,n}\,\psi_{m+1,n}, \tag{1.66}$$

where $\alpha_{m+1,n}$ are the approximation coefficients at resolution 2^{m+1}, while $\beta_{m+1,n}$ are the coefficients of details. The functions $\varphi_{m+1,n}$ and $\psi_{m+1,n}$ represent the scaling and wavelet function, respectively. The recursive realization of discrete wavelet transform in different levels can be written as:

$$\begin{aligned}\alpha_{m,n}(f) &= \sum_k h_{2n-k}\alpha_{m-1,k}(f),\\ \beta_{m,n}(f) &= \sum_k g_{2n-k}\alpha_{m-1,k}(f),\end{aligned} \tag{1.67}$$

where h and g are low-pass and high-pass filters, respectively (Fig. 1.19), often referred to as *analysis filters*: $h_i = 2^{1/2}\int \varphi(x-i)\varphi(2x)dx$, $g_i = (-1)^i h_{-i+1}$.

Since h and g are defined from the orthonormal basis functions, they provide exact reconstruction:

$$\alpha_{m-1,i}(f) = \sum_n h_{2n-i}\alpha_{m,n}(f) + \sum_n g_{2n-i}\beta_{m,n}(f). \tag{1.68}$$

Theoretically, for many orthonormal basis wavelet functions, there are large number of filters that can be used for their implementation. In practice, FIR (finite impulse response) filters are used to implement the wavelets efficiently.

Synthesis filters (h' and g') are used for the signal reconstruction. Namely, the signal decomposition is done by using (1.67), while the reconstruction is now given by the following expression:

$$\alpha_{m-1,i}(f) = \sum_n \alpha_{m,n}(f)h'_{2n-i} + \sum_n \beta_{m,n}(f)g'_{2n-i}. \tag{1.69}$$

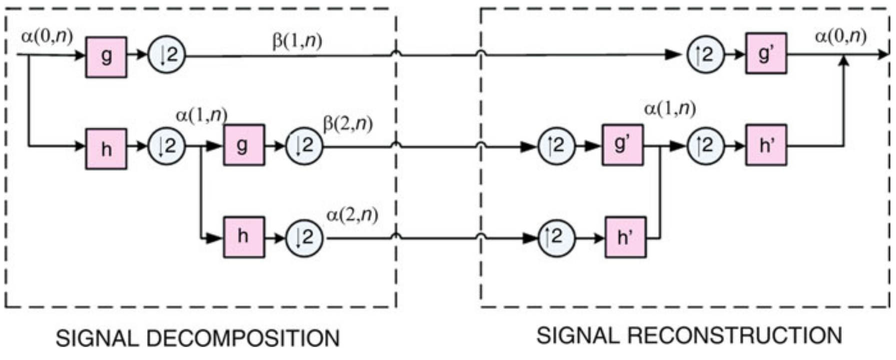


Fig. 1.20 Wavelet decomposition and reconstruction of signals using a filter bank

If $(h',g') = (h,g)$ holds, the filters are orthogonal. Otherwise, they are biorthogonal. Figure 1.20 illustrates the concept of filter banks. The input signal $\alpha(0,n)$ is filtered in parallel by a low-pass filter h and a high-pass filter g. The signal is downsampled after passing through the filters. Therefore, the output from the analysis filters at the first level of decomposition is given by:

$$(\downarrow \alpha(1,n)) = (\ldots,\alpha(1,-6),\alpha(1,-4),\alpha(1,-2),\alpha(1,0),\alpha(1,2),\alpha(1,4),\alpha(1,6),\ldots)$$

Before passing through the synthesis filters, the signal has to be upsampled:

$$\begin{aligned}(\uparrow \alpha(1,n)) = (&\ldots,\alpha(1,-6),0,\alpha(1,-4),0,\alpha(1,-2),0,\alpha(1,0),0,\alpha(1,2),\\ &0,a(1,4),0,\alpha(1,6),\ldots)\end{aligned}$$

1.8.7 Daubechies Orthogonal Filters

The Daubechies filter family is often used for the construction of orthogonal discrete wavelets. Suppose that the filter bank consists of the analysis filters h and g and the synthesis filters h' and g' of length N (N is even). The impulse responses of filters are then given by:

$$\begin{aligned}h &=(h(0),h(1),\ldots,h(N-1)),\\ g &=(g(0),g(1),\ldots,g(N-1)),\\ h' &=(h'(0),h'(1),\ldots,h'(N-1)),\\ g' &=(g'(0),g'(1),\ldots,g'(N-1)).\end{aligned}$$

The Daubechies filters satisfy the following conditions:

1. The vector $\boldsymbol{h}$ is normalized;
2. For each integer that satisfies $1 \leq n < N/2$, the vector formed by the first $2n$ elements of h should be orthogonal to the vector containing the last $2n$ elements of the same h;
3. The filter $\boldsymbol{h}'$ is the flipped version of $\boldsymbol{h}$***;***
4. Vector $\boldsymbol{g}$ is formed based on $\boldsymbol{h}'$ by multiplying the vector elements with -1 on even positions;
5. Vector $\boldsymbol{g}'$ is obtained from $\boldsymbol{h}$ by inverting the sign of the elements on odd positions;
6. The frequency response of the filter is equal to $\sqrt{2}$ for $\omega = 0$:

$$H(0) = \sqrt{2}.$$

7. The kth derivative of the filter is equal to zero for $\omega = \pi$:

$$H^{(k)}(\pi) = 0 \quad for \quad k = 0, 1, 2, \ldots, \frac{N}{2} - 1.$$

Suppose that the length of the filters is $N = 4$. Then, by using the above conditions we get:

Condition 1: $h^2(0) + h^2(1) + h^2(2) + h^2(3) = 1$,
Condition 2: $h(0)h(2) + h(1)h(3) = 0$,
Condition 3: $h' = (h(3), h(2), h(1), h(0))$,
Condition 4: $g = (h(3), -h(2), h(1), -h(0))$,
Condition 5: $g' = (-h(0), h(1), -h(2), h(3))$.
Condition 6: The filter h in the frequency domain can be written in the form of Fourier series:

$$H(\omega) = h(0) + h(1)e^{-i\omega} + h(2)e^{-2i\omega} + h(3)e^{-3i\omega}.$$

It is necessary to construct a filter that satisfies the condition $H(0) = \sqrt{2}$. If we set $\omega = 0$, we get:

$$\sqrt{2} = h(0) + h(1) + h(2) + h(3).$$

Based on the condition 7, we obtain:

$$\left.\begin{array}{l} H^{(0)}(\omega) = h(0) + h(1)e^{-i\omega} + h(2)e^{-2i\omega} + h(3)e^{-3i\omega} \\ H^{(0)}(\pi) = 0 \end{array}\right\} \boxed{h(0) - h(1) + h(2) - h(3) = 0}$$

$$\left.\begin{array}{l} H^{(1)}(\omega) = ih(1)e^{-i\omega} + 2ih(2)e^{-2i\omega} + 3ih(3)e^{-3i\omega} \\ H^{(1)}(\pi) \end{array}\right\} \boxed{-h(1) + 2h(2) - 3h(3) = 0}$$

Table 1.1 Coefficients of Daubechies D4 low-pass analysis filter *h*

h(0)	*h(1)*	*h(2)*	*h(3)*
−0.12941	0.224144	0.836516	0.482963

Hence, a system of equations is obtained, which can be used to calculate the coefficients:

$$\boxed{\begin{array}{l} h^2(0)+h^2(1)+h^2(2)+h^2(3)=1 \\ h(0)h(2)+h(1)h(3)=0 \\ h(0)+h(1)+h(2)+h(3)=\sqrt{2} \\ h(0)-h(1)+h(2)-h(3)=0 \\ -h(1)+2h(2)-3h(3)=0 \end{array}}$$

The system has two solutions. The first solution represents the coefficients of a low-pass analysis filter, and the second solution represents the coefficients of a low-pass synthesis filter (Table 1.1).

1.8.8 Two-Dimensional Signals

The two-dimensional discrete wavelet transform is generally used to decompose two-dimensional signals (e.g., images). Consider the two-dimensional separable scaling and wavelet functions. They can be represented as the product of one-dimensional functions $\varphi(x,y)=\varphi(x)\varphi(y)$ and $\psi(x,y)=\psi(x)\psi(y)$, enabling the application of the one-dimensional discrete wavelet transform separately to the rows and columns of a two-dimensional matrix. Several different approaches to analyze two-dimensional signals using the discrete wavelet transform are described below.

- *Standard wavelet decomposition*
  The first step of decomposition involves creating a low-frequency subband L_1 and a high-frequency subband H_1. The same procedure is then carried out over low-frequency subband L_1 by forming subbands L_2 and H_2. We continue this procedure until we reach a desired number of subbands. The second step involves the same procedure for the columns. The end result is a low-pass coefficient in the upper left corner. The decomposition is illustrated in Fig. 1.21.
- *Quincunx decomposition*
  This decomposition uses each low-frequency subband on the level *i* (L_i) and divides it into subbands L_{i+1} and H_{i+1} (subbands on the level $i+1$). Figure 1.22 illustrates the Quincunx decomposition.

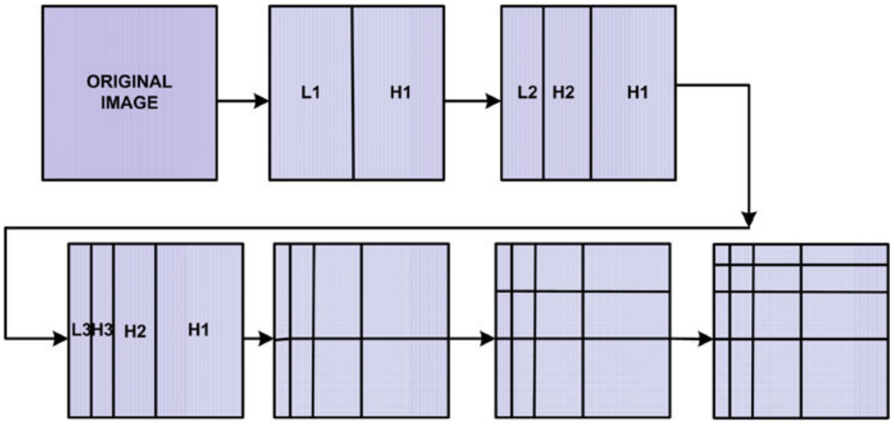


Fig. 1.21 Standard wavelet decomposition

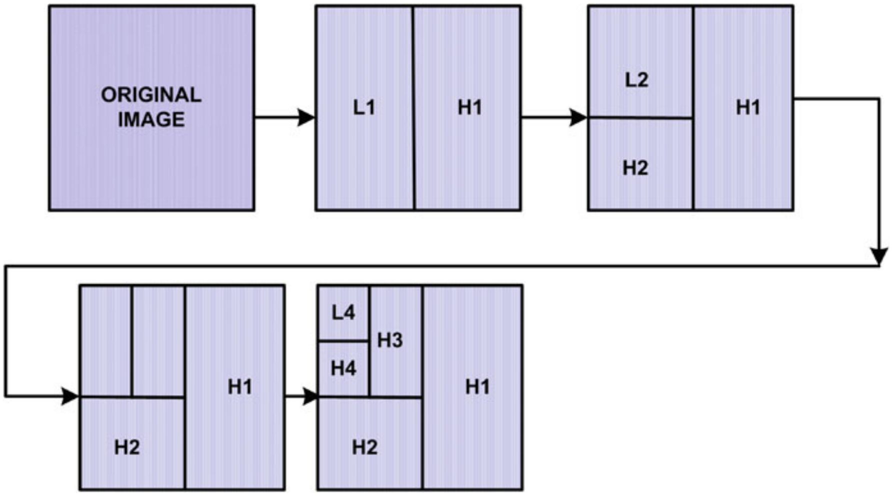


Fig. 1.22 Quincunx decomposition

- *Pyramidal wavelet decomposition*
  The most commonly used decomposition method in practical applications is the pyramidal decomposition as shown in Fig. 1.23. Suppose that the image dimensions are $M \times N$. Initially, the one-dimensional wavelet transform is performed for image rows and subbands L_1 and H_1 are obtained. Then the wavelet transform is performed for each column resulting in four subbands LL_1, LH_1, HL_1, HH_1 with dimensions equal to $M/2 \times N/2$. The LL_1 subband represents a version of the original image with lower resolution. The LL_1 subband is then

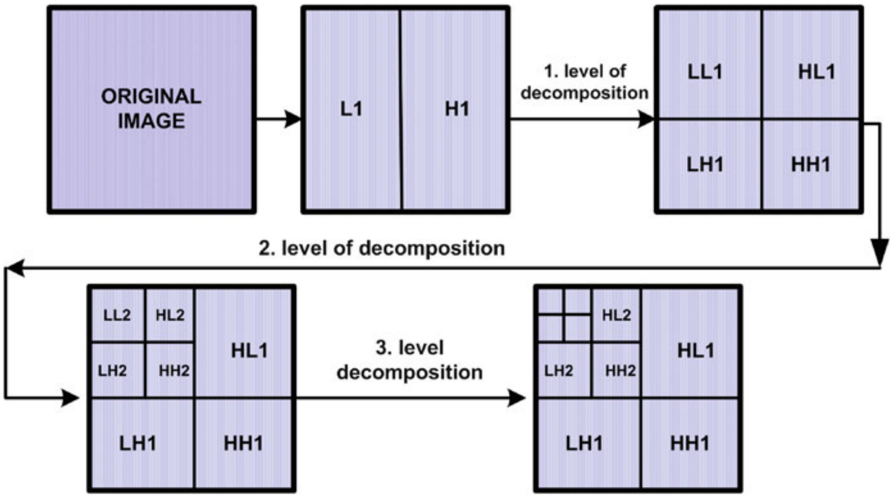


Fig. 1.23 Pyramidal wavelet decomposition

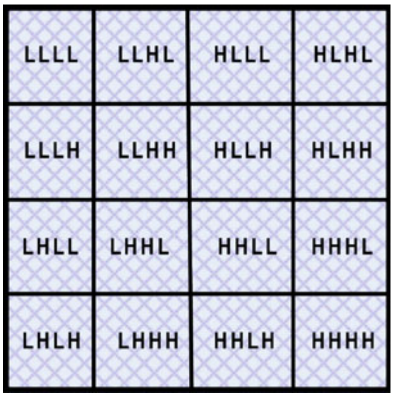


Fig. 1.24 Uniform wavelet decomposition

further decomposed into subbands LL_2, LH_2, HL_2 and HH_2. If further decomposition is needed, it would be based on the low-pass subbands LL_i.

- *Uniform wavelet decomposition*
  This decomposition is initially performed for rows and columns, producing the four subbands. Then, the same procedure is repeated for each subband and 16 new subbands are obtained. Figure 1.24 illustrates this process for two levels of decomposition.

1.9 Signal Decomposition Using Hermite Functions

Projecting signals using the Hermite functions is widely used in various image and signal processing applications (e.g., image filtering, texture analysis, speaker identification). The Hermite functions allow us to obtain good localization of the signals in both the signal and transform domains. These functions are defined as follows (Fig. 1.25):

$$\Psi_0(x) = \frac{1}{\sqrt[4]{\pi}} e^{-x^2/2}, \qquad \Psi_1(x) = \frac{\sqrt{2}x}{\sqrt[4]{\pi}} e^{-x^2/2},$$
$$\Psi_p(x) = x\sqrt{\frac{2}{p}}\Psi_{p-1}(x) - \sqrt{\frac{p-1}{p}}\Psi_{p-2}(x), \quad \forall p \geq 2. \tag{1.70}$$

1.9.1 One-Dimensional Signals and Hermite Functions

Assume that X is a discrete one-dimensional signal (of length M) that will be expanded by using the Hermite functions. The first step in the Hermite projection method is to remove the baseline, defined as:

$$x(i) = X(1) + \frac{X(M) - X(1)}{M} \cdot i, \tag{1.71}$$

where $i = 1,\ldots, M$. The baseline is then subtracted from the original signal:

$$f(i) = X(i) - x(i). \tag{1.72}$$

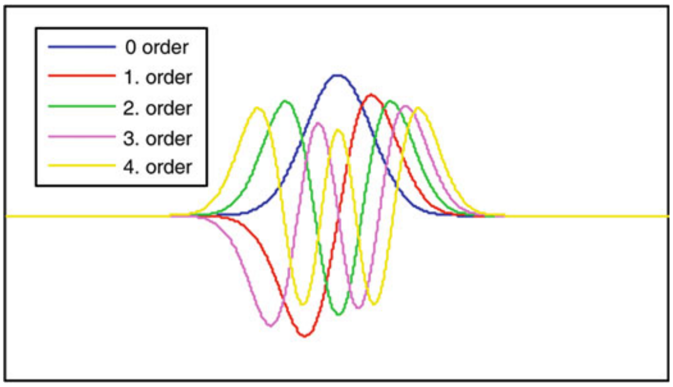


Fig. 1.25 Illustration of the first few Hermite functions

Table 1.2 Hermite polynomials of orders from 1 to 10 and the corresponding zeros

Hermite polynomials	Zeros
$H_1(x) = 2x$	0
$H_2(x) = 4x^2 - 2$	±0.707
$H_3(x) = 8x^3 - 12x$	±1.2247, 0
$H_4(x) = 16x^4 - 48x^2 + 12$	±1.6507, ±0.5246
$H_5(x) = 32x^5 - 160x^3 + 120x$	±2.0202, ±0.9586, 0
$H_6(x) = 64x^6 - 480x^4 + 720x^2 - 120$	±2.3506, ±1.3358, ±0.4361
$H_7(x) = 128x^7 - 1,344x^5 + 3,360x^3 - 1,680x$	±2.6520, ±1.6736, ±0.8163, 0
$H_8(x) = 256x^8 - 3,584x^6 + 13,440x^4 - 13,440x^2 + 1,680$	±2.9306, ±1.9817, ±1.1572, ±0.3812
$H_9(x) = 512x^9 - 9216x^7 + 48,384x^5 - 80,640x^3 + 30,240x$	±3.1910, ±2.2666, ±1.4686, ±0.7236, 0
$H_{10}(x) = 1,024x^{10} - 23,040x^8 + 161,280x^6 - 403,200x^4 + 302,400x^2 - 30,240$	±3.4362, ±2.5327, ±1.7567, ±1.0366, ±0.3429

Then, the decomposition by using N Hermite functions is defined as follows:

$$f(i) = \sum_{p=0}^{N-1} c_p \psi_p(i), \tag{1.73}$$

where the coefficients c_p are obtained as:

$$c_p = \int_{-\infty}^{\infty} f(i)\psi_p(i)\mathrm{d}i. \tag{1.74}$$

The Gauss-Hermite quadrature technique can be used to calculate the Hermite expansion coefficients as follows:

$$c_p \approx \frac{1}{M}\sum_{m=1}^{M} \mu_{M-1}^{p}(x_m)f(x_m), \tag{1.75}$$

where the points x_m ($m = 1,\ldots,M$) are obtained as zeros of an Mth order Hermite polynomial: $H_M(x) = (-1)^M e^{x^2} \frac{d^M\left(e^{-x^2}\right)}{dx^M}$. The Hermite polynomials of orders from 1 to 10, as well as the corresponding zeros, are given in Table 1.2.

The constants $\mu^p_{M-1}(x_m)$can be obtained using the Hermite functions:

$$\mu^p_{M-1}(x_m) = \frac{\psi_p(x_m)}{(\psi_{M-1}(x_m))^2}. \tag{1.76}$$

Figure 1.26 depicts the original signal whose length is equal to 126 samples, and three reconstructed signals using 126, 90, and 70 Hermite functions, respectively.

In the first case, the reconstructed signal is approximately equal to the original signal. However, the signal can be successfully reconstructed with a smaller number of Hermite functions (while losing some finer details). The fact that the signal can be represented with a fewer number of Hermite coefficients makes this method attractive for signal compression.

1.9.2 Two-Dimensional Signals and Two-Dimensional Hermite Functions

For two-dimensional signals such as images, we use the two-dimensional Hermite functions defined as follows:

$$\Psi_{kl}(x,y) = \frac{(-1)^{k+l} e^{x^2/2+y^2/2}}{\sqrt{2^{k+l}k!l!\pi}} \frac{d^k\left(e^{-x^2}\right)}{\mathrm{d}x^k} \frac{d^l\left(e^{-y^2}\right)}{\mathrm{d}y^l}. \tag{1.77}$$

Some examples of two-dimensional Hermite functions are illustrated in Fig. 1.27.

Two-dimensional functions can be evaluated as a composition of one-dimensional Hermite functions:

$$\Psi_{kl}(x,y) = \Psi_k(x)\Psi_l(y) = \frac{(-1)^k e^{x^2/2}}{\sqrt{2^k k!\sqrt{\pi}}} \frac{d^k\left(e^{-x^2}\right)}{\mathrm{d}x^k} \frac{(-1)^l e^{y^2/2}}{\sqrt{2^l l!\sqrt{\pi}}} \frac{d^l\left(e^{-y^2}\right)}{\mathrm{d}y^l}.$$

In the sequel, we consider the Hermite projection method along one coordinate. For the two-dimensional signal of size $M_1 \times M_2$, the signal baseline is defined as:

$$b_y(x) = F(1,y) + \frac{F(M_1,y) - F(1,y)}{M_1} \cdot x, \tag{1.78}$$

where $F(x,y)$ is the two-dimensional signal, and $x = 1,\ldots, M_1$ and $y = 1,\ldots, M_2$. The baseline $b_y(x)$ is calculated for a fixed value of y. The corresponding matrix

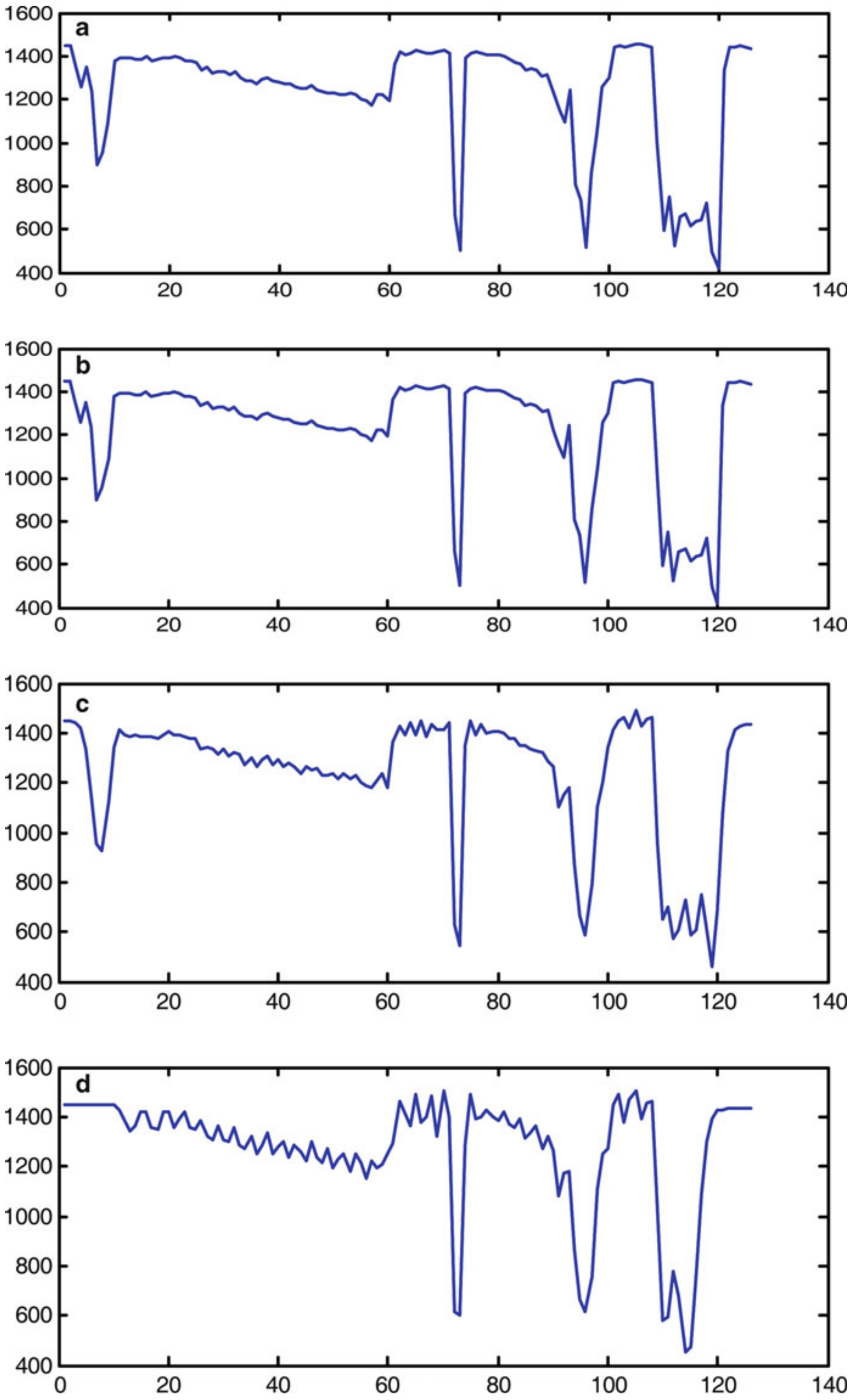


Fig. 1.26 (**a**) Original signal of length 126 samples, (**b**) Reconstructed signal with 126 functions, (**c**) Reconstructed signal with 90 functions, (**d**) Reconstructed signal with 70 functions

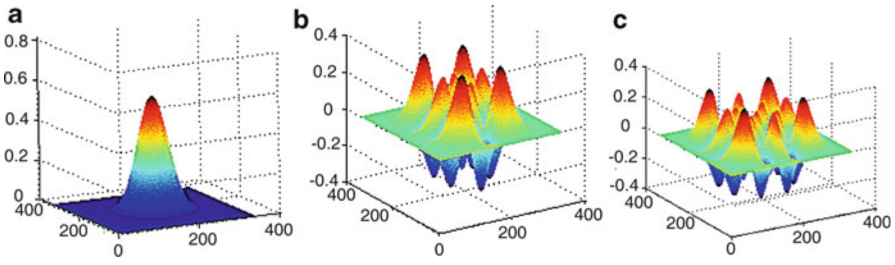


Fig. 1.27 Examples of two-dimensional Hermite functions: (**a**) $\Psi_{00}(x,y)$, (**b**) $\Psi_{24}(x,y)$, (**c**) $\Psi_{44}(x,y)$

$b(x,y)$ contains all vectors $b_y(x)$ for $y = 1,\ldots,M_2$. Then, we subtract the baselines from the original signal:

$$f(x,y) = F(x,y) - b(x,y). \tag{1.79}$$

The signal decomposition using N Hermite functions is defined as follows:

$$f_y(x) = \sum_{p=0}^{N-1} c_p \psi_p(x), \tag{1.80}$$

where $f_y(x) = f(x,y)$ is valid for fixed y. The coefficients are equal to:

$$c_p = \int_{-\infty}^{\infty} f_y(x)\psi_p(x)dx. \tag{1.81}$$

Using the Gauss-Hermite quadrature rule, the coefficients can be calculated as follows:

$$c_p \approx \frac{1}{M_1} \sum_{m=1}^{M_1} \mu_{M_1-1}^{p}(x_m) f_y(x_m), \tag{1.82}$$

where constants $\mu_{M_1-1}^{p}(x_m)$ can be obtained by using Hermite functions as in the one-dimensional case.

To illustrate this concept, the original image and three reconstructed images are shown in Fig. 1.28.

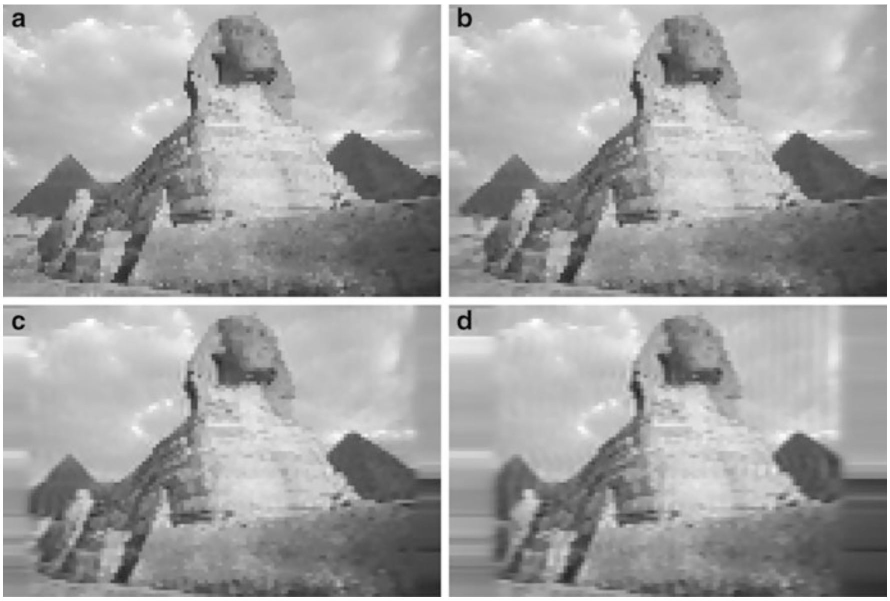


Fig. 1.28 (**a**) Original image, reconstructed image by using Hermite projection method along the rows: (**b**) with 100 Hermite functions, (**c**) 80 Hermite functions, (**d**) 60 Hermite functions

1.10 Examples

1.1. According to the Nyquist-Shannon sampling theorem, what should be the maximal value of frequency $f_{\max}$ for the signal whose sampling interval is $\Delta t = 0.001$, so that the signal is completely determined by its samples?

Solution:

$$f_{\max} \leq \frac{1}{2\Delta t} = \frac{1}{2 \cdot 0.001} = 500 \text{ Hz}$$

1.2. (a) If the maximal signal frequency is $f_{\max} = 4$ KHz, what should be the maximal sampling interval T according to the sampling theorem?
(b) For the signals whose sampling intervals are given as $T_1 = 2{\cdot}T_0$ and $T_2 = T_0/2$, specify the maximal values of signal frequency having in mind that the sampling theorem has to be satisfied?

Solution:

(a) For the known maximal signal frequency, the sampling interval can be calculated according to the relation:

$$T \leq \frac{1}{2 \cdot f_{\max}} = \frac{1}{2 \cdot 4 \cdot 10^3} = 125\,\mu\text{s}.$$

(b) For the sampling interval $T_1 = 2{\cdot}T_0$, we have:

$$f_{\max} \le \frac{1}{2\cdot T_1} = \frac{1}{2\cdot 2\cdot T_0} = \frac{1}{2\cdot 2\cdot 125\cdot 10^{-6}} = 2\text{ kHz}.$$

In the case $T_2 = T_0/2$, the maximal frequency of the considered signal can be calculated as:

$$f_{\max} \le \frac{1}{2\cdot T_0/2} = \frac{1}{2\cdot 125/2\cdot 10^{-6}} = 8\text{ kHz}.$$

1.3. Consider the signal in the form $y = \sin(150{\cdot}\pi{\cdot}t)$, where $t = -5{:}0.001{:}5$. What is the frequency f of the sinusoidal signal? What can be the maximal signal frequency if the sampling theorem is satisfied?

Solution:

The frequency of the considered sinusoidal signal can be determined as follows:

$$\omega = 2\cdot\pi\cdot f = 150\cdot\pi \Rightarrow f = 150/2 = 75\text{ Hz}.$$

The maximal frequency in this example is given by:

$$f_{\max} \le \frac{1}{2\cdot\Delta t} = \frac{1}{2\cdot 0.001} = 500\text{ Hz}.$$

1.4. Plot the Fourier transform of the signal $y = \sin(150\pi t)$, $t\in(-1,1)$, by using Matlab. The sampling interval should be set as: $T = 1/1{,}000$.

Solution:

First, we should check if the sampled signal satisfies the sampling theorem, i.e., we check if the condition $f \le f_{max}$ is satisfied.

The maximal frequency of the signal is: $f_{max} \le 1/(2T) = 500$ Hz, while the sinusoidal signal frequency is obtained as: $2\pi f = 150\pi \Rightarrow f = 75$ Hz, and satisfies the condition $f \le f_{max}$.

For the Fourier transform calculation, we use *fft* Matlab function. In order to centralize the zero-frequency component of the Fourier spectrum, the *fftshift* function is used as well. Matlab code is given in the sequel:

```
t=-1:1/1000:1;
y=sin(150*pi*t);
F=fft(y);
F=fftshift(F);
plot(abs(F))
```

1.5. Calculate the Fourier transform of the signal $y = \sin(150{\cdot}\pi{\cdot}t)$, $t = -5{:}0.001{:}5$. Is the same sampling interval appropriate to satisfy the sampling theorem even for signals: $y = \sin(600{\cdot}\pi{\cdot}t)$, $y = \sin(1{,}800{\cdot}\pi{\cdot}t)$? If the answer is yes, plot the spectra of the considered signals by using Matlab.

Solution:

$$f_{\max} = \frac{1}{2 \cdot \Delta t} = \frac{1}{2 \cdot 0.001} = 500 \text{ Hz}$$

$$\omega_1 = 2\pi f_1 = 150\pi \Rightarrow f_1 = 150/2 = 75 \text{ Hz}$$

$$\omega_2 = 2\pi f_2 = 600\pi \Rightarrow f_2 = 600/2 = 300 \text{ Hz}.$$

In the third case, the sampling theorem is not satisfied:

$$\omega_3 = 2\pi f_3 = 1,800\pi \Rightarrow f_3 = 1,800/2 = 900 \text{ Hz}$$

$f_3 > f_{\max}$

Matlab code:

```
t=-5:0.001:5;
y1=sin(150*pi*t);
F1=fft(y1);
plot (abs(F1))
y2=sin(600*pi*t);
F2=fft(y2);
figure,plot (abs(F2))
```

1.6. Consider the signal $y = \sin(150 \cdot \pi \cdot t)$ with an additive white Gaussian noise (zero mean value $\mu = 0$, and variance $\sigma = 1$) and plot the illustration of its spectrum via the Fourier transform calculation.

Solution:

```
y=sin(150*pi*t);
noise=randn(1,length(y));
yn=y+noise;
sound(yn);
F=fftshift(fft(yn));
plot (abs(F))
```

1.7. Design a simple low-pass filter with a cutoff frequency $f_c = 1{,}102{,}5$ Hz for the signal having 44,000 samples, sampled at the frequency 22,050 Hz.

Solution:

Let us denote the signal by y, while $f_s = 22{,}050$ Hz represents the sampling frequency. The maximal signal frequency is: $f_{\max} = f_s/2 = 11{,}025$ Hz. The Fourier transform of y contains 44,000 coefficients: 22,000 coefficients are located at the positive and 22,000 coefficients at negative frequencies (Fig. 1.29).

Hence, the following proportion holds:

$$f_{\max} : f_c = 22{,}000 : n$$

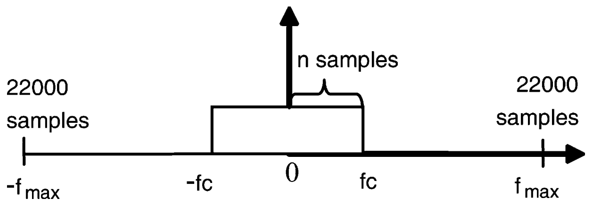


Fig. 1.29 Illustration of the filter function

and, consequently, we obtain $n = 2{,}200$.

The corresponding filter transfer function in the frequency domain can be defined as:

H = [zeros(1,22000–2200) ones(1,4400) zeros(1,22000–2200)];

In order to obtain the filtered signal, the Fourier transform of y should be multiplied by the filter function H, and then the inverse Fourier transform should be performed.

1.8. Make the illustrations for the Fourier transform and the ideal time-frequency representation, if the signal is given in the form:

(a) $y_a = e^{j\omega_1 t} + e^{j\omega_2 t}; t \in (0,2)$,
(b) $y_b = y_1 + y_2,\ y_1 = e^{j\omega_1 t} for\, t \in (0,1),\ \ y_2 = e^{j\omega_2 t} for\ t \in (1,2)$.

We may assume that $\omega_1 < \omega_2$.

Solution (Fig. 1.30):

1.9. Based on the ideal time-frequency representation of a certain signal $f(t)$, define each of the signal components and signal itself. Unit amplitudes and zero initial phases are assumed (Fig. 1.31).

Solution:

The analytic form of signal $f(t)$ can be defined as:

$$f(t) = \begin{cases} e^{j\omega_1 t} + e^{j\omega_2 t}, & t \in (0, t_1) \\ e^{j\omega_3 t}, & t \in (t_1, t_2) \\ e^{j\omega_1 t} + e^{j\omega_3 t}, & t \in (t_2, t_3) \end{cases}.$$

1.10. Consider a constant-frequency modulated signal and demonstrate how the window width influences the resolution of the spectrogram. The signal is given in the form:

$$f(n) = \begin{cases} e^{j15nT}, n = 1, \dots, 127 \\ e^{j5nT}, n = 128, \dots, 256 \end{cases}, \quad \text{where } T = 0.25.$$

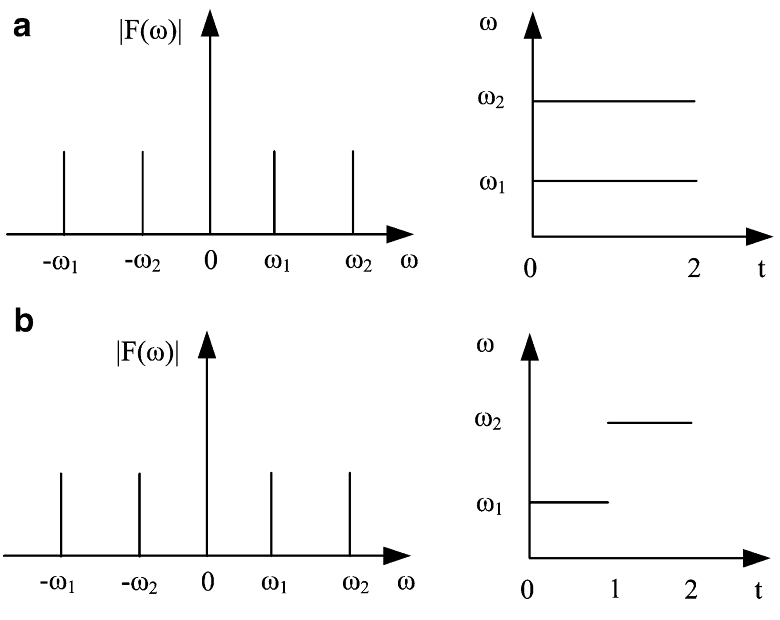


Fig. 1.30 Fourier transform and the ideal time-frequency distribution for signal: (**a**) y_a, (**b**) y_b

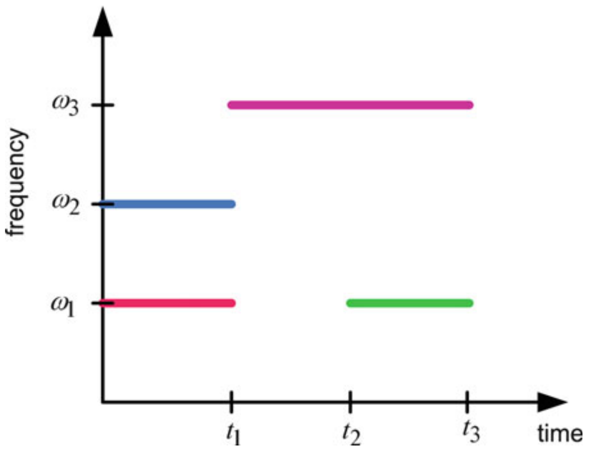


Fig. 1.31 Ideal time-frequency representation of signal $f(t)$

Solution:

The discrete version of the considered signal can be created in Matlab as follows:

```
T=0.25;
for n=1:256
    if n<128; f(n)=exp(j*15*n*T);
      else
```

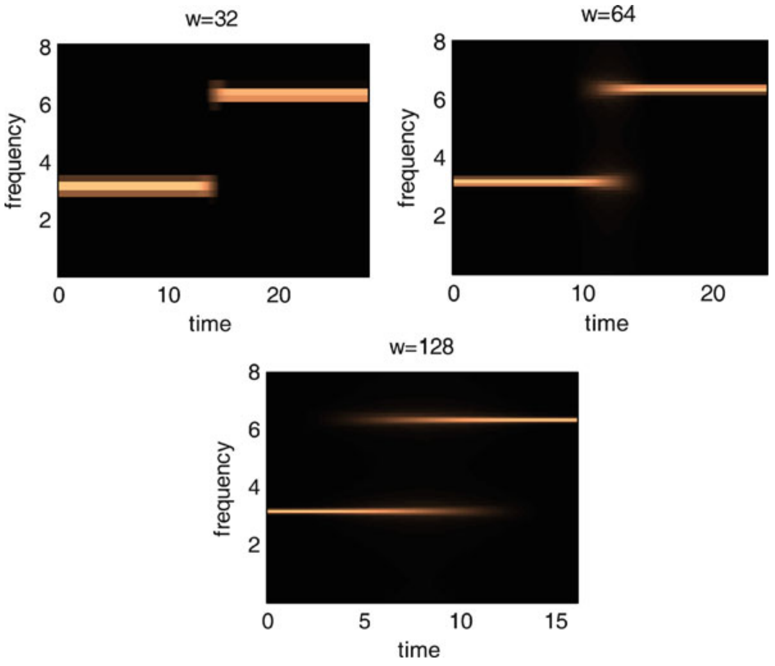


Fig. 1.32 Spectrograms for different window widths

```
        f(n)=exp(j*5*n*T);
    end
end
```

The spectrogram calculation in Matlab can be done by using the inbuilt function *specgram* as follows:

```
[s,F,T] = specgram(f,w,freq,w,w-1);
imagesc(T,F,abs(s))
```

Hereby, to change the window width w in a few realizations one should take, for instance, the following values: $w = 32$, $w = 64$, $w = 128$ (Fig. 1.32). Note that *freq* does not influence the spectrogram; it just scales the y axes (*freq* = 8 is used in this example).

1.11. Write the Matlab code that calculates the S-method. Also, the signal is loaded from the separate file signal.m, and it is defined as: $f(t) = e^{j(2\sin(\pi t)+11\pi t)} + e^{j2\pi\left(t^2+3t\right)}$. The signal length is $N = 128$ samples, t = −2:2 with sampling interval $\Delta t = 4/N$, the window width is $M = 128$ samples, while $L_d = 5$. The Gaussian window should be used.

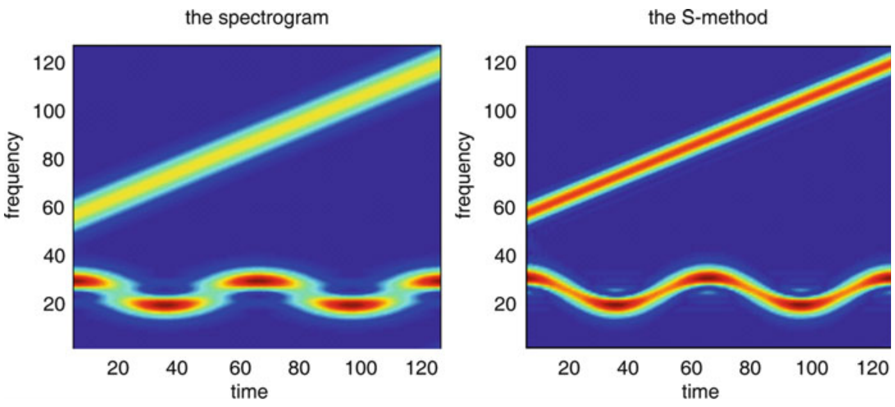


Fig. 1.33 Spectrogram and S-method

Solution:

The Matlab file *signal.m* creates the signal:

```
function f = signal(t)
f = exp(j*2*(sin(pi*t) + 11*pi*t)) + exp(j*2*pi*(t.
^2 + 3*t));
```

The Matlab code for the S-method calculation is given in the sequel (Fig. 1.33).

```
clear
f=[];
N=128;   %signal length
M=128;   %window width
Ld=5;    % parameter which determines the frequency domain
         % window width in the S-method calculation
t=-2+4/N:4/N:2;

% signal

for m=-M/2:1:M/2-1;
tau=2*m/N;
f=[f;signal(t+4*tau)];
end

% Calculating STFT and Spectrogram (SPEC)

for i=1:N
  w=gausswin(length(f(:,1)),10); % Gaussian window
  STFT(:,i)=fftshift(fft(f(:,i).*w));
  SPEC(:,i)=abs(STFT(:,i)).^2;
end
```

```
% Calculating the S-method (SM)

SFP=STFT;
SFN=STFT;
SM=SPEC;
for L=1:Ld
SFP=[zeros(1,N);SFP(1:N-1,:)];
SFN=[SFN(2:N,:);zeros(1,N)];
SM=SM+2*real([SFP.*conj(SFN)]);
end

% Plotting the spectrogram and the S-method

figure(1),pcolor(abs(SPEC)),shading interp
figure(2),pcolor(abs(SM)),shading interp
```

1.12. Observe a general form of a constant amplitude signal with a phase function $\phi(t)$: $f(t) = Ae^{j\phi(t)}$. Prove that the second and higher phase derivatives cause the spreading of the concentration around the instantaneous frequency in the case of the STFT?

Solution:

The STFT of the signal $f(t)$ is defined as:

$$\mathrm{STFT}(t,\omega) = \int_{-\infty}^{\infty} f(t+\tau)w(\tau)e^{-j\omega\tau}\mathrm{d}\tau = \int_{-\infty}^{\infty} A\cdot e^{j\phi(t+\tau)}w(\tau)e^{-j\omega\tau}\mathrm{d}\tau. \quad (1.83)$$

By applying the Taylor series expansion to the phase function, we obtain:

$$\phi(t+\tau) = \phi(t) + \phi'(t)\tau + \phi''(t)\tau^2/2! + \cdots \quad (1.84)$$

The short-time Fourier transform can be rewritten in the form:

$$STFT(t,\omega) = Ae^{j\phi(t)}\int_{-\infty}^{\infty} e^{j\phi'(t)\tau}w(\tau)e^{j(\phi''(t)\tau^2/2!+\cdots)}e^{-j\omega\tau}\mathrm{d}\tau.$$

We can further develop the above expression as:

$$\mathrm{STFT}(t,\omega) = Ae^{j\phi(t)}\mathrm{FT}\left\{e^{j\phi'(t)\tau}\right\}_{*\omega}\mathrm{FT}\{w(\tau)\}_{*\omega}\mathrm{FT}\left\{e^{j\phi''(t)\tau^2/2!+\cdots}\right\}.$$

Finally, we obtain the STFT in the form:

$$\mathrm{STFT}(t,\omega) = 2\pi Ae^{j\phi(t)}\delta(\omega-\phi'(t))_{*\omega}W(\omega)_{*\omega}\mathrm{FT}\left\{e^{j\phi''(t)\tau^2/2!+\cdots}\right\}. \quad (1.85)$$

Note that the last term in the expression for the STFT contains second and higher phase derivatives, and thus, we may conclude that this term will produce spreading of the concentration around the instantaneous frequency $\omega = \phi'(t)$.

1.13. For the signal from the previous example, analyze and derive the influence of the higher order phase derivatives in the case of the Wigner distribution.

Solution:

The Wigner distribution is defined as:

$$\begin{aligned}\mathrm{WD}(t,\omega) &= \int_{-\infty}^{\infty} f(t+\tau/2)f^*(t-\tau/2)e^{-j\omega\tau}\mathrm{d}\tau \\ &= \int_{-\infty}^{\infty} A^2 e^{j\phi(t+\tau/2)}e^{-j\phi(t-\tau/2)}e^{-j\omega\tau}\mathrm{d}\tau.\end{aligned} \tag{1.86}$$

The Taylor series expansion of the moment phase function results in:

$$\begin{aligned}\phi(t+\tau/2)-\phi(t-\tau/2) = \Bigg(\phi(t)+\phi'(t)\tau/2+\sum_{k=2}^{\infty}\phi^{(k)}(t)(\tau/2)^k/k!\Big)- \\ -\Big(\phi(t)-\phi'(t)\tau/2+\sum_{k=2}^{\infty}(-1)^k\phi^{(k)}(t)(\tau/2)^k/k!\Bigg)\end{aligned}$$

By using the Taylor series expansion terms in the definition of the Wigner distribution, we obtain:

$$\mathrm{WD}(t,\omega) = A^2\int_{-\infty}^{\infty} e^{\left(j\phi'(t)\tau+2\sum\limits_{k=1}^{\infty}\frac{\phi^{(2k+1)}(t)}{(2k+1)!}\left(\frac{\tau}{2}\right)^{2k+1}-j\omega\tau\right)}\mathrm{d}\tau, \tag{1.87}$$

Or, in other words:

$$\mathrm{WD}(t,\omega) = 2\pi A^2\delta(\omega-\phi'(t))_{*\omega}\mathrm{FT}\left\{e^{2j\sum\limits_{k=1}^{\infty}\frac{\phi^{(2k+1)}}{(2k+1)!}\left(\frac{\tau}{2}\right)^{2k+1}}\right\}. \tag{1.88}$$

Hence, we may see that only the odd phase derivatives are included in the spread factor, causing inner-interferences and spreading of the concentration in the time-frequency domain.

1.14. Consider a signal in the form: $x(t) = \left(f(kt)e^{j\frac{At^2}{2}}\right) * \frac{1}{\sqrt{2\pi jB}}e^{j\frac{t^2}{2B}}$, where * denotes the convolution. Prove that the Wigner distribution of $x(t)$ is equal to the Wigner distribution of $f(t)$ in the rotated coordinate system:

$$\mathrm{WD}_x(t,\omega) = \mathrm{WD}_f(t\cos\alpha-\omega\sin\alpha, \omega\cos\alpha+t\sin\alpha),$$

where $k = 1/\cos\alpha$, $B = \sin\alpha/\cos\alpha$, $A = -\sin\alpha\cos\alpha$.

Solution:

The signal $x(t)$ can be written as a convolution of two signals:

$$x(t) = f_1(t)^* f_2(t),$$

where $f_1(t) = f(kt)e^{jAt^2/2}$ and $f_2(t) = \frac{1}{\sqrt{2\pi jB}}e^{jt^2/2B}$.

It means that the Fourier transform of $x(t)$ can be written as:

$$X(\omega) = F_1(\omega)F_2(\omega) = F_1(\omega)e^{jB\omega^2/2}, \tag{1.89}$$

where,

$$F_2(\omega) = \mathrm{FT}\left\{\frac{1}{\sqrt{2\pi jB}}e^{jt^2/2B}\right\} = e^{jB\omega^2/2}. \tag{1.90}$$

Furthermore, the Wigner distribution of $x(t)$ can be obtained by using $X(\omega)$ as follows:

$$\begin{aligned}
\mathrm{WD}_x(t,\omega) &= \int_{-\infty}^{\infty} X(\omega+\theta/2)X^*(\omega-\theta/2)e^{j\theta t}\mathrm{d}\theta \\
&= \int_{-\infty}^{\infty} F_1(\omega+\theta/2)F_2(\omega+\theta/2)F_1{}^*(\omega-\theta/2)F_2{}^*(\omega-\theta/2)e^{j\theta t}\mathrm{d}\theta \\
&= \int_{-\infty}^{\infty} F_1(\omega+\theta/2)F_1{}^*(\omega-\theta/2)e^{-jB(\omega+\theta/2)^2/2+jB(\omega-\theta/2)^2/2}e^{j\theta t}\mathrm{d}\theta \\
&= \int_{-\infty}^{\infty} F_1(\omega+\theta/2)F_1{}^*(\omega-\theta/2)e^{-jB\omega\theta}e^{j\theta t}\mathrm{d}\theta
\end{aligned}$$

Hence, the following relation holds:

$$\mathrm{WD}_x(t,\omega) = \mathrm{WD}_{f_1}(t-B\omega,\omega). \tag{1.91}$$

Furthermore, we calculate the Wigner distribution $\mathrm{WD}_{f_1}(t,\omega)$ of the signal $f_1(t)$ as follows:

$$\begin{aligned}
\mathrm{WD}_{f_1}(t,\omega) &= \int_{-\infty}^{\infty} f(k(t+\tau/2))f^*(k(t-\tau/2))e^{jA(t+\tau/2)^2/2}e^{-jA(t-\tau/2)^2/2}e^{-j\omega\tau}\mathrm{d}\tau = \\
&= \int_{-\infty}^{\infty} f(k(t+\tau/2))f^*(k(t-\tau/2))e^{jAt\tau}e^{-j\omega\tau}\mathrm{d}\tau = \\
&= \int_{-\infty}^{\infty} f(k(t+\tau/2))f^*(k(t-\tau/2))e^{-jk\tau((\omega-At)/k)}\mathrm{d}\tau.
\end{aligned}$$

The Wigner distribution $\mathrm{WD}_{f_1}(t,\omega)$ can be, thus, expressed as:

$$\mathrm{WD}_{f_1}(t,\omega) = \mathrm{WD}_f(kt, \omega/k - At). \tag{1.92}$$

Consequently, from (1.91) and (1.92) we have:

$$\mathrm{WD}_x(t,\omega) = \mathrm{WD}_f(kt - B(\omega/k - Akt), \omega/k - Akt), \tag{1.93}$$

$$\text{or}\quad \mathrm{WD}_x(t,\omega) = \mathrm{WD}_f[(1+BA)kt - B\omega/k, \omega/k - Akt].$$

By substituting the given parameters: $k = 1/\cos\alpha$, $B = \sin\alpha/\cos\alpha$, $A = -\sin\alpha\cos\alpha$, we obtain:

$$\mathrm{WD}_x(t,\omega) = \mathrm{WD}_f(t\cos\alpha - \omega\sin\alpha, \omega\cos\alpha + t\sin\alpha). \tag{1.94}$$

The rotation of the coordinate system is defined as:

$$\begin{bmatrix} t_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} t \\ \omega \end{bmatrix}.$$

1.15. By using the recursive procedure for the calculation of the Haar transform, perform the first level decomposition of a given 8×8 image. Use the one-dimensional decomposition of image rows in the first step, and then the decomposition of image columns.

10	10	10	10	26	10	10	10
10	10	10	10	26	10	10	10
10	10	10	10	26	10	10	10
10	10	10	10	26	10	10	10
10	10	10	10	26	10	10	10
10	10	10	10	26	10	10	10
18	18	18	18	26	18	18	18
10	10	10	10	26	10	10	10

Solution:

First, we perform the first level decomposition along the image rows. Hence, for each row, it is necessary to calculate the mean values and differences (details). Then the resulted matrix should be used to perform the decomposition along columns. The low-frequency image content is obtained in the first quadrant, while the remaining parts contain image details.

Decomposition of rows								*Decomposition of columns*							
10	10	18	10	0	0	8	0	10	10	18	10	0	0	8	0
10	10	18	10	0	0	8	0	10	10	18	10	0	0	8	0
10	10	18	10	0	0	8	0	10	10	18	10	0	0	8	0
10	10	18	10	0	0	8	0	14	10	20	14	0	0	6	0
10	10	18	10	0	0	8	0	0	0	0	0	0	0	0	0
10	10	18	10	0	0	8	0	0	0	0	0	0	0	0	0
18	18	22	18	0	0	4	0	0	0	0	0	0	0	0	0
10	10	18	10	0	0	8	0	4	4	2	4	0	0	−2	0

1.16. Consider the function $f(t)$ in the form:

$$f(t) = \begin{cases} t^2 + t, & 0 \le t < 1, \\ 0, & \text{otherwise.} \end{cases}$$

By using the Haar wavelets calculate the expansion coefficients:

$s_{j_0}(k) = \int f(t)\varphi_{j_0,k}(t)dt, \quad d_j(k) = \int f(t)\psi_{j,k}(t)dt$, for $j_0 = 0$.

Solution:

$$s_0(0) = \int_0^1 (t^2 + t)\varphi_{0,0}(t)\mathrm{d}t = \int_0^1 (t^2 + t)\mathrm{d}t = \frac{t^3}{3}\Big|_0^1 + \frac{t^2}{2}\Big|_0^1 = \frac{5}{6}$$

$$d_0(0) = \int_0^1 (t^2 + t)\psi_{0,0}(t)\mathrm{d}t = \int_0^{0.5} (t^2 + t)\mathrm{d}t - \int_{0.5}^1 (t^2 + t)\mathrm{d}t = -0.5$$

$$d_1(0) = \int_0^1 (t^2 + t)\psi_{1,0}(t)\mathrm{d}t = \int_0^{0.25} (t^2 + t)\sqrt{2}\mathrm{d}t - \int_{0.25}^{0.5} (t^2 + t)\sqrt{2}\mathrm{d}t = -\frac{3\sqrt{2}}{32}$$

$$d_1(1) = \int_0^1 (t^2 + t)\psi_{1,1}(t)\mathrm{d}t = \int_{0.5}^{0.75} (t^2 + t)\sqrt{2}\mathrm{d}t - \int_{0.75}^1 (t^2 + t)\sqrt{2}\mathrm{d}t = -\frac{5\sqrt{2}}{32}$$

$$f(t) = \underbrace{\underbrace{\underbrace{\frac{5}{6}\varphi_{0,0}(t)}_{V_0} + \underbrace{\left(-\frac{1}{2}\psi_{0,0}(t)\right)}_{W_0}}_{V_1=V_0\oplus W_0} + \underbrace{\left(-\frac{3\sqrt{2}}{32}\psi_{1,0}(t) - \frac{5\sqrt{32}}{32}\psi_{1,1}(t)\right)}_{W_1}}_{V_2=V_1\oplus W_1=V_0\oplus W_0\oplus W_1} + \cdots$$

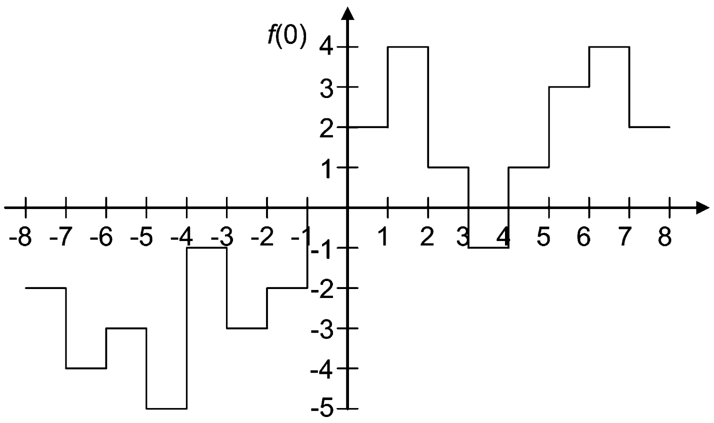


Fig. 1.34 Signal *f*(0) before decomposition

1.17. Consider the signal illustrated in Fig. 1.34 and perform the Haar wavelet decomposition (e.g., 3-level decomposition).
Solution (Fig. 1.35):

1.18. Starting from the dilation equation:

$$\varphi(t)=\sum_{k=0}^{N-1}s(k)\sqrt{2}\varphi(2t-k), \tag{1.95}$$

and using the filter coefficients $h(k)$, where $s(k)=\sqrt{2}h(k)$ and $\sum_{k=0}^{N-1}h(k)=1$, show that the Fourier transform $\Phi(\omega)$ of scaling function $\varphi(t)$ is equal to the product of filter frequency responses:

$$\Phi(\omega)=\prod_{j=1}^{\infty}H\left(\frac{\omega}{2^j}\right).$$

Solution:

The Fourier transform of the scaling function can be calculated as:

$$\begin{aligned}\Phi(\omega)&=\int_{-\infty}^{\infty}\varphi(t)e^{-j\omega t}\mathrm{d}t=2\sum_{k=0}^{N-1}h(k)\int_{-\infty}^{\infty}\varphi(2t-k)e^{-j\omega t}\mathrm{d}t\\&=\sum_{k=0}^{N-1}h(k)\int_{-\infty}^{\infty}\varphi(x)e^{-j\omega(x+k)/2}\mathrm{d}x=\sum_{k=0}^{N-1}h(k)e^{-j\omega k/2}\int_{-\infty}^{\infty}\varphi(x)e^{-j\omega x/2}\mathrm{d}x.\end{aligned} \tag{1.96}$$

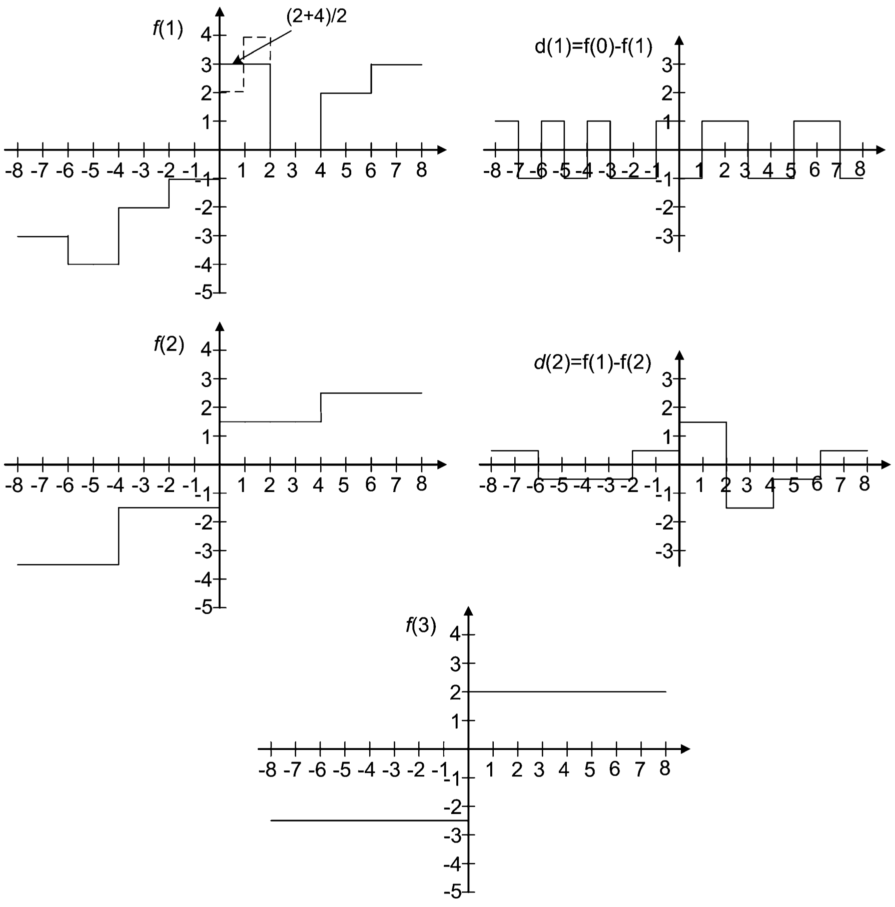


Fig. 1.35 Signal decomposition

From (1.96), we may observe that:

$$\Phi(\omega) = \Phi\left(\frac{\omega}{2}\right) \sum_{k=0}^{N-1} h(k) e^{-j\omega k/2} = \Phi\left(\frac{\omega}{2}\right) H\left(\frac{\omega}{2}\right). \tag{1.97}$$

Hence, by applying the recursion we obtain:

$$\Phi(\omega) = H\left(\frac{\omega}{2}\right) H\left(\frac{\omega}{4}\right) ... H\left(\frac{\omega}{2^n}\right) \Phi\left(\frac{\omega}{2^n}\right). \tag{1.98}$$

Having in mind that: $\lim_{n\to\infty} \Phi\left(\frac{\omega}{2^n}\right) = \Phi(0) = \int_{-\infty}^{\infty} \varphi(t)\mathrm{d}t = 1$, the Fourier transform of the scaling function is obtained in the form:

$$\Phi(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right). \tag{1.99}$$

1.19. Determine the Hermite expansion coefficients, for a short signal $f(n)$ given below.

$f = [1332.4 \quad 1313.4 \quad 1148.4 \quad 1243.2 \quad 735.7 \quad 861.9 \quad 1261.1 \quad 1438.1$
$1443.9 \quad 1454.1];$

Solution:

The signal consists of $M = 10$ samples. The zeros of the Hermite polynomial of order ten are (Table 1.2):

$x_m = [-3.4362 \quad -2.5327 \quad -1.7567 \quad -1.0366 \quad -0.3429 \quad 0.3429$
$1.0366 \quad 1.7567 \quad 2.5327 \quad 3.4362];$

The first 10 Hermite functions, calculated for x_m, are given below:

ψ_0	0.0021	0.0304	0.1605	0.4389	0.7082	0.7082	0.4389	0.1605	0.0304	0.0021
ψ_1	−0.0100	−0.1089	−0.3989	−0.6434	−0.3435	0.3435	0.6434	0.3989	0.1089	0.0100
ψ_2	0.0328	0.2542	0.5871	0.3566	−0.3830	−0.3830	0.3566	0.5871	0.2542	0.0328
ψ_3	−0.0838	−0.4368	−0.5165	0.2235	0.3877	−0.3877	−0.2235	0.5165	0.4368	0.0838
ψ_4	0.1753	0.5622	0.1331	−0.4727	0.2377	0.2377	−0.4727	0.1331	0.5622	0.1753
ψ_5	−0.3060	−0.5098	0.3141	0.1100	−0.3983	0.3983	−0.1100	−0.3141	0.5098	0.3060
ψ_6	0.4471	0.2323	−0.4401	0.3657	−0.1382	−0.1382	0.3657	−0.4401	0.2323	0.4471
ψ_7	−0.5378	0.1575	0.1224	−0.3044	0.3941	−0.3941	0.3044	−0.1224	−0.1575	0.5378
ψ_8	0.5058	−0.4168	0.3041	−0.1843	0.0617	0.0617	−0.1843	0.3041	−0.4168	0.5058
ψ_9	−0.3123	0.3491	−0.3672	0.3771	−0.3815	0.3815	−0.3771	0.3672	−0.3491	0.3123

Furthermore, the constants $\mu_{M-1}^{p}(x_m)$ are calculated by using the Hermite functions:

$$\mu_{M-1}^{p}(x_m) = \frac{\psi_p(x_m)}{(\psi_{M-1}(x_m))^2}, \quad p = 0, \ldots, N-1.$$

The obtained matrix is:

μ_9^0	0.0210	0.2494	1.1904	3.0868	4.8662	4.8662	3.0868	1.1904	0.2494	0.0210
μ_9^1	−0.1022	−0.8934	−2.9573	−4.5253	−2.3598	2.3598	4.5253	2.9573	0.8934	0.1022
μ_9^2	0.3362	2.0864	4.3533	2.5082	−2.6317	−2.6317	2.5082	4.3533	2.0864	0.3362
μ_9^3	0.8598	−3.5851	−3.8294	1.5719	2.6636	−2.6636	−1.5719	3.8294	3.5851	0.8598
μ_9^4	1.7980	4.6137	0.9867	−3.3244	1.6333	1.6333	−3.3244	0.9867	4.6137	1.7980
μ_9^5	−3.1384	−4.1838	2.3289	0.7735	−2.7366	2.7366	−0.7735	−2.3289	4.1838	3.1384
μ_9^6	4.5848	1.9062	−3.2628	2.5718	−0.9492	−0.9492	2.5718	−3.2628	1.9062	4.5848
μ_9^7	−5.5153	1.2929	0.9076	−2.1412	2.7076	−2.7076	2.1412	−0.9076	−1.2929	5.5153
μ_9^8	5.1871	−3.4203	2.2549	−1.2959	0.4237	0.4237	−1.2959	2.2549	−3.4203	5.1871
μ_9^9	−3.2023	2.8647	−2.7229	2.6520	−2.6212	2.6212	−2.6520	2.7229	−2.8647	3.2023

The resulting vector of the Hermite expansion coefficients c is (for the sake of simplicity the constants are written with two-decimal places):

$$\mathrm{c} = [-701.61 \quad 90.30 \quad 140.84 \quad 77.5 \quad -140.56 \quad 2.08 \quad 94.06 \quad -54.75 \quad -52.74 \quad 88.06];$$

1.20. In this example, we provide the Matlab code for the Hermite projection method, which is used to obtain the illustrations in Fig. 1.26.

```
N=126;    % signal length
n=70;     % the number of Hermite functions

% the function that calculates the zeros of the Hermite
polynomial
xm=hermite_roots(N);

% function that calculates Hermite functions

y=psi(n,xm);
% Loading a one-dimensional signal
load sig1.mat
x=signal1;


% Removing the baseline

i=1:N;
baseline=x(1)+(x(N)-x(1))/N.*i;
f=x-baseline;

% Calculating Hermite coefficients

for i=1:n
mi(i,:)=y(i,:)./(y(N,:)).^2;
```

```
Mi(i)=1/N*sum(mi(i,:).*f);
end
c=Mi;
ff=zeros(1,length(xm));
for ii=1:length(xm)
for i=1:n
   ff(ii)=ff(ii)+c(i)*y(i,ii);
end
end

% Signal reconstruction

ss=ff+baseline;
figure,plot((ss))
```

Matlab function *psi.m* that is used for the recursive calculation of the Hermite functions is given in the sequel:

```
function y=psi(n,x);
Psi=zeros(n,length(x));
psi0=1./(pi^(1/4)).*exp(-x.^2/2);
psi1=sqrt(2).*x./(pi^(1/4)).*exp(-x.^2/2);
Psi(1,:)=psi0; Psi(2,:)=psi1;

for i=2:180
Psi(i+1,:)=x.*sqrt(2/i).*Psi(i,:)-sqrt((i-1)/(i)).*Psi
(i-1,:);
end
y=Psi;
```

References

1. Amin MG, Williams WJ (1998) High spectral resolution time-frequency distribution kernels. IEEE Trans Signal Process 46(10):2796–2804
2. Bastiaans MJ, Alieva T, Stankovic LJ (2002) On rotated time-frequency Kernels. IEEE Signal Process Lett 9(11):378–381
3. Boashash B, Ristic B (1998) Polynomial time-frequency distributions and time-varying higher order spectra: application to the analysis of multicomponent FM signals and to the treatment of multiplicative noise. Signal Process 67(1):1–23
4. Boashash B (2003) Time-frequency analysis and processing. Elsevier, Amsterdam
5. Cohen L (1989) Time-frequency distributions – a review. Proc IEEE 77(7):941–981
6. Daubechies I (1992) Ten lectures on wavelets. Society for Industrial and Applied Mathematics, Philadelphia
7. Dudgeon D, Mersereau R (1984) Multidimensional digital signal processing. Prentice Hall, Englewood Cliffs

8. Fugal L (2009) Conceptual wavelets in digital signal processing. Space and Signal Technical Publishing, San Diego
9. González RC, Woods R (2008) Digital image processing. Prentice Hall, Englewood Cliffs
10. Hlawatsch F, Boudreaux-Bartels GF (1992) Linear and quadratic time-frequency signal representations. IEEE Signal Process Mag 9(2):21–67
11. Kortchagine D, Krylov A (2000) Projection filtering in image processing. In: Proceedings of the tenth international conference on computer graphics and applications (GraphiCon'2000), Moscow, Russia, pp 42–45
12. Kortchagine D, Krylov A (2005) Image database retrieval by fast Hermite projection method. In: Proceedings of the fifteenth international conference on computer graphics and applications (GraphiCon'2005), Novosibirsk, Russia, pp 308–311
13. Krylov A, Kortchagine D (2006) Fast Hermite projection method. In: Proceedings of the third international conference on image analysis and recognition (ICIAR'2006), vol 1, Povoa de Varzim, Portugal, pp 329–338
14. Mallat S (1999) A wavelet tour of signal processing, 2nd edn. Academic, San Diego
15. Oppenheim A (1978) Applications of digital signal processing. Prentice Hall, Englewood Cliffs
16. Orović I, Orlandić M, Stanković S, Uskoković Z (2011) A virtual instrument for time-frequency analysis of signals with highly non-stationary instantaneous frequency. IEEE Trans Instrum Meas 60(3):791–803
17. Orović I, Stanković S, Stanković LJ, Thayaparan T (2010) Multiwindow S-method for instantaneous frequency estimation and its application in radar signal analysis. IET Signal Process Special Issue Time-Freq Approach Radar Detect Imaging Classif 4(4):363–370
18. Percival DB, Walden AT (2006) Wavelet methods for time series analysis. Cambridge University Press, Cambridge
19. Radunović D (2005) Talasići (Wavelets). Akademska misao (in Serbian), Beograd
20. Stanković LJ (1994) A method for time-frequency signal analysis. IEEE Trans Signal Process 42(1):225–229
21. Stanković LJ (1994) Multitime definition of the Wigner higher order distribution: L-Wigner distribution. IEEE Signal Process Lett 1(7):106–109
22. Stanković S (2010) Time-frequency analysis and its application in digital watermarking (Review paper). EURASIP J Adv Signal Process, Special Issue Time-Freq Anal Appl Multimedia Signals 2010: Article ID 579295, 20
23. Stanković S, Orović I, Krylov A (2010) Video frames reconstruction based on time-frequency analysis and Hermite projection method. EURASIP J advances in signal processing, Special Issue Time-Freq Anal Appl Multimedia Signals, Article ID 970105, 11
24. Stanković S, Orović I, Ioana C (2009) Effects of Cauchy integral formula on the precision of the IF estimation. IEEE Signal Process Lett 16(4):327–330
25. Stanković S, Stanković LJ (1997) An architecture for the realization of a system for time-frequency signal analysis. IEEE Trans Circuit Syst Part II 44(7):600–604
26. Stollnitz EJ, DeRose TD, Salesin DH (1995) Wavelets for computer graphics: a primer, part 1. IEEE Comput Graph Appl 15(3):76–84
27. Stollnitz EJ, DeRose TD, Salesin DH (1995) Wavelets for computer graphics: a primer, part 2. IEEE Comput Graph Appl 15(4):75–85
28. Strutz T (2009) Lifting parameterization of the 9/7 wavelet filter bank and its application in lossless image compression. In: ISPRA'09, Cambridge, UK, pp 161–166
29. Strutz T (2009) Wavelet filter design based on the lifting scheme and its application in lossless image compression. WSEAS Trans Signal Process 5(2):53–62
30. Sydney Burus C, Gopinath RA, Guo H (1998) Introduction to wavelets and wavelet transforms: a primer. Prentice-Hall, Inc, Upper Saddle River
31. Veterli M, Kovačević J (1995) Wavelets and subband coding. Prentice Hall, Englewood Cliffs
32. Viswanath G, Sreenivas TV (2002) IF estimation using higher order TFRs. Signal Process 82 (2):127–132

Chapter 2
Digital Audio

2.1 The Nature of Sound

The sound is created as a result of wave fluctuations around the vibrating material. The propagation speed, frequency, and sound pressure level are important sound features. For example, the sound propagation speed through the air under normal atmospheric conditions is 344 m/s. Since, in this chapter, we will focus our attention to specific types of audio signals such as speech and music, let us consider their frequency characteristics. Music is defined as the sound that has a distinct periodicity. Its frequency ranges from 20 to 20 KHz; while in the case of speech, the frequency ranges from 50 to 10 KHz. It is important to note that the human auditory system is most sensitive to frequencies from 700 to 6,600 Hz.

Let us observe what affects the perception of sound in the human auditory system. If we consider a closed room, as shown in Fig. 2.1, the auditory system receives direct and reflected waves. Reflected waves are delayed in comparison to the direct waves. The number of reflected waves and their respective delays depend on the geometry of the room.

The position of the sound source is perceived based on the delays between the direct and reflected waves detected by left and right ear. The time delay between two ears is about 0.7 ms. Here, it is interesting to mention some effects that appear as a result of the stereo nature of the human auditory system. For example, if one signal channel is delayed for 15 ms with respect to the other, it will be perceived as a signal with lower amplitude, although both signals are actually of the same amplitude. Hence, this effect can be reduced by increasing the amplitude of delayed signal. However, the auditory system registers two different sounds if the delay exceeds 50 ms.

The sound pressure level (SPL) is another key characteristic of audio signals. The SPL is the ratio of the measured sound pressure to the reference pressure ($P_0 = 20\mu\text{Pa}$). The reference pressure denotes the lowest SPL that can be registered

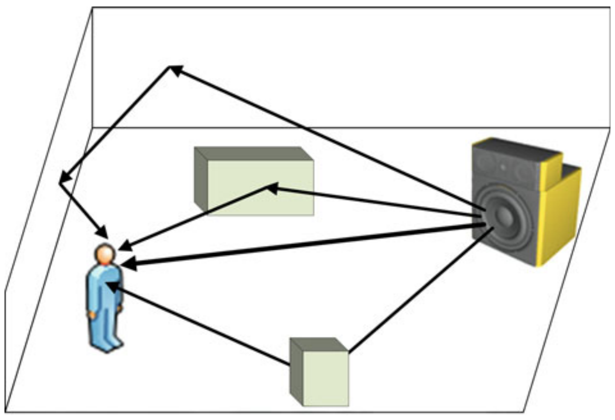


Fig. 2.1 An illustration of sound propagation within a closed room

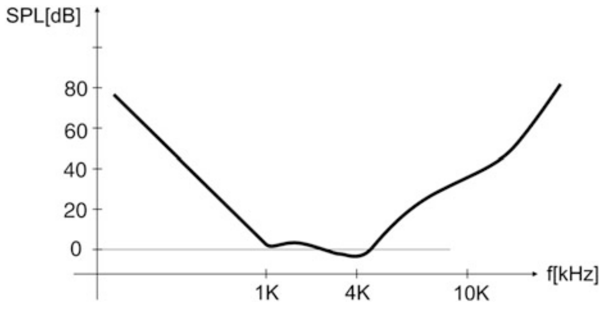


Fig. 2.2 The Fletcher curve

by the auditory system in a noise-free environment. The sound pressure is calculated as follows:

$$\mathrm{SPL} = 20\log_{10}\frac{P}{P_0}[\mathrm{dB}]. \tag{2.1}$$

In addition to these characteristics of acoustic signals, the Fletcher curve, shown in Fig. 2.2, is a measure of SPL over the frequency spectrum for which a listener perceives a constant loudness when presented with pure steady tones. From Fig. 2.2, it can be observed that the human auditory system has a nonlinear sensitivity to the frequency.

2.2 Development of Systems for Storing and Playback of Digital Audio

The first system for audio recording and playback dates back to 1877 (the Edison phonograph). The first gramophone dates back to 1893. Electrical playback systems began replacing mechanical systems in 1925. The broadcast of AM (amplitude modulated) audio signals began in 1930. The LP (Long Play) system with a playback time of about 25 min was developed in 1948. This brief review of some of the inventions testifies that the audio industry has developed significantly over the last 100 years. For example, the first gramophones could play recordings about 2 min long, and the system used 78 rpm. The frequency range of the system was 200 Hz–3 KHz and its dynamic range was 18 dB. The later systems had the extended frequency range (30–15 KHz), with the dynamic range being 65 dB.

Efforts to improve the performance of audio devices have led to the use of tape recorders during the 1960s and 1970s. The development of compact disc (CD) began during 1970s, when Mitsubishi, Sony, and Hitachi demonstrated the digital audio disc (DAD). DAD was 30 cm in diameter. Philips and Sony continued to work together on this system. As a result, they produced a disc with a diameter of 12 cm in the early 1980s. The capacity of the disc was 74 min. A further development of the CD technology led to the development of mini discs, digital versatile discs (DVD), super audio CDs (SACDs).

Along with the development of digital audio devices, there was a growing need to develop systems for digital audio broadcasting (DAB). The used bandwidth is 1.54 MHz. The frequency blocks are arranged as: 12 frequency blocks in the range 87–108 MHz, 39 blocks in the VHF band (174–240 MHz) and 23 frequency blocks in the L band (1,452–1,492 MHz). An example of DAB system is given in Fig. 2.3,

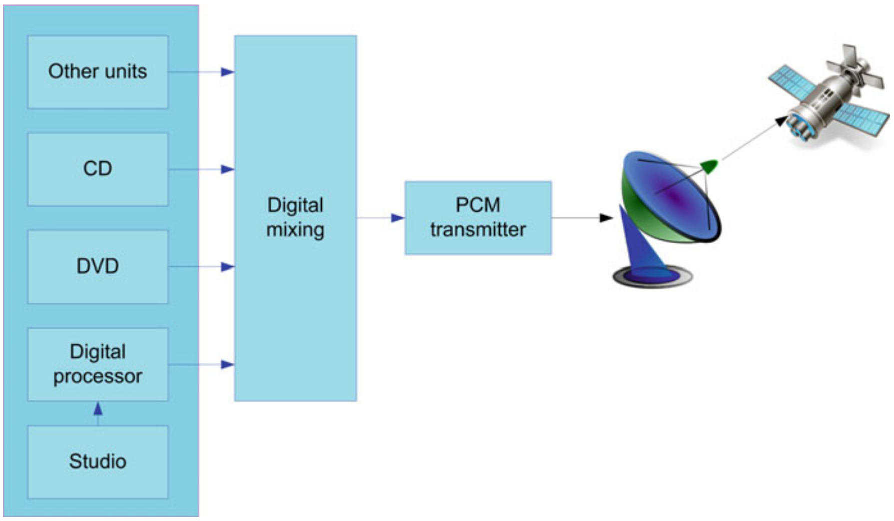


Fig. 2.3 Block diagram of a DAB system

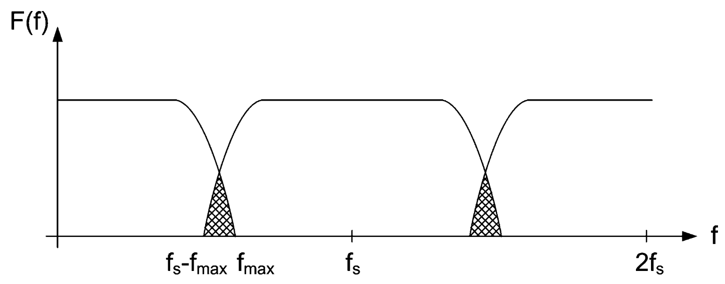


Fig. 2.4 Aliasing effects

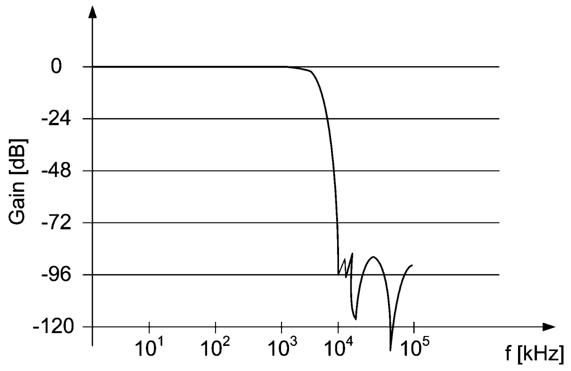


Fig. 2.5 An example of antialiasing filter with a steep transition

showing the general principle of combining different signals and their transmission in digital form.

2.3 Effects of Sampling and Quantization on the Quality of Audio Signal

Sampling is the first step in digitalization of analog signals. Recall that sampling causes periodic extensions in the frequency domain. If the discretization is performed according to the sampling theorem, then the basic part of the signal spectrum will not overlap with periodic extensions. However, if the sampling rate is not sufficiently high, then there is a spectrum overlap (or aliasing) (Fig. 2.4).

The signal spectrum is extracted by using antialiasing filters with steep transition from pass to stop regions (a filter example is shown in Fig. 2.5). Note that filters with steep transitions are usually the higher order ones.

In many real applications, it is necessary to use more economic versions of antialiasing filters of lower orders. Therefore, the sampling rate is increased beyond what is required by the sampling theorem in order to allow for less steeper

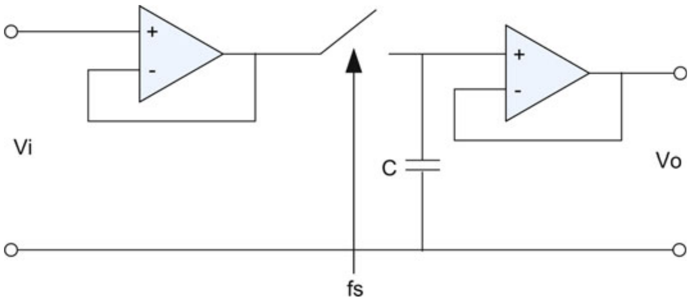


Fig. 2.6 A circuit for signal sampling

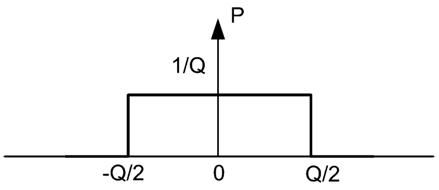


Fig. 2.7 The probability density function of the quantization error

transitions. For example, the sampling frequency used for a CD is equal to 44.1 KHz, although the maximum frequency we want to reproduce is 20 KHz.

A sample and hold circuit that can be used for sampling of analog signals is shown in Fig. 2.6. A switching element is controlled by the signal *fs*, which defines the sampling frequency. The operational amplifier provides high resistance, and thus a large time constant for the capacitor C to discharge. Thus, the voltage on the capacitor is changed slightly between the two control pulses *fs*.

The next step after the sampling process is the quantization. Analog signals can have infinitely many different values, but the number of quantization levels is limited. As a result, the signal after quantization can meet only a certain degree of accuracy, as defined by the number of quantization levels. In other words, the quantization introduces the quantization noise. A relationship between the signal-to-noise ratio (S/N or SNR) and the number of bits (which is determined by the number of quantization levels) can be easily determined. Suppose the probability density function of quantization error is uniform, as shown in Fig. 2.7.

The number of quantization levels in an n-bit system is denoted as $M = 2^n$. Now, consider a sinusoidal signal with the amplitude $V/2$. Then, the quantization interval is $Q = V/(M - 1)$. Since the quantization noise is uniformly distributed in the range $[-Q/2, Q/2]$, the quantization noise power is equal to:

$$N = \frac{2}{Q}\int_0^{Q/2} x^2 \mathrm{d}x = \frac{2}{Q}\frac{(Q/2)^3}{3} = \frac{Q^2}{12}. \tag{2.2}$$

On the other hand, the power of a sinusoidal signal is equal to:

$$P = \frac{1}{2\pi}\int_{0}^{2\pi}\left(\frac{V}{2}\right)^2 \sin^2 x\mathrm{d}x = \frac{1}{2\pi}\frac{V^2}{4}\int_{0}^{2\pi}\frac{1-\cos 2x}{2}\mathrm{d}x = \frac{V^2}{8}. \tag{2.3}$$

Therefore, S/N in the n-bit system is given by:

$$S/N = \frac{P}{N} = \frac{V^2/8}{V^2/(2^{2n}/12)} = \frac{3}{2}2^{2n}, \tag{2.4}$$

or equivalently:

$$S/N[\mathrm{dB}] = 10\log\frac{S}{N} = 10\log\frac{3}{2} + 10\log 2^{2n} = 1.76 + 6n. \tag{2.5}$$

For example, if we use 16 bits to quantize the signal, then S/N $\approx$ 98 dB.

2.3.1 Nonlinear Quantization

The previous section discussed a uniform quantization approach (where each quantization interval Q is identical). However, we can assign the quantization levels in a nonlinear manner. For instance, the quantization levels can be adjusted according to the input signal amplitude, such that a small amplitude signal will have smaller quantization intervals, and vice versa.

A process of nonlinear quantization of a variable x can be described as follows: First x is transformed (compressed) by using the nonlinear function f (i.e., $f(x)$), which is then linearly quantized. The quantized values are then processed (expanded) by the inverse nonlinear function f^{-1}. Lastly, for a nonlinear quantizer, we have:

$$Q(x) = f^{-1}(Q_u(f(x))), \tag{2.6}$$

where $Q_u(x)$ denotes a linear quantizer. A typical function for nonlinear quantization is the A-law, which is defined as follows:

$$F(x) = \begin{cases} Ax/(1+\ln A) \text{ for } 0 < x \leq V/A, \\ V(1+\ln(Ax/V))/(1+\ln A) \text{ for } V/A \leq x \leq V, \end{cases} \tag{2.7}$$

where A is a constant that controls the compression ratio, while the peak magnitude of the input signal is labeled as V. $A = 87.6$ is often used in practice.

Figure 2.8 depicts the process of nonlinear quantization. The x-axis represents the normalized amplitude of the input signal, while the y-axis represents the values

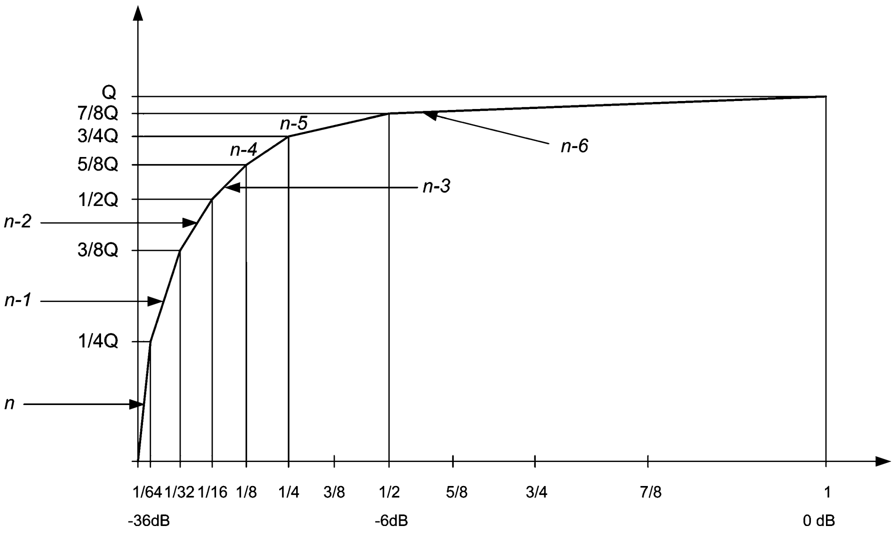


Fig. 2.8 Nonlinear quantization

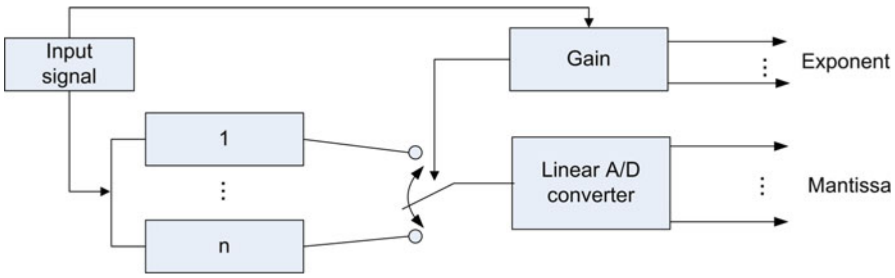


Fig. 2.9 Floating-point conversion

of quantization intervals. For example, when the signal amplitude drops four times (-12 dB), the quantization interval is 3/4Q.

The concept of nonlinear quantization is applied in other schemes such as the floating-point conversion, which is used in professional audio systems. The principle of floating-point conversion is shown in Fig. 2.9.

This system is based on the principle of a logarithmic scale. Namely, the signal is sent through several parallel circuits with different gains ensuring that the input to linear A/D converter is always a signal whose level is suitable for linear conversion. The converted part of the signal is called the mantissa.

Information on the amplitude of the signal is provided through the second part of the system, whose output is a binary value called the exponent. Note that with three bits of the exponent, we can achieve a conversion of signals with the following

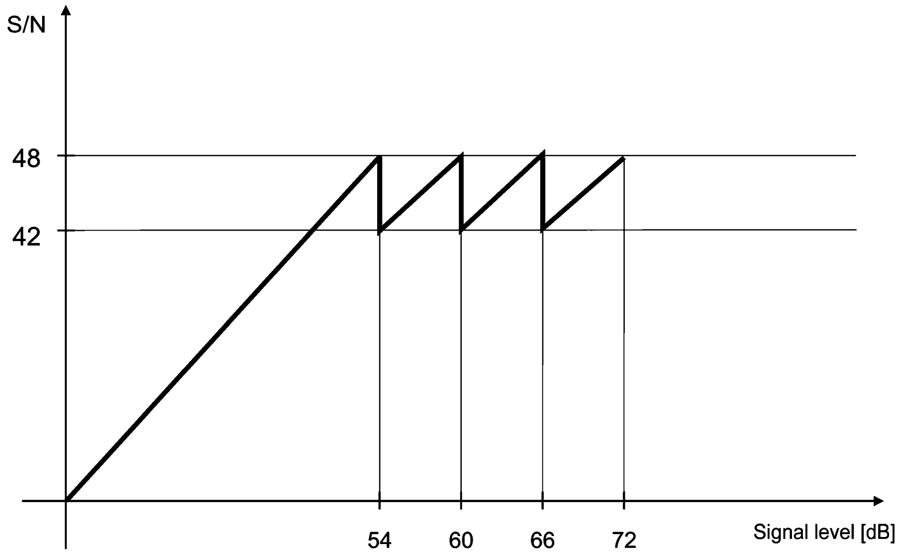


Fig. 2.10 S/N ratio for a considered floating-point converter (8-bit mantissa and 3-bit exponent)

gains: 0, 6, 12, 18, 24, 30, 36, and 42 dB. Hence, we can effectively digitize signals with very different amplitude levels, which is often a practical demand for audio signals. A typical S/N curve for a signal based on an 8-bit mantissa and a 3-bit exponent is illustrated in Fig. 2.10.

It should be noted that although it is an 11-bit system, the S/N is between 42 and 48 dB, and its maximum value is defined by the mantissa.

2.3.2 *Block Floating-Point Conversion*

This is a special case of floating-point conversion, used when a low bandwidth is required. Namely, the exponent is not associated with every sample, but it is done for a block of successive samples. In this way, a considerable bit rate reduction is enabled. This technique is also known as near-instantaneous companding.

2.3.3 *Differential Pulse Code Modulation*

Using the previous conversion techniques, we analyze each sample separately in order to prepare it for transmission. In the case of differential pulse code modulation (DPCM), we transmit the differences between neighboring samples.

This modulation is a form of predictive coding in which the prediction for the current sample is carried out on the basis of the previous sample. It is particularly

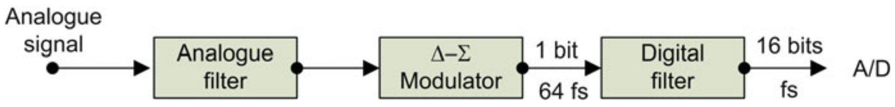


Fig. 2.11 Single-bit A/D converter

efficient when a small sampling period is used, since the differences between adjacent samples are very small and practically related to a single bit (the least significant bit). Sigma delta converters are used for this type of conversion. Note that the serial bit stream is impractical, and therefore digital filters (decimation filters) are usually applied to convert the serial stream into a multibit format (e.g., 16 bits for the CD system). A block scheme of a single-bit A/D converter is shown in Fig. 2.11.

2.3.4 Super Bit Mapping

In the CD technology, audio signals are usually encoded with 16 bits. In some cases (e.g., professional audio studios), 20 bits are used for encoding of audio signal. Then, the super bit mapping is used to convert 20-bit signals to 16-bit signals. The additional four bits are used to increase the accuracy of the least significant bits of the 16-bit signal.

2.4 Speech Signals

The system for generating speech signals is illustrated in Fig. 2.12. We can see that the lungs initialize the air flow through the trachea and larynx to the mouth. The lips form a longitudinal wave that will spread further through the air.

Note that the air flow is modulated by passing through the larynx and the vocal folds. Therefore, the vocal folds generate waves that pass through the mouth and the nasal cavity. The observed system for the voice production can be viewed through two subsystems called glottal and vocal tract. The glottal tract (up to the beginning of the pharynx) generates waves under the influence of the vocal folds, while the vocal tract works as a set of resonators and filters, which modulate and shape the wave in order to make specific sounds.

As the speech sounds can be divided into vowels and consonants, it is necessary to describe how they are formed within the speech production system. When generating vowels, the vocal folds resonate and produce quasi-periodic oscillating impulses that continue to be shaped in the vocal tract where the oral cavity acts as a resonator. During this process, some of the frequencies are attenuated, while others are amplified. By examining the spectrum of vowels, we can notice some harmonics that dominate over other components. These harmonics are called the formants

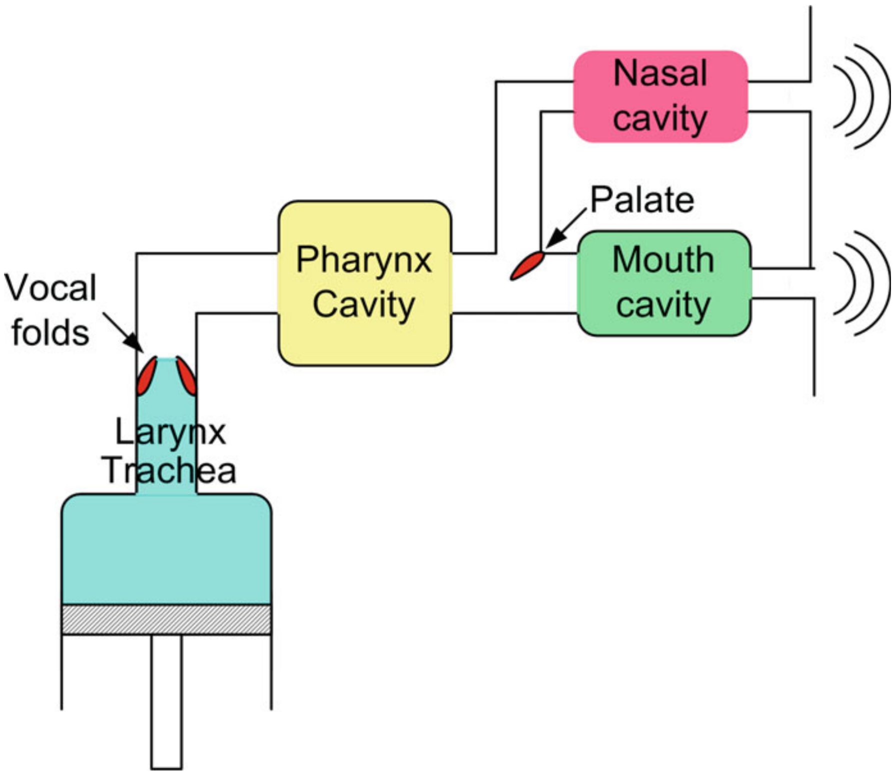


Fig. 2.12 Illustration of the speech generation system

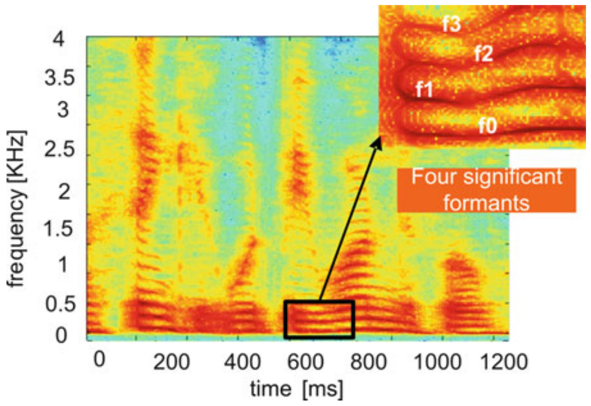


Fig. 2.13 Time-frequency representation of speech formants

and they actually represent the resonant frequencies of vocal tract. When analyzing the speech signal, we can often observe the first four formants. The structure of formants in the time-frequency domain is shown in Fig. 2.13.

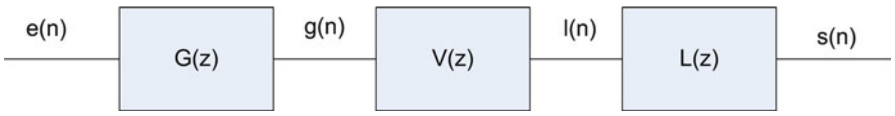


Fig. 2.14 A model of the speech production system

The strongest formants for the vowel *A* range from 700 to 1,000 Hz. For the vowel *I* these formants are in the range of 200–400 Hz and 2,200–3,200 Hz, while for the vowel *O* they are restricted to frequencies from 400 to 800 Hz.

Consonants can be divided into voiced and voiceless consonants. In the case of voiced consonants, the vocal folds produce noise, which is then modulated in the vocal tract. Although, the noise spectrum is mainly spread and continuous, the specific components representing a certain form of formants appear as well. Voiceless consonants occur only in the oral cavity, when the vocal folds are not active.

Let us define some of the most important features of the formant, since it represents an important voice characteristic. The formant frequency is the maximum frequency within the frequency band defined by the formant. The formant bandwidth is defined as the frequency region in which the amplification differs less than 3 dB from the amplification at the peak (central) frequency of the formant.

Having in mind the characteristics of the speech production system, the speech signal can have a variety of values due to its continuous nature. However, from the perceptual point of view, we are able to distinguish just a finite number of sounds, since there is a limited set of meaningful information contained in speech. In this way, we consider only the functional units called phonemes. Note that the same phoneme can occur in different forms, which have no impact on its meaning. In other words, the strength and timbre of the voice will not affect the understanding of phonemes and will not change their functional value.

2.4.1 *Linear Model of Speech Production System*

Based on the previous analysis, we can model the speech production system as shown in Fig. 2.14.

The transfer functions of the glottal tract, the vocal tract, and the lips are denoted by $G(z)$, $V(z)$, and $L(z)$, respectively. $e(n)$ is the input excitation signal, which can be modeled as a train of Dirac impulses for voiced sounds or Gaussian noise for unvoiced sounds. Based on the system in Fig. 2.14, we can write:

$$S(z) = E(z)G(z)V(z)L(z). \tag{2.8}$$

By introducing the inverse filter:

$$A(z) = \frac{1}{G(z)V(z)L(z)}, \tag{2.9}$$

where $A(z)$ has a form of all-zero filter $A(z) = 1 + \sum_{i=1}^{p} a_i z^{-i}$, we can write the following relation:

$$E(z) = A(z)S(z). \tag{2.10}$$

In other words, if z^{-1} is interpreted as the unit delay operator: $z^{-1}s(n) = s(n-1)$, then the previous relation can be written as the autoregressive model of order *p:*

$$s(n) + \sum_{i=1}^{p} a_i s(n-i) = e(n). \tag{2.11}$$

We can model every 700 Hz with one pair of poles.

Let us consider now the impact of the glottal tract and mouth. The speech production system can be observed from the glottal wave $g(n)$. Moreover, the characteristics of the glottal wave are known and given by:

$$g(t) = \begin{cases} \sin^2 \dfrac{\pi t}{2T_p}, & \text{for } 0 \le t \le T_p, \\ \cos \dfrac{\pi\left(t - T_p\right)}{2T_n}, & \text{for } T_p < t \le T_c, T_c = T_p + T_n, \\ 0, \text{ for } T_c < t \le T, \end{cases} \tag{2.12}$$

where $T_p = 3.25$ ms, $T_n = 1.25$ ms, and the pitch period (time interval between two consecutive periodic excitation cycles) is $T = 8$ ms. The glottal tract can be modeled by the following transfer function:

$$Hg(z) = \frac{1}{(1 - qz^{-1})^2}, \tag{2.13}$$

which attenuates –12 dB/oct. (for $q \approx 1$). The influence of radiation from the lips can be approximated by:

$$L(z) = 1 - z^{-1}. \tag{2.14}$$

Since a linear model of the speech production system is assumed, the transfer functions $L(Z)$ and $V(Z)$ in Fig. 2.14 can replace the positions. Thus, as the input of $V(z)$, we have:

$$\left(1 - z^{-1}\right)g(n) = g(n) - g(n-1) = g'(n), \tag{2.15}$$

where $g'(n)$ is a differentiated glottal wave. When considering the remaining part of the system, we get:

$$s(n) = V(z)g'(n). \tag{2.16}$$

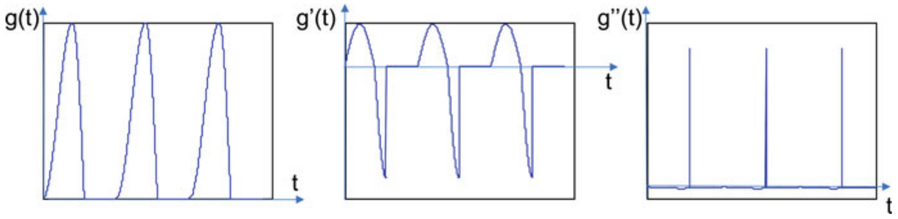


Fig. 2.15 Excitation signals $g(t)$, $g'(t)$, $g''(t)$

Next, an additional differentiation can be performed, which will result in:

$$s'(n) = V(z)g''(n). \tag{2.17}$$

Assuming that:

$$V(z) = \frac{1}{A_p(z)} = \frac{1}{1 + \sum_{i=1}^{p} a_i z^{-i}}, \tag{2.18}$$

we can obtain the final model of the speech production system:

$$s'(n)A_p(z) = s'(n) + \sum_{i=1}^{p} a_i s'(n-i) = g''(n). \tag{2.19}$$

It is important to emphasize that $g''(n)$ is the excitation signal that can be approximated in the form of the Dirac pulse train (Fig. 2.15). Then the signal $s'(n)$ is the pre-emphasized signal $s(n)$, with no influence of the glottal wave and radiation. This model also represents the autoregressive model of the order p, as the one defined by (2.11).

2.5 Voice Activity Analysis and Detectors

Recall that different speech sounds are formed by forcing the air through the vocal system. They could be classified as voiced and unvoiced speech sounds, as shown in Fig. 2.16. Voiced speech parts are generated by vocal folds vibrations that cause the periodical air oscillations. As a result, a sequence of air pulses is created, which excites the vocal tract and produces the acoustically filtered output. On the other hand, the unvoiced sounds are usually generated by forcing the air through certain constrictions in the vocal tract.

The voiced sounds are characterized by a significant periodicity in the time domain, with the pitch (fundamental) frequency. Note that, the unvoiced sounds

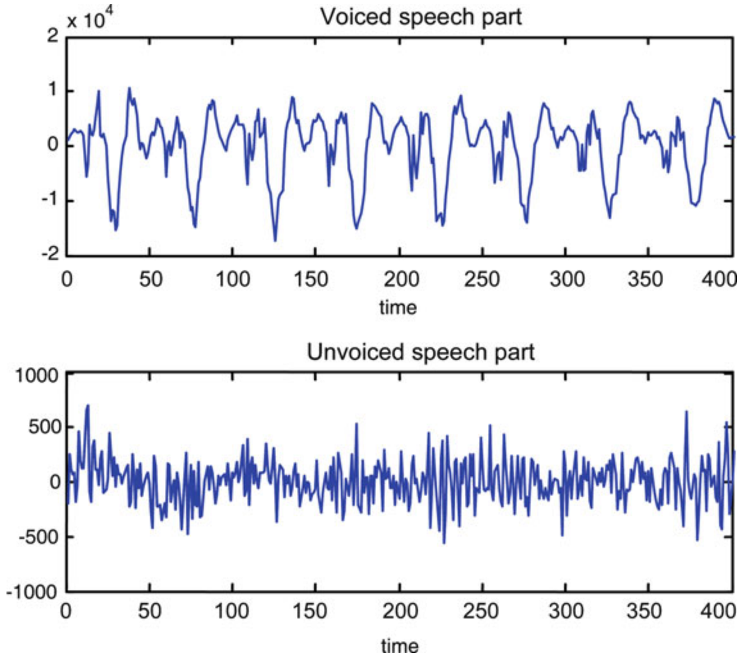


Fig. 2.16 An illustration of different speech parts

have a more noisy-like nature. Also, the voiced parts are characterized by significantly higher energy compared to the unvoiced sounds. As mentioned before, the voiced sounds contain formants in the frequency domain. Formants are very important in the speech analysis and applications (e.g., speech coding). Frequency components of unvoiced sounds are generally low-energy components located mostly at the high frequencies. Due to the significant differences between voiced and unvoiced speech parts, some applications employ the sounds classification as a preprocessing step. The classification of voiced and unvoiced sounds can be done by using voice activity detectors. These detectors are based on voice activity indicators (energy, zero-crossing rate, prediction gain, etc.) combined with thresholding to decide between voiced and unvoiced option. Some of the existing voice activity indicators are described in the sequel.

Energy

Before processing, the speech signals are usually divided into frames with a certain number of samples. The length of the frame is determined such that the statistical signal characteristics are almost constant within the frame. The simplest way to make differentiation between the voiced and unvoiced parts is the frame energy, which is defined as:

$$E(n) = \sum_{k=n-N+1}^{n} s^2(k), \tag{2.20}$$

where s denotes the speech signal, N is the length of frame, while n is the endpoint of the frame. The voiced parts have the energy that is several times higher than the unvoiced parts energy.

Instead of energy, one can use the magnitudes of the frame samples:

$$\mathrm{MA}(n) = \sum_{k=n-N+1}^{n} |s(k)|. \tag{2.21}$$

Zero-Crossing Rate

Due to the presence of low-frequency pitch component, the voiced sounds are characterized by a low zero-crossing rate compared to the unvoiced sounds. For a certain frame, the zero-crossing rate can be calculated as follows:

$$\mathrm{ZC}(n) = \frac{1}{2} \sum_{k=n-N+1}^{n} \left|\mathrm{sgn}(s(k)) - \mathrm{sgn}\big(s(k-1)\big)\right|. \tag{2.22}$$

Prediction Gain

As previously mentioned, the linear prediction algorithm is commonly used in the analysis and synthesis of speech signals. This method provides the extraction of certain sound characteristics that can be used for the voiced/unvoiced speech classification. The prediction of discrete signal $s(n)$ based on the M samples can be defined as:

$$\widehat{s}(k) = -\sum_{i=1}^{M} a_i s(k-i), \quad k = n-N+1, ..., n, \tag{2.23}$$

where a_i, $i = 1,\ldots,M$ are estimated linear prediction coefficients of the autoregressive model, while M is the order of the prediction system. For a nonstationary signal such as speech, the linear prediction is performed separately for each frame.

The estimation of linear prediction parameters is based on the criterion of mean square prediction error:

$$J = E\{e^2(k)\} = E\left\{\left(s(k) + \sum_{i=1}^{M} a_i s(k-i)\right)^2\right\}. \tag{2.24}$$

The optimal linear prediction coefficients are obtained by solving the system of equations based on the partial derivatives of the error function J with respect to parameters a_m, for $m = 1, 2,\ldots,M$:

$$\frac{\partial J}{\partial a_m} = 2E\left\{\left(s(k) + \sum_{i=1}^{M} a_i s(k-i)\right) s(k-m)\right\} = 0. \tag{2.25}$$

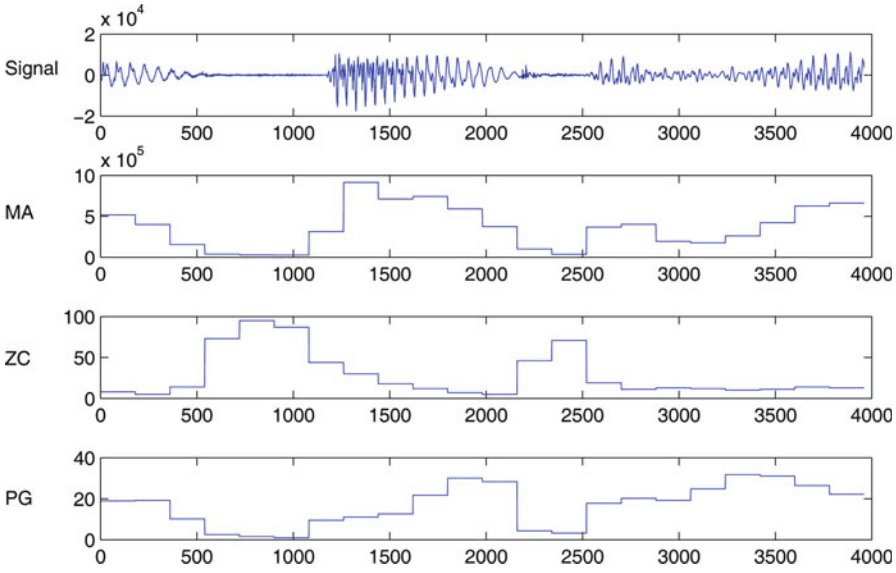


Fig. 2.17 The outputs of the voice activity indicators based on the magnitudes (*MA*), zero-crossing rate (*ZC*), and prediction gain (*PG*)

The prediction gain is defined as the ratio between the signal energy and the prediction error:

$$\mathrm{PG}[n] = 10\log_{10}\left(\frac{\sum\limits_{k=n-N+1}^{n} s^2(k)}{\sum\limits_{k=n-N+1}^{n} e^2(k)}\right). \tag{2.26}$$

This parameter can be used as an indicator of differences between the voiced and unvoiced speech parts. It is known that voiced sounds achieve higher prediction gain compared to the unvoiced ones for at least 3 dB. The periodicity of voiced frames causes a stronger correlation between the frame samples. On the other hand, random nature of unvoiced parts makes the prediction less efficient.

The outputs of the considered voiced/unvoiced sounds indicators for the frames with 180 samples (22.5 ms when the sampling rate is 8 KHz) are illustrated in Fig. 2.17.

The simple versions of the voice activity detectors assume one of these indicators as the input signal. As in the standard classification problems, here also it is necessary to define suitable thresholds to separate the voiced and unvoiced speech parts. The thresholds setting is based on the analysis of large signal sets, with the aim to minimize the classification errors. In the practical applications, the considered detectors could be combined to improve the performance of the detection system.

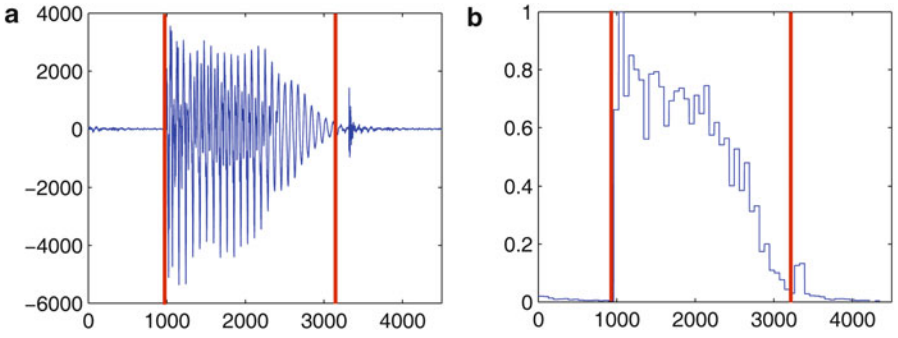


Fig. 2.18 (**a**) Speech signal, (**b**) Energy-entropy feature for speech frames

2.5.1 Word Endpoints Detector

The start and endpoints of words can be detected by using a word endpoints detector. One realization of this detector is based on the energy-entropy signal feature. The signal is first divided into time frames that are 8 ms long (e.g., 64 samples long for a speech signal sampled at 8 KHz). The energy of frame E_i is calculated according to (2.20). On the other hand, the probability density function for the speech spectrum $S(\omega)$ is obtained by normalizing the frequency content within the frame. Hence, for the i-th frame, we have:

$$p_i = \frac{S(\omega_i)}{\sum_{k=1}^{N} S(\omega_k)}, \tag{2.27}$$

where N is the number of components within the frame. The energy-entropy feature can be calculated as follows:

$$\mathrm{EEF}_i = (1 + |E_i \cdot H_i|)^{1/2}, \tag{2.28}$$

where H_i represents the entropy of the i-th frame defined as:

$$H_i = \sum_{k=1}^{K} p_k \log p_k. \tag{2.29}$$

Energy-entropy features for the consecutive frames of speech signal are illustrated in Fig. 2.18.

By using the energy-entropy feature, the start and the endpoint of a spoken word can be determined as follows:

$$\begin{aligned} t_s &= \arg\min_i \{\mathrm{EEF}(i) > T_1\}, 1 \le i \le N \\ t_e &= \arg\max_i \{\mathrm{EEF}(i) > T_2\}, 1 \le i \le N \end{aligned} \tag{2.30}$$

where N is the total number of considered speech frames, while T_1 and T_2 are thresholds for the start and endpoint, respectively. The thresholds can be set empirically, based on various experiments with different speech signals and speakers. The typical values for the thresholds are $T_1 = 0.16$ and $T_2 = 0.17$. The resulting word endpoints are illustrated in Fig. 2.18.

2.6 Speech and Music Decomposition Algorithm

The singular value decomposition (SVD) has been used in numerous practical applications for characterization of signals and their components. The SVD has been applied on the time-frequency distributions to extract features used for signal characterization. Most of the procedures are based on the use of singular values. However, significant information about the patterns embedded in the matrix can be obtained by using the left and right singular vectors, especially those corresponding to the largest singular values. Namely, the left and right singular vectors contain the information about time and frequency domain of the signal, respectively. Here, the SVD is used to extract speech and musical components from the autocorrelation function. The autocorrelation function is obtained by using the inversion of suitable time-frequency distribution, as described in the sequel.

2.6.1 Principal Components Analysis Based on SVD

SVD transforms the original correlated variables into the uncorrelated set of variables. It allows one to identify the direction along which the data variations are dominant. For a certain matrix S, SVD is defined as follows:

$$S = U\Sigma V^T, \tag{2.31}$$

where Σ is a diagonal matrix of singular values. Σ is of the same size as S, and the values are sorted in decreasing order along the main diagonal. The U and V are orthonormal matrices whose columns represent left and right singular vectors, respectively. If S is $M \times N$ matrix ($M > N$), then the size of U is $M \times M$, Σ is an $M \times N$ matrix, while V is an $N \times N$ matrix. A memory-efficient method known as economy-sized SVD is computed as follows:

– Only N columns of U are computed.
– Only N rows of Σ are computed.

2.6.2 Components Extraction by Using the SVD and the S-Method

The audio signals, such as the speech and musical signals are multicomponent signals: $f(n) = \sum_{c} f_c(n)$. Let us consider the inverse Wigner distribution for a separately observed c-th signal component:

$$f_c(n+m)f_c^*(n-m) = \frac{1}{N+1} \sum_{k=-N/2}^{N/2} \mathrm{WD}_c(n,k) e^{j\frac{2\pi}{N+1}k2m}. \tag{2.32}$$

By replacing $n+m=p$ and $n-m=q$, we obtain:

$$f_c(p)f_c^*(q) = \frac{1}{N+1} \sum_{k=-N/2}^{N/2} \mathrm{WD}_c\left(\frac{p+q}{2}, k\right) e^{j\frac{2\pi}{N+1}(p-q)k}. \tag{2.33}$$

The left-hand side corresponds to the autocorrelation matrix:

$$R_c(p,q) = f_c(p)f_c^*(q),$$

where $f_c(p)$ is a column vector, whose elements represent the signal terms, and f_c* (q) is a row vector, with complex conjugate elements. For a sum of M signal components, the total autocorrelation matrix becomes:

$$\begin{aligned} \sum_{c=1}^{M} R_c(p,q) &= \frac{1}{N+1} \sum_{k=-N/2}^{N/2} \sum_{c=1}^{M} \mathrm{WD}_c\left(\frac{p+q}{2}, k\right) e^{j\frac{2\pi}{N+1}(p-q)k} \\ &= \frac{1}{N+1} \sum_{k=-N/2}^{N/2} \mathrm{SM}\left(\frac{p+q}{2}, k\right) e^{j\frac{2\pi}{N+1}(p-q)k}. \end{aligned} \tag{2.34}$$

Applying SVD to the autocorrelation matrix, we get:

$$R(p,q) = \sum_{c=1}^{M} R_c(p,q) = U\Sigma V^T. \tag{2.35}$$

Furthermore, we observe the case when the time-frequency distribution is represented by a square matrix, i.e., time and frequency dimensions are the same. Consequently, the autocorrelation function $R(p, q)$ will be the symmetric square matrix with the symmetry axis along the main diagonal. Therefore, we have $U = V$ are the matrices containing eigenvectors, while $\Sigma = \Lambda$ is the eigenvalue matrix:

$$U\Sigma V^T = U\Lambda U^T. \tag{2.36}$$

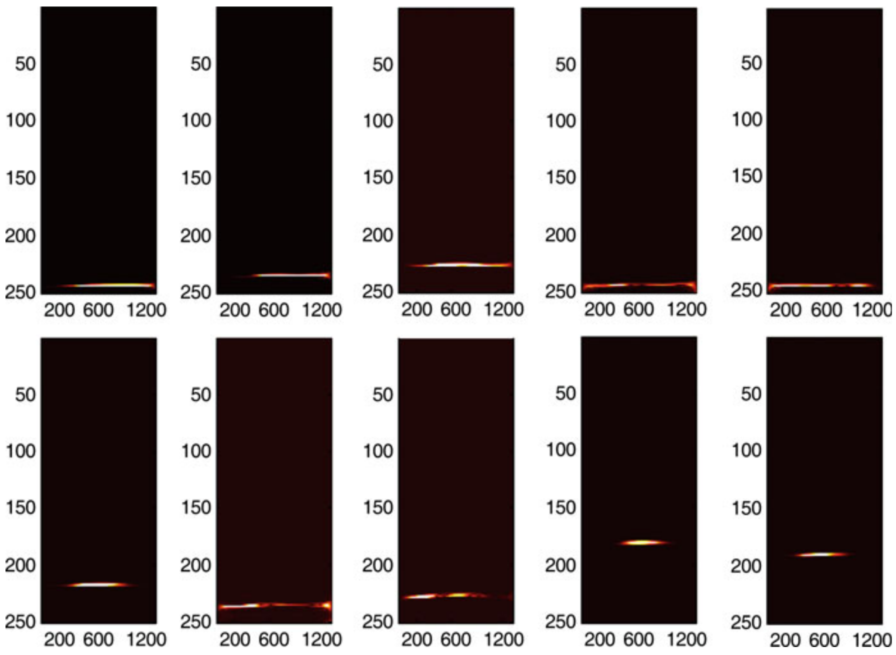


Fig. 2.19 The formants isolated by using the eigenvalues decomposition method

Hence, the autocorrelation matrix $R(p, q)$ can be decomposed as follows:

$$R(p,q) = \sum_{c=1}^{M} R_c(p,q) = \sum_{j=1}^{M} \lambda_j u_j(n) u^*{}_j(n), \tag{2.37}$$

where λ_j are the eigenvalues and $u_j(n)$ are the eigenvectors of the autocorrelation matrix R. Note that the eigenvectors correspond to the signal components, while the eigenvalues are related to the components energy.

The speech formants, separated by using the eigenvalue decomposition, are shown in Fig. 2.19 (the formants at positive frequencies are shown). Now, it is possible to arbitrarily combine the components that belong to the low-, middle-, or high-frequency regions. Consequently, an arbitrary time-frequency mask (Fig. 2.20) can be made and used in speech processing applications.

Let us consider a violin signal with a number of closely spaced components, as it can be seen from Fig. 2.21. The eigenvalue decomposition method is applied in the same way as in the case of speech signal. The extracted components are shown in Fig. 2.22. It is important to note that, due to the specific nature of audio signals, the perfect signal reconstruction from its separated components is not fully attainable.

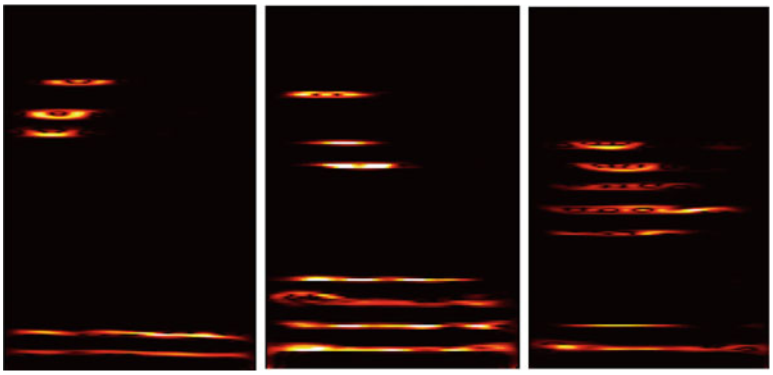


Fig. 2.20 Illustrations of different components combinations selected by a few time-frequency masks

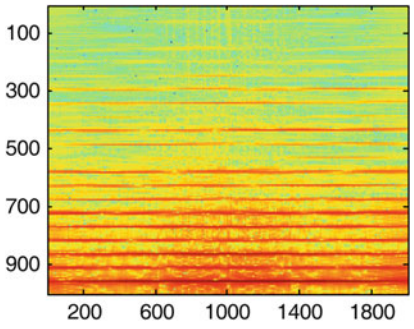


Fig. 2.21 The time-frequency representation of the violin signal obtained by using the S-method

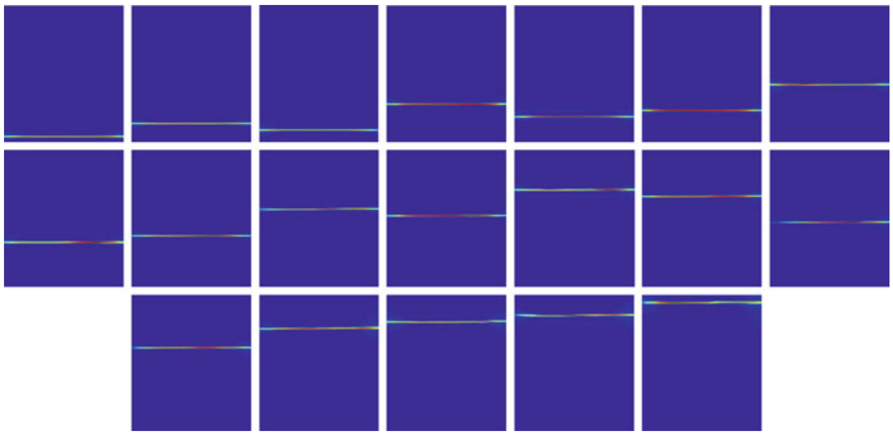


Fig. 2.22 Separated components of violin signal

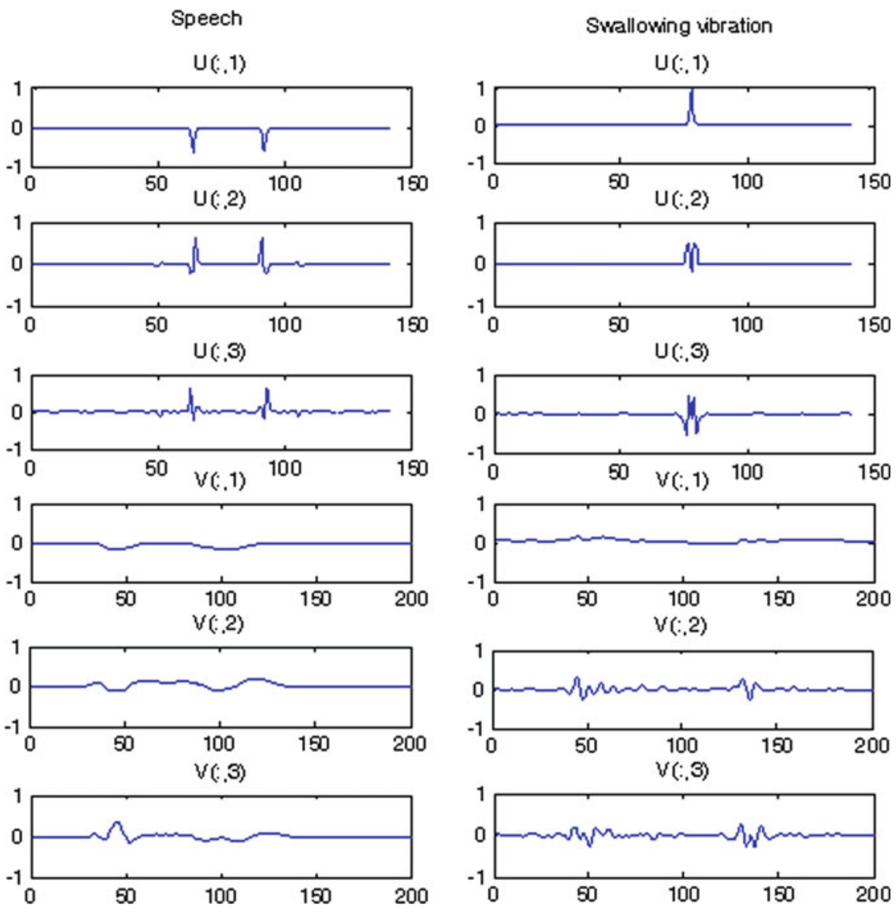


Fig. 2.23 Left and right vectors of speech (*left column*) and swallowing sound (*right column*)

The characteristics of a signal in the time-frequency domain could also be analyzed by using the SVD technique. Namely, the right and left singular vectors (within the matrices U and V) contain information about signal characteristics along the time and frequency axes, respectively. Usually, only a few singular vectors (left and right) associated with the highest singular values are observed, since they bring most of the information about the signal structure. Namely, the specific features obtained from these vectors can be combined with neural networks for analysis and classification of sounds. An example of differences between the first three singular vectors, obtained from the time-frequency region containing speech and swallowing sounds (recorded by the microphone on the neck) are illustrated in Fig. 2.23.

2.7 Psychoacoustic Effects

It was mentioned earlier that the ear is not equally sensitive to different frequencies. The sensitivity function (shown in Fig. 2.2) is obtained experimentally and is given by the following expression:

$$T(f) = 3.64\left(\frac{f}{1,000}\right)^{-0.8} - 6.5e^{-0.6(f/1,000-3.3)^2} + 10^{-3}\left(\frac{f}{1,000}\right)^{4}\ \text{dB}. \quad (2.38)$$

Let us perform now a detailed analysis of the auditory system. It is composed of the outer (lobe) ear, the middle ear, and the inner ear, as illustrated in Fig. 2.24. The auditory system up to the inner ear can be simply represented as a combination of a horn and open pipes.

Sound waves, collected by the ear shell, are forwarded over the ear channel. In the inner ear there is the organ of Corti, which contains the fibrous elements with different lengths and resonant frequencies. These elements are connected to the auditory nerve that is used to convey any information to the brain. As a consequence of the applied sound wave, the mechanical vibrations are passed through the ossicles to the cochlea causing the basilar membrane to oscillate. The parts of basilar membrane resonate depending on the frequencies (Fig. 2.24). In the case of high frequencies, the resonance is produced in the front part of basilar membrane, while in the case of low frequencies, it occurs in the rear part.

The hearing system works effectively as a filter bank. We devote our attention to a particular sound only after our brain focuses on it.

2.7.1 *Audio Masking*

We have already stated that there is a threshold value of SPL below which we cannot hear a beep. However, even the components above this threshold can be non-audible if they are masked by other components. Masking effects can be either in the time and/or in the frequency domain. In the case of frequency masking, tones

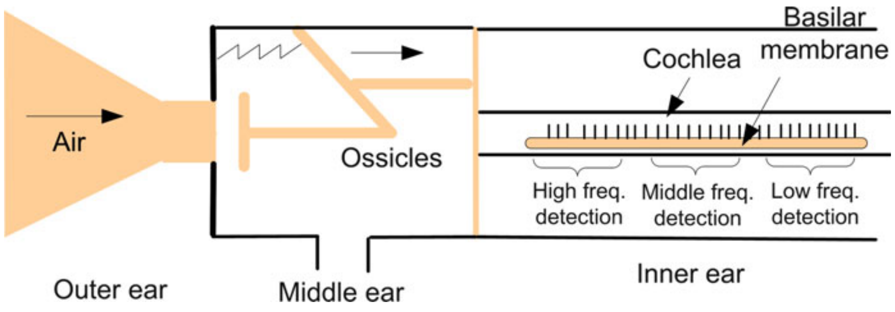


Fig. 2.24 Illustration of human auditory system

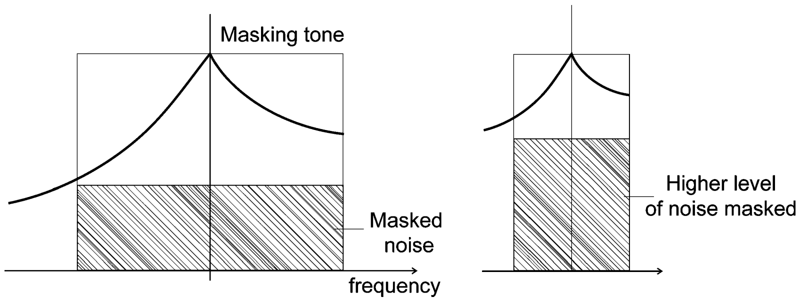


Fig. 2.25 Masking noise

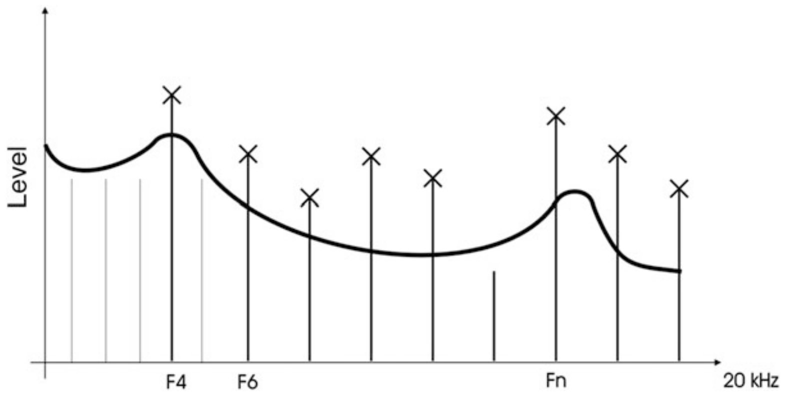


Fig. 2.26 An illustration of audio masking

with greater intensity can mask lower intensity tones at neighboring frequencies. Therefore, if we know a value of the threshold below which the adjacent frequencies become non-audible, then we can ignore those frequencies without sacrificing the quality of the sound, as shown in Fig. 2.25. This is particularly important when applied to each of the critical frequency bands, where we can say that the ear is equally sensitive. The sensitivity is different for different critical bands.

It should be mentioned that the width of the critical frequency bands varies from a few hundred Hz at lower frequencies to several KHz at higher frequencies. An overview of the 25 experimentally determined critical bands will be given in the following section.

A masking curve is illustrated in Fig. 2.26. Note that the samples below the masking curve are dismissed and only the samples that are not masked are considered for encoding and transmission. In addition to frequency masking, we can use time masking where the threshold is defined as a function of time. For example, let us assume that we have a signal with a dominant frequency f at time t. Then, it is possible to determine the masking threshold for the interval $(t, t + \Delta t)$ for which the signal becomes non-audible at the given frequency f or adjacent frequencies.

2.8 Audio Compression

Based on the aforementioned characteristics of the audio signal, we can conclude that storing high quality digital audio signals requires a large memory space. Therefore, the transmission of such signals also requires a network with large bandwidth. The reduction of the required bandwidth and memory space, while maintaining high audio quality, can be achieved by compression algorithms. Recent advances in computer technology have prompted significant improvements in compression algorithms. Also, there is a growing need to transfer large amount of data over the network. Hence, the compression algorithms have a significant economic impact related to various storage media or better utilization of network connections.

Data compression is performed by a circuit called the encoder. After transmission over a communication channel, the data are restored back into its basic form by using decoders. The encoder is generally much more complex and expensive than the decoder. However, a single encoder can be used to provide data to a large number of decoders.

A compression ratio is the ratio of the compressed signal size versus the original signal size. This ratio is often referred to as a coding gain. The compression is especially important in the Internet-based communications and applications. The need for efficient compression algorithms is also growing in radio broadcasting, as we are trying to use the available bandwidth more efficiently.

2.8.1 Lossless Compressions

Compression, in general, can be divided into lossless and lossy compression. In lossless compression, the information before and after compression must be identical. To achieve lossless compression, we use algorithms such as Huffman coding and LZW coding. Lossless compression algorithms have limited compression abilities. If the audio signal is compressed by using lossless compression techniques, then we refer to it as heavy, due to a low compression ratio.

Figure 2.27 illustrates the concept of entropy as the information content without redundancy. Namely, if we transmit the amount of information smaller than the information content or entropy, we actually introduce the artifacts. This is called lossy compression. Otherwise, the compression scheme is lossless when it is possible to recover the signal by uncompressing, i.e., the compressed signal has the same entropy as the original. We can conclude that the redundancy is actually a difference between the information rate and the overall bit rate. An ideal coder should provide the information bit rate defined by the entropy.

The relationship between the compression ratio and the complexity of the compression system is depicted in Fig. 2.28. In order to maintain the quality of signal under high compression ratio, we have to increase the complexity of the system.

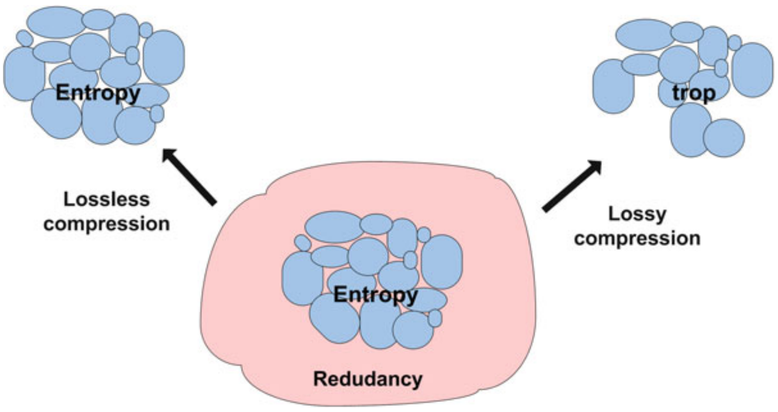


Fig. 2.27 Lossless and lossy compressions

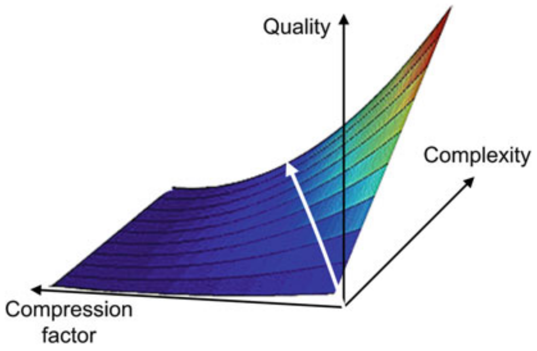


Fig. 2.28 Quality of a compressed signal depends on the complexity of the compression system and the compression ratio

2.8.1.1 LZ-77

LZ-77 algorithms achieve compression by replacing repeated occurrences of data with references to a single copy of those existing earlier in the input (uncompressed) data stream. It is especially important to determine the optimal length of the sequence that is encoded. A very short or very long sequence can cause negative effects on compression.

Pointers can be encoded with 12 bits such that the first 8 bits are used to denote the number of characters we have to go back, and the last 4 bits are used to denote the length of the sequence. In some cases, the pointers are encoded with 18 bits, where the first 12 bits determine the position, and the last 6 bits denote the length of the sequence. Encoding an entire sentence is performed by inserting 1 in front of an uncompressed part and 0 in front of a compressed part.

For the sake of simplicity, let us illustrate this compression principle on the text by using the following sentence:

she_sells_seashells_by_the_seashore

The letters *_se* (from the word seashells) are found in the word *_sells* and they are replaced by the pointer (6,3) meaning that we go back six characters and take the following three characters. The sequence *she* (from seashells) is found in the word *she* and can be replaced by a pointer (13,3), meaning that we go back 13 characters and take the next 3 characters. The procedure continues until we reach the end of the sentence, which we are encoding. The sentence can be then encoded as follows:

she_sells_<6, 3>a<13, 3><10, 4>by_t<23, 5><17, 3>ore

Since the pointers are encoded with 12 bits, in this short example we can reduce the amount of information by 76 bits (out of 280).

2.8.1.2 LZW Coding

LZW coding is a generalization of the LZ-77 coding, and it is based on defining a code book (dictionary) of words and strings found in the text. Strings are placed in the dictionary. Since the first 255 entries found in the dictionary are assigned to single characters, the first available index in the dictionary is actually 256. The dictionary is formed by initially indexing any two-character string found in the message. Then, we continue with three-character string, and so on. For example, let us consider the previous example:

she_sells_seashells_by_the_seashore

256 ->sh	**<sh>e_sells_ seashells_by_the_seashore**
257 ->he	**s<he>_sells_ seashells_by_the_seashore**
258 ->e_	**sh<e_ >sells_ seashells_by_the_seashore**
259 ->_s	**she< _s >ells_ seashells_by_the_seashore**
260 ->se	**she_ < se >lls_ seashells_by_the_seashore**
261 ->el	**she_s <el>ls_ seashells_by_the_seashore**
262 ->ll	**she_se <ll>s_ seashells_by_the_seashore**
263 ->ls	**she_sel<ls>_ seashells_by_the_seashore**
264 ->s_	**she_sell<s_> seashells_by_the_seashore**

The next two characters are "_s", but they already exist in the dictionary under the number 259. This means that we can now place the three characters "_se" as a new entry in the dictionary and then continue with the strings of two characters:

265 ->_se	**she_sells<_se>ashells_by_the_seashore**
266 ->ea	**she_sells_s<ea>shells_by_the_seashore**
267 ->as	**she_sells_se<as>hells_by_the_seashore**

The next two characters “sh” are already indexed in the dictionary under 256. Therefore, we add a new three-character string “she”:

268 ->she she_sells_sea<she>lls_by_the_seashore

The string “el” is already in the dictionary with the label (261), and therefore we add “ell":

269 ->ell she_sells_seash<ell>s_by_the_seashore

The string “ls” is already in the dictionary with the label (263), and we add “ls_”, and then continue with the string of two characters:

270 ->ls_	**she_sells_seashel<ls_>by_the_seashore**
271 ->_b	**she_sells_seashells<_b>y_the_seashore**
272 ->by	**she_sells_seashells_<by>_the_seashore**
273 ->y_	**she_sells_seashells_b<y_>the_seashore**
274 ->_t	**she_sells_seashells_by<_t>he_seashore**
275 ->th	**she_sells_seashells_by_<th>e_seashore**

As the string “he” is already in the dictionary with the label (257), “he_” is added:

276 ->he_ she_sells_seashells_by_t<he_>seashore

String “_s” is already in the dictionary with the label (259), as well as the string “_se” with the label (265). Thus, we add a new string with four characters “_sea”:

277 ->_sea	**she_sells_seashells_by_the<_sea>shore**
278 ->ash	**she_sells_seashells_by_the_se<ash>ore**
279 ->ho	**she_sells_seashells_by_the_seas<ho>re**
280 ->or	**she_sells_seashells_by_the_seash<or>e**
281 ->re	**she_sells_seashells_by_the_seasho<re>**

Finally, the dictionary will contain the following strings:

256 ->sh	**269 ->ell**
257 ->he	**270 ->ls_**
258 ->e_	**271 ->_b**
259 ->_s	**272 ->by**
260 ->se	**273 ->y_**
261 ->el	**274 ->_t**
262 ->ll	**275 ->th**
263 ->ls	**276 ->he_**
264 ->s_	**277 ->_sea**
265 ->_se	**278 ->ash**
266 ->ea	**279 ->ho**
267 ->as	**280 ->or**
268 ->she	**281 ->re**

In parallel to forming the dictionary, the encoder continuously transmits characters until it encounters the string that is in the dictionary. Then, instead of sending the string, the index from the dictionary is sent. This process is repeated until the whole message is transmitted. It means that the compressed messages in our case will be:

she_sells<259>ea<256><261><263>_by_t<257><265><267>hore

Note that it is not necessary to send to the decoder the dictionary created by the encoder. While reading and decoding the message, the decoder creates the dictionary in the same way as the encoder.

Let us consider another example:

Thomas_threw_three_free_throws

256 ->th	**<Th>omas_threw_three_free_throws**
257 ->ho	**T<ho>mas_threw_three_free_throws**
258 ->om	**Th<om>as_threw_three_free_throws**
259 ->ma	**Tho<ma>s_threw_three_free_throws**
260 ->as	**Thom<as>_threw_three_free_throws**
261 ->s_	**Thoma<s_>threw_three_free_throws**
262 ->_t	**Thomas<_t>hrew_three_free_throws**
263 ->thr	**Thomas_<thr>ew_three_free_throws**
264 ->re	**Thomas_th<re>w_three_free_throws**
265 ->ew	**Thomas_thr<ew>_three_free_throws**
266 ->w_	**Thomas_thre<w_>three_free_throws**
267 ->_th	**Thomas_threw<_th>ree_free_throws**
268 ->hr	**Thomas_threw_t<hr>ee_free_throws**
269 ->ree	**Thomas_threw_th<ree>_free_throws**
270 ->e_	**Thomas_threw_thre<e_>free_throws**
271 ->_f	**Thomas_threw_three<_f>ree_throws**
272 ->fr	**Thomas_threw_three_<fr>ee_throws**
273 ->ree_	**Thomas_threw_three_f<ree_>throws**
274 ->_thr	**Thomas_threw_three_free<_thr>ows**
275 ->ro	**Thomas_threw_three_free_th<ro>ws**
276 ->ow	**Thomas_threw_three_free_thr<ow>s**
277 ->ws	**Thomas_threw_three_free_thro<ws>**

The dictionary is formed as follows:

256 ->th	**267 ->_th**
257 ->ho	**268 ->hr**
258 ->om	**269 ->ree**
259 ->ma	**270 ->e_**
260 ->as	**271 ->_f**
261 ->s_	**272 ->fr**
262 ->_t	**273 ->ree_**
263 ->thr	**274 ->_thr**
264 ->re	**275 ->ro**
265 ->ew	**276 ->ow**
266 ->w_	**277 ->w**

while the coded message is:

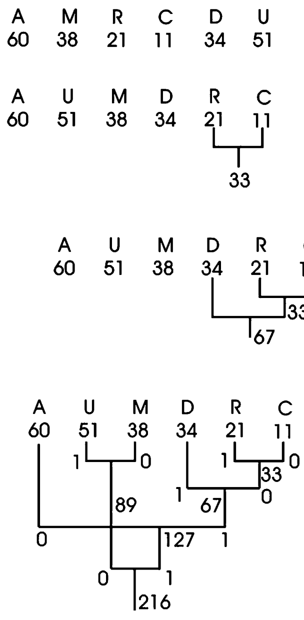
Thomas_<256>rew<262>h<264>e_f<269><267>rows

2.8.1.3 Huffman Coding

The idea behind the Huffman coding is to encode each character with a code word whose length is inversely proportional to the probability of occurrence of that character. In other words, if a character appears more frequently, it should be encoded with the shortest possible code word.

The characters are first sorted according to the number of occurrences (NO). Then, we observe a pair of characters with the lowest NO. If the logical value of "1" is assigned to the character with a higher NO, then "0" is assigned to the character with a lower NO. The cumulative NO for the two characters is calculated and it replaces this pair in the next iterations. The next character is used in the new iteration and its NO is compared with the smaller one, between NO for another character and cumulative NO from the previous iteration. Again, "1" is assigned to the higher NO, while "0" is assigned to lower NO. The procedure is repeated until we get the entire tree. Each branch within the tree corresponds to one character, and it is uniquely determined by the resulting sequence of logical values "1" and "0."

Consider an example with the following characters A, M, R, C, D, and U. Assume that the NOs of characters in a text are: A = 60, M = 38, R = 21, C = 11, D = 34, U = 51. For Huffman coding we will form the following tree:



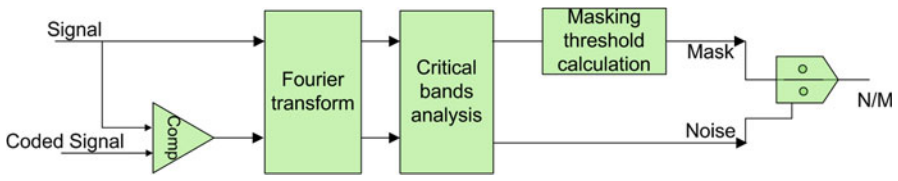


Fig. 2.29 The system for measuring the noise/masking ratio

Therefore, the binary combinations denoting each of the characters are given as:

A	*U*	*M*	*D*	*R*	*C*
10	01	00	111	1101	1100

2.8.2 Lossy Compressions

The idea of lossy compression is based on the perceptual characteristics. Namely, the information that is least important, from a perceptual point of view, is omitted. For lossy compressions we utilize our understanding of psychoacoustics (e.g., the auditory system responds differently to different frequencies and some sounds may be masked by the others). Therefore, this coding method is often referred to as the perceptual coding. MPEG (Moving Picture Experts Group) compression algorithms represent the important and widely used cases of lossy compression.

The amount of compressed data depends on the signal nature (i.e., the encoding may have a variable compression factor) that causes variable bit rate through the channel. In practice, it is often required that coders have a constant compression factor in order to transmit at a constant rate.

In order to use the perceptual coding, it is important to adjust and calibrate correctly the microphone gain and the volume control of the reproduction system. The overall gains should be adjusted to the human hearing system such that the coder uses the SPL, which is actually heard. Otherwise, we might have a situation that the low gain from the microphone is interpreted as low SPL, which further causes inappropriate masking of the coded signal. Thus, the compression systems must include the calibration model based on human hearing system. In addition to calibration, an important role in perceptual coding has a masking model. The accuracy of the model used for the separation of relevant and irrelevant components is of particular importance. Based on this model, we decide to ignore a certain amount of information that will not affect the signal quality. The most reliable approach for assessing the quality of the masking is listening. Such methods are usually expensive to carry out. Therefore, systems have been developed to measure the quality of sound masking. A system based on noise measurements is shown in Fig. 2.29.

Table 2.1 Critical bands

Subband	F_l	F_c	F_u	Bandwidth (Hz)
1	0	50	100	100
2	100	150	200	100
3	200	250	300	100
4	300	350	400	100
5	400	450	510	110
6	510	570	630	120
7	630	700	770	140
8	770	840	920	150
9	920	1,000	1,080	160
10	1,080	1,170	1,270	190
11	1,270	1,370	1,480	210
12	1,480	1,600	1,720	240
13	1,720	1,850	2,000	280
14	2,000	2,150	2,320	320
15	2,320	2,500	2,700	380
16	2,700	2,900	3,150	450
17	3,150	3,400	3,700	550
18	3,700	4,000	4,400	700
19	4,400	4,800	5,300	900
20	5,300	5,800	6,400	1,100
21	6,400	7,000	7,700	1,300
22	7,700	8,500	9,500	1,800
23	9,500	10,500	12,000	2,500
24	12,000	13,500	15,500	3,500
25	15,500	18,775	22,050	6,550

The system compares the original and coded signals and determines the noise introduced by encoder. The lower branch performs the noise analysis and provides the critical band spectrum of the noise. The blocks in the upper branch of the system are used to calculate the masking threshold of the input signal. The ratio noise/masking (N/M) is obtained at the output of the observed system (Fig. 2.29). This ratio is a quality measure of masking. Smaller values denote more accurate masking models.

2.8.2.1 Critical Subbands and Perceptual Coding

The spectrum of the audio signal can be divided into the subbands (critical bands) within which is assumed that the human hearing system has equal sensitivity for all frequencies. Table 2.1 provides the lower (F_l) and upper (F_u) limit frequencies, the center frequency (F_c), and the bandwidth for each critical band.

Thus, the auditory system can be approximately modeled through a filter bank. However, implementing selected critical bands would be a demanding task.

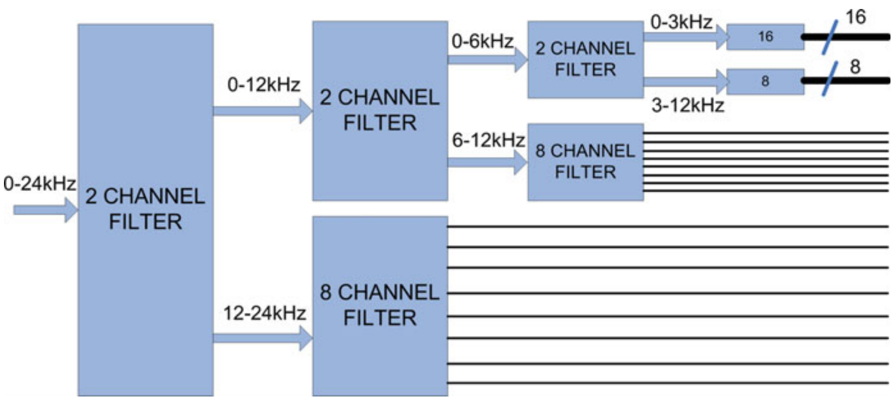


Fig. 2.30 Dividing the spectrum into critical bands by using a filter bank

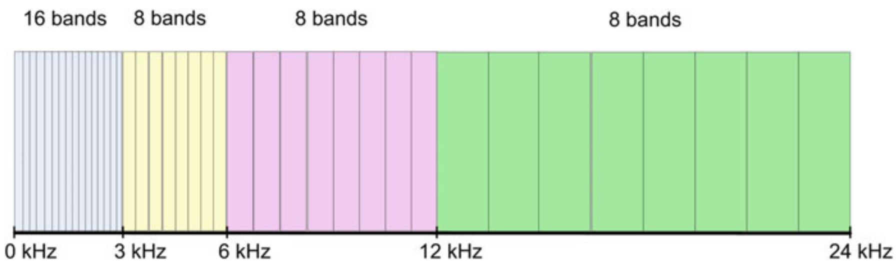


Fig. 2.31 Illustration of the critical bands

Hence, we can obtain a simpler system with some approximations as shown in Fig. 2.30.

At each filtering stage, the signal bandwidth is halved, allowing us to decrease the sampling frequency[1] by 2. The high-frequency part of the spectrum is obtained as a difference between the input and the filtered spectrum (low-frequency part). In this way, a ladder scheme of the spectrum partition into critical bands is obtained. The frequency subbands are illustrated in Fig. 2.31.

The scale used to number these critical bands is known as the Bark scale named after the German scientist Barkhausen. The scale depends on the frequencies (expressed in Hz) and can be approximately given by:

$$B\,(\text{Bark}) = \begin{cases} \dfrac{f}{100} \text{ for } f < 500 \text{ Hz}, \\ 9 + 4\log_2\left(\dfrac{f}{1,000}\right) \text{ for } f \geq 500 \text{ Hz}, \end{cases} \tag{2.39}$$

[1] For the signal with spectrum bandwidth B, the sampling frequency is $f_s = 2B$ if $(2f_c + B)/2B$ is an integer (f_c is the central frequency).

where B is the number of the critical band. It is often used by the following approximate relation:

$$B(\text{Bark}) = 13\arctan(0.76(f(\text{Hz})/1{,}000)) + 3.5\arctan\left((f(\text{Hz})/7{,}500)^2\right)$$

For example, the frequency of 200 Hz can be represented by 2 from the Bark scale, while the frequency of 2,000 Hz can be represented by 13 from the Bark scale.

To obtain the frequency in Hz from the Bark scale, we can use the following relationship:

$$f(\text{Hz}) = 1{,}000\left\{((\exp(0.219\cdot B)/352) + 0.1)\cdot B - 0.032\cdot\exp\left(-0.15\cdot(B-5)^2\right)\right\}$$

Figure 2.32 shows the masking effects versus the frequency expressed in KHz and the Bark scale. In both cases, the dotted line shows the curve representing a hearing threshold in quiet. Figure 2.32a also depicts masking curves for samples at frequencies 1, 4, and 8 KHz, respectively. Similarly, Fig. 2.32b shows the masking curve for different ranges on the Bark scale.

Consider the following example. Amplitude levels in certain frequency bands are provided in the Table 2.2.

Note that the amplitude level in the 12th band is 43 dB. Suppose that it masks all components below 15 dB in the 11th band and the components below 17 dB in the 13th band.

- The signal level in the 11th band is 25 dB (>15 dB) and this band should be encoded for transmission. However, the quantization noise of 12 dB will be masked, and therefore, we can use 2 bits less to represent the samples in this band.
- The signal level in the 13th band is 14 dB (<17 dB). Hence, the components in the 13th band are masked and there is no need to transmit this band.

2.8.3 MPEG Compression

In 1988, the ISO (International Standards Organization) and IEC (International Commission Electrotechnical) have begun to establish international standards for audio compression. As a result, they established guidelines for MPEG audio compression, which is currently used for the audio coding in digital audio broadcasting (DAB). Algorithms for MPEG audio compression were derived from Masking-pattern Universal Subband Integrated Coding And Multiplexing (MUSICAM) algorithm. A block diagram for an audio compression coder based on the MUSICAM is shown in Fig. 2.33.

MUSICAM compresses audio data such that the optimal bit rate is approximately 700 Kb/s. In parallel to the MUSICAM, a compression algorithm known as

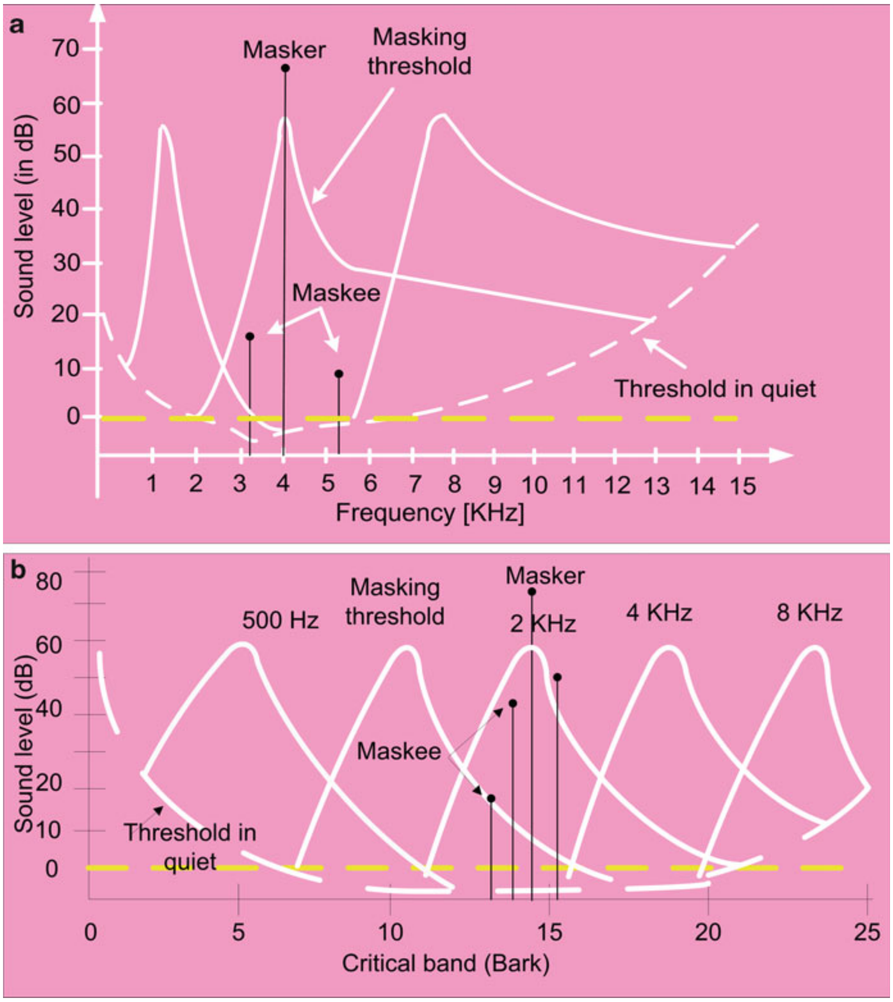


Fig. 2.32 Illustration of the effects of masking tones: (**a**) Masking in frequency, (**b**) Masking of the bandwidth range

Table 2.2 An example of amplitude levels in different frequency bands

Band	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Level (dB)	12	5	3	1	2	12	8	28	19	10	25	43	14	2	6	35

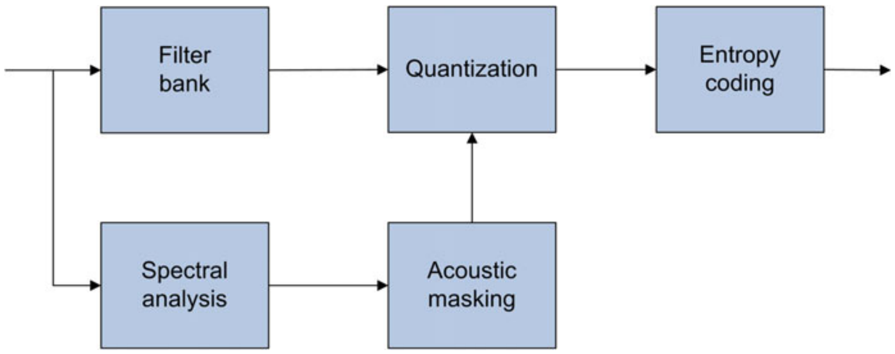


Fig. 2.33 Block diagram of MUSICAM-based coder

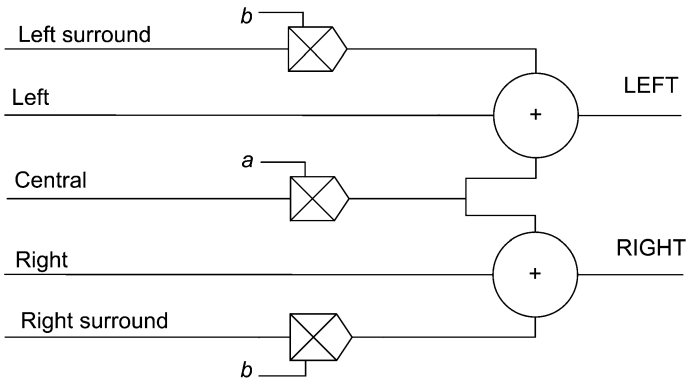


Fig. 2.34 A scheme for converting from five channels into two channels

Adaptive Spectral Perceptual Entropy Coding (ASPEC) was developed. Its main goal was to achieve high compression factors in order to facilitate transmission of audio signals over the ISDN lines. By combining MUSICAM and ASPEC, MP3 (MPEG layer III) algorithm was created. In other words, while the MPEG layer I and layer II represent simplified versions of MUSICAM, MP3 combines the best features of MUSICAM and ASPEC. The layers of MPEG audio coding deal with signals having maximal frequencies: 16, 22.05, and 24 KHz and supports bit rates of 32, 48, 56, 64, 96, 112, 128, 192, 256, and 384 Kb/s. MPEG layer I is based on two channels (i.e., a stereo signal), while MPEG layer II can handle a five-channel audio signal. MPEG layer II can also convert a five-channel signal into a two-channel signal, and such a system is illustrated in Fig. 2.34.

The compression algorithm known as AC-3, developed by Dolby Laboratories, is also used in North America. At the beginning, the AC-3 was developed as a compression scheme that provides the surround sound for the theater and cinema. Nowadays, it is usually referred as Dolby Digital and can be found in the HDTV, home theaters, DVD players, some TV receivers, etc.

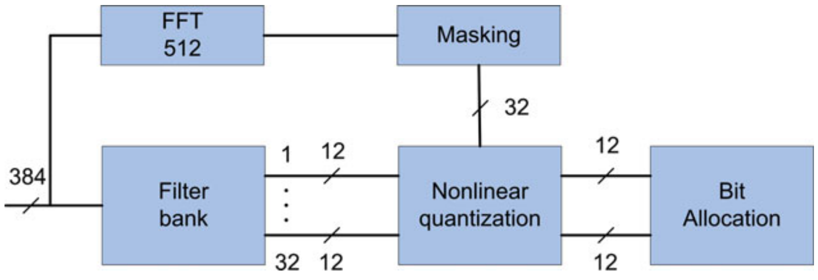


Fig. 2.35 A block diagram of MPEG layer I compression

2.8.3.1 MPEG Layer I

As already noted, the MPEG layer I is a simplified version of the MUSICAM algorithm. According to the MPEG layer I algorithm, an audio signal is divided into 32 subbands. All 32 subbands are of the same width, which is one of the drawbacks of this compression scheme, since the bandwidths of the critical bands are frequency dependent. Thus, subbands can be either too wide at lower frequencies or too narrow at higher frequencies. In order to compensate the imprecision caused by the uniform subbands width, audio masking is used. The Fourier transform has an important role in the audio masking (it is computed by the FFT algorithm). A block scheme for MPEG layer I compression is given in Fig. 2.35.

The signal compression is carried out in blocks of 384 samples. After coding, we obtain 32 blocks with 12 samples corresponding to the width of 8 ms at the sampling frequency of 48 KHz. The FFT is calculated for 512 points in order to obtain higher resolution. This provides a more accurate model of masking. The data in each block are encoded according to the maximum signal value in that block. A 6-bit scale factor is assigned to each block and it is applied to all the 12 block samples. The gain step between two successive 6-bit combinations is 2 dB, thus providing a 128 dB of dynamic range. Having in mind the nature of audio signals, the number of bits reserved for samples will vary for the 32 different blocks, but the total length of 32 blocks has to be equal for each coded block (the size of the output block with 384 samples is fixed for a certain bit rate).

The bit allocation is used to determine the structure of binary code words for the appropriate subband. Namely, four bits are used to describe the samples code length. The length can vary between 1 and 15 bits (i.e., from the combination 0000–1110). The combination 0000 denotes that each of the 12 samples within the block can be encoded with one bit, while 1110 denotes that we need 15 bits for each sample in the block. The combination 1111 is not used in order to avoid possible conflict with the synchronization code. There is also a special code if all samples in the block are equal to zero. Hence, for each block it is necessary to send 4 allocation bits and 6 bits that define the amplification factor (Fig. 2.36).

Note that the block length of 8 ms is quite long to avoid premasking effects that may appear due to the abrupt changes in signal followed by silence at the transition between two blocks. This phenomenon can be avoided by comparing the values

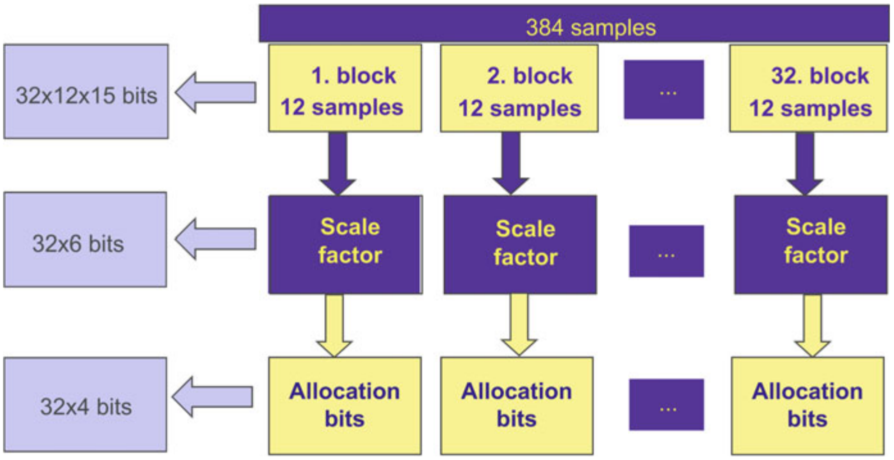


Fig. 2.36 A part of the bits packing structure in MPEG layer I

in the neighboring blocks. A significant difference between consecutive blocks indicates transient in the signal. A typical value of the compression factor in MPEG layer I is 1:4, and the bit rate is 384 Mb/s.

2.8.3.2 MPEG Layer II

The MPEG layer II is an improved version of the MPEG layer I algorithm, which almost completely utilizes the MUSICAM algorithm. The scheme with 32 blocks is also used for this compression. However, the total frequency range is divided into three parts: low, medium, and high (Fig. 2.37). Given the different sensitivities of the auditory system to these three parts, the number of bits used for encoding will be different in each part. Namely, the low-frequency range uses up to 15 bits, the mid-frequency range uses up to 7 bits, and the high-frequency range uses up to 3 bits. In addition, 4 bits are needed for bit allocation in the low-frequency band, while the middle and high-frequency ranges use 3 and 2 allocation bits, respectively. The input blocks contain 1,152 samples, and since they split into three new blocks, each of them will contain 384 samples. In such a way, we get a structure that corresponds to the previously described code scheme for the MPEG layer I. The masking procedure is done by using the FFT algorithm with 1,024 samples. The compression ratio of the MPEG layer II is approximately equal to 6–8 times.

2.8.3.3 MPEG Layer III (MP3)

Unlike the previous two compression algorithms, MP3 is based on ASPEC and MUSICAM. Namely, compression is carried out using samples in the transform domain, and the structure of the blocks resembles the previous algorithms. MP3 uses the blocks containing 1,152 samples divided into 32 subbands. The transformation

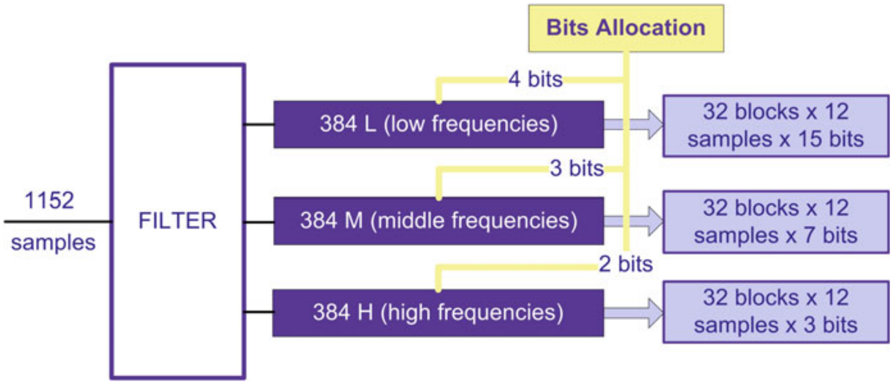

Bits Allocation
4 bits
3 bits
2 bits
1152 samples
FILTER
384 L (low frequencies)
384 M (middle frequencies)
384 H (high frequencies)
32 blocks x 12 samples x 15 bits
32 blocks x 12 samples x 7 bits
32 blocks x 12 samples x 3 bits

Fig. 2.37 Dividing the frequency range in MPEG layer II

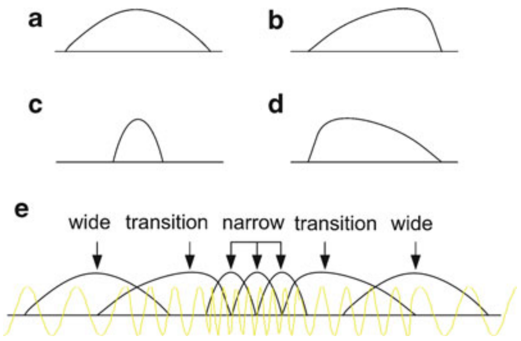


Fig. 2.38 Windows used by MP3 algorithm: (**a**) Wide window, (**b**) Narrow window, (**c**) and (**d**) Transition windows, (**e**) Shifting from wide to narrow window and vice versa

from the time to the frequency domain is performed using the modified discrete cosine transform (MDCT). It is important to note that the MP3 algorithm does not use the fixed-length windows, but they are either 24 ms or 8 ms long. The windows of short duration are used when there are sudden signal changes, since shorter windows ensure a good time resolution. Wider windows are used for slowly varying signals. Figure 2.38 shows window forms used in the MP3 compression.

The algorithm based on variable window widths provides a better quality of the compressed signals. However, it should be noted that a choice of an appropriate window function depends on a more complex psychoacoustical model than the models used in the MPEG layer I and layer II algorithms. Namely, the complexity of the psychoacoustical model is increased due to the use of the MDCT. A block diagram of the MP3 compression is shown in Fig. 2.39.

It should be mentioned that the MP3 coding also uses blocks for entropy coding based on Huffman code. The MP3 was developed primarily for Internet applications and provides high compression ratio (about 12 times) with a good quality of the reproduced signal.

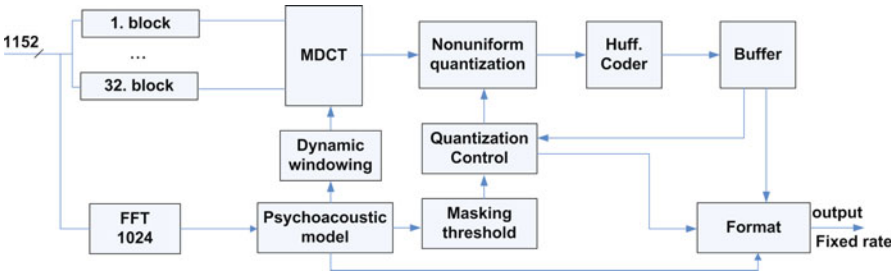


Fig. 2.39 A block diagram of the system for MP3 compression

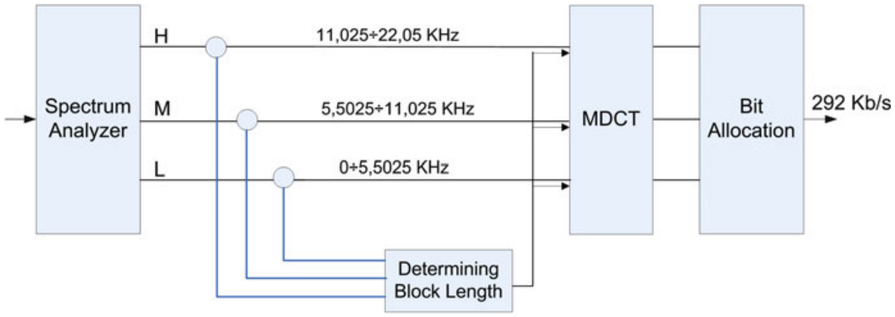


Fig. 2.40 A block scheme of ATRAC compression system

2.8.4 *ATRAC Compression*

The ATRAC compression algorithm is used for mini-discs in order to store the same amount of audio signals and with same quality as in the case of the CD, but on the significantly smaller disc area. ATRAC stands for Adaptive Transform Acoustic Coder. Using filters, the range of the input signal is divided into three subband (0–5.5, 5.5–11, and 11–22 KHz). Each subband is passed to the MDCT processors. The first subband has 20 blocks, while the other two contain 16 blocks each. Such a resolution corresponds to the sensitivity of the auditory system. The time slot for the analysis can vary from 1.45 to 11.6 ms by using the increments of 1.45 ms. In this way, the time-frequency plane of the signal is divided into a number of different areas, which enable successful compression, taking into account the difference in sensitivity of the auditory system in different parts of the time-frequency plane. The ATRAC compression reduces the bit rate from 1.4 Mb/s to 292 Kb/s. A block scheme of ATRAC compression system is shown in Fig. 2.40. Figure 2.41 demonstrates the division of the time-frequency plane as required by the ATRAC compression algorithm.

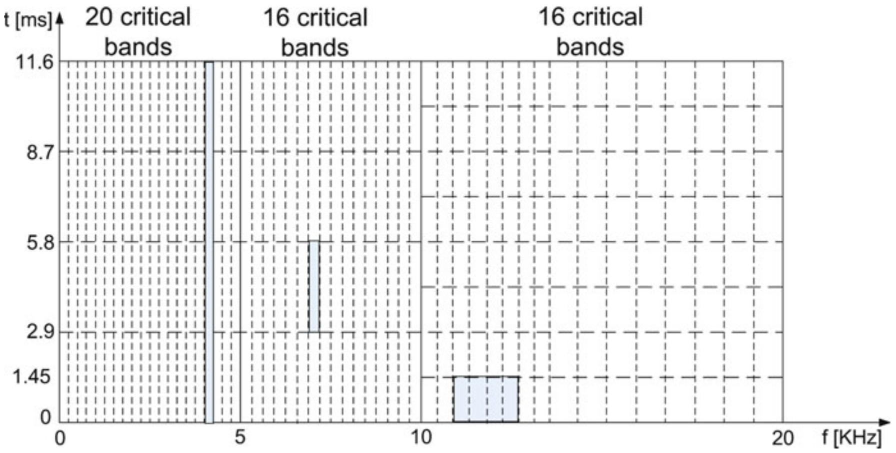


Fig. 2.41 Division of the time-frequency plane in the ATRAC algorithm

2.9 Examples

2.1. The sound pressure level for a signal is SPL $=$ 20 dB. If the reference level of pressure is $P_o = 20$ μPa, calculate the value of the pressure P in Pascals.

Solution:

$$\text{SPL} = 20\ \text{dB}$$
$$P_o = 20\mu\text{Pa}$$

$$\text{SPL} = 20 \cdot \log_{10}(P/P_o)$$
$$20 = 20 \cdot \log_{10}(P/P_o) \quad => \quad \log(P/P_o) = 1 \quad => \quad P/P_o = 10$$

$$P = P_o \cdot 10 = 20 \cdot 10^{-6} \cdot 10\ \text{Pa} = 2 \cdot 10^{-4}\text{Pa} = 0.2\ \text{mPa}$$

2.2. If the signal to quantization noise is $S/N = 61.76$ dB, determine the number of bits used for signal representation?

Solution:

$S/N = 1.76 + 6 \cdot n$ $\quad n$ - number of bits used to represent signal
$6 \cdot n = 60 =>$ $\quad n = 10$ bits

2.3. A 13-bit signal is obtained at the output of the floating-point converter, with the signal to noise ratio $S/N = 61.76$ dB. Determine the number of bits used to represent mantissa, and the number of bits used for exponent?

Solution:

$$\begin{aligned} &S/N = 61.76\text{ dB} \\ &6 \cdot n = 60 => \quad n = 10 \text{ bits for mantissa} \\ &\qquad\qquad\qquad\quad m = 13 - 10 = 3 \text{ bits for exponent} \end{aligned}$$

2.4. The communication channel consists of three sections. The average level of transmission power is 400 mW. The first section introduces 16 dB attenuation compared to the average power level, the second introduces 20 dB gain compared to the first section, while the third introduces attenuation of 10 dB compared to the second section. Determine the signal power at the output of each channel section.

Solution:

$$P_0 = 400\,\text{mW}$$

First section:

$$16\,\text{dB} = 10\log\left(\frac{P_0}{P_1}\right) = 10\log\left(\frac{400}{P_1}\right) \Rightarrow \text{P}_1 = 10.0475\text{ mW}$$

Second section:

$$20\,\text{dB} = 10\log\left(\frac{P_2}{P_1}\right) = 10\log\left(\frac{P_2}{10.0475}\right) \Rightarrow \text{P}_2 = 1004.75\text{ mW}$$

Third section:

$$10\,\text{dB} = 10\log\left(\frac{P_2}{P_3}\right) = 10\log\left(\frac{1004.75}{P_3}\right) \Rightarrow \text{P}_3 = 100.475\text{ mW}$$

2.5. Load the signal *speech_dft.wav* in Matlab. Make a new signal y that will contain 2 s of the original speech signal, and listen to the resulting signal.

Solution:

```
[y,fs]=wavread('speech_dft.wav');
length(y)
  ans = 110033
fs= 22050
y=y(1:2*fs);
soundsc(y,fs)
```

2.6. For a signal obtained in the previous example, design a low-pass filter in Matlab, with the cut-off frequency $f_c = 735$ Hz.

Solution:

The sampling frequency of the considered signal is 22,050 Hz. The total length of the signal is 44,100 samples. Hence, the Fourier transform will produce 44,100 samples

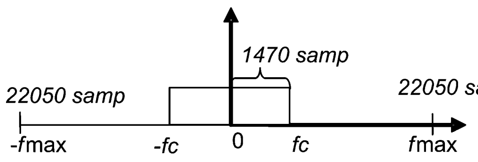


Fig. 2.42 Filter function

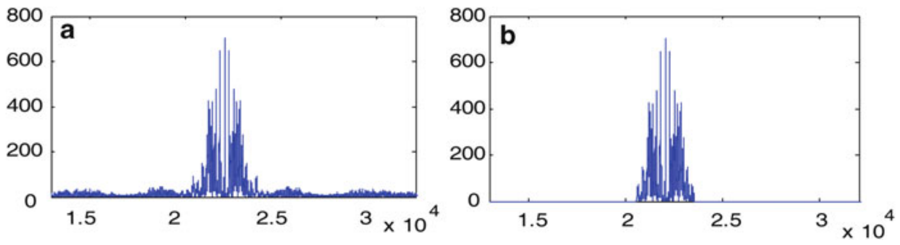


Fig. 2.43 (**a**) Fourier transform of the original signal, (**b**) Fourier transform of the filtered signal

in the frequency domain, from which 22,050 samples are related to positive and 22,050 to negative frequencies (Fig. 2.42).

$$f_{\max} = f_s/2 = 11,025 \text{ Hz};$$

In the frequency range between zero and the cut-off frequency $f_c = 735$ Hz, we have: 22,050·(735/11,025) = 1,470 samples

The filtering operation can be done by using Matlab as follows (Fig. 2.43):

```
F=fftshift(fft(y));  % Fourier transform of the signal
figure(1), plot((abs(F)))
% Filter transfer function
H=[zeros(1,20580) ones(1,2940) zeros(1,20580)];
G=F.*H';        % Signal filtering in the frequency domain
figure(2), plot(abs(G));
% The filtered signal is obtained by applying the inverse Fourier
   transform
yg=ifft(fftshift(G));
soundsc(real(yg),fs)
```

2.7. For the speech signal used in previous examples, design the band-pass filter with the band frequencies defined by 1,102.5 Hz and 2,205 Hz.

Solution:

The cutoff frequencies of the band-pass filter are: $fc_1 = 1{,}102.5$ Hz and $fc_2 = 2{,}205$ Hz, while the maximal signal frequency is $f_{\max} = 11{,}025$ Hz (Fig. 2.44).

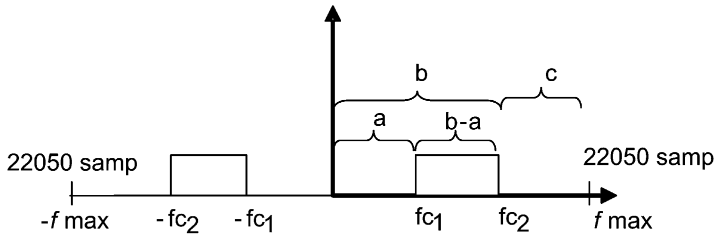


Fig. 2.44 Parameters of band-pass filter function

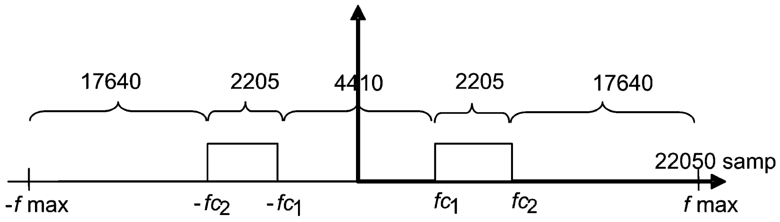


Fig. 2.45 Filter transfer function

Hence, we made the proportions as:

$$\begin{aligned} fc_1 : \mathrm{a} = f_{\max} : 22{,}050 \quad &\Rightarrow \mathrm{a} = 2{,}205 \\ fc_2 : \mathrm{b} = f_{\max} : 22{,}050 \quad &\Rightarrow \mathrm{b} = 4{,}410 \end{aligned}$$

The number of samples passing unchanged through the filter is $b - a = 2{,}205$. Note that the length between the cutoff frequency fc_2 and the maximal signal frequency $f_{\max}$ is:

$$C = 22{,}050 - \mathrm{b} = 22{,}050 - 4{,}410 = 17{,}640 \text{ samples.}$$

The filter transfer function in Matlab is given by (Fig. 2.45):

```
>>H=[zeros(1,17640) ones(1,2205) zeros(1,4410) ones(1,2205)
zeros(1,17640)];
```

Finally, we can perform signal filtering in the frequency domain by using the filter transfer function H:

```
>>G=F.* H';
```

The filtered signal is obtained by applying the inverse Fourier transform to the filtered signal spectrum:

```
>>yg=ifft(fftshift(G));
>>soundsc(real(yg),fs)
```

2.8. By using the speech signal "*speech_dft.wav*"in Matlab, realize the echo by using a 0.2 s delay, while the echo amplitude is decreased for 50 %. Listen to the achieved echo effect.

Solution:

Echo effect can be realized in a way that we make two versions of the original signal: one is obtained by adding a zero sequence at the beginning of the original signal, while the other is obtained by adding zeros at the end of the considered signal. The signal with echo effect is obtained as a sum of two modified signal versions.

The length of the zero sequence is defined by the delay which is equal to 0.2 s. Since the sampling frequency for the observed speech signal is 22,050 Hz, the delay 0.2 s corresponds to 4,410 samples. The echo realization in Matlab can be done as follows:

```
[y,fs]=wavread('speech_dft.wav');
y1=[zeros(1,4410) y'];
y2=[y' zeros(1,4410)];
echo=0.5*y1+y2;
soundsc(echo,fs)
```

2.9. By using the linear prediction coefficients given by vector *a*, and the set of 20 signal samples (vector *f*), determine the 14th signal sample and the prediction error.

$$a=[-1.7321\ 0.9472\ -0.3083\ 0.0748\ -0.0812\ 0.1260\ 0.2962\ -0.3123\ 0.0005\ldots$$
$$\ldots 0.0216\ -0.1595\ 0.2126\ -0.0496]$$

$$f=[-2{,}696\ -2{,}558\ -2{,}096\ -1{,}749\ -1{,}865\ -2{,}563\ -2{,}280\ -1{,}054\ -635\ -41\ldots$$
$$\ldots 1{,}695\ 3{,}645\ 5{,}150\ 6{,}188\ 5{,}930\ 4{,}730\ 3{,}704\ 3{,}039\ 2{,}265\ 1{,}159]$$

Solution:

Based on the linear prediction analysis, the estimated value of 14th sample is calculated according to:

$$\widehat{f}(n) = -\sum_{i=1}^{L} a_i f(n-i)$$

For $n = 14, L = 13$, we have: $\widehat{f}(14) = -\sum_{i=1}^{13} a_i f(14-i) = 6{,}064.$

The prediction error is: $e(14) = f(14) - \widehat{f}(14) = 6{,}188 - 6{,}064 = 124.$

2.10. For a given set *f* of signal samples and the corresponding prediction errors (given by vector *e*) calculated as in the previous example, determine the value of prediction gain.

$$e = 10^3 \cdot [-0.0095 \quad -0.7917 \quad -1.1271 \quad -0.3273 \quad 0.0907 \quad -0.1379$$
$$-0.1106 \quad 0.1444 \quad -0.1762 \quad 0.5057]$$

$$f = [6,188 \quad 5,930 \quad 4,730 \quad 3,704 \quad 3,039 \quad 2,265 \quad 1,159 \quad 168 \quad -434 \quad 120]$$

Solution:

The prediction gain for the observed set of samples given in f can be calculated as:

$$\text{PG} = 10\log_{10}\left(\frac{\sum_{k=1}^{10} f^2(k)}{\sum_{k=1}^{10} e^2(k)}\right) = 17.27\text{dB}.$$

2.11. For a set of 10 samples (given below), calculate the value of energy-entropy feature EEF.

$$f = [-437 \quad -97 \quad -3 \quad -163 \quad 182 \quad 143 \quad 225 \quad -242 \quad -262].$$

Solution:

First, we calculate the energy E of the frame:

$$E = \sum_{k=1}^{10} f_k{}^2 = 269,291.$$

The Fourier transform coefficients of the signal f are:

$$F(\omega) = [-2.5300 \quad -5.8848 + 1.0069i \quad 1.4868 - 7.6610i \quad 3.0148 - 0.1360i$$
$$3.2532 + 0.1677i \quad \ldots -5.5100 \quad 3.2532 - 0.1677i \quad 3.0148 + 0.1360i$$
$$1.4868 + 7.6610i \quad -5.8848 - 1.0069i]$$

The probability density function is calculated as:

$$p = F(\omega) / \sum_{k=1}^{10} F(\omega),$$

and the corresponding vector p is obtained:

$$p = [0.0526 \;\; 0.1240 \;\; 0.1621 \;\; 0.0627 \;\; 0.0677 \;\; 0.1145 \;\; 0.0677 \;\; 0.0627 \;\; 0.1621 \;\; 0.1240].$$

The entropy of the observed frame is calculated as:

$$H = \sum_{k=1}^{10} p_k \log p_k = -0.9651.$$

Finally, the energy-entropy feature can be calculated as:

$$\text{EEF} = (1 + |E \cdot H|)^{1/2} = 509.8067.$$

2.12. Write the Matlab code for the word endpoints detector based on the energy-entropy feature.

Solution:

```
%% load test speech signal in vector f
k=1;
   for i=1:64: round(length(f)/64)*64
     E(k)=sum(f(i:i+63).^2);
     X=fft(f(i:i+63));
     p=(abs(X)./sum(abs(X)));
     H(k)=sum(p.*log10(p));
     EEF(k)=sqrt(1+abs(E(k).*H(k)));
     k=k+1;
   end
   for i=0:length(EEF)-1
   s(1+i*64:i*64+64)=EEF(i+1);
   end
figure(1),plot(real(s)./max(real(s)))
```

2.13. In this example, a short Matlab code for the time-frequency based eigenvalue decomposition is provided. We assume that the S-method is calculated in advance (Chap. 1).

Solution:

```
%Sm is the S-method matrix
%% Calculation of the auto-correlation matrix
R=zeros(N+1);
   for n=1:N+1;
     v=N+n;
     k=n;
     for m=1:N+1;
       R(n,m)=Sm(v,k);
       v=v-1;k=k+1;
     end
   end
```

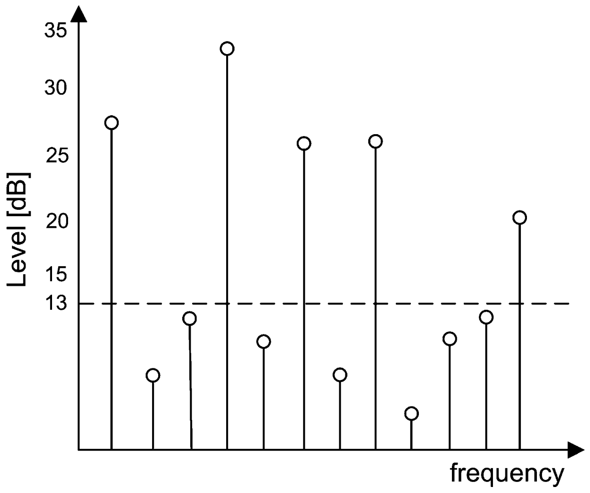


Fig. 2.46 Subband samples and masking level

```
% Eigenvalues matrix D and eigenvectors V
[V,D]=eigs(R,Nc,'lm',opt); %columns of V are eigenvectors
D=abs(diag(D));
```

2.14. For the given subband samples, determine the number of bits that will be transmitted, if we know that the samples below 13 dB are masked by the neighboring subband (as shown in Fig. 2.46). Assume that the signal samples are originally represented by 8 bits.

Solution:

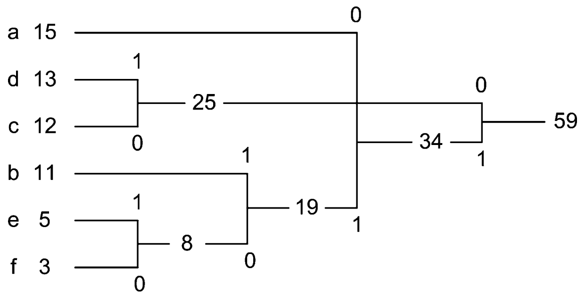
Due to the audio masking effects, only the samples that are above the masking level will be transmitted. Due to the masking of tones below 13 dB, the quantization noise of 12 dB is masked as well. Therefore, we use two bits less to represent the samples, and the total number of transmitted bits is:

$$5\text{ samples} \cdot (8-2)\text{b} = 30\text{ b}$$

2.15. Perform the Huffman coding algorithm, for the symbols whose numbers of occurrences within a certain sequence are given below.

$$\text{Number of occurrences}: \quad \begin{array}{l} a \rightarrow 15 \\ b \rightarrow 11 \\ c \rightarrow 12 \\ d \rightarrow 13 \\ e \rightarrow 5 \\ f \rightarrow 3 \end{array}$$

Fig. 2.47 An example of Huffman coding



Solution:

In order to perform Huffman coding, the numbers of occurrences for symbols a, b, c, d, e, and f are first sorted in decreasing order. Then the coding is performed according to the scheme in Fig. 2.47:

Thus, the symbols are coded as follows:

$$a \rightarrow 10 \quad d \rightarrow 01 \quad c \rightarrow 00 \quad b \rightarrow 111 \quad e \rightarrow 1101 \quad f \rightarrow 1100$$

2.16. Consider the sequence *this_image_is_damaged*. Code the sequence by using the LZ-77 code. Determine the number of bits that can be saved by applying this coding algorithm. Assume that the pointers are represented by 12 bits.

Solution:

The sequences can be coded as follows:

this_image_is_damaged
*this_image_*(9,3)*da*(10,4)*d*

(9,3) → 00001001 0011
(10,4) → 00001010 0100
Before LZ77: 21·8 bits=168 bits (21 characters including spaces)
After LZ 77: 14·8b+24b= 136 b
168 − 136=32 (19%)

2.17. Perform the LZW coding of the sequence: *strange strategic statistics*.

Solution:

strange_strategic_statistics

256 ->st	**<st>range_strategic_statistics**
257 ->tr	**s<tr>ange_strategic_statistics**
258 ->ra	**st<ra>nge_strategic_statistics**
259 ->an	**str<an>ge_strategic_statistics**
260 ->ng	**stra<ng>e_strategic_statistics**
261 ->ge	**stran<ge>_strategic_statistics**
262 ->e_	**strang<e_>strategic_statistics**

263 ->_s	**strange<_s>trategic_statistics**
264 ->str	**strange_<str>ategic_statistics**
265 ->rat	**strange_st<rat>egic_statistics**
266 ->te	**strange_stra<te>gic_statistics**
267 ->eg	**strange_strat<eg>ic_statistics**
268 ->gi	**strange_strate<gi>c_statistics**
269 ->ic	**strange_strateg<ic>_statistics**
270 ->c_	**strange_strategi<c_>statistics**
271 ->_st	**strange_strategic<_st>atistics**
272 ->ta	**strange_strategic_s<ta>tistics**
273 ->at	**strange_strategic_st<at>istics**
274 ->ti	**strange_strategic_sta<ti>stics**
275 ->is	**strange_strategic_stat<is>tics**
276 ->sti	**strange_strategic_stati<sti>cs**
277 ->ics	**strange_strategic_statist<ics>**

Coded sequence:

strange_<256><258>tegic<263> tati<256><269>s

2.18. Determine the bit rate (in Kb/s) for the following cases:

(a) Speech signal with the maximal frequency 10 KHz, while the samples are coded by using 12 b/sample;
(b) Musical signal with the maximal frequency 20 KHz, coded using 16 b/sample. How much memory is required to store 10 min of this stereo music?

The speech and musical signals are sampled according to the sampling theorem.

Solution:

(a) Speech signal:

$f_{\max} = 10$ KHz $\quad = > f_s \geq 2 \cdot f_{\max} = 20$ KHz. Let us consider f_s =20 KHz.

Therefore, we have:

$$(20{,}000 \text{ samples/s})\cdot(12 \text{ b/sample}) = 240 \text{ Kb/s}$$

(b) Musical signal:

$$f_{\max} = 20 \text{ KHz} \quad = > f_s \geq 2 \cdot f_{\max} = 40 \text{ KHz}$$

mono signal: $(40,000 \text{ samples/s}) \cdot (16 \text{ b/sample}) = 640 \text{ Kb/s}$
stereo signal: $2 \cdot 640 \text{ Kb/s} = 1,280 \text{ Kb/s}$

Memory requirements:

$$1,280 \text{ Kb/s} \cdot 10\text{min} = 1,280 \text{ Kb/s} \cdot 600\text{s} = 768{,}000 \text{ Kb}$$

$$768{,}000 \text{ Kb}/8 = 93,750 \text{ KB}$$

2.19. Consider a stereo signal, sampled at 44.1 KHz, and coded by using 16 b/sample. Calculate the memory requirements for storing 1 min of this audio format? What time is required to download 1 min of audio content from the Internet if the connection speed is 50 Kb/s?

Solution:

The sampling rate for the considered signal is 44,100 samples per second. This number is multiplied by 2 due to stereo format, so that we have 88,200 samples per second. Since each sample is coded with 16 bits, the total number of bits used to represent 1 s of this audio format is:

$$88,200 \cdot 16 = 1,411,200 \text{ b/s}$$

Furthermore, 60 s of audio contains:

$$1{,}411{,}200\text{b/s}\cdot 60 \text{ s} = 84{,}672{,}000\text{b},$$

or equivalently,

$$\frac{84,672,000}{8} = 10{,}584{,}000 \text{ B} = 10{,}336 \text{ KB} = 10 \text{ MB}.$$

The time required for a download of 1 min long audio content is:

$$\frac{84,672,000 \text{ b}}{50,000 \ \frac{\text{b}}{\text{s}}} = 1,693.44 \text{ s} = 28.22 \text{ min}$$

2.20. If the sampling frequency of a signal is $f_s = 32{,}000$ Hz, determine the frequency bandwidth of each subband in the case of the MPEG layer I compression algorithm.

Solution:

$$f_s = 32 \text{ KHz}$$
$$f_{\max} = f_s/2 = 16 \text{ KHz}$$

In the MPEG layer I compression algorithm the total frequency bandwidth is divided into 32 subbands. Hence, each of the subbands has the following width:

$$16{,}000/32 = 500 \text{ Hz}.$$

2.21. Calculate the bit rate of the compressed 16-bit stereo signal if the sampling frequency is:

(a) 32 KHz, (b) 44.1 KHz, (c) 48 KHz.

Assume that the MPEG layer I compression factor is 1:4.

Solution:

(a) $$\frac{16\text{b} \cdot 2 \cdot 32{,}000\frac{1}{\text{s}}}{4} = 256{,}000\ \text{b/s} = 256\ \text{Kb/s}.$$

(b) $$\frac{16\text{b} \cdot 2 \cdot 44{,}100\frac{1}{\text{s}}}{4} = 352{,}800\ \text{b/s} = 352.8\ \text{Kb/s}.$$

(c) $$\frac{16\ \text{b} \cdot 2 \cdot 48000\frac{1}{\text{s}}}{4} = 384{,}000\ \text{b/s} = 384\ \text{Kb/s}.$$

2.22. Consider 1,152 signal samples and show that MPEG layer II compression algorithm provides considerable savings compared to the MPEG layer I algorithm, even in the case when the samples are coded with the maximal number of bits in each subband.

Solution:

MPEG layer I algorithm:

1152 samples = 3 block x 384 samples
384 samples = 32 block x 12 samples
4 allocation bits are assigned to each block
Maximal number of bits that is available for coding of samples is 15
6 bits that corresponds to scale factor is assigned to each block
3·32·4 b + 3·32·6 b + 3·32·12·15 = 18240 b

MPEG layer II algorithm:

The signal with 1,152 samples is divided into three parts: 384 samples belonging to low frequencies, 384 middle frequency samples and 384 samples corresponding to high frequencies.
We assign 4 allocation bits, for each low-frequency block and consequently we have 15 b/sample at most;
3 allocation bits are assigned to each of 32 middle frequency blocks, meaning that at most 7 b/samples are available;
Finally, high-frequency blocks get 2 allocation bits each, and this means at most 3 b/sample;
The scale factor requires 6 bits per block.
Therefore, the total number of bits that is required for coding the set of 1,152 samples is:
$32 \cdot 4 + 32 \cdot 3 + 32 \cdot 2 + 3 \cdot 32 \cdot 6 + 32 \cdot 12 \cdot 15 + 32 \cdot 12 \cdot 7 + 32 \cdot 12 \cdot 3 = 10{,}464\ \text{b}$
The savings can be calculated as a difference between the number of required bits: 18,240–10,464 = 7,776 b.

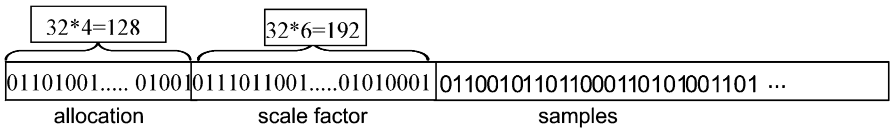


Fig. 2.48 Illustration of MPEG layer I sequence part

2.23. Consider a simplified part of the sequence obtained by using the MPEG layer I algorithm and determine the value of the third sample in the first block (from 32 blocks)? (Fig. 2.48)
Solution:

First 4 allocations bits – 0110 – correspond to the first block.

The sequence 0110 determines the samples within the considered block are coded by using 6 + 1 = 7 b/sample. Hence, we have:

I sample: 0110010
II sample: 1101100
III sample: 0110101
The value of the third signal sample is 53.

The scale factor is defined by the sequence 011101, i.e. the scaling factor is 29·2 dB = 58 dB.
2.24. The signal with maximal frequency 24 KHz is coded by using the MPEG layer II algorithm and the achieved bit rate is 192 Kb/s. Calculate the number of bits required for representation of the constant-length block used as a coding unit.

Solution:

$f_{max} = 24$ KHz $\Rightarrow f_s = 48$ KHz, or in other words 1 s of the signal consists of 48,000 samples.

The total number of bits for the coding block within the MPEG layer II algorithm is:

$$n = \frac{1,152 \ \text{samples} \cdot 192,000 \ \text{b/s}}{48,000 \ \text{samples/s}} = 4,608 \ \text{b}.$$

References

1. Bosi M, Goldberg RE (2003) Introduction to digital audio coding and standards. Springer, New York
2. Chu WC (2003) Speech coding algorithms. Wiley, Hoboken
3. Gibson J, Berger T, Lookabaugh T, Baker R, Lindbergh D (1998) Digital compression for multimedia: principles and standards. Morgan Kaufmann, San Francisco
4. Hankersson D, Greg AH, Peter DJ (1997) Introduction to information theory and data compression. CRC Press, Boca Raton
5. Hassanpour H, Mesbah M, Boashash B (2004) Time-frequency feature extraction of newborn EEG seizure using SVD-based techniques. EURASIP J Appl Signal Process 16:2544–2554

6. Hoeg W, Lauterbach T (2003) Digital audio broadcasting: principles and applications of digital radio. Wiley, Chichester
7. Kaplan R (1997) Intelligent multimedia systems. Willey, New York
8. Kovačević B, Milosavljević M, Veinović M, Marković M (2000) Robustna Digitalna Obrada Signala. Akademska misao, Beograd
9. Maes J, Vercammen M, Baert L (2002) Digital audio technology, 4th edn. In association with Sony, Focal Press
10. Mataušek M, Batalov V (1980) A new approach to the determination of the glottal waveform. IEEE Trans Acoust Speech Signal Process ASSP-28(6):616–622
11. Painter T (2000) Perceptual coding of digital audio. Proc IEEE 88(4):451–513
12. Pan D (1995) A tutorial on MPEG/audio compression. IEEE Multimedia 2(2):60–74
13. Pohlmann KC (2005) Principles of digital audio. McGraw-Hill, New York
14. Salomon D, Motta G, Bryant D (2009) Handbook of data compression. Springer, London
15. Sayood K (2000) Introduction to data compression, 2nd edn. Morgan Kaufmann, San Francisco
16. Smith MT (1999) Audio engineer's reference book, 2nd edn. Focal Press, Oxford
17. Spanias A, Painter T, Atti V (2007) Audio signal processing and coding. Wiley-Interscience, Hoboken
18. Stanković LJ (1994) A method for time-frequency signal analysis. IEEE Trans Signal Process 42(1):225–229
19. Stanković S, Orović I (2010) Time-frequency based speech regions characterization and eigenvalue decomposition applied to speech watermarking. EURASIP J Adv Signal Process, Special Issue on Time-Frequency Analysis and its Application to Multimedia signals, Article ID 572748, Pages(s) 10 pages
20. Steinmetz R, Nahrstedt K (2004) Multimedia systems. Springer-Verlag, Berlin Heidelberg
21. Vetterli M, Kovačević J (1995) Wavelets and subband coding. Prentice-Hall, Englewood Cliffs
22. Watkinson J (2001) The art of digital audio, 3rd edn. Focal Press, London
23. Watkinson J (2001) The MPEG handbook. Focal Press, Oxford
24. Wong DY, Markel JD, Gray AH (1979) Least squares glottal inverse filtering from the acoustic speech waveform. IEEE Trans Acoust Speech Signal Process ASSP-27(4):350–355

Chapter 3
Storing and Transmission of Digital Audio Signals

In this chapter, we consider the widely used media for storing digital data. Special attention is given to CD, Mini Disc (MD), DVD (concepts of data writing and reading processes are considered), as well as to the coding principles. Different error correction and interleaving algorithms such as cyclic redundancy check, cross interleaving, Reed–Solomon code, and Eight-to-Fourteen Modulation are presented. Also, the basic concepts of the digital audio broadcasting system are considered.

3.1 Compact Disc: CD

The basic characteristics of a CD are provided in Table 3.1.

A CD has 20,625 tracks, where the distance between tracks is 1.6 μm. The audio storage space is placed between the lead-in and the lead-out area, having diameters of 46 and 116 mm, respectively. The lead-in area contains information about the CD content, the length and the starting time of audio sequences. The lead-out area provides the information that the playback is completed. The internal structure of the CD is given in Fig. 3.1.

On the CD surface, there are pits and a flat layer called land. The pits can have one of nine different lengths, from T3 to T11 (Table 3.2). The smallest pit size is 0.833×0.5 μm. However, the pit and land lengths may vary, depending on the disc writing (turning) speed while recording. For example, T3 pit's length is 833 nm for the writing speed 1.2 m/s, while for the speed 1.4 m/s it is 972 nm.

Laser rays that fall on the land of the CD are reflected with the same path and the same intensities, while the intensities of rays scattered from the bumps are lower. The intensity of reflected beam is detected as one of the logical values (1 or 0). Figure 3.2 illustrates the laser beam reflections from a CD.

It is noteworthy that a CD is not sensitive to some amount of dust, fingerprints, and scratches. One reason that a CD has a good performance in terms of sensitivity

Table 3.1 Basic features of CD

Characteristics	Values
Frequency range	20 Hz–20 KHz
Dynamic range	≈96 dB
Diameter	12 cm
Playing time	60–74 min

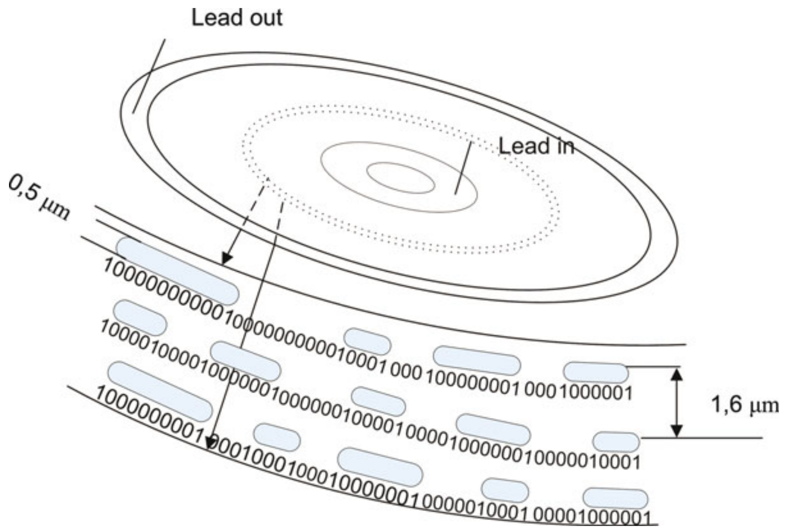


Fig. 3.1 The structure of CD

Table 3.2 Lengths of pits

Pits length		Size in nm
T3 = 10001		833
T4 = 100001		1,111
T5 = 1000001		1,388
...	...	...
T11 = 1000000000001		3,054

to dust is a protective layer of 1.2 mm thickness, which completely passes the laser beam through. For example, if there is a dust grain on the protective layer, the laser ray will pass to the focal point without obstacles as long as the dust grain diameter is less than 0.8 mm. The intensity of the reflected ray will correspond to the same logical value as in the case of reflection from the clean surface, Fig. 3.2.

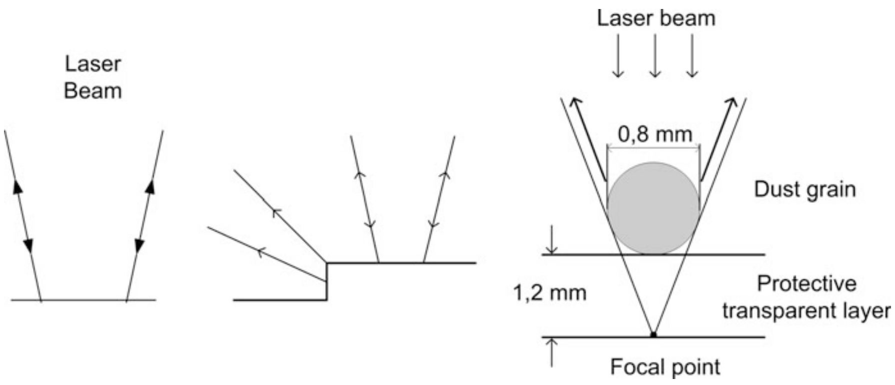


Fig. 3.2 Reflections from a CD

3.1.1 *Encoding CD*

In order to be resistant to scratches and other errors, an efficient coding scheme is applied. Namely, the audio signal stored on a CD is encoded within four steps:

1. Cross-Interleaved Reed–Solomon Coding (CIRC)
2. Generating a control word
3. EFM encoding
4. Generating synchronization word

When passing through the CD encoding system, a bit rate for a 16-bit stereo audio signal changes from $1.4112 \cdot 10^6$ to $4.3218 \cdot 10^6$ b/s. To easily understand the coding schemes used for CD, let us first briefly consider cyclic redundancy check (CRC) and interleaving.

3.1.1.1 Cyclic Redundancy Check

CRC is a general method used for error detection. The method relies on the division of a polynomial corresponding to the original sequence by another predefined polynomial function, resulting in a residue, which is actually a CRC. The length of a residue is always smaller than the length of the polynomial. CRC coding can also be done by using the exclusive OR operation, but both polynomials have to be represented as binary sequences.

Let us assume that the message is 11010011101100, while the divisor sequence is equal to 1011 (as a polynomial it is defined by $x^3 + x + 1$). Now, the EX OR operation should be carried out between the original sequence and the divisor sequence (from left to right). It should be mentioned that if the first bit in the original sequence is equal to 0, then we begin the EX OR operation on the next bit that has the value 1. The second step is to move the divisor sequence by one position to the right and perform the EX OR operation again. This procedure is repeated until the sequence 1011 reaches the end of the original sequence, as illustrated in the example. At the end, a binary residual sequence is obtained, representing the CRC function.

```
11010011101100
1011
______________________________
01100011101100
 1011
______________________________
00111011101100
  1011
______________________________
00010111101100
   1011
______________________________
00000001101100
       1011
______________________________
00000000110100
        1011
______________________________
00000000011000
         1011
______________________________
00000000001110
          1011
______________________________
00000000000101(the remaining 3 bits)
```

Typical polynomials used in the CRC encoding are given in Table 3.3.

3.1.1.2 Interleaving

Interleaving is an approach that arranges the data in noncontiguous order to decrease the effects of errors. This enables us to possibly recover damaged information by an interpolation method. For example, consider a signal with 24 samples and divide it into blocks of 12 samples. A simple interleaving can be obtained by reordering samples, as shown in Fig. 3.3. In this case, interleaving is based on moving the first 6 samples within the block to the right by $i - 1$, where i is the sample position (e.g., the third sample is moved for $3 - 1 = 2$ positions to the right).

Table 3.3 Some of the polynomials used for CRC coding

Code	Polynomial
CRC-1	$x + 1$
CRC-4 ITU	$x^4 + x + 1$
CRC-5 ITU	$x^5 + x^4 + x^2 + x1$
CRC-8-CCITT	$x^8 + x^2 + x + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16 CCIT	$x^{16} + x^{12} + x^5 + 1$
CRC-16 IBM	$x^{16} + x^{15} + x^2 + 1$

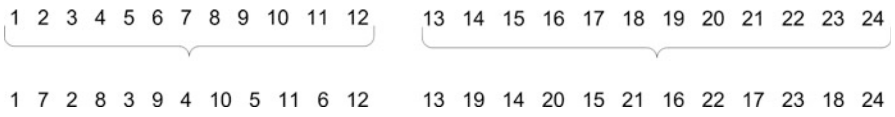


Fig. 3.3 An example of interleaving

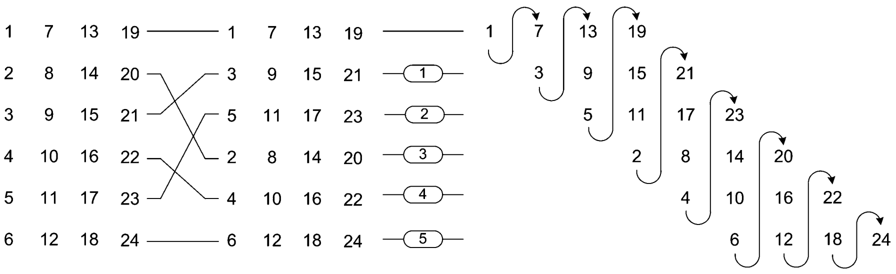


Fig. 3.4 Interleaving based on the delay lines

Another simple example of interleaving, which is closer to the concept used in the CIRC encoding, is shown in Fig. 3.4. Note that the distance between the consecutive samples is increased.

Each row has a different delay (the first row has no delay, the second row has a unit delay, etc.).

3.1.1.3 Cross-Interleaved Reed–Solomon Coding

Consider now the interleaving procedure used in the CD coding, which is considerably more complex and is illustrated in Fig. 3.5.

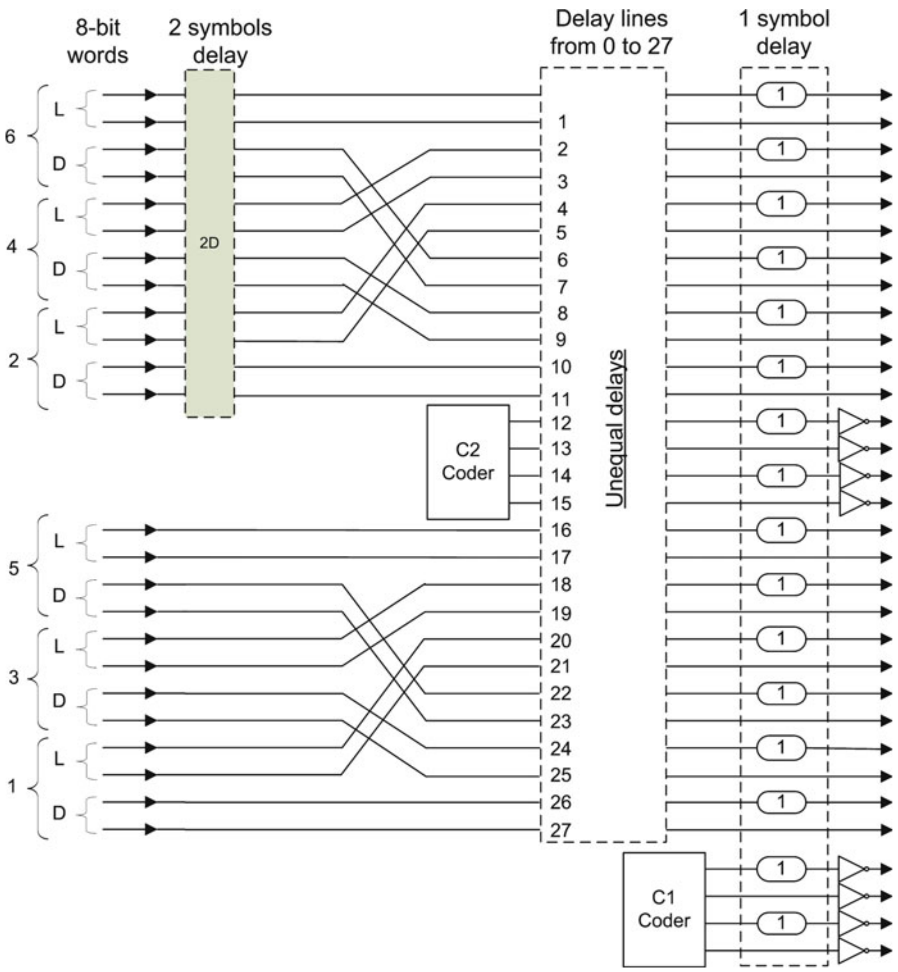


Fig. 3.5 Interleaving used in CD coding

The structure is based on a group of six samples for the left and six samples for the right channel of stereo audio signals. Each sample is represented by 16 bits. Odd and even samples are separated. From each sample, two 8-bit words are formed (24 words in total). Then, all even samples are delayed by two symbols.

Figure 3.6 illustrates an example depicting how even the part of the system with a two-symbol delay can be useful to reconstruct the damaged part of the signal. Labels L_i and R_i represent the left and right ith sample, respectively. Shaded parts denote damaged samples. In the lower part of Fig. 3.6, the delay compensation is performed and the samples are synchronized according to their initial order. Based on the even samples, the damaged odd samples are reconstructed by interpolation, and vice versa.

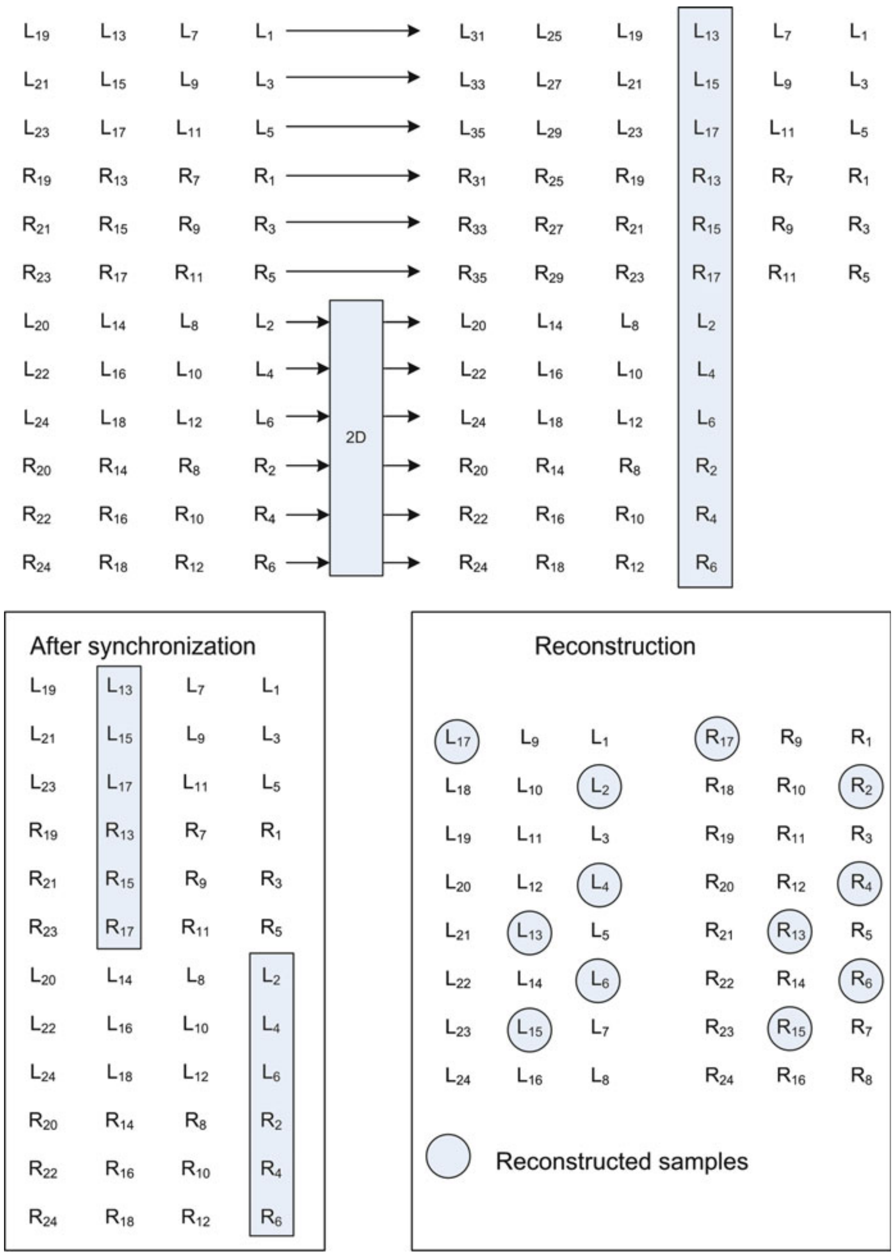


Fig. 3.6 The reconstruction principle of the damaged signal part

Table 3.4 The Galois field $GF(2^3)$

Exponential	Algebraic	Binary
0	0	000
1	1	001
a	a	010
a^2	a^2	100
a^3	$a+1$	011
a^4	$a \cdot a^3 = a^2 + a$	110
a^5	$a^2 \cdot a^3 = a^2 + a + 1$	111
a^6	$a \cdot a^5 = a^3 + a^2 + a = a + 1 + a^2 + a = a^2 + 1$	101
a^7	$a \cdot a^6 = a \cdot (a^2 + 1) = a + 1 + a = 1$	001

The C2 encoder, shown in Fig. 3.5, generates four Q words that are 8-bits long. These words represent the parity bytes used to increase the distance between the odd and even samples and allows for the errors detection. Additional interleaving is performed after the C2 encoder, which arranges the order and distances between the existing 28 words. The introduced delay between the words is used to dissipate the error across distant positions in order to increase the ability to recover as many samples as possible. After the interleaving subsystem, the C1 encoder generates four P words (P parity bytes). Therefore, the CIRC encoder ends up with 32 words from the initial 24 input words, introducing the redundancy of 8 words and increasing the bit rate from $1.4112 \cdot 10^6$ to $1.8816 \cdot 10^6$ b/s. The resultant 32 sequences are included within the unit called frame.

The procedure for determining P and Q parity words is made by using the Reed–Solomon code. It is based on the finite field arithmetic, which is usually referred to as Galois fields. A finite field of q elements is denoted as $GF(q)$. The field $GF(q)$ always contains at least one element, called a primitive element, with the order $(q - 1)$. If a is a primitive in $GF(q)$, then $(q - 1)$ consecutive powers of a: $\{1, a, a^2, \ldots, a^{q-2}\}$ must be distinct and they are $(q - 1)$ nonzero elements of $GF(q)$. The "exponential representation" allows in describing the multiplication operation as an addition: $a^x a^y = a^{x+y}$. A primitive element is a root of a *primitive polynomial* $p(x)$. For example, if we consider the polynomial: $p(x) = x^3 + x + 1$, then $a^3 + a + 1 = 0$. Note that the addition is done as the XOR operation.

The Reed–Solomon code uses the Galois field in the form $GF(2^k)$, where the elements of the field are represented by k bits. The 3-bit terms given in Table 3.4 describe a *Galois field* $GF(2^3)$.

In order to understand how to obtain P and Q words, let us consider one simplified, but illustrative example. Suppose that we have five data words labeled as A, B, C, D, and E (3-bit words are used). Then, we set the following equations:

$$A \oplus B \oplus C \oplus D \oplus E \oplus P \oplus Q = 0, \tag{3.1}$$

$$a^7A \oplus a^6B \oplus a^5C \oplus a^4D \oplus a^3E \oplus a^2P \oplus aQ = 0, \tag{3.2}$$

where a^i are the above defined constants. By solving the equations simultaneously, the expressions for P and Q word are obtained. Hence, (3.2) is divided by a, and then Q is replaced by $A \oplus B \oplus C \oplus D \oplus E \oplus P$ (since from (3.1) $Q = A \oplus B \oplus C \oplus D \oplus E \oplus P$ holds):

$$\begin{aligned}&a^6A \oplus a^5B \oplus a^4C \oplus a^3D \oplus a^2E \oplus aP \oplus Q = \\ &= a^6A \oplus a^5B \oplus a^4C \oplus a^3D \oplus a^2E \oplus aP \oplus A \oplus B \oplus C \oplus D \oplus E \oplus P \\ &\Rightarrow \left(a^6 \oplus 1\right)A \oplus \left(a^5 \oplus 1\right)B \oplus \left(a^4 \oplus 1\right)C \oplus \left(a^3 \oplus 1\right)D \oplus \left(a^2 \oplus 1\right)E = (a \oplus 1)P.\end{aligned} \tag{3.3}$$

By using the binary representation of constants from the Table 3.4, (3.3) can be simplified as:

$$\begin{aligned}a^2A \oplus a^4B \oplus a^5C \oplus aD \oplus a^6E = a^3P \\ P = a^6A \oplus aB \oplus a^2C \oplus a^5D \oplus a^3E,\end{aligned} \tag{3.4}$$

where $a^2/a^3 = a^{-1} = a^{7-1} = a^6$. Similarly, by multiplying (3.1) by a^2, we have:

$$\begin{gathered}a^2A \oplus a^2B \oplus a^2C \oplus a^2D \oplus a^2E \oplus a^2P \oplus a^2Q = 0 \quad \Rightarrow \\ a^7A \oplus a^6B \oplus a^5C \oplus a^4D \oplus a^3E \oplus \left(a^2A \oplus a^2B \oplus a^2C \oplus a^2D \oplus a^2E \oplus a^2Q\right) \oplus aQ = 0 \quad \Rightarrow \\ \left(a^7 \oplus a^2\right)A \oplus \left(a^6 \oplus a^2\right)B \oplus \left(a^5 \oplus a^2\right)C \oplus \left(a^4 \oplus a^2\right)D \oplus \left(a^3 \oplus a^2\right)E \oplus \left(a \oplus a^2\right)Q = 0\end{gathered}$$

Again using the binary representation of constants, the Q word is obtained as:

$$\begin{aligned}a^6A \oplus B \oplus a^3C \oplus aD \oplus a^5E = a^4Q \quad \Rightarrow \\ Q = a^2A \oplus a^3B \oplus a^6C \oplus a^4D \oplus aE.\end{aligned} \tag{3.5}$$

In order to detect errors, two syndromes are considered:

$$\begin{aligned}S_1 &= A' \oplus B' \oplus C' \oplus D' \oplus E \oplus P' \oplus Q', \\ S_2 &= a^7A' \oplus a^6B' \oplus a^5C' \oplus a^4D' \oplus a^3E \oplus a^2P' \oplus aQ',\end{aligned} \tag{3.6}$$

where A',B',. . .,Q' denote received words that may contain an error. Assume that the error occurred in the word C ($C' = C + G$), while the other words are without errors. Then, we obtain:

$$\begin{aligned}S_1 &= A \oplus B \oplus (C + G) \oplus D \oplus E \oplus P \oplus Q = G, \\ S_2 &= a^7A \oplus a^6B \oplus a^5(C + G) \oplus a^4D \oplus a^3E \oplus a^2P \oplus aQ = a^5G,\end{aligned} \tag{3.7}$$

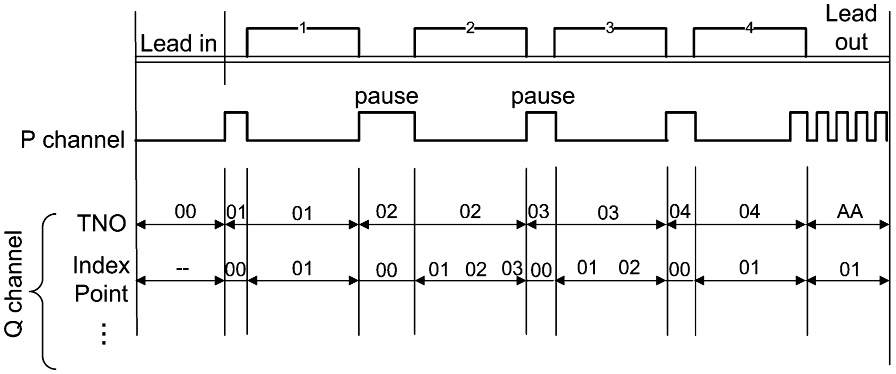


Fig. 3.7 Illustration of timing diagrams for *P* and *Q* channels

or $S_2 = a^5 S_1$. Therefore, the error is equal to the syndrome S_1 and the error location is obtained based on the weighting coefficient. After calculating the coefficient as: $a^x = \frac{S_2}{S_1}$, and concluding that $a^x = a^5$ holds, one may know that an error occurred within the *C* word, because *C* is multiplied by a^5.

3.1.1.4 Generating Control Word

The next step in the CD coding procedure is a control word generation. The control word is added to each block of 32 words. This word consists of the codes *P*, *Q*, *R*, *S*, *T*, *U*, *V*, *W*. Note that the choice of *P* and *Q* labels is made a bit unadvisedly, since we used them to obtain new code sequences independent of *P* and *Q* words generated in CIRC. *P* can have values 0 or 1. From Fig. 3.7, we can observe that *P* has value 1 between two sequences recorded on CD and value 0 during the sequence duration. Switching from 0 to 1 with frequency equal to 2 Hz in the lead-out area indicates the end of the disc. The *Q* word specifies the number of audio channels. It should be noted that the total length of these subcodes is 98 bits, which means that it can be read from 98 frames. After adding the control word, the bit rate is increased to:

$$33/32 \cdot 1.8816 \cdot 10^6 \mathrm{b/s} = 1.9404 \cdot 10^6 \mathrm{b/s}.$$

An example of the *P* and *Q* words is given in Fig. 3.7.

The *Q* word in the BCD format contains the current track number (01, 02, 03, etc.), the index number, running time, etc. Track number (TNO) represents the current track number and ranges from 01 to 99. The TNO within the lead in area has the value 00. The index point is a two-digit number in the BCD format, and within the sequences, it can be up to 99 index points. During a pause, the index point is equal to 00, while the index point at the beginning of each sequence is equal to 01. Also, the index point in the lead out area is equal to 01. Setting up the values for

Table 3.5 Examples of extending 8-bit words to 14-bit words

8-bit word	14 bit words
00000011	00100100000000
01001110	01000001001000
10101010	10010001000100
11110010	00000010001001

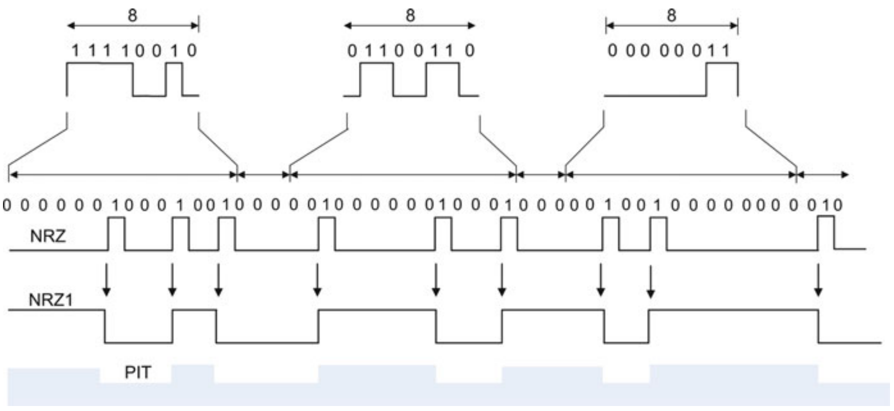


Fig. 3.8 An example of EFM encoding

index pointers is a way to divide the sequence into smaller parts. Index pointers are primarily intended for CDs with long sequences (e.g., a classical music CD), since they allow direct access to some parts of the sequence. However, they are rarely used nowadays.

Other subcodes are used for transmitting additional information such as text and information on duration of individual sequences.

After we determine the control word, the EFM (**E**ight to **F**ourteen **M**odulation) is used to convert 8-bit symbols into 14-bit symbols. Observe that with 8 bits we can make 256 combinations, while with 14 bits we can achieve 16,384 combinations. The basic idea of EFM coding is to map 8-bit words into 14-bit words such that the number of inversions between consecutive bits is reduced, i.e., the distance between transitions on the disc surface is increased (logical value 1 is used to determine the transitions). An example of an EMF mapping is shown in Table 3.5, while the EFM encoding procedure is illustrated in Fig. 3.8.

Note that the 14-bit signals are separated by using three merging bits to additionally reduce the distance between consecutive values 1. In other words, the initial 8-bit sequences are extended to 17 bits and represented by the NRZ code. Then the sequence of bits from the NRZ code is transferred into the NRZ1 code, such that each value 1 in the NRZ code makes the transition in NRZ1, as shown in Fig. 3.8. The NRZ1 sequence defines the position of pits when writing data to a CD. The minimum duration of the NRZ1 signals is 3 T (3 clock periods) and the maximum duration is 11 T.

The bit rate after this coding stage is:

$$17/8 \cdot 1.9404 \cdot 10^6 \mathrm{b/s} = 4.12335 \cdot 10^6 \mathrm{b/s}.$$

Finally, the CD encoding process ends with a synchronization (sync) word. This word is added after each frame to indicate the beginning of the frame, but also serves to control the spinning motor speed. The sync word consists of 12 values equal to 1, another 12 values 0, and 3 filter bits, making a total of 27 bits. Hence, from the previously achieved 561 bits per frame, now we get 588 bits within the frame (33 words·17 bits = 561).

The final bit rate is:

$$4.12335 \cdot 588/561 = 4.3218 \cdot 10^6 \mathrm{b/s}.$$

3.2 Mini Disc

Mini Disc (MD) has a diameter of 6.4 cm, almost twice smaller than a CD, with the same playing time of 74 min. The sound quality is almost identical to the CD audio quality. The structure of MD is depicted in Fig. 3.9.

Sophisticated compression algorithms are needed to reduce the amount of information that has to be stored in order to retain a high-quality sound on MDs. For this purpose, the MD uses ATRAC compression, described in the previous chapter. Note that the sampling frequency used for MDs is the same as for CDs (44.1 KHz), and the track width is 1.6 μm.

Data recording is done through the magnetization performed by the magnetic head. The magnetization is done at the specific temperature, which is above the Curie point (about 185 °C). Note that the materials that are easily magnetized are not used for manufacturing of MDs due to the possibility of data loss in the presence of an external magnetic field. Therefore, even when exposed to an external

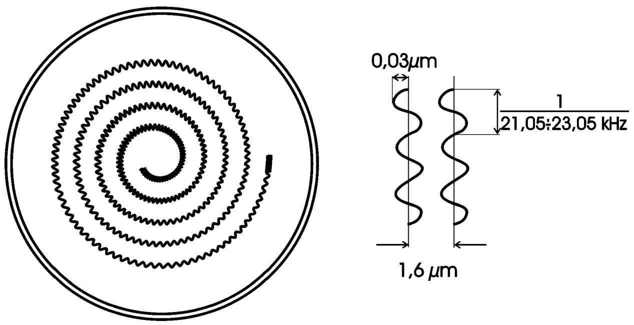


Fig. 3.9 The structure of MD

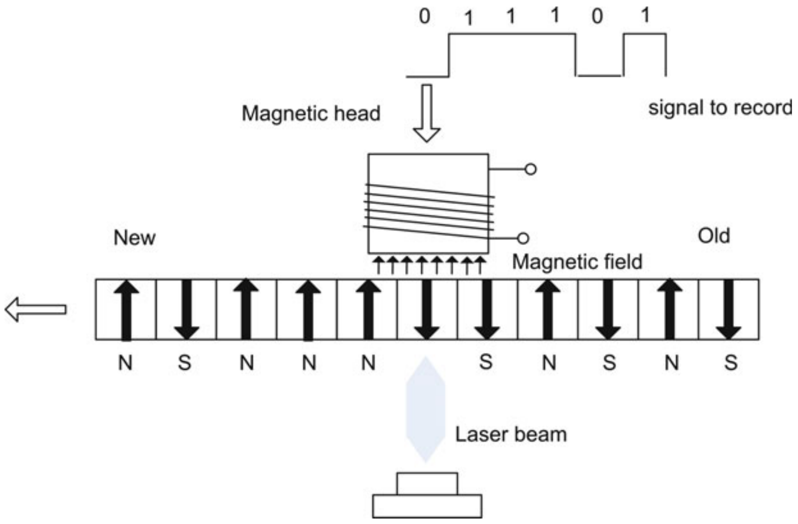


Fig. 3.10 Recording the data on the MD

magnetic field, MDs will not lose its contents, unless the required temperature is achieved. A system for the MD magnetization is illustrated in Fig. 3.10.

When recording the data, the laser heats the magnetic layer up to the Curie temperature. Then, the magnetic head, placed on the opposite disc surface, performs the magnetization by producing the correct polarity for each logical value (north or south, depending on the bit 1 or 0). The laser beam is reflected from the magnetic layer while reading the data. The polarization of the laser beam is changed based on the orientation of the magnetic layer (Fig. 3.11). An optical device with a polarizing filter collects reflected polarized signals. When the laser beam passes through the filter, the intensity changes according to the laser beam polarization, and the output signal is generated.

MDs use the Advanced CIRC (ACIRC) for encoding, which is similar to the CIRC used in CDs. It also uses the EFM encoding, along with the ATRAC compression, which is not used in CDs.

The antishock system is an important part of MDs as it enables the system to recover from any shocks during playback. This system is based on the RAM, allowing recovery from the shock with duration of several seconds.

A block diagram of the entire MD system is illustrated in Fig. 3.12.

When data are written to the MD (upper part in Fig. 3.12), the digital audio signal is fed to the ATRAC encoder. The ATRAC data compression is performed and the data are loaded to the antishock system, and further through the EFM/ ACIRC encoder (which includes interleaving, error detection and EFM coding). The signal from the EFM/ACIRC encoder is used to control the magnetic head when recording the data.

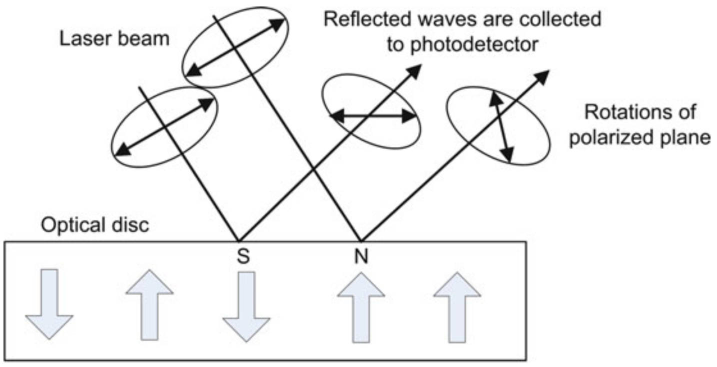

Laser beam
Reflected waves are collected to photodetector
Rotations of polarized plane
Optical disc
S
N

Fig. 3.11 Reflections of the laser beam in the case of MD

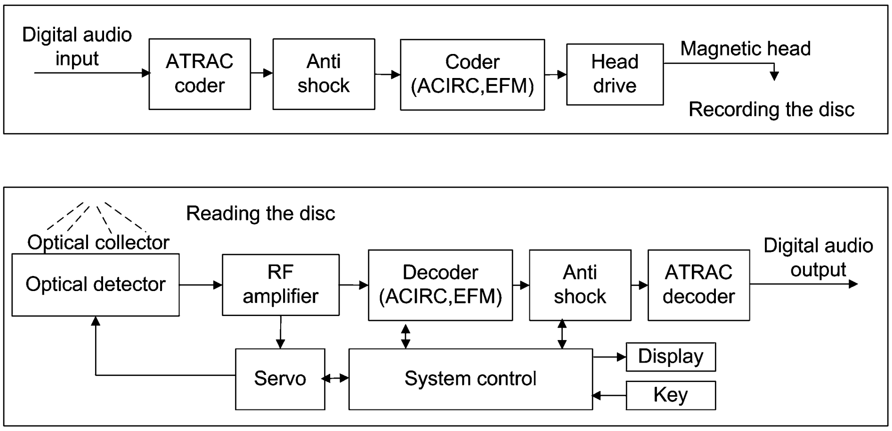


Fig. 3.12 A block diagram of the MD system

Reproduction or reading of the data starts at the unit called the optical signal collector, shown in the lower part of Fig. 3.12. A special device within this unit is called the optical detector. Then, the signal is amplified by the radio frequency (RF) amplifier and fed to the EFM/ACIRC decoder. The data are decompressed using the ATRAC decoder that follows the antishock system. The output of the ATRAC decoder is a digital audio signal.

3.3 Super Audio CD (SACD)

SACD provides a high-quality sound, with the option of multichannel records. The diameter of SACD is the same as of a CD, while the width of pit lane is less than in the case of CD (the track width is 0.74 μm and the length of a pit is 0.40 μm).

The sampling frequency is 2.8224 MHz and the 1-bit DSD encoding is used. The maximum frequency of the reproduced sound is up to 100 KHz and with 120 dB dynamic range. Data are protected with the SACD watermarking techniques.

The memory space required to store 74 min stereo audio recording is $(2 \cdot 74 \cdot 60 \cdot 2.8224 \cdot 10^6)/8\ \mathrm{B} = 2.9$ GB. Hence, in the case of six-channel format, the required memory capacities would be certainly much larger. Therefore, SACDs use lossy compression (e.g., AC3) or lossless compression based on the complex algorithms with adaptive prediction and entropy coding.

3.4 DVD-Audio

DVD-audio (DVD hereinafter) is also used to record high-quality sound with the sampling frequency of 192 KHz and 24 bit data format. DVD allows the signal-to-noise ratio of $S/N = 146$ dB. The capacity of a DVD is 4.7 GB and its diameter is 8 or 12 cm. The maximum number of channels is 6. Based on these requirements, a DVD cannot store 74 min of high quality music within 4.7 GB of memory space. Therefore, the data have to be compressed. For this purpose, the lossless compression called Meridian Lossless Packing (or Packed PCM) has been developed. It is based on three lossless techniques: Infinite Impulse Response (IIR) waveform predictor selected from a set of predetermined filters to reduce the intersample correlation, lossless interchannel decorrelation, and Huffman coding. This compression algorithm compresses the original data by 50 %. However, even with this high compression ratio, it is not possible to record six channels with sampling frequency of 192 KHz and 24 bits. Therefore, the channels reflecting the influence of the environment (surround sound) use different sampling frequencies. For example, the direct left, right, and center channels are characterized by 24-bit format and the sampling frequency of 192 KHz, while the signals in the remaining three channels (representing the surround effects) have a sampling frequency 96 KHz and they are coded by 16 bits.

3.5 Principles of Digital Audio Broadcasting: DAB

Before we consider the main characteristics of DAB systems, let us review some basic facts about the FM systems. In order to receive an FM signal with a stable high quality, the fixed and well-directed antennas are required. For example, it is impossible to achieve this condition with car antennas. Also, due to the multipath propagation, the waves with different delays (i.e., different phases) can cause a significant amplitude decrease and, hence, the poor reception of such signals (Fig. 3.13).

The DAB system can avoid the aforementioned problems. Consider a DAB system given in Fig. 3.14.

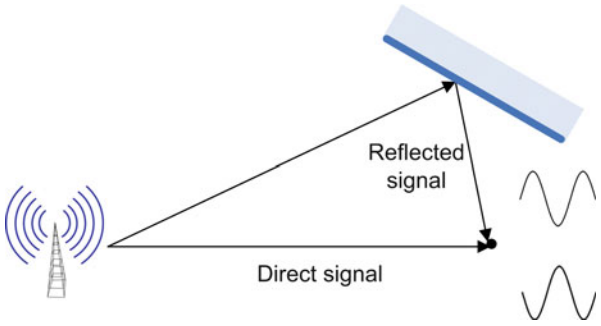


Fig. 3.13 Direct and reflected FM signals with equal strength are nulled when 180° phase difference occur

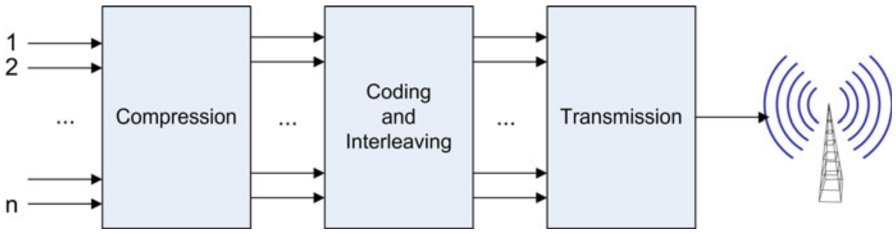


Fig. 3.14 A block scheme of DAB system

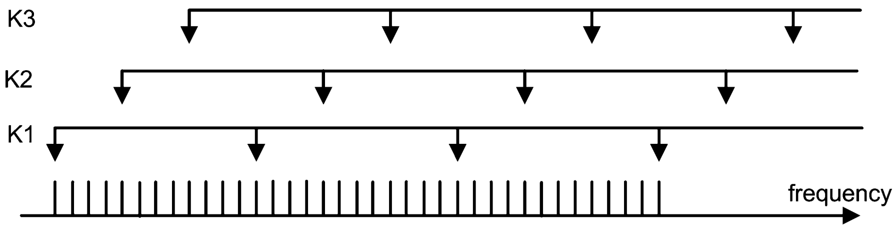
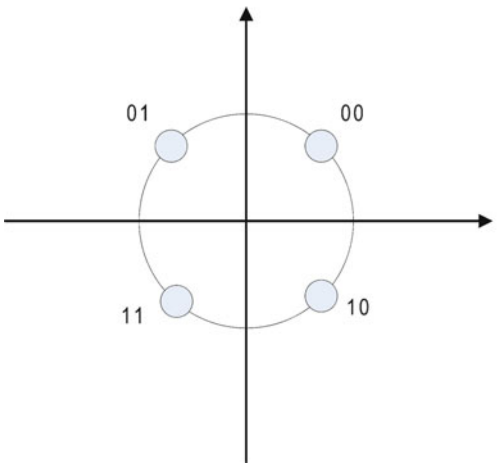
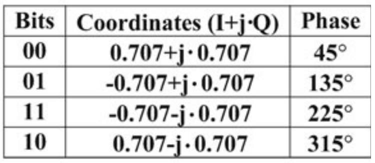


Fig. 3.15 An illustration of channel interleaving

The first block compresses the data, which are then forwarded to the second block. The second block encodes the data in order to become less sensitive to noise. Lastly, the signal is forwarded to a transmitter that broadcasts the data. Figure 3.15 shows the channel interleaving process used to combine data from different channels into one transmission channel.

If interference occurs, it will not damage only the signal in one channel, but will be scattered across all channels. Hence, a significant damage of signal belonging to only one channel is avoided.





Bits	Coordinates (I+j·Q)	Phase
00	0.707+j·0.707	45°
01	-0.707+j·0.707	135°
11	-0.707-j·0.707	225°
10	0.707-j·0.707	315°

Fig. 3.16 Diagram and table for QPSK modulation

3.5.1 Orthogonal Frequency-Division Multiplexing (OFDM)

OFDM is used to achieve more efficient bandwidth utilization for DAB systems. Binary data sequence is first divided into pairs of bits, which are then forwarded to the QPSK modulator (some systems use QAM or other modulation schemes). This means that two bits are mapped into one of the four phase values, as illustrated in the diagram in Fig. 3.16.

This produces the complex QPSK symbols. If the changes in the phase of the received signal are used instead of the phase itself, the scheme is called the differential QPSK (DQPSK). It depends on the difference between successive phases. In DQPSK the phase shifts are 0°, 90°, 180°, 270°, corresponding to the data '00', '01', '11', '10', respectively.

Each of the symbols obtained after QPSK modulation is multiplied by a subcarrier frequency:

$$s_k(t) = A_k e^{j\phi_k} e^{j2\pi f_k t}, \tag{3.8}$$

where A_k i ϕ_k are the amplitude and the phase of a QPSK symbol. For example, symbols obtained by using the QPSK modulation have the constant amplitude and their phases can have one of the four possible values. If we assume that we have N subcarriers, then one OFDM symbol will be in the form:

$$s(t) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} A_k e^{j(2\pi f_k t + \phi_k)}, \quad 0<t<T, \tag{3.9}$$

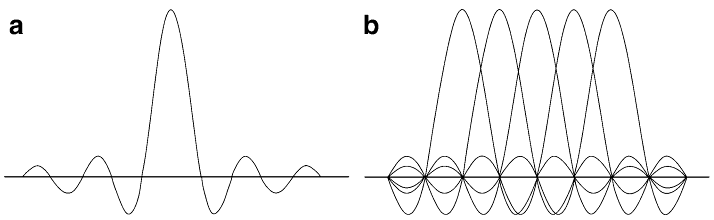


Fig. 3.17 An OFDM spectrum: (**a**) One subcarrier, (**b**) Five subcarriers

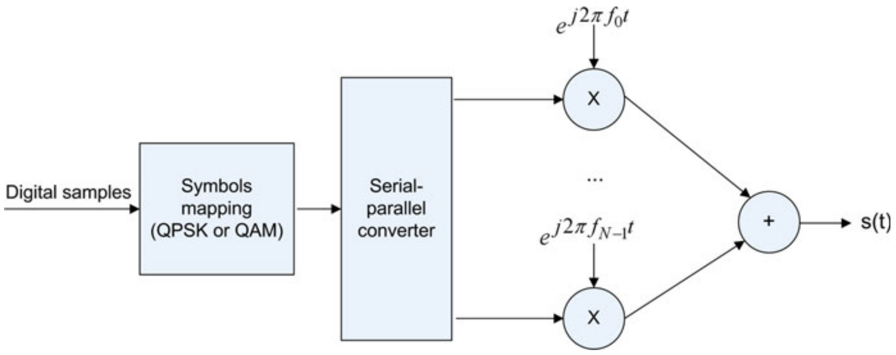


Fig. 3.18 A simplified block diagram of OFDM system

where $f_k = f_0 + k\Delta f = f_0 + k\frac{1}{NT_s}$, T_s is the length of the symbols (e.g., the QPSK symbols), while $T = N{\cdot}T_s$ is the OFDM symbol duration. The carrier frequency is f_0, while the subcarriers are separated by $1/T$. The subcarriers are transmitted in mutually orthogonal frequencies, so that the subcarriers are peak centered at the positions where other subcarries pass through zero (Fig. 3.17). Note that the OFDM symbol corresponds to the definition of the inverse Fourier transform. Comparing to the previously used form of the Fourier transform, ω_{k} is replaced by $2\pi f_k$, and consequently $1/N$ is replaced by $1/\sqrt{N}$.

The spectrum of an individual subcarrier is of the form $\sin(x)/x$ and it is centered at the subcarrier frequency.

A simplified scheme including QPSK and OFDM modulator is given in Fig. 3.18. Note that an OFDM system should include additional elements, such as pilot symbols, guard intervals, etc., but here, we only deal with the basic OFDM concepts.

We saw that the OFDM modulation can be performed by calculating the inverse Fourier transform. Demodulation is achieved by dividing the signal into the parts that are equal in duration to OFDM symbols. Then, the Fourier transform is performed and we can identify the subcarrier frequencies. The resulting signal is obtained by calculating the phases of the components on the subcarrier frequencies.

3.6 Examples

3.1. Starting from the sequence:

$$\{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\},$$

perform a simple interleaving procedure defined as follows: the sequence is divided into four-sample segments, and then the first interleaved block is formed by taking the first elements from each segment, the second block is formed from the elements on the second position, and so on. Determine the output sequence.

Solution (Fig. 3.19):

3.2. Consider the following input sequence:

$$\{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16\}$$

The interleaving procedure is defined as follows:

- The input samples are placed to the 4 × 4 matrix, by filling the matrix rows.
- The matrix rows are reordered according to the principle 4–2–3–1 (the new order of rows).
- The columns are reordered by using the same rule.

Determine the output sequence obtained by reading the columns of the resulting matrix.

Solution:

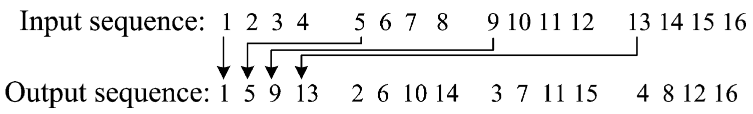
$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array} \Rightarrow \begin{array}{cccc} 13 & 14 & 15 & 16 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \end{array} \Rightarrow \begin{array}{cccc} 16 & 14 & 15 & 13 \\ 8 & 6 & 7 & 5 \\ 12 & 10 & 11 & 9 \\ 4 & 2 & 3 & 1 \end{array}$$

Output sequence is: {16,8,12,4,14,6,10,2,15,7,11,3,13,5,9,1}.

3.3. Consider a 16-bit audio stereo signal and calculate how much does the bit rate change when passing through the first three CD encoding stages (CIRC, Generating control word, EFM), if the starting bit rate at the input of the coder is $1.4112 \cdot 10^6$ b/s.

Solution:

CIRC: At the input of the CIRC coder, we have 24 words (8 bits each). The coder C2 generates 4 Q words (8 bits each), while the coder C1 generates 4 P words (8 bits each).



Input sequence: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Output sequence: 1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16

Fig. 3.19 Example of interleaving

At the output of the CIRC Coder, we have 32 words.

24 words produce the bit rate equal to $1.4112 \cdot 10^6$ b/s $\Rightarrow$
32 words produce the following bit rate:

$$(32/24) \cdot 1.4112 \cdot 10^6 = 1.8816 \cdot 10^6 \text{b/s}.$$

Generating control word: The 8-bit control word (P, Q, R, S, T, U, V, W) is assigned to each block.

Before generating the control word, the bit rate was $1.8816 \cdot 10^6$ b/s. At the end of this stage the bit rate becomes:

$$(33/32) \cdot 1.8816 \cdot 10^6 \text{b/s} = 1.9404 \cdot 10^6 \text{b/s}.$$

EFM: In this stage 8-bit symbols are firstly extended into 14-bit symbols, and then three additional bits are embedded between 14-bit combinations. Hence, instead of 8-bit words, we have 17-bit words at the output of the EFM coder, which results in the bit rate:

$$(17/8) \cdot 1.9404 \cdot 10^6 \text{b/s} = 4.1233 \cdot 10^6 \text{b/s}.$$

3.4. Consider the Super audio CD with sampling frequency 2.8224 MHz and 1-bit DSD coding. It is recommended that the 74 min of an audio is stored within 4.7 GB. Is it enough memory to store the considered six-channel audio format?

Solution:

$$f_s = 2.8224 \text{ MHz} \quad \Rightarrow \quad 2.8224 \cdot 10^6 \text{samples/s}.$$

$$74 \text{ min} = 74 \cdot 60 = 4440 \text{ s}$$

Memory requirements: $6 \cdot 4440 \cdot 2.8224 \cdot 10^6 \cdot 1\text{b} = 75188 \cdot 10^6 \text{b}$,

$$\frac{75188 \cdot 10^6}{8} = 9.4 \cdot 10^9 \text{B} = 8.75 \, \text{GB}.$$

Hence, 4.7 GB is not enough to store 74 min of the considered audio format.

3.5. In the case of DVD, the samples of direct left, right, and central channel are coded by using 24 bits, while the sampling frequency is 192 KHz. The samples of the three environmental channels are coded by using 16 bits (the sampling frequency is 96 KHz). Calculate the memory requirements for storing 10 min of audio on DVD.

Solution:

In the considered case, we have:

- Three channels with $192 \cdot 10^3$ samples/s, each coded by using 24 bits
- Three channels with $96 \cdot 10^3$ samples/s, each coded by using 16 bits

Memory requirements:

$$3 \cdot 24 \cdot 192 \cdot 10^3 \cdot 10 \cdot 60 + 3 \cdot 16 \cdot 96 \cdot 10^3 \cdot 10 \cdot 60 =$$
$$= 8294400 \cdot 10^3 + 2764800 \cdot 10^3 = 11059200 \cdot 10^3 \text{ b}$$
$$(11059200 \cdot 10^3/8)/(1024^3) = 1.29 \text{ GB}$$

3.6. Having in mind that the sector of a CD contains 98 frames, each frame contains 588 bits, and each sample is coded by using 16 bits (the sampling frequency is 44.1 KHz, stereo channel), calculate the number of sectors that are processed/read within 2 s.

Solution:

One sector of a CD contains the following number of bits:

$$98 \text{ frames} \cdot 588 \text{ b} = 57624 \text{ b}.$$

The bit rate for a considered stereo signal is:

$$2 \cdot 44100 \cdot 16 = 1411200 \text{ b/s}.$$

Hence, the total number of sectors that are read in 2 s is:

$$2 \cdot 1411200/57624 = 49 \text{ sectors}.$$

3.7. By using the CD bit rate equal to $4.3218{\cdot}10^6$ b/s, calculate the number of bits which are used for (P, Q, R, S, T, U, V, W) words within 1 s of the audio signal stored on CD?

Solution:

The total number of frames: $\frac{4.3218 \cdot 10^6 \text{ b/s}}{588 \text{ b/frames}} = 7,350 \text{ frames/s}$

The total number of sectors is: $\frac{7,350}{98} = 75.$

Each sector contains one of each word type: P, Q, R, S, T, U, V, W. Hence, the total number of bits used to represent these words is:

$$75 \text{ sectors} \cdot 98 \text{ b/word} \cdot 8 \text{ words} = 58,800 \text{ b}$$

3.8. Consider a 16-bit sequence: 1001101010101010, which is fed to the QPSK modulator. The resulting QPSK sequence duration is $T = 4.984$ ms and it is the input of OFDM block. Assuming that the carrier frequency is $f_0 = 1$ KHz, while the inverse Fourier transform is calculated in 768 points, determine the frequencies of subcarriers f_1 and f_2.

Solution:

At the output of QPSK modulator, we obtain the sequence of eight symbols. The QPSK symbol duration is:

$$T_S = \frac{T}{8} = \frac{4.984 \text{ ms}}{8} = 0.623 \text{ ms}.$$

The frequency of the k-th subcarrier is given by:

$$f_k = f_0 + k\Delta f = f_0 + k\frac{1}{NT_S},$$

Hence, the frequencies of the first two subcarriers are obtained as:

$$f_1 = f_0 + \Delta f = f_0 + \frac{1}{NT_S} = 1 \text{ KHz} + \frac{1}{768 \cdot 0.623 \text{ ms}} = 1002.09 \text{ Hz},$$

$$f_2 = f_0 + 2\Delta f = f_0 + 2\frac{1}{NT_S} = 1 \text{ KHz} + \frac{2}{768 \cdot 0.623 \text{ ms}} = 1004.18 \text{ Hz}.$$

3.9. Determine the frequency of the subcarrier $k = 5$ within a certain OFDM system, if the carrier frequency is $f_0 = 2{,}400$ MHz, while the symbols rate is $f_S = 2$ MHz and the total number of subcarriers is $N = 200$.

Solution:

The frequency of the k-th subcarrier can be calculated as:

$$f_k = f_0 + k\Delta f = f_0 + k\frac{1}{NT_S},$$

where:$f_S = \frac{1}{T_S}$. The frequency of fifth subcarrier is then:

$$f_5 = f_0 + 5\Delta f = f_0 + 5\frac{f_S}{N} = 2{,}400 \text{ MHz} + 5\frac{2 \text{ MHz}}{200} = 2400.05 \text{ MHz}.$$

3.10. Determine the number of subcarriers in the OFDM system if the OFDM symbol duration is 3.2 μs, while the total transmission bandwidth is $B = 20$ MHz.

Solution:

For a given OFDM symbol duration $NT_S = 3.2$ μs, we can calculate subcarrier spacing:

$$\Delta f = \frac{1}{NT_S} = \frac{1}{3.2\ \mu\text{s}} = 312.5 \text{ KHz}.$$

The number of subcarriers in the OFDM system can be obtained as:

$$N_{sc} = \frac{B}{\Delta f} = 64.$$

References

1. Bahai A, Saltzberg BR, Ergen M (2004) Multi-carrier digital communications, 2nd edn. Springer, New York
2. Bosi M, Goldberg RE (2003) Introduction to digital audio coding and standards. Springer, New York
3. Frederiksen FB, Prasad R (2002) An overview of OFDM and related techniques towards development of future wireless multimedia communications. In: IEEE Radio and Wireless Conference, RAWCON 2002, pp 19–22
4. Immink KAS (2002) A survey of codes for optical disk recording. IEEE J Sel Areas Commun 19(4):756–764
5. Hoeg W, Lauterbach T (2003) Digital audio broadcasting: principles and applications of digital radio. Wiley, Chichester
6. Lin TC, Truong TK, Chang HC, Lee HP (2011) A future simplification of procedure for decoding nonsystematic Reed-Solomon codes using the Berlekamp-Massey algorithm. IEEE Trans Commun 59(6):1555–1562
7. Maes J, Vercammen M, Baert L (2002) Digital audio technology, 4th edn. In association with Sony, Focal Press
8. Orović I, Zarić N, Stanković S, Radusinović I, Veljović Z (2011) Analysis of power consumption in OFDM systems. J Green Eng 1(1):477–489
9. Painter T (2000) Perceptual coding of digital audio. Proc IEEE 88(4):451–513
10. Roth R (2006) Introduction to coding theory. Cambridge University Press, Cambridge
11. Shieh W, Djordjević I (2009) Orthogonal frequency division multiplexing for optical communications. Academic, London
12. Spanias A, Painter T, Atti V (2007) Audio signal processing and coding. Wiley-Interscience, Hoboken
13. Watkinson J (2001) The art of digital audio, 3rd edn. Focal Press, Oxford
14. Wicker SB, Bhargava VK (1999) Reed-Solomon codes and its applications. Wiley, New York

Chapter 4
Digital Image

4.1 Fundamentals of Digital Image Processing

An image can be represented as a two-dimensional analog function ***f*(*x*,*y*)**. After digitalization, a digital image is obtained and it is represented by a two-dimensional set of samples called pixels. Depending on the number of bits used for pixel representation, a digital image can be characterized as:

- Binary image – each pixel is represented by using one bit
- Computer graphics – four bits per pixel are used
- Grayscale image – eight bits per pixel are used
- Color image – each pixel is represented by using 24 or 32 bits

Increasing the number of bits reduces the quantization error, i.e., increases the SNR by 6 dB per bit.

Grayscale image with N_1 rows and N_2 columns contains $N_1 \times N_2$ spatially distributed pixels, and it requires $8 \times N_1 \times N_2$ bits for representation. Color images are represented by using three matrices (for three color channels). Hence, if 8 bits per pixel are used, we need $3 \times 8 \times N_1 \times N_2$ bits of memory to store a color image.

In addition to the spatial distribution of pixels, which provides the information about the positions of grayscale values, a pixel value distribution in different image regions can be analyzed as well. Such a distribution can be described by the joint density function:

$$p(x_i) = \sum_{k=1}^{N} \pi_k p_k(x_i), \quad i = 1, 2, \ldots, M, \tag{4.1}$$

DOI 10.1007/978-1-4614-4208-0_4, © Springer Science+Business Media, LLC 2012

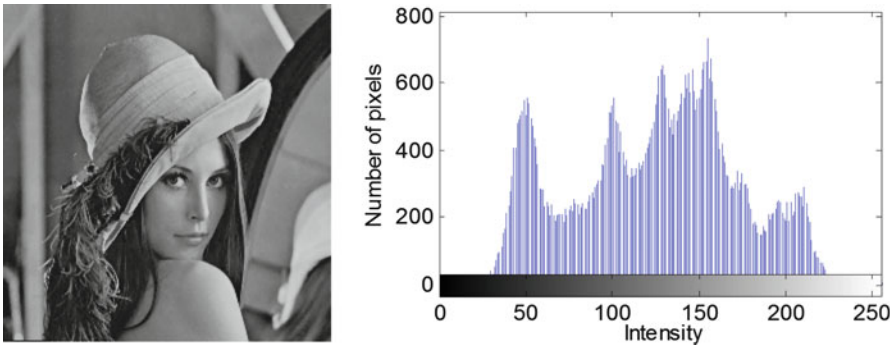


Fig. 4.1 Histogram of "Lena" image

where x_i represents the gray level of the i-th pixel, $p_k(x_i)$ is the probability density function (pdf) for a region k, and π_k is a weighting factor. The pdf for a region k can be described by the generalized Gaussian function:

$$p_k(x_i) = \frac{\alpha\beta_k}{2\Gamma(1/\alpha)} e^{[-|\beta_k(x_i-\mu_k)|^\alpha]}, \quad \alpha>0, \quad \beta_k = \frac{1}{\sigma_k}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{\frac{1}{2}}, \tag{4.2}$$

where Γ is the gamma function and μ_k represents the mean value. The variance σ_k is used to calculate β_k. For $\alpha>>1$, the pdf becomes uniform. For $\alpha = 2$, the Gaussian distribution is obtained, while for $\alpha = 1$ the Laplace distribution follows. The generalized Gaussian pdf is suitable, because it can be used to describe the image histogram. The image histogram provides important information about the occurrence of certain pixel values, and as such, plays an important role in image analysis. The histogram of a grayscale image "Lena" is given in Fig. 4.1.

4.2 Elementary Algebraic Operations with Images

Consider two images of the same dimensions, whose pixels at an arbitrary position $(\boldsymbol{i,j})$ are denoted as $\boldsymbol{a(i,j)}$ for the first and $\boldsymbol{b(i,j)}$ for the second image. Addition or subtraction of two images is done by adding or subtracting the corresponding pixels of an image, so that the resulting pixel is given in the form: $\boldsymbol{c(i,j)} = \boldsymbol{a(i,j)} \pm \boldsymbol{b(i,j)}$. Multiplying the image by a constant term $\boldsymbol{k}$ can be written as $\boldsymbol{c(i,j)} = \boldsymbol{ka(i,j)}$. However, if we want to represent the result of these and other operations as a new image, we must perform quantization (i.e., rounding to integer values) and limit the results in the range of 0–255 (grayscale image is assumed).

Consider now the grayscale images "Baboon" and "Lena" (Fig. 4.2).

Let us perform the following operation: $\boldsymbol{c(i,j)} = \boldsymbol{a(i,j)} + \boldsymbol{0.3b(i,j)}$, where $\boldsymbol{a(i,j)}$ denotes the pixel belonging to the "Lena" image, while $\boldsymbol{b(i,j)}$ belongs to the "Baboon" image. The result is the image shown in Fig. 4.3.

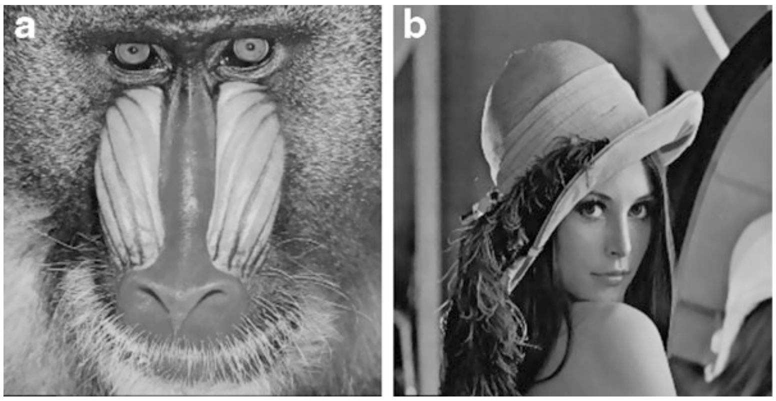


Fig. 4.2 (**a**) Grayscale image “Baboon”, (**b**) grayscale image “Lena”

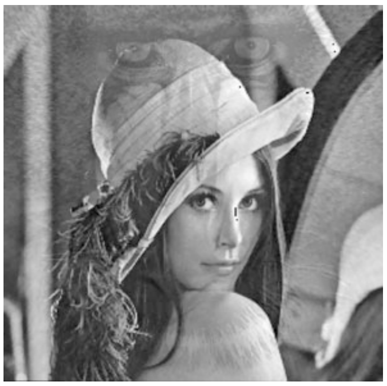


Fig. 4.3 The resulting image obtained by adding 30% of “Baboon” to “Lena”

To obtain a negative of a grayscale image, we use the following relation:

$$\boldsymbol{n}(\boldsymbol{i},\boldsymbol{j}) = 255 - \boldsymbol{a}(\boldsymbol{i},\boldsymbol{j}).$$

The negative image of “Lena” is shown in Fig. 4.4.

Clipping (cutting the pixels values over a certain level $c_{\max}$ and below a certain level $c_{\min}$) is another mathematical operation used in image processing, and it is defined as:

$$a(i,j) = \begin{cases} c_{\max}, & a(i,j) > c_{\max}, \\ a(i,j), & c_{\max} \geq a(i,j) \geq c_{\min}, \\ c_{\min}, & a(i,j) < c_{\min}. \end{cases} \tag{4.3}$$

For example, consider clipping of image “Lena” with $c_{\min} = 100$, $c_{\max} = 156$. The result of clipping is shown in Fig. 4.5.

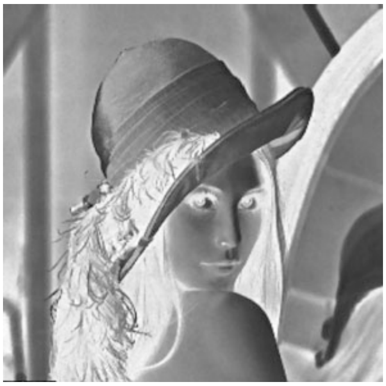


Fig. 4.4 Negative of “Lena” image

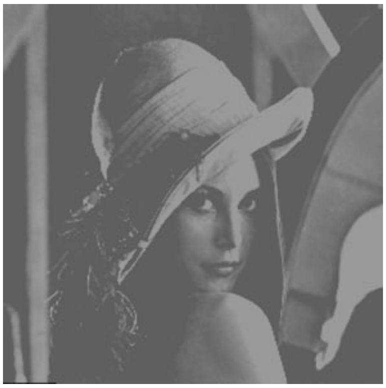


Fig. 4.5 Clipped “Lena” image

4.3 Basic Geometric Operations

Translation of an image $\boldsymbol{a}(\boldsymbol{i},\boldsymbol{j})$ with dimensions $N_1 \times N_2$ can be represented as moving the pixels in one or both directions for a certain number of positions. In the example shown in Fig. 4.6, we translated the image by embedding 31 rows and 31 columns of black color (zero value), while omitting the last 31 rows and columns. If we would like a white surface to appear after translation, the zero values should be replaced by the maximum values (e.g., value 255).

For an image $a(x,y)$ the coordinates can be written by the vector $\begin{bmatrix} x \\ y \end{bmatrix}$. Then the image rotation can be defined by:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{4.4}$$

where $\begin{bmatrix} X \\ Y \end{bmatrix}$ are the new coordinates after rotation (Fig. 4.7).

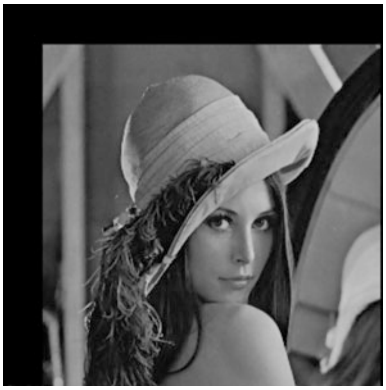


Fig. 4.6 "Lena" image translated for 31 columns and 31 rows

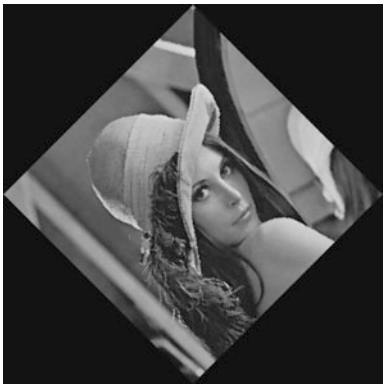


Fig. 4.7 "Lena" is rotated by 45°

After image rotation, we need to transform points from the polar coordinate system to the rectangular coordinate system. In general, this transform is performed with certain approximations.

4.4 The Characteristics of the Human Eye

By considering the characteristics of human visual system, we can define different image processing algorithms that will meet important perceptual criteria.

One of the specific features of the human eye is sensitivity to the change of light intensity. Specifically, the eye does not perceive the changes in light intensity linearly, but logarithmically. It means that at lower intensity human eye can notice very small changes in brightness, while at high intensity even a much bigger change can hardly be registered.

There are two types of cells in the eye: elongated (rod cells or rods) and cone-like (cone cells or cones). There are about 125 million rods and about 5 million cones.

The rods just detect the amount of light, while the cones detect colors. An eye is not equally sensitive to three primary colors: red, green, and blue. The relative ratio of these sensitivities is:

$$\text{Red} \; : \; \text{Green} \; : \; \text{Blue} = 30\% : 59\% : 11\%$$

An eye is able to identify approximately between 40 and 80 shades of gray, while for color images it can recognize between 15 and 80 million colors. The light entering the eye is detected by the cones and rods. The image in the brain is actually obtained as the sum of images in primary colors. TV sets (CRT, LCD, and plasma), monitors, video projectors follow the human three-color model.

It is interesting to note that various models are used to measure the image quality in different applications. Namely, the image quality can be represented by three dimensions: *fidelity*, *usefulness*, and *naturalness*. For example, the usefulness is a major metric for medical imaging, the fidelity is the major metric for paintings, while the naturalness is used in virtual reality applications.

4.5 Color Models

Color is one of the most important image characteristics. It is generally invariant to translation, rotation, and scaling. The color image can be modeled using various color systems. RGB is one of the commonly used color systems. It can be represented by the color cube as shown in Fig. 4.8. The gray level is defined by the line R = G = B. Although the RGB model is based on the human perception of colors, and thus, it has been used for displaying images (monitors, TV, etc.), other color systems have also been defined in order to meet various constraints that exist in the applications.

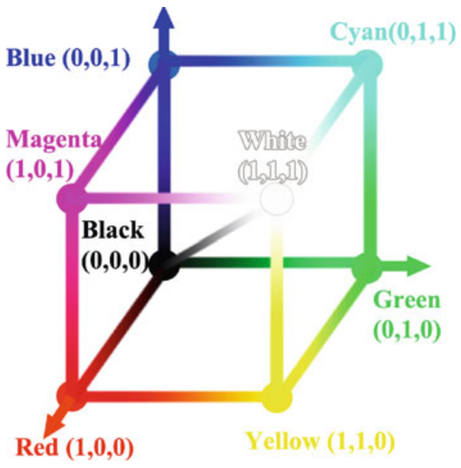


Fig. 4.8 Color cube

The RGB model is based on the fact that color can be viewed as a vector function of three coordinates for each position within the image. Sometimes this model is called the additive model, because the image is obtained by adding the components in primary colors. Each point in the image can be represented by the sum of values of the three primary colors (R, G, B). A size of an RGB digital image depends on how many bits we use for quantization. For example, for $n = 8$ bits, the values range from 0 to 255. In the RGB model, the value 0 (coordinate $= 0$) means the absence of color, while the value 255 (coordinate $= 1$) denotes the color with maximum intensity. Thus, we conclude that (0,0,0) represents black and (1,1,1) represents white. When converting a color image to a grayscale one, the luminescence is calculated as the mean value of the RGB components. By combining two of the three primary colors (R, G, B), we get the colors used in the CMY color model, and white color as a sum of all three colors:

$$\begin{array}{ll} \mathrm{G+B=C(cyan);} & \mathrm{R+B=M(magenta);} \\ \mathrm{R+G=Y(yellow);} & \mathrm{R+G+B=W(white).} \end{array}$$

In Fig. 4.8, the color cube is shown in rectangular coordinates. It illustrates the relative position of the RGB and CMY color model.

4.5.1 CMY, CMYK, YUV, and HSV Color

The coordinate system in the color space can be formed by using three noncollinear color vectors. Thus, if we choose the basis vectors as follows: C – Cyan, M – Magenta and Y – Yellow, the CMY color model is obtained. This model is basically the most commonly used in printers, because the white is obtained by the absence of colors. Even though, black is obtained by combining all three colors, the printers usually have a separate cartridge for the black color. The CMY model including the black color is called the CMYK color model. K is used to refer to the black color. The connection between the CMY and RGB models is evident from the color cube:

$$\mathrm{C = 1 - R, M = 1 - G, Y = 1 - B,} \tag{4.5}$$

while the CMYK model can be obtained as:

$$\mathrm{K = min(C, M, Y),\ C = C - K, M = M - K, Y = Y - K.} \tag{4.6}$$

Another commonly used system is YUV. Here, the color is represented by three components: luminance (Y) and two chrominance components (U and V). The YUV is obtained from the RGB by using the following equations:

$$\begin{aligned} \mathrm{Y} &= 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}, \\ \mathrm{U} &= 0.564(\mathrm{B} - \mathrm{Y}), \\ \mathrm{V} &= 0.713(\mathrm{R} - \mathrm{Y}). \end{aligned} \tag{4.7}$$

It is interesting to note that in the case of R = G = B, we have Y = R = G = B, which is actually the luminance component, while U = 0, V = 0.

Special efforts have been made to define the color systems that are more uniform from the standpoint of perceptual sensitivity, such as *L*u*v** and *L*a*b** systems. Perceptually uniform means that two colors that are equally distant in the color space are equally distant perceptually, which is not the case with the RGB or CMY models (the calculated distance between two colors does not correspond with the perceived difference between the colors). In the *L*a*b** model the perceptual color difference is represented by the Euclidean distance:

$$\begin{aligned}\Delta E^*{}_{ab} &= \left(\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}\right)^{\frac{1}{2}} \quad \text{where}\\ \Delta L^* &= L_1{}^* - L_2{}^*\\ \Delta a^* &= a_1{}^* - a_2{}^*\\ \Delta b^* &= b_1{}^* - b_2{}^* \end{aligned} \tag{4.8}$$

The *L*a*b** model can be obtained from the RGB by the following transformations:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix}, \tag{4.9}$$

$$\begin{aligned} L* &= 25\left(\frac{100Y}{Y_0}\right)^{\frac{1}{3}} - 16,\\ a* &= 500\left[\left(\frac{X}{X_0}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_0}\right)^{\frac{1}{3}}\right],\\ b* &= 200\left[\left(\frac{Y}{Y_0}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_0}\right)^{\frac{1}{3}}\right]. \end{aligned} \tag{4.10}$$

The condition $1 \le 100Y \le 100$ should be satisfied in (4.10). (X_0,Y_0,Z_0) is the value representing reference white. On the basis of this system, we can introduce the HSV color system that is more oriented towards the perceptual model. The HSV color system is represented by a cylindrical coordinate system as shown in Fig. 4.9.

This system is based on the three coordinates: *H*, *S*, and *V*. *H* is a measure of the spectral composition of color, while S provides information about the purity of color, or more accurately, it indicates how far is the color from the gray level, under the same amount of luminescence. *V* is a measure of the relative luminescence. The component *H* is measured by the angle around the *V* axis, ranging from 0 (red) to 360°.

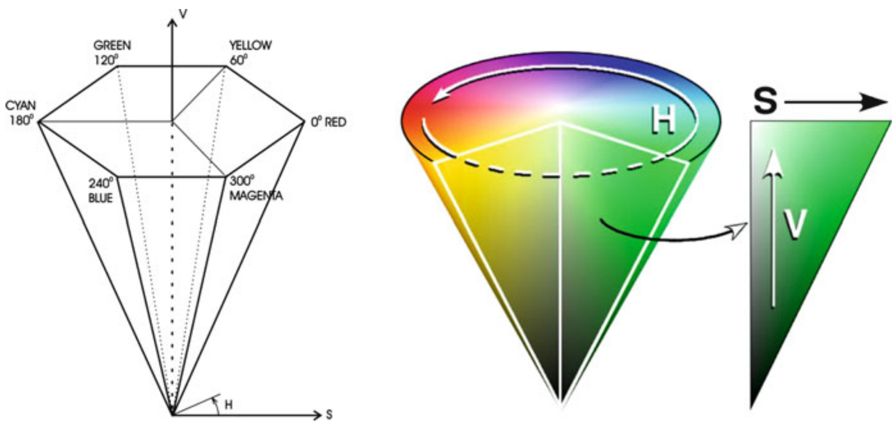


Fig. 4.9 The HSV color model

Along the V axis, the luminance is changed from black to white. The value of the H, S, and V can be defined by using the RGB model as follows:

$$H_1 = \cos^{-1}\left(\frac{\frac{1}{2}[(\mathrm{R}-\mathrm{G})+(\mathrm{R}-\mathrm{B})]}{\sqrt{(\mathrm{R}-\mathrm{G})^2+(\mathrm{R}-\mathrm{B})(\mathrm{G}-\mathrm{B})}}\right), \quad \begin{array}{ll} H = H_1 & \text{for B} \leq \text{G}, \\ H = 360^\circ - H_1 & \text{for B>G}, \end{array} \tag{4.11}$$

$$S = \frac{\max(\mathrm{R},\mathrm{G},\mathrm{B}) - \min(\mathrm{R},\mathrm{G},\mathrm{B})}{\max(\mathrm{R},\mathrm{G},\mathrm{B})}, \tag{4.12}$$

$$V = \frac{\max(\mathrm{R},\mathrm{G},\mathrm{B})}{255}. \tag{4.13}$$

The HSV model is suitable for face detection and tracking algorithms. The thresholds that define the human face color are defined as:

$$\begin{aligned} &340^\circ \leq H \leq 360^\circ \quad \text{and} \quad 0^\circ \leq H \leq 50^\circ, \\ &S \geq 20\%, \\ &V \geq 35\%. \end{aligned} \tag{4.14}$$

Having in mind the coordinate system of this color model, we may observe that the previously given intervals are wide, which may lead to the false detection of the object that actually does not represent the face, but have a similar color information. In order to avoid this possibility, additional analyses are required.

4.6 Filtering

4.6.1 Noise Probability Distributions

Image noise may occur during image transmission over a communication channel. The most common types of noise are impulse noise and Gaussian noise. Impulse noise is manifested as a set of black and white pulses in the image (Fig. 4.10). It occurs as a result of atmospheric discharges, or due to electromagnetic field generated by various appliances.

If impulse noise takes two fixed values: a (negative impulse) and b (positive impulse), with equal probabilities $p/2$, we will have the two-sided impulse noise model. In this case, an image with impulse noise can be defined as:

$$f_I(i,j) = \begin{cases} a, & \text{with a probability } p/2; \\ b, & \text{with a probability } p/2; \\ f(i,j), & \text{with a probability } (1-p). \end{cases} \tag{4.15}$$

Thermal noise is usually modeled as the white Gaussian one and its distribution is given by:

$$P_g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{4.16}$$

where μ is the mean, while σ^2 denotes the variance of noise (σ is the standard deviation of noise). Figure 4.11 demonstrates image “Lena” affected by white Gaussian noise.

Beside the impulse and Gaussian noise, the uniformly distributed noise can appear. The gray level values of the noise are evenly distributed across a specific

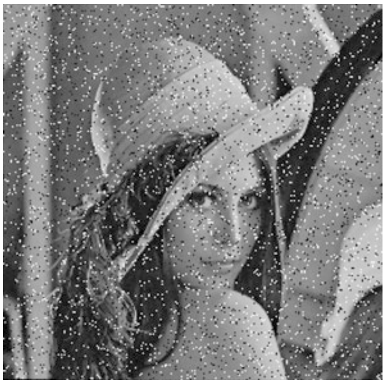


Fig. 4.10 “Lena” affected by an impulse noise with density 0.05

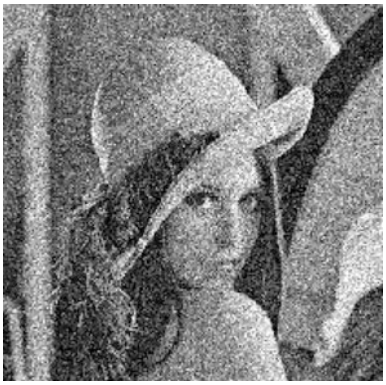


Fig. 4.11 "Lena" affected by zero-mean white Gaussian noise whose variance is equal to 0.02

range. The quantization noise can be approximated by using uniform distribution. The corresponding pdf is defined as:

$$P_u(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \le x \le b \\ 0, & \text{otherwise.} \end{cases} \tag{4.17}$$

The mean value and variance of the uniform density function are:

$$\mu = (a+b)/2 \text{ and } \sigma^2 = (b-a)^2/12, \text{ respectively.}$$

Radar images may contain noise characterized by the Rayleigh distribution:

$$P_{\mathrm{R}}(x) = \begin{cases} \frac{2}{\beta}(x-\alpha)e^{-(x-\alpha)^2/\beta}, & \text{for } x \ge \alpha \\ 0, & \text{otherwise,} \end{cases} \tag{4.18}$$

with the mean equal to $\mu = \alpha + \sqrt{\pi\beta/4}$ and the variance $\sigma^2 = \beta(4-\pi)/4$.

4.6.2 *Filtering in the Spatial Domain*

Filtering of noisy images intends to reduce noise and to highlight image details. For this purpose, the commonly used filters in the spatial domain are the mean and median filters. Use of these filters depends on the nature of the noise that is present within the image. Spatial domain filters are especially suitable in the cases when additive noise is present.

4.6.2.1 Mean Filter

Mean filters are used to filter the images affected by the Gaussian white noise, since it is based on calculating the average pixel intensity within an image part captured by a specified window. The filter should use a small number of points within the

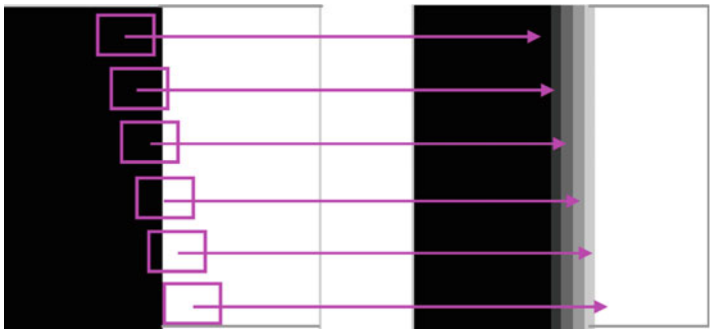


Fig. 4.12 Illustration of blurring after applying mean filter on the image edge (the mask size used is 5 × 5)

window to avoid blurring of image details. Note that a larger window would provide better noise filtering.

Consider a window of the size $(2N_1 + 1) \times (2N_2 + 1)$. The signal $f(i,j)$ is affected by the noise $n(i,j)$ and the noisy signal is:

$$x(i,j) = f(i,j) + n(i,j). \tag{4.19}$$

The output of the arithmetic mean filter is defined by the relation:

$$x_f(i,j) = \frac{1}{(2N_1+1)(2N_2+1)} \sum_{n=i-N_1}^{i+N_1} \sum_{m=j-N_2}^{j+N_2} x(n,m), \tag{4.20}$$

where $x(n,m)$ is the pixel value within the window, while the impulse response of the filter is $h(i,j) = 1/((2N_1+1)(2N_2+1))$. The output of this filter is actually the mean value of pixels captured by the window. For a window size 3 × 3, we deal with 9 points, while the 5 × 5 window includes 25 points. From the aspect of noise reduction, the second window will be more effective. However, it will introduce more smoothed edges and blurred image (Fig. 4.12).

As an example, "Lena" image affected by a zero-mean Gaussian noise with variance 0.02 and its filtered versions are shown in Fig. 4.13.

Instead of the arithmetic mean filter, the geometric mean can be used as well, where the filter output is given by:

$$x_f(i,j) = \left(\prod_{n=i-N_1}^{i+N_1} \prod_{m=j-N_2}^{j+N_2} x(n,m) \right)^{\frac{1}{(2N_1+1)(2N_2+1)}}. \tag{4.21}$$

The geometric mean filter introduces less blurring and preserves more image details (Figs. 4.14 and 4.15).

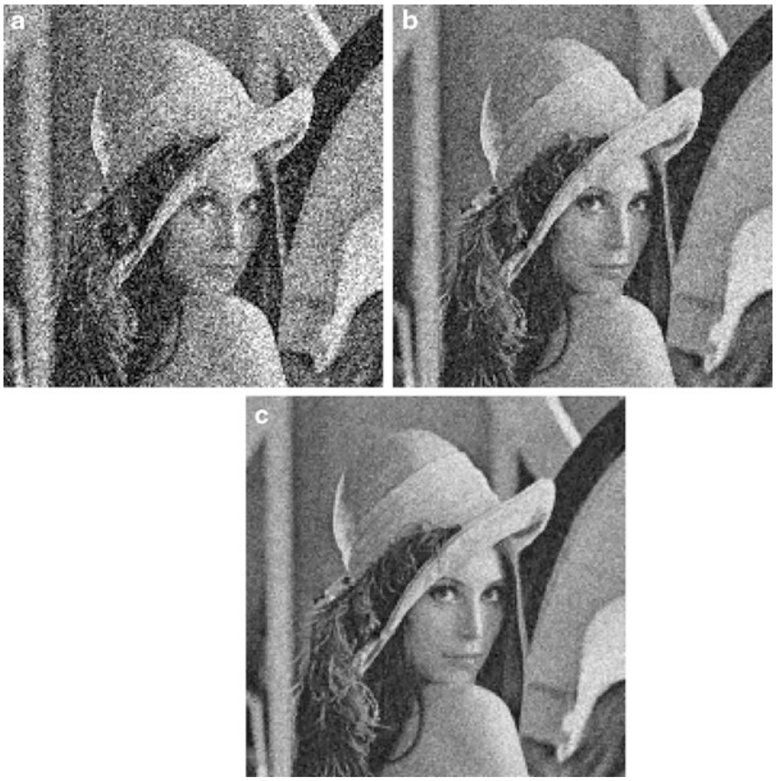


Fig. 4.13 (**a**) “Lena” affected by Gaussian noise with zero mean and variance 0.02, (**b**) filtered image obtained by using mean filter of size 3 × 3, (**c**) filtered image obtained by using mean filter of size 5 × 5

4.6.2.2 Median Filter

Median filters are used to filter out the impulse noise. Consider a sequence with an odd number of elements. After sorting the elements, the median value is obtained as the central element. In a sequence with an even number of elements, the median is calculated as the mean of two central elements of the sorted sequence.

Example

The sequence is given as:

$$3 \quad 14 \quad 7 \quad 1 \quad 5$$

Sort the numbers in the ascending order:

$$1 \quad 3 \quad \underline{\mathbf{5}} \quad 7 \quad 14$$

and then the median is central element 5.

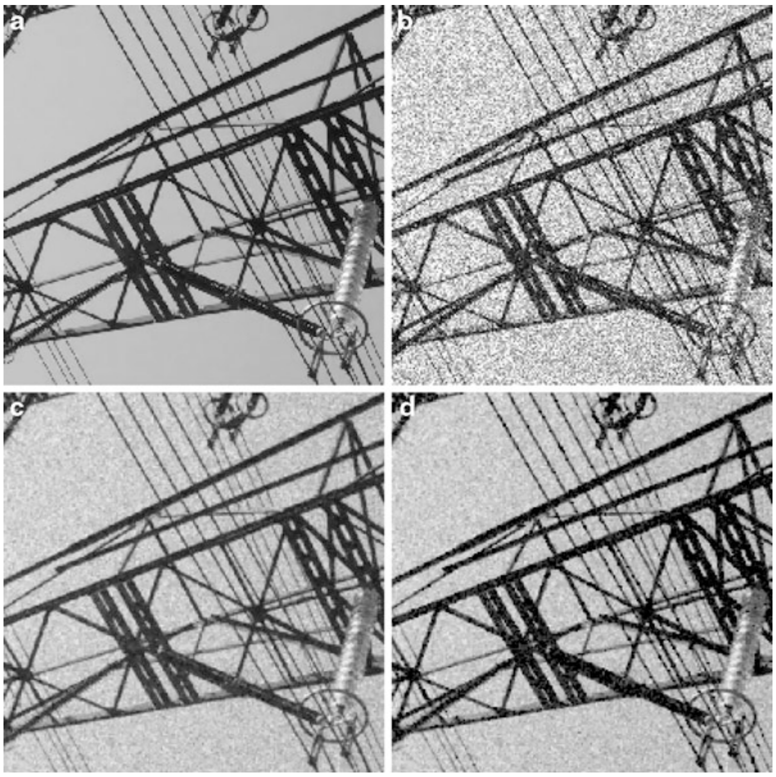


Fig. 4.14 (**a**) Original image, (**b**) noisy image (Gaussian noise with 0.05 mean and variance 0.025), (**c**) image filtered by using arithmetic mean of size 3 × 3, (**d**) image filtered by using geometric mean of size 3 × 3

Consider now a sequence with an even number of elements:

1 12 7 4 9 2

Sort the elements in ascending order:

1 2 $\underline{\mathbf{4}}$ $\underline{\mathit{7}}$ *9 12*

4 and 7 are the two central elements, and the median is equal to their mean value, or 5.5.

The median filter is applied in image denoising by using a rectangular window that slides over the entire image. The elements captured by the window are reordered as a vector $\mathbf{x}$:$\{x(k), k\in[1,N]\}$ and then the median value x_m for the vector is calculated as follows:

$$\begin{aligned} x_m &= \operatorname{med}(x(1),\ldots,x(k),\ldots,x(N)) \\ &= \begin{cases} x_s(\lfloor N/2 \rfloor + 1), & N \text{ is odd,} \\ \frac{x_s(N/2)+x_s(N/2+1)}{2}, & N \text{ is even,} \end{cases} \end{aligned} \tag{4.22}$$

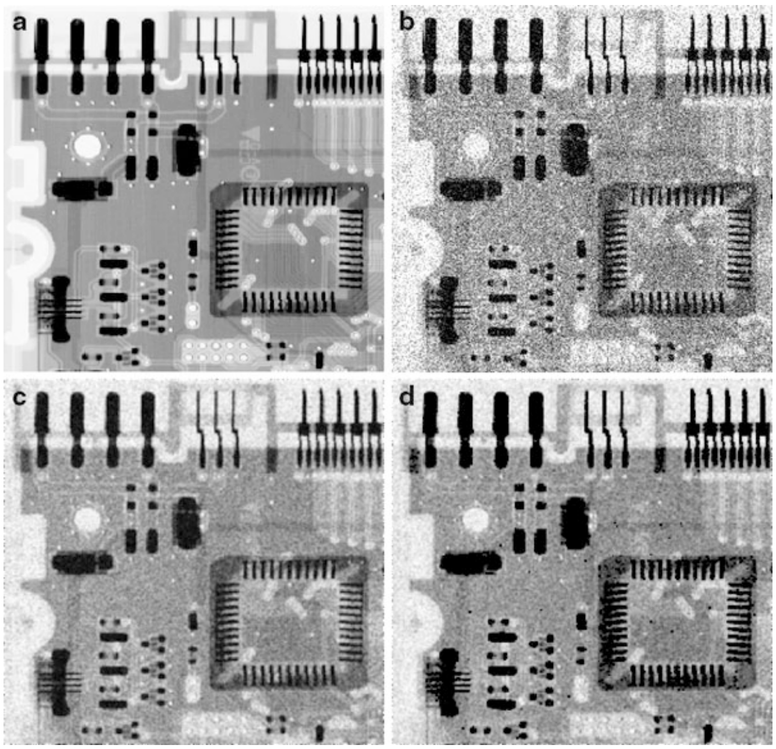


Fig. 4.15 (**a**) Original image, (**b**) noisy image (Gaussian noise with 0.05 mean and variance 0.025), (**c**) image filtered by using arithmetic mean of size 3 × 3, (**d**) image filtered by using geometric mean of size 3 × 3

where x_s is the sorted version of **x**. Another way to calculate the median of a matrix is to calculate the median value for columns and then for rows (or vice versa). Generally, these two approaches usually do not produce exactly the same result.

Suppose that the filter window covers $(2\,N_1 + 1)(2\,N_2 + 1)$ pixels. The pixel $x(i, j)$ is the central one in the filter window. From all the pixels within the window, we form a matrix:

$$\begin{bmatrix} x(i-N_1, j-N_2) & \ldots & x(i-N_1, j) & \ldots & x(i-N_1, j+N_2) \\ \vdots & & \vdots & & \vdots \\ x(i, j-N_2) & \ldots & x(i, j) & \ldots & x(i, j+N_2) \\ \vdots & & \vdots & & \vdots \\ x(i+N_1, j-N_2) & \ldots & x(i+N_1, j) & \ldots & x(i+N_1, j+N_2) \end{bmatrix}$$

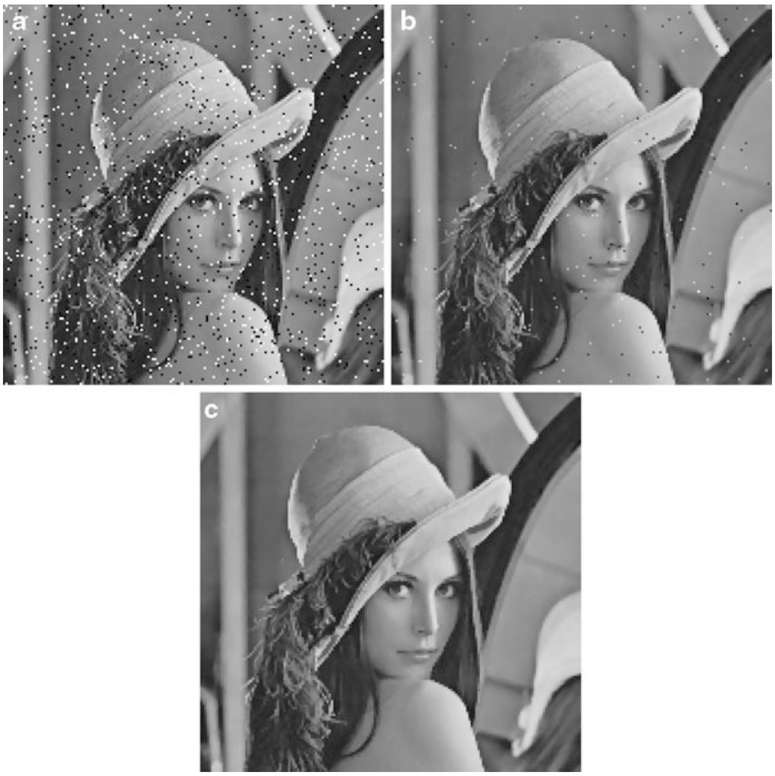


Fig. 4.16 (**a**) "Lena" affected by impulse noise with density 0.05, (**b**) denoised image (median filter of size 3 × 3 is used), (**c**) denoised image (5 × 5 median filter)

The first step is to sort the elements within columns and to determine the median for each column. The second step uses the median values of columns and calculates the median again. Mathematically, it is described as:

$$\begin{aligned} q_n &= \mathrm{med}\{x(i-N_1,j), x(i-N_1+1,j), \ldots, x(i+N_1,j)\}, \quad \text{for } n=j, \\ q(i,j) &= \mathrm{med}\{q_n; n \in (j-N_2, \ldots, j+N_2)\}. \end{aligned} \tag{4.23}$$

Therefore, $q(i,j)$ represents the output of separable median filter. An application of the median filter to image "Lena" affected by the impulse noise is illustrated in Fig. 4.16.

The α-trimmed mean filter has been introduced as a good compromise between the median and arithmetic mean filter. Namely, after sorting the windowed pixels, we discard a few lowest and highest samples, while the remaining pixels are averaged. The α-trimmed mean filter can be defined as:

$$x_\alpha(i,j) = \frac{1}{(N-2[\alpha N])} \sum_{n=[\alpha N]+1}^{N-[\alpha N]} x_s(n), \tag{4.24}$$

where $x_s(n)$ is the vector of sorted pixels from the window $N_1 \times N_2$, $N = N_1N_2$, $[\alpha N]$ denotes rounding to the greatest integer not greater than αN. The parameter α takes the values from the range: $0 \le \alpha < 0.5$. Note that this filter form corresponds to the median for $\alpha = 0.5$ (for odd N), while for $\alpha = 0$ it performs as a moving average filter. Alternatively, we can apply the same operation separately on the rows and columns as follows:

$$x_\alpha(i,j) = \frac{1}{(N_1 - 2[\alpha N_1])(N_2 - 2[\alpha N_2])} \sum_{n=[\alpha N_1]+1}^{N_1-[\alpha N]} \sum_{m=[\alpha N_2]+1}^{N_2-[\alpha N]} x(m,n). \tag{4.25}$$

4.6.3 Filtering in the Frequency Domain

Filters in the frequency domain are designed on the basis of a priori knowledge about signal frequency characteristics. The most significant frequency content of images is mostly concentrated at low frequencies. Therefore, in many applications, the images are usually filtered with low-pass filters. The ideal rectangular separable low-pass filter has the following transfer function:

$$H(\omega_1,\omega_2) = \begin{cases} 1, & |\omega_1| \le W_1 \text{ and } |\omega_2| \le W_2 \\ 0, & \text{otherwise.} \end{cases} \tag{4.26}$$

A band-pass filter can be defined as:

$$H(\omega_1,\omega_2) = \begin{cases} 1, & W_{11} \le |\omega_1| \le W_{12},\ W_{21} \le |\omega_2| \le W_{22} \\ 0, & \text{otherwise.} \end{cases} \tag{4.27}$$

In addition to rectangular, a circular low-pass filter can be used:

$$H(\omega_1,\omega_2) = \begin{cases} 1, & \omega_1^2 + \omega_2^2 \le W, \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

Filtering images with a high-pass filter provides high-frequency components that contain the image details:

$$H(\omega_1,\omega_2) = \begin{cases} 1, & (|\omega_1| > W_1 \text{ and } |\omega_2| > W_2) \text{ or } \left(\omega_1^2 + \omega_2^2 > W\right) \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

4.6.4 Image Sharpening

A blurred noisy image in the Fourier domain can be written as:

$$X(u,v) = H(u,v)F(u,v) + N(u,v), \tag{4.30}$$

where $X(u,v)$ is the Fourier transform of blurred image, $H(u,v)$ is the impulse response of the system that induces blurring (degradation), $F(u,v)$ is the Fourier transform of the original image, and $N(u,v)$ is the Fourier transform of noise. For example, if the blurring is produced by the uniform linear motion between the image and the sensor (during image acquisition) along the x axis, then the degradation function can be defined by:

$$H(u,v) = T\frac{\sin(\pi nu)}{\pi nu}e^{-j\pi nu}, \tag{4.31}$$

where n is the distance of pixels displacement, while T is the duration of the exposure. Sharpening of the image is achieved based on the following relation:

$$F(u,v) = \frac{X(u,v)}{H(u,v)}. \tag{4.32}$$

4.6.5 Wiener Filtering

The Wiener filter is defined in the theory of optimal signal estimation. It is based on the equation:

$$f_e(i,j) = L[f(i,j)], \tag{4.33}$$

where L is a linear operator meaning that the estimated values are a linear function of the original (degraded) values. The estimated values are obtained such that the mean square error:

$$E\left\{(f(i,j) - f_e(i,j))^2\right\}, \tag{4.34}$$

is minimized. The Wiener filter in the frequency domain is obtained in the form:

$$H_w(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_n(u,v)}{S_f(u,v)}}, \tag{4.35}$$

where $S_n(u,v)$ and $S_f(u,v)$ represent the power spectrum of the noise and the signal, respectively: $S_n(u,v) = |N(u,v)|^2$, $S_f(u,v) = |F(u,v)|^2$. $H^*(u,v)$ is the complex conjugate of the degradation function and for $H^*(u,v) = 1$, (4.35) becomes:

$$H_w(u,v) = \frac{S_f(u,v)}{S_f(u,v) + S_n(u,v)} = 1 - \frac{S_n(u,v)}{S_x(u,v)}. \tag{4.36}$$

It is assumed that the signal and noise are uncorrelated: $S_x(u,v) = S_f(u,v) + S_n(u,v)$. Note that when S_n tends to zero, the filter function is approximately equal to 1 (no modification of the signal), while in the case when S_f tends to zero, the filter function is zero. Thus, the filtered spectrum is:

$$F_e(u,v) = H_w(u,v)X(u,v), \tag{4.37}$$

where $X(u,v)$ is the spectrum of noisy image. The noise measurement should be performed when the signal is not present (e.g., consider a communication channel without signal during an interval of time), and then the estimated noise spectrum is available for calculating $H_w(u,v)$.

4.7 Enhancing Image Details

An image can be represented in terms of its illumination and reflectance components:

$$a(i,j) = a_i(i,j)a_r(i,j), \tag{4.38}$$

where a_i is the illumination describing the amount of incident light on the observed scene, while a_r is the reflectance component describing the amount of light reflected by the objects. It is usually assumed that the scene illumination varies slowly over space, while the reflectance varies rapidly especially on the transitions between different objects. Hence, the illumination and the reflectance components are associated with low and high frequencies, respectively. Usually, the goal is to extract the reflectance component and to minimize the illumination effect, which can be done by using the logarithm to transform multiplicative into additive procedure:

$$\log(a(i,j)) = \log(a_i(i,j)) + \log(a_r(i,j)). \tag{4.39}$$

Having in mind their frequency characteristics, we can separate these two components of images and emphasize the details.

4.8 Analysis of Image Content

The distribution of colors and textures are considered as two important features for the analysis of image content.

4.8.1 The Distribution of Colors

The distribution of colors can be described by using histogram. If we want to search an image database, we can achieve it by comparing the histogram of the sample image Q and the histogram of each image I from the database. Suppose that both histograms have N elements. Comparison is done by calculating the total number of pixels that are common to both histograms:

$$S = \sum_{i=1}^{N} \min(I_i, Q_i). \tag{4.40}$$

This amount is often normalized by the total number of pixels in one of the two histograms. Having in mind that this method is computationally demanding, the modified forms have been considered. Namely, by using a suitable color model, an image can retain its relevant properties even with a coarser representation. Hence, a significant computational savings can be achieved.

A computationally efficient method for comparison of color images can be obtained if colors are represented by fewer bits. For example, if each color is reduced to 2 bits, then we have 64 possible combinations in the case of three colors.

The colorfulness of images can be described by using the color coherence vectors. Assume that the total number of colors is N, the color coherence vectors for images Q and I are given by:

$$\left[\left(\alpha_1^Q, \beta_1^Q\right), \ldots, \left(\alpha_N^Q, \beta_N^Q\right)\right] \text{ and } \left[\left(\alpha_1^I, \beta_1^I\right), \ldots, \left(\alpha_N^I, \beta_N^I\right)\right],$$

where α_i and β_i represent the number of coherent and incoherent pixels for color i, respectively. Coherent pixels are those that belong to a region characterized by the same color. A difference between two images can be calculated by using the following formula:

$$\operatorname{dist}(Q, I) = \sum_{i=1}^{N} \left(\left| \frac{\alpha^Q{}_i - \alpha^I{}_i}{\alpha^Q{}_i + \alpha^I{}_i + 1} \right| + \left| \frac{\beta^Q{}_i - \beta^I{}_i}{\beta^Q{}_i + \beta^I{}_i + 1} \right| \right). \tag{4.41}$$

4.8.2 Textures

A texture is an important characteristic of the image surface. There are different methods and metrics for texture analysis. For instance, the textures can be described by using the following properties: contrast, directionality, and coarseness.

The *contrast* can be quantified by the statistical distribution of pixel intensities. It is expressed as:

$$\mathrm{Con} = \frac{\sigma}{K^{1/4}}, \tag{4.42}$$

where σ is the standard deviation, K is the kurtosis, defined by:

$$K = \frac{\mu_4}{\sigma^4}, \tag{4.43}$$

where μ_4 is the fourth moment about the mean. The presented definition is a global measure of the contrast obtained for the entire image.

Coarseness represents a measure of texture granularity. It is obtained as the mean value calculated over windows of different sizes $2^k \times 2^k$, where k is usually between 1 and 5. The windowing and averaging is done for each image pixel. Consider a pixel at the position (x,y). The mean value within the window of size $2^k \times 2^k$ is defined as:

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} \frac{a(i,j)}{2^{2k}}, \tag{4.44}$$

where $a(i,j)$ is the grayscale pixel value at the position (i,j). Then the differences between mean values in the horizontal and vertical directions are calculated as follows:

$$\begin{aligned} D_{kh} &= \left|A_k\left(x+2^{k-1},y\right) - A_k\left(x-2^{k-1},y\right)\right|, \\ D_{kv} &= \left|A_k\left(x,y+2^{k-1}\right) - A_k\left(x,y-2^{k-1}\right)\right|. \end{aligned} \tag{4.45}$$

Using the above equations, we choose the value of k, which yields the maximum values for D_{kh} and D_{kv}. The selected k is used to calculate the *optimization parameter*:

$$g(x,y) = 2^k. \tag{4.46}$$

Finally, the measure of granulation can be expressed in the form:

$$\mathrm{Gran} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} g(i,j). \tag{4.47}$$

In order to reduce the number of calculations, the measure of granularity can be calculated for lower image resolution.

Directionality is the third important texture feature. As a measure of directionality, at each pixel we calculate a gradient vector, whose amplitude and angle are given as:

$$\begin{aligned} |\Delta G| &= (|\Delta_H| + |\Delta_V|)/2, \\ \varphi &= \arctan\left(\frac{\Delta_V}{\Delta_H}\right) + \frac{\pi}{2}, \end{aligned} \tag{4.48}$$

where horizontal Δ_H and vertical Δ_V differences are calculated over 3×3 window around a pixel. After determining the above parameters for each pixel, we can draw a histogram of angle values φ, taking only those pixels where $|\Delta G|$ is larger than a given threshold. The resulting histogram will have dominant peaks for highly directional images, while for nondirectional images it will be flatter.

4.8.3 Co-occurrence Matrix

A simplified method to measure the contrast of textures can be performed by using the co-occurrence matrices. First, we form the co-occurrence matrices, that show how many times *y* values appear immediately after the *x* values. For example, consider the following sample matrix:

1	1	1	2	3
1	1	1	2	3
1	1	1	2	3
1	1	1	3	4

The corresponding co-occurrence matrix is then obtained as:

x y	1	2	3	4
1	8	3	1	0
2	0	0	3	0
3	0	0	0	1
4	0	0	0	0

Let us analyze the numbers in the matrix. The number 8 means that 1 occurs eight times after 1, while 3 denotes that value 2 occurs three times after 1. The expression for the measure of the texture contrast is given by:

$$\mathrm{Con} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} (x-y)^2 c(x,y), \tag{4.49}$$

where $c(x,y)$ represents the co-occurrence matrix of size $N \times N$. If there are significant variations in the image, $c(x,y)$ will be concentrated outside the main diagonal and contrast measures will have greater values. The co-occurrence matrix with values concentrated on its diagonal corresponds to a homogeneous region.

There are other useful features that can be computed from the co-occurrence matrix, as listed below:

$$\begin{aligned}
&\text{Energy :} && \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} c^2(x,y),\\
&\text{Entropy :} && -\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} c(x,y)\log_2 c(x,y),\\
&\text{Homogeneity :} && \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \frac{c(x,y)}{1+|x-y|},\\
&\text{Correlation :} && \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \frac{(x-\mu_x)(y-\mu_y)c(x,y)}{\sigma_x\sigma_y}.
\end{aligned}$$

4.8.4 Edge Detection

Edge detection plays an important role in a number of applications. Consider an image with pixels $a(i,j)$. Edges of the image should be obtained by simple differentiation. However, bearing in mind that the image is always more or less affected by noise, the direct application of differentiation is not effective. For this purpose, several algorithms have been defined, and among them the most commonly used one is based on the Sobel matrices (for vertical and horizontal edge). Specifically, the image is analyzed pixel by pixel using the Sobel matrix as a mask. The matrix elements are the weights that multiply the pixels within the mask. Then the sum is calculated by adding all the obtained values. The resulting value is compared with a threshold. If it is greater than the threshold, the central pixel belongs to the edge, and vice versa.

The Sobel matrices for vertical and horizontal edges are given by:

$$S_v = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The edges are obtained by:

$$L(i,j) = \sum_{m=-1}^{1}\sum_{n=-1}^{1} a(i+m,j+n)S(m+2,n+2), \tag{4.50}$$

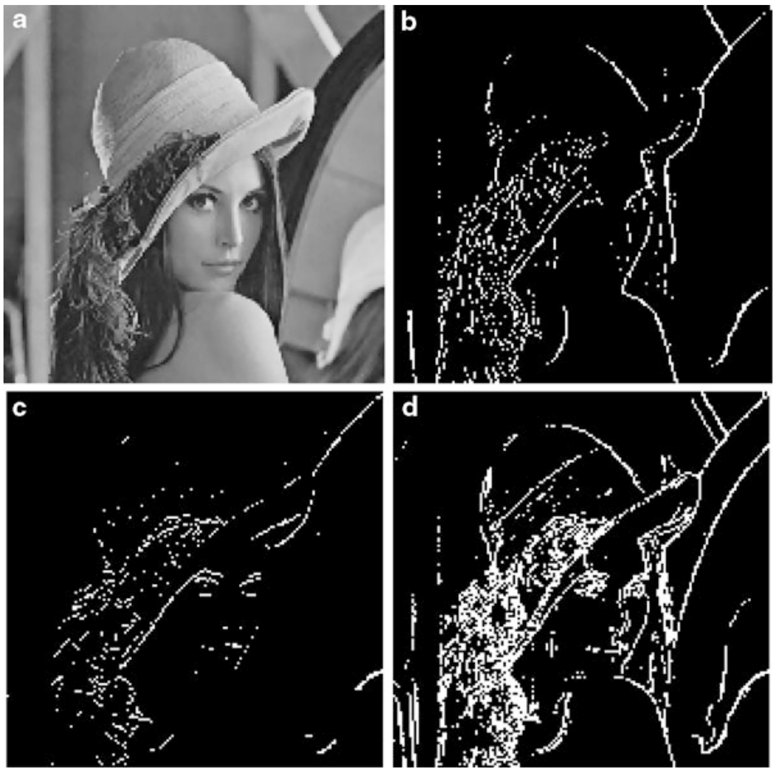


Fig. 4.17 Illustration of edge detection: (**a**) Original image, (**b**) L_v, (**c**) L_h, (**d**) L

where $S(m,n)$ is a filtering function (e.g., the Sobel matrix S_h or S_v). After calculating L_h and L_v (using the horizontal and vertical matrix), the overall L is calculated as follows:

$$L = \sqrt{{L_h}^2 + {L_v}^2}. \tag{4.51}$$

The obtained values (for all pixels) are compared with a threshold and the results are represented in a binary form. An example of edge detection is illustrated in Fig. 4.17. For simplicity, the threshold was set to 100 for the entire image.

However, the local threshold values are frequently used in practical applications. They are calculated based on the mean response of the edge detector around the current pixel. For example, a threshold value can be calculated as:

$$T(i,j) = \bar{L}(i,j)(1+p) = \frac{1+p}{2N+1} \sum_{k=i-N}^{i+N} \sum_{l=j-N}^{j+N} L(k,l), \tag{4.52}$$

where p has a value between 0 and 1.

4.8.5 The Condition of the Global Edge (Edge-Based Representation: A Contour Image)

An algorithm for edge-based image representation is described in the sequel. First, the image is normalized by applying the affine transformation which results in the square image of the size 64 × 64. Then the gradient is calculated for each pixel:

$$\partial_{i,j} = \frac{|\Delta p_{i,j}|}{|I_{i,j}|}, \tag{4.53}$$

where the numerator represents the maximum of the differences between the intensity of a given pixel and the intensity of its neighbors. The denominator is the local power of intensity values. The calculated gradient is compared with the sum of the mean and the variance of original image:

$$\partial_{i,j} \geq \mu + \sigma. \tag{4.54}$$

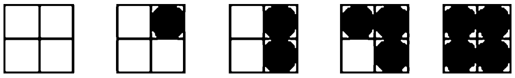
Pixels that fulfill the condition (4.54) are called the global edge candidates. Now, from pixels selected as the global edge candidates, we reselect the pixels for which:

$$\partial_{i,j} \geq \mu_{i,j} + \sigma_{i,j}, \tag{4.55}$$

holds, where μ and σ are mean and variance, respectively, of the local gradient to its neighbors. They are called the local edge candidates.

4.8.6 Dithering

One of the properties of the human eye is that when observing a small area from a long distance, it perceives just the overall intensity as a result of averaging granular details. This feature is used in dithering, where a group of points represents a color. Consider a simple example by using four points:



We see that with only two values, we can create five different colors (from pure white to pure black). If this 2 × 2 structure is used with the three primary colors, we can get 125 color combinations.

4.9 Image Compression

Multimedia information is very demanding on the memory space and usually needs much processing power. Additionally, it may require higher bit rates compared to the available bit rates of communication channels. All of these aspects lead to the inevitable use of compression algorithms. As already mentioned, data compression can be performed as lossless compression and lossy compression. This section considers compression algorithms for digital images. Special attention will be devoted to JPEG and JPEG2000 compression.

4.9.1 JPEG Image Compression Algorithm

Note that the JPEG algorithm can achieve significant compression ratio while maintaining high image quality. Therefore in this chapter, we will discuss in details the elements of JPEG encoder. JPEG algorithm can be analyzed across several blocks used for image compression. These blocks can be summarized as follows: a block performing DCT on the 8 × 8 image blocks, quantization block, zigzag matrix scanning, and an entropy coding block (Fig. 4.18).

The DCT of an 8 × 8 image block is defined by:

$$\mathrm{DCT}(k_1,k_2)=\frac{C(k_1)}{2}\frac{C(k_2)}{2}\times\sum_{i=0}^{7}\sum_{j=0}^{7}a(i,j)\cos\left(\frac{(2i+1)k_1\pi}{16}\right)\cos\left(\frac{(2j+1)k_2\pi}{16}\right)\tag{4.56}$$

where:

$$C(k_1)=\begin{cases}\frac{1}{\sqrt{2}}, & \text{for } k_1=0\\ 1, & \text{for } k_1>0\end{cases},\quad C(k_2)=\begin{cases}\frac{1}{\sqrt{2}}, & \text{for } k_2=0\\ 1, & \text{for } k_2>0\end{cases}.$$

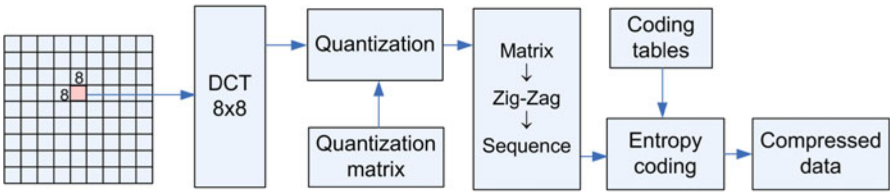


Fig. 4.18 JPEG encoder block diagram

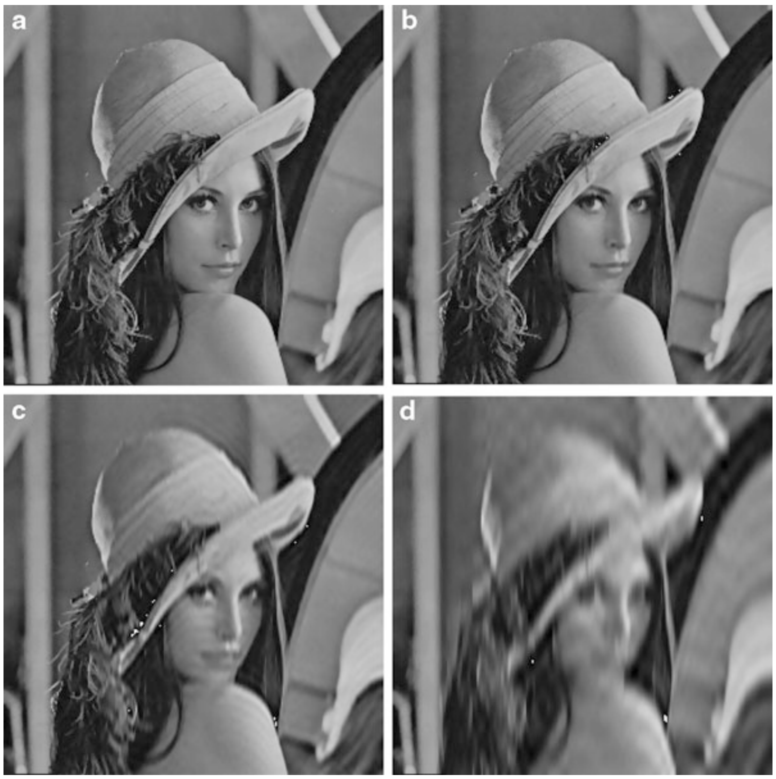


Fig. 4.19 (**a**) Original image "Lena" (**b**) image based on the first 128 × 128 DCT coefficients, (**c**) image based on the first 64 × 64 DCT coefficients, (**d**) image based on the first 25 × 25 DCT coefficients

The DCT coefficient (0,0) is called the DC component, and it carries an information about the mean value of 64 coefficients. The remaining 63 coefficients are the AC coefficients.

The samples of the grayscale image whose values are in the range $[0, 2^n - 1]$ (n is number of bits used to represent samples), are shifted to the range $[-2^{n-1}, 2^{n-1} - 1]$, and then the DCT is applied. Hence, in the case of 8-bit samples, the shifted range is $[-128, 127]$. The corresponding DCT coefficients will be in the range $[-1024, 1023]$ and they require additional 3 bits. To encode the DC coefficient of a current block, we subtract its value from the DC coefficient in the previous block, and then encode their difference.

Before introducing the quantization matrix, let us show that the most important transform coefficients of images are concentrated at low frequencies. For this purpose, we will analyze the image "Lena" (of the size 256 × 256 pixels). After applying the DCT, we take the first 128 × 128 coefficients, then the first 64 × 64 coefficients, and finally the first 25 × 25 coefficients. Applying the inverse DCT, we reconstruct the images shown in Fig. 4.19. Note that, although the number of coefficients is significantly decreased, the image retains much of the information.

Fig. 4.20 Quantization matrix obtained for *quality* = 2

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

The previous fact was extensively investigated in order to determine the optimal block size and to provide the efficient energy compaction within the smallest number of coefficients. Even though the 32 × 32 and 16 × 16 blocks slightly improve the coding gain compared to the 8 × 8 blocks, the JPEG compression still uses 8 × 8 blocks due to a quite easier calculation. Namely, the 8 × 8 blocks provides an optimal trade-off between the computational complexity, prediction gain and energy compaction with as smallest artifacts as possible. Therefore, the algorithm for JPEG image compression first decomposes an image into 8 × 8 blocks. Next, the DCT is calculated for each 8 × 8 block. The DCT coefficients are divided by weighting coefficients, representing the elements of quantization matrix. Therefore, we have:

$$\mathrm{DCT}_q(k_1,k_2)=\mathrm{round}\left\{\frac{\mathrm{DCT}(k_1,k_2)}{Q(k_1,k_2)}\right\}, \tag{4.57}$$

where Q is a quantization matrix, while DCT_q are the quantized coefficients. A simplified example for calculating coefficients of a matrix that can be used for quantization is given by the following code:

```
$for\ i = 0 : N-1$
    $for\ j = 0 : N-1$
$Q(i+1, j+1) = 1 + [(1+i+j)^* quality];$
    $end$
$end$
```

The *quality* parameter ranges from 1 to 25. Higher values denote better compression, but worse image quality. The compression matrix for *quality* = 2 is given in Fig. 4.20.

In practical applications, the quantization matrices are derived from the experimental quantization matrix given in Fig. 4.21. The experimental quantization matrix is defined for the 50% compression ratio (quality factor − QF = 50).

Fig. 4.21 Coefficients of the quantization matrix Q_{50}

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

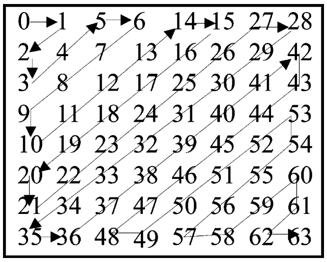


Fig. 4.22 Zigzag reordering

Using the matrix Q_{50}, we can obtain matrices for other compression degrees as follows:

$$Q_{\mathrm{QF}} = \mathrm{round}(Q_{50} \cdot q), \tag{4.58}$$

where q is defined as:

$$q = \begin{cases} 2 - 0.02\mathrm{QF}, & \text{for QF} \geq 50, \\ \frac{50}{\mathrm{QF}}, & \text{for QF}<50. \end{cases}$$

The DCT coefficients of 8×8 blocks are divided by the corresponding coefficients of quantization matrices and rounded to the nearest integer values.

After quantization, the zigzag reordering is applied to the 8×8 matrix to form a vector of 64 elements. This reordering allows the values to be sorted from the low-frequency coefficients toward the high-frequency ones. A schematic of zigzag reordering is shown in Fig. 4.22.

Next, the entropy coding is applied based on the Huffman coding. Each AC coefficient is encoded with two symbols. The first symbol is defined as: $(a,b) = (runlength,size)$. The *runlength* provides the information about the number of consecutive zero coefficients preceding the non-zero AC coefficient. Since it is encoded with 4 bits, it can be used to represent no more than 15 consecutive zero coefficients. Hence, the symbol (15,0) represents 16 consecutive zero AC coefficients and it can be up to three (15,0) extensions. This symbol also contains information on the number of bits required to represent the coefficient value (*size*). The second symbol is the amplitude of the coefficient (which is in the range $[-1023, 1024]$) that can be represented with up to 10 bits.

Table 4.1 Encoding of the coefficients amplitudes

Amplitude range	Size
−1,1	1
−3,−2,2,3	2
−7,−6,−5,−4,4,5,6,7	3
−15,. . .,−8,8,. . .,15	4
−31,. . .,−16,16,. . .,31	5
−63,. . .,−32,32,. . .,63	6
−127,. . .,−64,64,. . .,127	7
−255,. . .,−128,128,. . .,255	8
−511,. . .,−256,256,. . .,511	9
−1,023,. . .,−512,512,. . .,1,023	10

For example, if we have the following sequence of coefficients: 0,0,0,0,0,0,0,239, we code it as: (7,8) (239).

The symbol (0,0) denotes the end of the block (EOB).

Since there is a strong correlation between the DC coefficients from adjacent blocks, the differences between DC coefficients are coded instead of their values. The DC coefficients are in the range $[-2048\ ,\quad 2047]$ and are coded by two symbols: the first symbol is the number of bits (*size*) used to represent the amplitude, while the second symbol is the amplitude itself.

The amplitude for both DC and AC coefficients are encoded by using the variable-length integer code, as shown in the Table 4.1.

Consider an example of JPEG compression applied to the 8×8 block. The pixel values (Fig. 4.23a) from range [0,255] are initially shifted to range [−128,127] (Fig. 4.23b). The values of DCT coefficients are shown in Fig. 4.23c. Note that, for the sake of simplicity, the DCT coefficients are rounded to the nearest integers. The quantization matrix is used with a *quality factor* QF = 70% (Fig. 4.23d) and the quantized DCT coefficients are shown in Fig. 4.23e.

The zigzag sequence is obtained in the form:

18, −1, 1, −1, 4, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, −1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

and by using symbols for DC and AC coefficients, we obtain the intermediate symbol sequence:

(5)(18), (0,1)(−1), (0,1)(1), (0,1)(−1) (0,3)(4), (0,3)(4), (1,1)(1), (10,2)(2), (1,1)(1), (3,1)(1), (0,1)(1), (6,1)(−1), (0,0)

The symbols for AC components ((*a*,*b*) = (*runlength*,*size*)) are coded by using the Huffman tables, specified by the JPEG standard and given at the end of this chapter (for luminance component). The symbols used in this example are provided in the Table 4.2 (the list of all symbols is given in Table 4.4).

a

177	153	151	143	142	145	150	158
159	148	148	145	144	153	147	149
154	155	151	154	157	146	156	146
152	159	157	155	151	147	159	151
142	154	144	143	144	144	168	156
160	150	136	136	148	153	161	153
150	142	142	142	156	152	161	161
135	131	150	154	151	144	146	160

b

49	25	23	15	14	17	22	30
31	20	20	17	16	25	19	21
26	27	23	26	29	18	28	18
24	31	29	27	23	19	31	23
14	26	16	15	16	16	40	28
32	22	8	8	20	25	33	25
22	14	14	14	28	24	33	33
7	3	22	26	23	16	18	32

c

180	-10	21	4	3	6	-6	8
10	28	9	4	-2	3	3	-3
-7	0	2	0	15	1	19	0
0	0	26	14	-3	0	-1	0
0	1	0	-17	0	3	0	0
13	5	3	3	3	0	-2	4
0	6	1	3	4	1	-2	1
0	-6	0	-3	0	0	-7	4

d

10	7	6	10	14	24	31	37
7	7	8	11	16	35	36	33
8	8	10	14	24	34	41	34
8	10	13	17	31	52	48	37
11	13	22	34	41	65	62	46
14	21	33	38	49	62	68	55
29	38	47	52	62	73	72	61
43	55	57	59	67	60	62	59

e

18	-1	4	0	0	0	0	0
1	4	1	0	0	0	0	0
-1	0	0	0	1	0	0	0
0	0	2	1	0	0	0	0
0	0	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig. 4.23 (**a**) 8 × 8 image block, (**b**) values of pixels after shifting to the range [−128, 127], (**c**) DCT coefficients (rounded to integers) for the given block, (**d**) quantization matrix QF = 70, (**e**) DCT coefficients after quantization

Table 4.2 Code words for the symbols obtained in the example

Symbol (*a*,*b*)	Code word
(0,1)	00
(0,2)	01
(0,3)	100
(1,1)	1100
(3,1)	111010
(6,1)	1111011
(10,2)	1111111111000111
(0,0) EOB	1010

The entire 8 × 8 block in encoded form is given by:

(101)(10010) (00)(0) (00)(1) (00)(0) (100)(100) (100)(100) (1100)(1) (1111111111000111)(10) (1100)(1) (111010)(1) (00)(1) (1111011)(0) (1010)

Note that in this example, we have coded the DC coefficient value, not the DC coefficients difference, since we have examined a single block.

Decoding is performed using the blocks in Fig. 4.24.

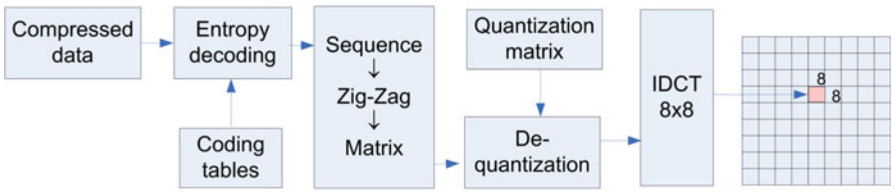


Fig. 4.24 JPEG decoder

We first return the sequence of samples into the matrix form. Next, we perform dequantization, followed by the inverse DCT. In other words, after we get:

$$\mathrm{DCT}_{dq} = \mathrm{DCT}_q(k_1, k_2) \cdot Q(k_1, k_2), \tag{4.59}$$

we apply the inverse DCT transformation:

$$a(i,j) = \sum_{k_1=0}^{7} \sum_{k_2=0}^{7} \frac{C(k_1)}{2} \frac{C(k_2)}{2} \mathrm{DCT}_{\mathrm{dq}}(k_1, k_2) \cos\left(\frac{(2i+1)k_1\pi}{16}\right) \cos\left(\frac{(2j+1)k_2\pi}{16}\right) \tag{4.60}$$

It is obvious that quantization/dequantization procedures and rounding procedures introduce an error proportional to the quantization step.

In order to illustrate the efficiency of JPEG compression in terms of the compromise between the compression factor and image quality, the examples of compressed images with different qualities are shown in Fig. 4.25.

4.9.2 JPEG Lossless Compression

The JPEG lossless compression provides a compression ratio approximately equal to 2:1. It uses a prediction approach to encode the difference between the current pixel X and the one predicted from three previous pixels A, B, C, as illustrated below:

	C	B	
	A	X	

The prediction sample X_p can be obtained using one of the formulas:

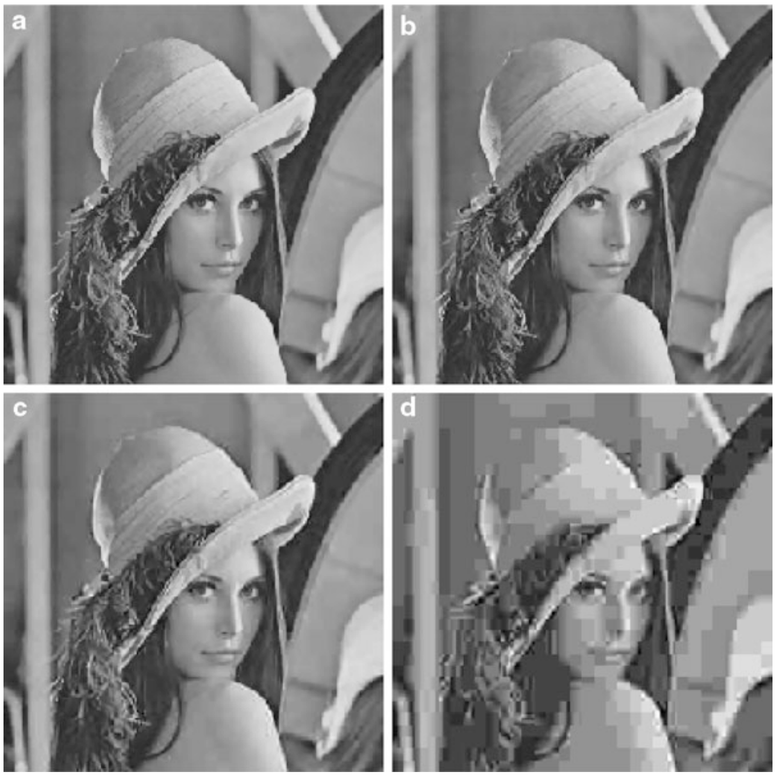


Fig. 4.25 (**a**) Original "Lena" image, (**b**) "Lena" image after applying JPEG compression with QF = 70%, (**c**) "Lena" image after JPEG compression with QF = 25%, (**d**) "Lena" image after JPEG compression with QF = 5%

Case	Prediction formula
1	$X_p = A$
2	$X_p = B$
3	$X_p = C$
4	$X_p = A + B - C$
5	$X_p = A + (B - C)/2$
6	$X_p = B + (A - C)/2$
7	$X_p = (A + B)/2$

Then the difference $\Delta X = X - X_p$ is encoded by using the Huffman code.

4.9.3 Progressive JPEG Compression

4.9.3.1 Spectral Compression

During the image transmission, it is often demanded that a receiver gradually improves the image resolution. Namely, a rough version of the image is first transmitted (which can be done with a high compression factor), and then we transmit the image details. This is achieved with progressive compression methods.

In the algorithm for progressive compression, coding is implemented by using several spectral bands. The bands are divided according to their importance. For example, the first band can be dedicated only to DC coefficients; the second band may be dedicated to the first two AC coefficients (AC1 and AC2); the third band contains the next four AC coefficients, while the fourth band may contain the remaining coefficients.

4.9.3.2 Successive Approximations

In this algorithm, the coefficients are not initially sent with the original values (i.e., they are sent with fewer bits). For example, we first send the coefficients with 2 bits left out (divided by 4), then with one bit left out (divided by 2), and finally at full resolution.

4.9.3.3 Combined Progressive Algorithms

This algorithm combines the two previously described algorithms. Specifically, all the coefficients are grouped into the spectral bands (as in the spectral algorithm), and then the information from all bands is sent with different resolutions (in terms of the number of bits), as in the second algorithm. An example of this algorithm is shown in Fig. 4.26.

4.9.4 JPEG Compression of Color Images

JPEG compression of color images can be performed by compressing each color channel as described for the grayscale images. In JPEG compression, the RGB model can be transformed into the YCrCb space (Fig. 4.27). The Y channel contains information about luminance, and Cr and Cb channels are related to the color along the axes red-green and yellow-blue, respectively. Then, each channel is treated separately, because it is not necessary to encode them with the same precision.

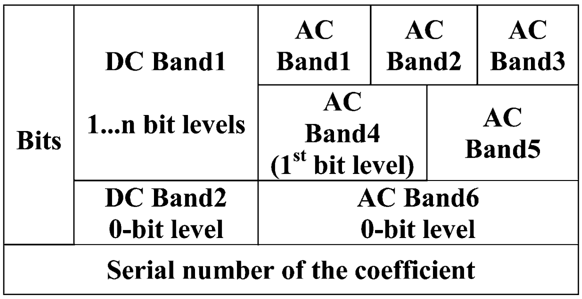


Fig. 4.26 An example of using the combined progressive JPEG algorithm

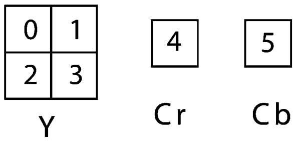


Fig. 4.27 Y, Cr, and Cb component

During decompression, the process is reversed: each channel is decoded individually and then the information is merged together. Lastly, we convert from the YCrCb to the RGB color space.

However, a drawback of this approach is that the color components appear sequentially until the complete image is displayed: the red color is displayed first, then the green color is joined, and finally the blue. In the real applications where the image is decompressed and displayed simultaneously, the interleaved ordering approach is used to combine data from different color channels. Let us consider the case of an image with four components of different resolutions. In addition, each component is divided into rectangular regions with resolutions $\{H_i,V_i\}$. Specifically, factors H_i and V_i define the horizontal and vertical resolutions, respectively, for each component (Fig. 4.28).

The coefficients are combined from the rectangular regions of different components. Each component has the same number of rectangular regions (e.g., six regions) as shown in Fig. 4.28. The basic coding units (Minimum Coded Units—MCU) are formed by using one region from each component. The coefficients in each region are sorted from left to right and from top to bottom. Each MCU can contain up to 10 coefficients.

In the example, in Fig. 4.28, the basic MCUs for encoding are:

$$\begin{aligned}
\text{MCU1} &= a^{1}{}_{00}a^{1}{}_{01}a^{1}{}_{10}a^{1}{}_{11}a^{2}{}_{00}a^{2}{}_{01}a^{3}{}_{00}a^{3}{}_{10}a^{4}{}_{00}\\
\text{MCU2} &= a^{1}{}_{02}a^{1}{}_{03}a^{1}{}_{12}a^{1}{}_{13}a^{2}{}_{02}a^{2}{}_{03}a^{3}{}_{01}a^{3}{}_{11}a^{4}{}_{01}\\
\text{MCU3} &= a^{1}{}_{04}a^{1}{}_{05}a^{1}{}_{14}a^{1}{}_{15}a^{2}{}_{04}a^{2}{}_{05}a^{3}{}_{02}a^{3}{}_{12}a^{4}{}_{02}\\
\text{MCU4} &= a^{1}{}_{20}a^{1}{}_{21}a^{1}{}_{30}a^{1}{}_{31}a^{2}{}_{10}a^{2}{}_{11}a^{3}{}_{20}a^{3}{}_{30}a^{4}{}_{10}
\end{aligned}$$

The described procedure ensures that an image is always displayed with all color components.

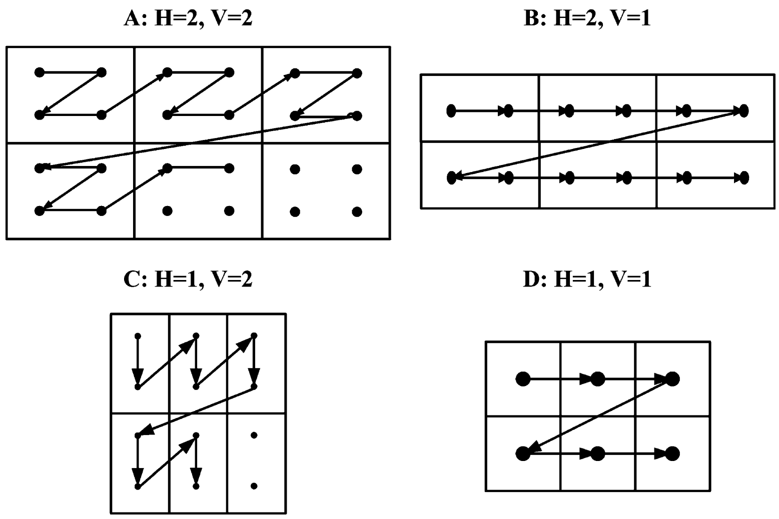


Fig. 4.28 An interleaving procedure for JPEG color image compression

4.9.5 *JPEG2000 Compression*

The standard JPEG algorithm based on the 8 $\times$ 8 DCT blocks often leads to visible block distortions (block effects). To avoid these effects, a JPEG2000 compression algorithm based on the wavelet transform is introduced.

In this algorithm, the entire system can be divided into three parts: (1) image preprocessing, (2) compression, and (3) encoding.

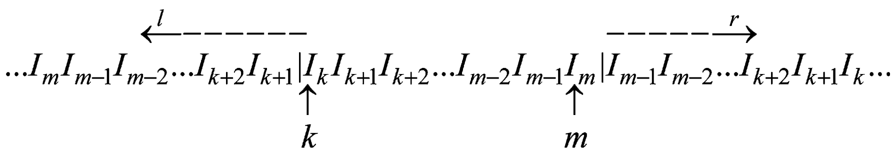
1. Image preprocessing contains some of the optional functions including:

   - Dividing large images into regions (this process is called tiling),
   - DC level shifting,
   - Color components transformation.

   Dividing large images (usually larger than 512 $\times$ 512) into smaller rectangular areas that are separately analyzed is required in order to avoid large buffers in the implementation of the algorithm.

   Similarly to the standard JPEG compression, DC level shifting ($I(x,y) = I(x,y) - 2^{n-1}$) is also used in order to obtain an image with dynamic range that is centered around zero. Values in the range $[0, 2^n - 1]$ are shifted to the values in the range $[-2^{n-1}, 2^{n-1} - 1]$ (n is the number of bits used to represent pixel values).

   This algorithm is defined for the color images consisting of three components. The JPEG2000 standard supports two-color transforms: the reversible color transform (RCT) and the irreversible color transform (ICT). RCT can be applied for both lossy and lossless compression, while ICT can be used only for lossy compression.



l
r
$\dots I_m I_{m-1} I_{m-2} \dots I_{k+2} I_{k+1} | I_k I_{k+1} I_{k+2} \dots I_{m-2} I_{m-1} I_m | I_{m-1} I_{m-2} \dots I_{k+2} I_{k+1} I_k \dots$
k
m

Fig. 4.29 Expansion of a sequence of pixels

The RCT transform (integer-to-integer) is defined as:

$$Y_r = \left\lfloor \frac{\mathrm{R} + 2\mathrm{G} + \mathrm{B}}{4} \right\rfloor, \quad U_r = \mathrm{B} - \mathrm{G}, \quad V_r = \mathrm{R} - \mathrm{G}. \tag{4.61}$$

In the case of RCT, the pixels can be exactly reconstructed by using the inverse RCT defined as follows:

$$\mathrm{R} = V_r + \mathrm{G}, \quad \mathrm{G} = Y_r - \left\lfloor \frac{U_r + V_r}{4} \right\rfloor, \quad \mathrm{B} = U_r + \mathrm{G}. \tag{4.62}$$

ICT is actually a YCrCb transform (real-to-real transform):

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cr} \\ \mathrm{Cb} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168 & -0.331 & 0.5 \\ 0.5 & -0.41 & -0.08 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix}, \tag{4.63}$$

while the inverse transform is given as:

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.402 \\ 1.0 & -0.344136 & -1.714136 \\ 1.0 & 1.772 & 0.0 \end{bmatrix} \begin{bmatrix} \mathrm{Y} \\ \mathrm{Cr} \\ \mathrm{Cb} \end{bmatrix}. \tag{4.64}$$

2. Compression
   JPEG2000 algorithm uses two types of the wavelet transforms. These are (9,7) floating-point wavelets (irreversible) and (5,3) integer wavelet transform (reversible). Only the (5,3) integer transform, which is fully reversible, can be used for lossless compression.

   Consider a sequence of pixels denoted as $I_k, I_{k+1}, I_{k+2}, \dots, I_m$ that belong to an image row. To calculate the wavelet transform, it is necessary to use a few pixels with indices less than k and greater than m. Before applying the wavelet transform, the considered area has to be expanded. An easy way to extend a sequence of pixels (I_k, I_{k+1}, $I_{k+2}, \dots$, I_{m-2}, I_{m-1}, I_m) is illustrated in Fig. 4.29.

After expanding the sequence of pixels, the wavelet coefficients are calculated. For the (5,3) integer wavelet transform, the coefficients are calculated according to:

$$\begin{aligned} d_{j-1,i} &= I_{j,2i+1} - \left\lfloor \frac{I_{j,2i+2} + I_{j,2i}}{2} \right\rfloor, \\ s_{j-1,i} &= I_{j,2i} + \left\lfloor \frac{d_{j-1,i} + d_{j-1,i-1} + 2}{4} \right\rfloor. \end{aligned} \tag{4.65}$$

Here, d coefficients represent high frequency components, while s coefficients represent low frequency components.

In the case of the (9,7) floating-point wavelet transform, wavelet coefficients are obtained as follows:

$$\begin{aligned} P_{j,2i+1} &= I_{j,2i+1} + \alpha\left(I_{j,2i} + I_{j,2i+2}\right), \\ P_{j,2i} &= I_{j,2i} + \beta\left(P_{j,2i-1} + P_{j,2i+1}\right), \\ d'_{j-1,i} &= P_{j,2i-1} + \gamma\left(P_{j,2i-2} + P_{j,2i}\right), \\ s'_{j-1,i} &= P_{j,2i} + \delta\left(d'_{j-1,i} + d'_{j-1,i+1}\right), \\ d_{j-1,i} &= -Kd'_{j-1,i}, \\ s_{j-1,i} &= (1/K)s'_{j-1,i}, \end{aligned} \tag{4.66}$$

where the constants (wavelet filter coefficients) for the JPEG2000 algorithm are: $\alpha = -1.586134342$, $\beta = -0.052980118$, $\gamma = 0.882911075$, $\delta = 0.443506852$, $K = 1.230174105$. As in (4.65), d are details, and s are low-frequency coefficients.

Consider a simple example with five consecutive pixels $I_{j,2i-2}, I_{j,2i-1}, I_{j,2i}, I_{j,2i+1}, I_{j,2i+2}$. The (9,7) wavelet coefficients can be calculated in four steps:

$$\begin{aligned} &\text{The first step is :} && P_{j,2i-1} = I_{j,2i-1} + \alpha I_{j,2i-2} + \alpha I_{j,2i}, \\ & && P_{j,2i+1} = I_{j,2i+1} + \alpha I_{j,2i} + \alpha I_{j,2i+2}, \\ &\text{The second step is :} && P_{j,2i-2} = I_{j,2i-2} + \beta P_{j,2i-3} + \beta P_{j,2i-1}, \\ & && P_{j,2i} = I_{j,2i} + \beta P_{j,2i+1} + \beta P_{j,2i-1}, \\ &\text{The third step is :} && d'_{j-1,i} = P_{j,2i-1} + \gamma\left(P_{j,2i-2} + P_{j,2i}\right), \\ & && d'_{j-1,i+1} = P_{j,2i+1} + \gamma\left(P_{j,2i} + P_{j,2i+2}\right), \\ &\text{The fourth step is :} && s'_{j-1,i} = P_{j,2i} + \delta\left(d'_{j-1,i} + d'_{j-1,i+1}\right), \\ & && s'_{j-1,i+1} = P_{j,2i+2} + \delta\left(d'_{j-1,i+1} + d'_{j-1,i+2}\right). \end{aligned}$$

At the end, d' coefficients are scaled by the parameter K and s' coefficients are scaled by the parameter $1/K$. This procedure is illustrated schematically in Fig. 4.30.

The considered one-dimensional wavelet transform is usually applied to the image rows and then to the columns, as it is illustrated in Fig. 4.31.

Subbands can be organized in one of the three ways, as illustrated in Fig. 4.32.

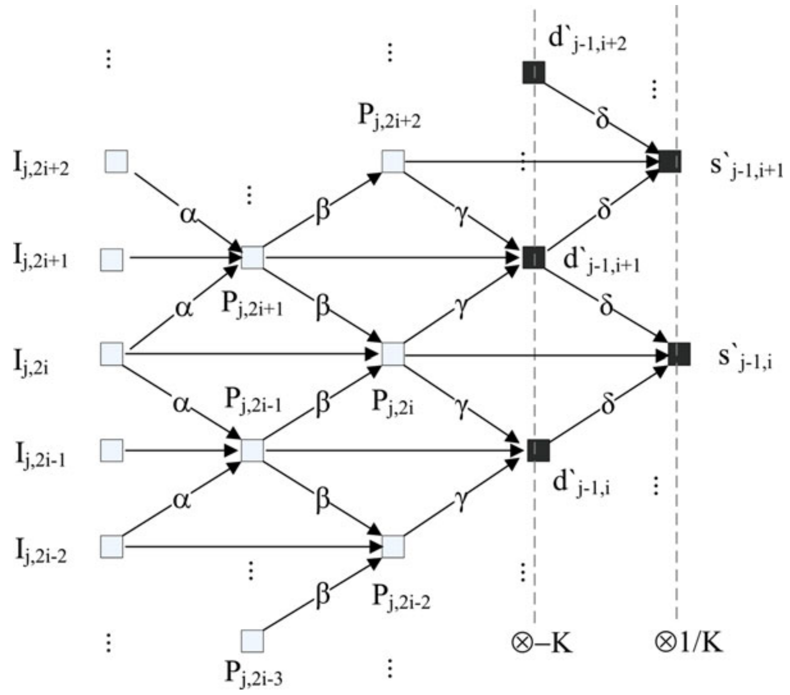


Fig. 4.30 Calculating the wavelet coefficients

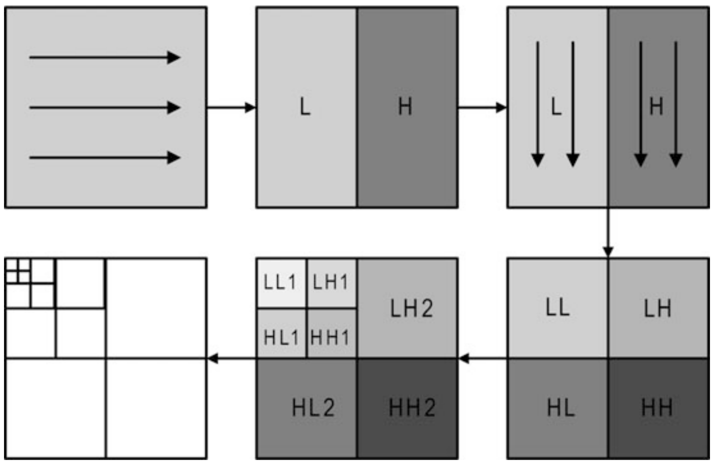


Fig. 4.31 Illustration of applying the wavelet transform to two-dimensional signals

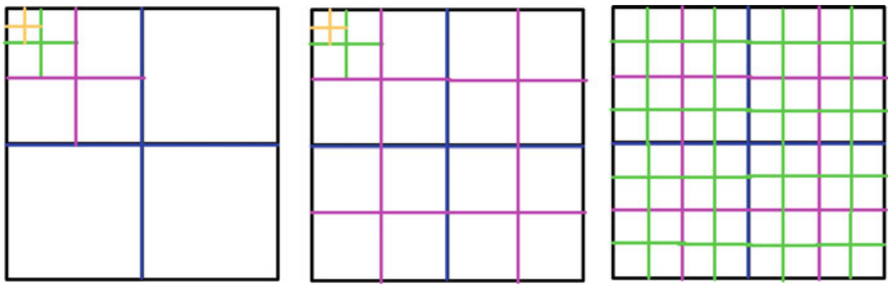


Fig. 4.32 Organization of subbands for JPEG2000

4.9.5.1 JPEG2000 Quantization

For each subband denoted by *b,* a quantization step Δ_b is used for quantization of the coefficients in the subband. Quantization is defined as follows:

$$Q_b(u,v) = \text{sign}(C_b(u,v))\left\lfloor \frac{|C_b(u,v)|}{\Delta_b} \right\rfloor, \tag{4.67}$$

where $C_b(u,v)$ is the original DWT coefficient from the subband b. The operator $\lfloor \cdot \rfloor$ represents rounding to integer number. Hence, the value of quantized coefficient is: $Q_b(u,v)\Delta_b$. The quantization step is defined as follows:

$$\Delta_b = 2^{R_b - \varepsilon_b}\left(1 + \frac{\mu_b}{2^{11}}\right), \tag{4.68}$$

where R_b represents the nominal dynamic range of the subband b. The parameter μ_b is the 11-bit mantissa and ε_b is the 5-bit exponent of the quantization step ($0 \le \mu_b < 2^{11}, 0 \le \varepsilon_b < 2^5$). The exponent/mantissa pairs (μ_b,ε_b) can be explicitly signaled in the bit stream syntax for every subband. The dynamic range depends on the number of bits used to represent the original image tile component and on the choice of the wavelet transform. For reversible compression, the quantization step-size is required to be 1.

The inverse quantization is defined as:

$$R_Q(u,v) = \begin{cases} (Q(u,v) + \delta)\Delta_b, & \text{for } Q(u,v) > 0, \\ (Q(u,v) - \delta)\Delta_b, & \text{for } Q(u,v) < 0, \\ 0, & \text{for } Q(u,v) = 0. \end{cases} \tag{4.69}$$

The reconstruction parameter is usually given by $0 \le \delta < 1$ and the most commonly used value is $\delta = 0.5$.

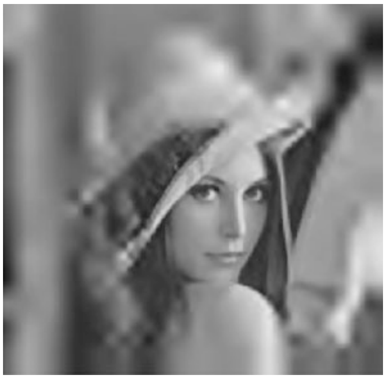


Fig. 4.33 Face is coded as ROI

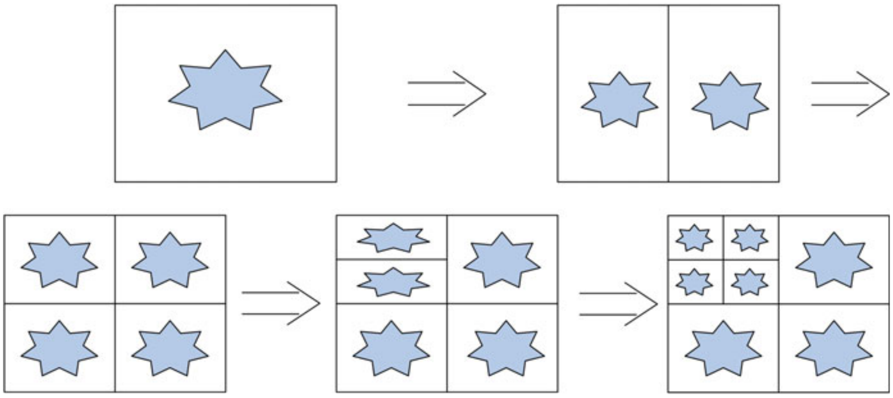


Fig. 4.34 ROI mask mapped into subbands

4.9.5.2 Coding the Regions of Interest

One of the important techniques in the JPEG2000 algorithm is coding the regions of interest (ROI). ROI is expected to be encoded with better quality than the other regions (e.g., image background). Coding of the ROI aims to scale the ROI coefficients and place them in the higher bit planes (comparing to the bit planes of other coefficients). Hence, the ROI coefficients will be progressively transmitted before the non-ROI coefficients. Consequently, ROI will be decoded before other image parts and with higher accuracy (Fig. 4.33).

The method based on scaling is implemented as follows:

1. First, the wavelet transform is calculated.
2. Then, we form a mask indicating the set of coefficients that belong to ROI. Specifically, the ROI mask is mapped from the pixels domain into each subband in the wavelet domain (Fig. 4.34).

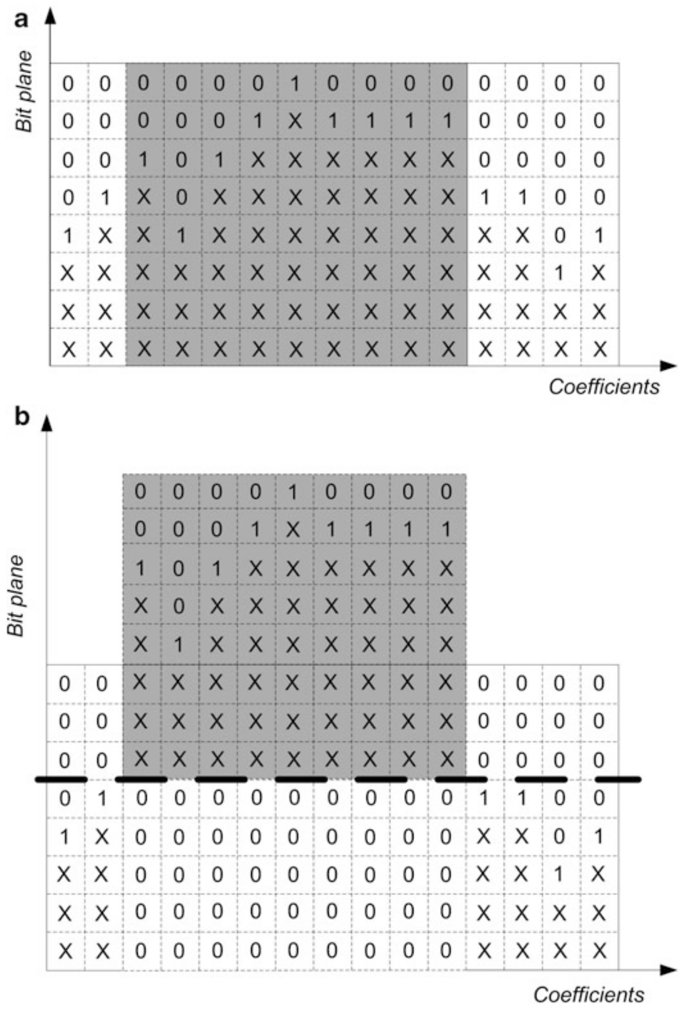


Fig. 4.35 (**a**) Quantized wavelet coefficients before scaling, (**b**) quantized wavelet coefficients after scaling

3. Wavelet coefficients are quantized.
4. ROI coefficients are scaled and shifted to higher bit planes (*MAXSHIFT*).
5. Finally, the coefficients are encoded.

The value of the scaling factor has to be sufficiently large to ensure that the lowest value in the ROI is greater than the largest value of the surrounding non-ROI coefficients. The scaling factor is transmitted with the coefficients in order to be able to reconstruct the original ROI values. An illustration of this process is shown in Fig. 4.35.

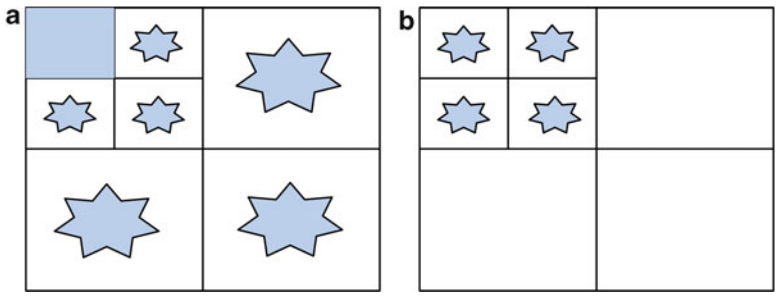


Fig. 4.36 (**a**) The low-frequency subband is codded together with ROI regions in other subbands, (**b**) arbitrary ROI mask

The most significant bit of each coefficient is indicated by "1" (the first non-zero bit in each column), while the following bits are denoted by "x" (could be either 0 or 1). The coefficients that belong to the ROI are shaded in gray. The bit planes that remain after scaling the ROI are filled by "0."

There are several ways to set the ROI masks. As shown in Fig. 4.36a, the low-frequency coefficients can be transmitted together with the ROI coefficients if the ROI masks are placed in all other subbands. Also, ROI regions can be only used in specific subbands as shown in Fig. 4.36b.

Areas and Code Blocks

To achieve more efficient coding, each wavelet decomposition subband can be further divided into precincts. The size of the precincts may vary at different levels of decomposition, but can be usually expressed as a power of 2. Areas on the same positions in different subbands are shaded in gray in Fig. 4.37. Each area is further divided into the code-blocks whose dimensions are also a power of 2. This division provides memory efficient implementation. Simple coders will use this division. On the other hand, sophisticated coders will use a large number of code-blocks to ensure that the decoder performs progressive decompression, as well as to provide higher bit rate, image zooming, and other operations when decoding only parts of the image.

The highlighted precincts in Fig. 4.37 correspond roughly to the same $N/2 \times N/2$ region in the original image (of size $N \times N$). Note that the code-blocks in all subbands are of the same size, except when their size is constrained by the subband precinct size, as in the low-frequency subbands in Fig 4.37.

4.9.5.3 Entropy Coding

The wavelet coefficients are coded by bit-planes using the arithmetic encoding scheme. The encoding is done from the most significant to the least significant

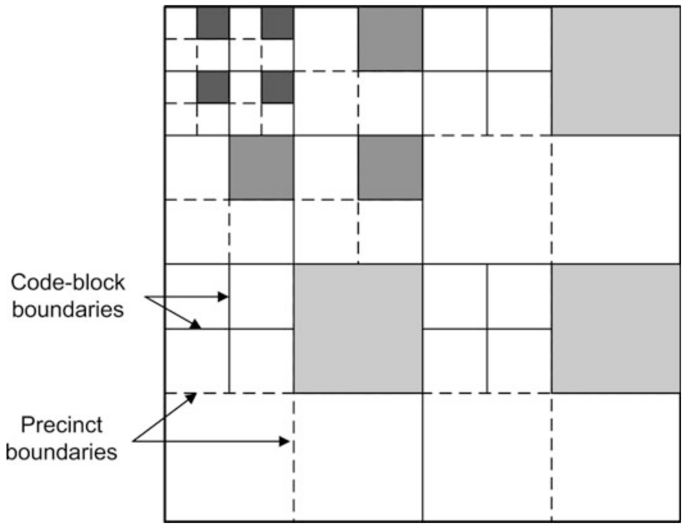


Fig. 4.37 An example of subbands, precincts, and code-blocks partition

bit-plane. We also have to determine the bit context, where the probability of occurrence for each bit is estimated. The bit value and its probability are forwarded to an arithmetic coder. Hence, unlike many other compression algorithms, which encode the coefficients of images, the JPEG2000 encodes the sequences of bits.

Each wavelet coefficient should have an indicator of importance (1 if the coefficient is significant, and 0 if not). At the beginning of coding, all wavelet coefficients are set to be insignificant. If there are bit-planes that contain all zero values, the number of these bit-planes is stored in the bit stream.

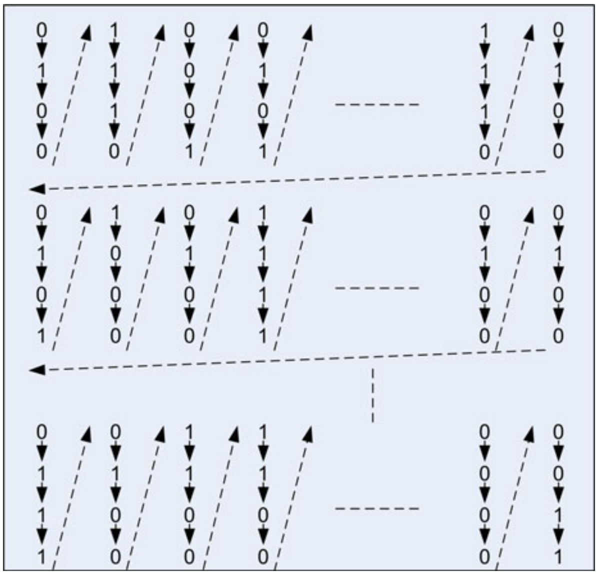
A most significant non-zero bit plane is encoded in one pass, which is called the *cleanup* pass. Each subsequent bit plane is encoded within three passes. Here, each bit-plane is divided into tracks containing four lines (rows), while the tracks are scanned by using the zigzag order (from left to right), as shown in Fig. 4.38.

At the beginning, all bits from the most significant non-zero bit-plane are fed to the encoder (they are encoded in one pass). During this step, the coefficient is denoted as significant if its bit is equal to 1. The coefficient remains significant until the end of the encoding process.

The remaining bit-planes are encoded one after the other (from most to least significant bit-planes), and within the three passes:

- The first pass involves the coefficients denoted as insignificant, which have at least one significant neighbor (one of its eight nearest neighbors is denoted as significant). Their bits from the observed bit plane are forwarded to the coder. As previously described, the coefficients whose bit is 1 are declared significant (a significance indicator is set to 1).
- The second pass encodes the bits of the coefficients that became significant in earlier steps, when passing through a previous bit planes.

Fig. 4.38 The bit-planes scan method



- The third step considers the bits omitted in the previous two steps. If the bit value in the third step is 1, the corresponding coefficient is declared significant.

Example: We will illustrate the bit-planes encoding process by considering separately only four isolated wavelet coefficients whose values are 9, 1, 2, and −6, Fig. 4.39. The coefficients are encoded with 9 bits (1 sign bit and 8 bits for value).

Therefore, 9 = 0 | 00001001, 1 = 0 | 00000001, 2 = 0 | 00000010, −6 = 1 | 00000110

Bit plane containing the sign bits is considered separately and it is ignored at the beginning. If we observe just the four given coefficients, the first four bit planes (planes from 7 to 4) are zero, so the encoding starts from the third plane.

Plane 3: The bit-plane 3 is the most significant non-zero bit plane and it is encoded in a single pass. One bit from each coefficient is brought to the encoder. Note that in the plane 3, a bit for the coefficient with value 9 has the logical value of 1 and the coefficient is declared as significant. The sign bit for the coefficient 9 is encoded immediately after this bit.

Plane 2: The bit-plane 2 is encoded after plane 3. We first encode the coefficients that are insignificant, but they are the neighbors to the significant ones. The coefficient 1 is insignificant, but it is the adjacent coefficient to 9, which is significant. Therefore, the bit 0 of the coefficient 1 is passed to the encoder. Note that none of the coefficients have been declared significant at this stage. The second step includes bits belonging to the significant coefficients. The coefficient 9 is significant, thus its bit 0 is passed to the encoder. Coefficients 2 and −6 are not significant, and they are not located next to the significant coefficients. Hence, they will be encoded in the third step. Since the bit of the coefficient −6 (plane 2) has value 1, this is declared significant, and its sign bit is encoded, as well.

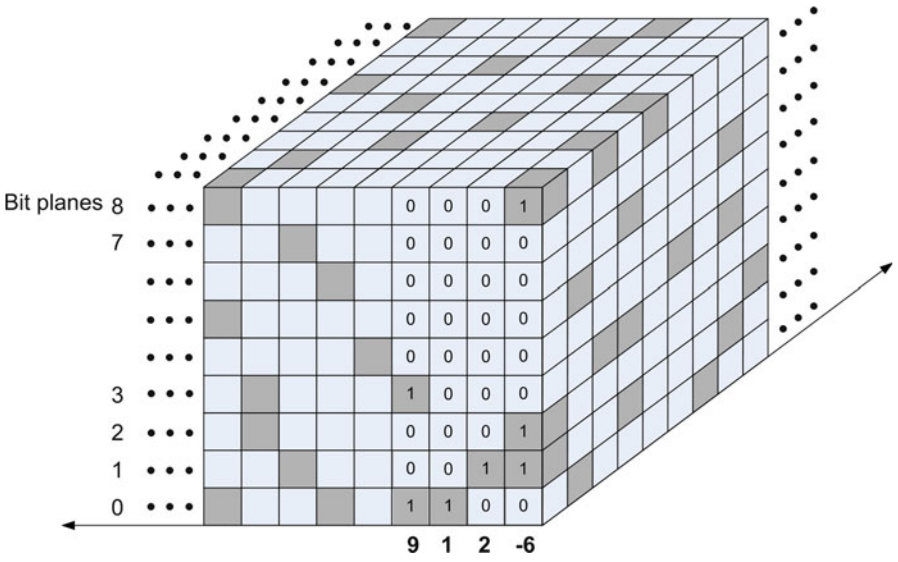


Fig. 4.39 Part of bit planes that contain four considered coefficients

Plane 1: The bits of the coefficients 1 and 2 are coded in the first pass (they are neighbors of significant coefficients), while the bits of the significant coefficients 9 and −6 will be encoded in the second pass. The bit for coefficient 2 is 1, so this coefficient becomes significant (its sign bit is passed to the encoder as well).

Plane 0: This plane is encoded last. The bit of coefficient 1 is encoded in the first pass. The coefficient 1 becomes significant and its sign bit is encoded. The coefficients 9, 2, and −6 are significant, and their bits are encoded in the second pass.

Arithmetic Coding

The sequence of bits from each plane is forwarded to an arithmetic coder. Here, we use the bit context to determine a probability of a binary symbol. One simple example of using arithmetic coding with known probabilities of symbols occurrences is illustrated in Fig. 4.40. Suppose that we want to encode the binary sequence 1011, where the probability of occurrence of symbol “1” is equal to 0.6, while for the symbol “0,” it is 0.4.

At the beginning, we have the interval [0, 1], which is divided into two intervals according to the probability of symbol occurrence, as shown in Fig. 4.40. The lower interval [0, 0.6], used to denote the symbol “1,” is then divided again into two intervals with the same proportions as in the previous case. The second symbol is “0,” and therefore, the upper interval [4.36, 4.6] is divided into two new intervals. We continue this process until the whole sequence is encoded. At the end, we get

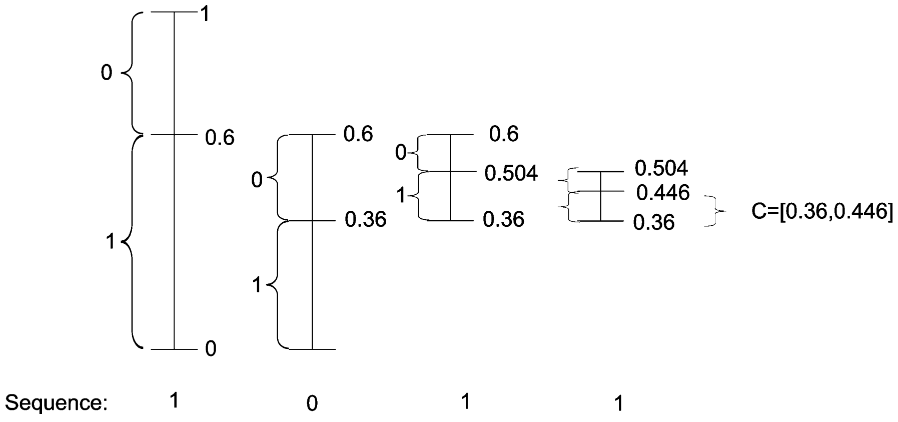


Fig. 4.40 An example of arithmetic coding

C = [0.36, 0.446]. Finally, the sequence 1011 is coded by using one value from the obtained interval, e.g., value 0.4. The interval should be available during the decoding process. Note that the encoding is usually applied to much longer sequences.

Determining the Context

The bit context is used in each step to estimate the probability for bit encoding. The following procedure describes how to determine the context of bits in the first pass. We consider eight neighboring coefficients around the wavelet coefficient X and their current indicators of importance (1 indicating a significant coefficient, and 0 indicating the insignificant one). Let us assume that the indicators of neighboring coefficients are denoted as in Fig. 4.41.

The context is selected based on the criteria listed in Table 4.3. Note that there are 9 contexts, and the criteria depend on the subband (LL, LH, HL, HH), which is encoded. For example, the context 0 represents the coefficient without any significant neighbor. JPEG2000 standard defines similar rules for determining the bit context for the other two passes.

Based on the estimated bit context, the probability estimation process (for encoding the bit) is done by using lookup tables.

4.9.6 *Fractal Compression*

Having in mind that nowadays the devices (e.g., printers) constantly increase the resolution, it is necessary to provide that once compressed image can be decompressed at any resolution. This can be accomplished by using the fractal compression.

Fig. 4.41 Significance indicators for eight neighboring coefficients

D0	V0	D1
H0	X	H1
D2	V1	D3

Table 4.3 Nine contexts for the first pass

LL and LH subbands			Context
$\sum H_i$	$\sum V_i$	$\sum D_i$	
2			**8**
1	**≥1**		**7**
1	**0**	**≥1**	**6**
1	**0**	**0**	**5**
0	**2**		**4**
0	**1**		**3**
0	**0**	**≥2**	**2**
0	**0**	**1**	**1**
0	**0**	**0**	**0**

HL subband			Context
$\sum H_i$	$\sum V_i$	$\sum D_i$	
	2		**8**
≥1	**1**		**7**
0	**1**	**≥1**	**6**
0	**1**	**0**	**5**
2	**0**		**4**
1	**0**		**3**
0	**0**	**≥2**	**2**
0	**0**	**1**	**1**
0	**0**	**0**	**0**

HH subband		Context
$\sum (H_i + V_i)$	$\sum D_i$	
	≥3	**8**
≥1	**2**	7
0	**2**	**6**
≥2	**1**	**5**
1	**1**	**4**
0	**1**	**3**
≥2	**0**	**2**
1	**0**	**1**
0	**0**	**0**

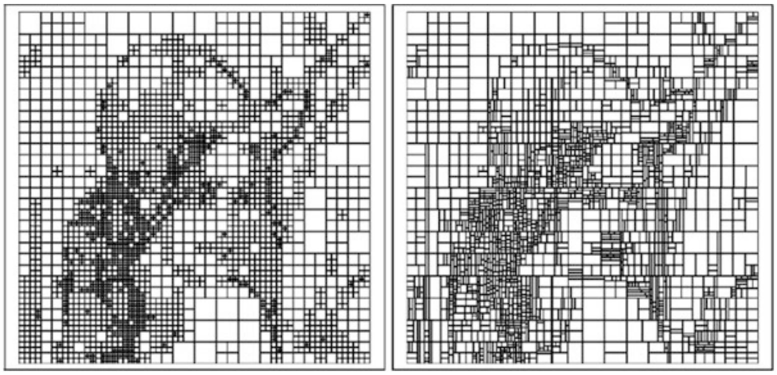


Fig. 4.42 "Lena" image divided into different fractals

In fractal compression, the entire image is divided into pieces (fractals). Using an affine transformation, we are able to mathematically rotate, scale, skew, and translate a function, and thus certain fractals can be used to "cover" the whole image.

Affine transformations for two-dimensional case are given by the relation:

$$W(x,y) = (ax + by + e, cx + dy + f), \tag{4.70}$$

or in the matrix form:

$$W\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \tag{4.71}$$

where the matrix with elements a, b, c, d determines the rotation, skew, and scaling, while the matrix with elements e and f defines the translation.

In the algorithm for fractal compression the entire image is firstly divided into nonoverlapping domain regions. Then, each region is divided into a number of predefined shapes (e.g., rectangles, squares, or triangles) as shown in Fig. 4.42.

The third step is to determine the affine transformations that closely match the domain regions. In the final step, the image is recorded in the Fractional Image Format (FIF) and it contains information about regions selection and the coefficients of affine transformation (for each region).

4.9.7 Image Reconstructions from Projections

Image reconstruction based on projections has important applications in various fields (e.g., in medicine when dealing with computer tomography, which is widely used in everyday diagnosis).

A theoretical approach to this problem is presented below. Consider an object in space, which can be described by the function *f*(*x*,*y*). The projection of function *f*(*x*,*y*) along an arbitrary direction AB (defined by an angle φ) can be defined as follows:

$$p_{\varphi}(u) = \int\limits_{\overline{AB}} f(x,y)\mathrm{d}l, \tag{4.72}$$

where $u = x\cos\varphi + y\sin\varphi$. Thus, (4.72) can be written as:

$$p_{\varphi}(u) = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} f(x,y)\delta(x\cos\varphi + y\sin\varphi - u)\mathrm{d}x\mathrm{d}y. \tag{4.73}$$

The Fourier transform of the projection is given by:

$$P_{\varphi}(\omega) = \int\limits_{-\infty}^{\infty} p_{\varphi}(u)e^{-j\omega u}\mathrm{d}u. \tag{4.74}$$

Furthermore, the two-dimensional Fourier transform of *f*(*x*,*y*) is defined as:

$$F(\omega_x,\omega_y) = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} f(x,y)e^{-j\left(\omega_x x+\omega_y y\right)}\mathrm{d}x\mathrm{d}y. \tag{4.75}$$

As a special case, we can observe $F(\omega_x,\omega_y)$ for $\omega_y = 0$:

$$F(\omega_x,0) = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} f(x,y)e^{-j\omega_x x}\mathrm{d}x\mathrm{d}y = \int\limits_{-\infty}^{\infty} p_0(x)e^{-j\omega_x x}\mathrm{d}x = P_0(\omega). \tag{4.76}$$

Hence, the Fourier transform of a two-dimensional object along the axis $\omega_y = 0$ is equal to the Fourier transform along the projection angle $\varphi = 0$. Consider now what happens in the rotated coordinate system:

$$\begin{pmatrix} u \\ l \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}. \tag{4.77}$$

In this case, (4.74) can be written as:

$$P_{\varphi}(\omega) = \int_{-\infty}^{\infty} p_{\varphi}(u) e^{-j\omega u} \mathrm{d}u = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(u,l) e^{-j\omega u} \mathrm{d}u\mathrm{d}l$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j\omega(x\cos\varphi + y\sin\varphi)} \mathrm{d}x\mathrm{d}y = F(\omega,\varphi), \tag{4.78}$$

where: $F(\omega,\varphi) = F(\omega_x,\omega_y)\Big|_{\substack{\omega_x=\omega\cos\varphi \\ \omega_y=\omega\sin\varphi}} \Big\rangle$.

Now, let us summarize the previous considerations. If we have the object projections, then we can determine their Fourier transforms. The Fourier transform of a projection represents the transform coefficients along the projection line of the object. By varying the projection angle from 0° to 180°, we obtain the Fourier transform along all the lines (e.g., we get the Fourier transform of the entire object), but in the polar coordinate system. To use the well-known FFT algorithm, we have to switch from polar to rectangular coordinate system. Then, the image of the object is obtained by calculating the inverse Fourier transform.

The transformation from the polar to the rectangular coordinate system can be done by using the nearest neighbor principle, or by using some other more accurate algorithms that are based on the interpolations.

4.10 Examples

4.1. Calculate the memory requirements for an image of size 256 $\times$ 256 pixels, in the case of:

(a) Binary image
(b) Grayscale image
(c) Color image

Solution:

(a) In the case of binary image each sample is represented by a single bit, and thus the required memory space is:

$$256{\cdot}256{\cdot}1 = 65{,}536 \text{ bits.}$$

(b) Grayscale image is usually represented by 8 bits per pixel, thus having 256 grayscale levels. The memory requirements for such a kind of image are:

$$256{\cdot}256{\cdot}8 = 524{,}288 \text{ bits.}$$

(c) Color image usually contains three different matrices for each color channel and requires three-times higher memory space than the grayscale image:

$$256{\cdot}256{\cdot}8{\cdot}3 = 1{,}572{,}864 \text{ bits.}$$

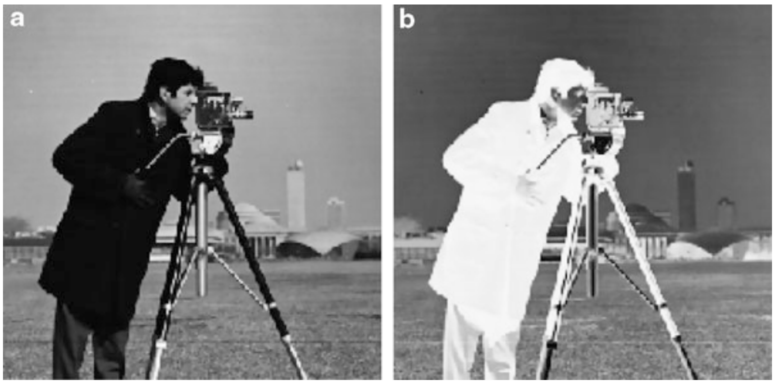


Fig. 4.43 (**a**) Original image cameraman, (**b**) negative of image cameraman

4.2. If the values of R, G, and B components in the RGB systems are known and for a certain pixel they are given by: R = 0.5, G = 0.2, B = 0.8, determine the corresponding values of the components in the YUV color model.

Solution:

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = 0.564(B - Y)$$

$$V = 0.713(R - Y)$$

$$Y = 0.299 \cdot 0.5 + 0.587 \cdot 0.2 + 0.114 \cdot 0.8 = 0.36$$

$$U = 0.564 \cdot (0.8 - 0.358) = 0.25$$

$$V = 0.713 \cdot (0.5 - 0.358) = 0.1$$

4.3. Write a Matlab code that will load color image (e.g., *lena.jpg*), determine the image size, and then convert the color image into a grayscale version by using the Matlab built-in function *rgb2gray*, as well as by using the formula: Grayscale $= \frac{R_{value}+G_{value}+B_{value}}{3}$.

Solution:

```
I=imread('lena.jpg');   % load image
size(I)                % image size
ans =
512 512 3
I1=rgb2gray(I);
figure, imshow(I1)           % show image
```

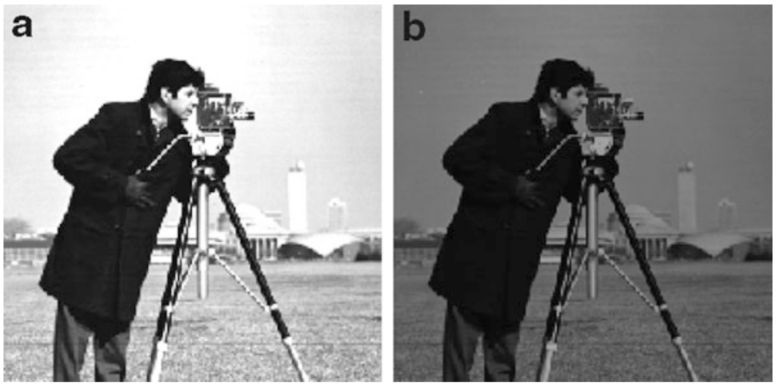


Fig. 4.44 Image lightening and darkening

```
I=double(I);
I_2=(I(:,:,1)+I(:,:,2)+I(:,:,3))/3;
figure, imshow(uint8(I_2));
```

Note: The color channels are obtained as: I(:,:,1), I(:,:,2), I(:,:,3).

4.4. Write a code in Matlab that will create a negative of image *"cameraman.tif"* (Fig. 4.43).

Solution:

```
I=imread('cameraman.tif');
I=double(I);
N=255-I;
figure, imshow(uint8(I))
figure, imshow(uint8(N))
```

4.5. Write a code in Matlab that will provide a simple image darkening and brightening procedure by decreasing/increasing original pixels values for 40% (Fig. 4.44).

Solution:

```
I=imread('cameraman.tif');
I=double(I);
I_B=I+0.4*I;          % brightening
figure(1), imshow(uint8(I_B))
I_D=I-0.4*I;          % darkening
figure(2), imshow(uint8(I_D))
```

4.6. Starting from the grayscale image "*cameraman.tif*," make a version of binary image by setting the threshold on value 128.

Solution:

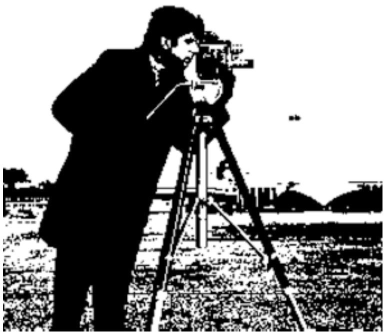


Fig. 4.45 Binary image cameraman

A binary image will have values 255 at the positions (i,j) where the original image has values above the threshold. On the remaining positions the pixels in the binary image will be 0:

$$\mathrm{B}(i,j) = \begin{cases} 255, & I(i,j)>\text{threshold} \\ 0, & \text{otherwise} \end{cases}.$$

The Matlab code that transforms grayscale into binary image is given in the sequel (Fig. 4.45).

```
I=imread('cameraman.tif');
[m,n]=size(I);
for i=1:m
for j=1:n
if I(i,j)>128
  I(i,j)=255;
  else
      I(i,j)=0;
      end
  end
end
figure, imshow(I)
```

4.7. Consider a color image "*lena.jpg*." Transform the image into grayscale one and add a white Gaussian noise with variance 0.02 (Fig. 4.46).

Solution:

```
I=imread('lena.jpg');
I=rgb2gray(I);
I1=imnoise(I,'gaussian',0,0.02);
figure, imshow(uint8(I1))
```

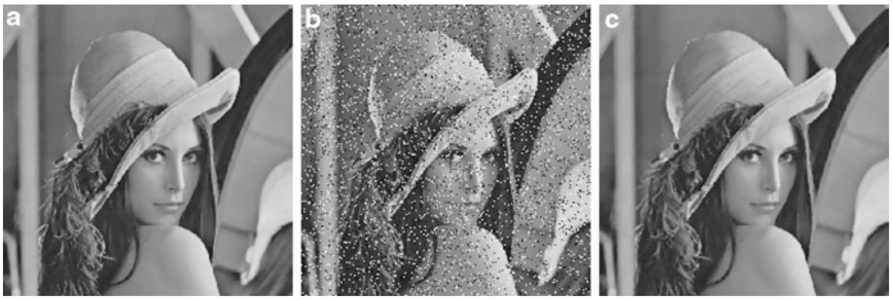


Fig. 4.46 (**a**) Original image, (**b**) noisy image, (**c**) filtered image

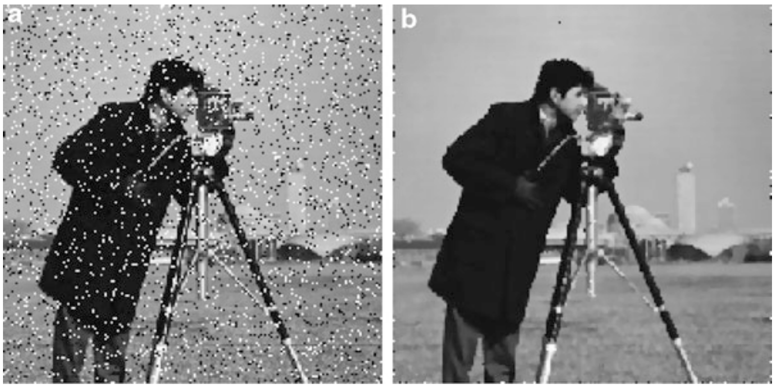


Fig. 4.47 (**a**) Noisy image, (**b**) filtered image

4.8. Calculate the mean and median values for vectors:
(a) $v_1 = [12\ 22\ 16\ 41\ -3]$; (b) $v_2 = [12\ 9\ 22\ 16\ 41\ -3]$;

Solution:

(a) $v_1 = [12\ 22\ 16\ 41\ -3]$;
mean $= 17.6$
sorted_$v_1 = [-3\ 12\ 16\ 22\ 41]$;
median $= 16$.

(b) $v_2 = [12\ 9\ 22\ 16\ 41\ -3]$
mean $= 16.17$
sorted_ $v_2 = [-3\ 9\ 12\ 16\ 22\ 41]$
median $= (12 + 16)/2 = 14$.

4.9. By using the Matlab function *imnoise*, add the impulse noise ("salt & pepper" with a density 0.1) to the image "*lena.jpg*." Then perform the image

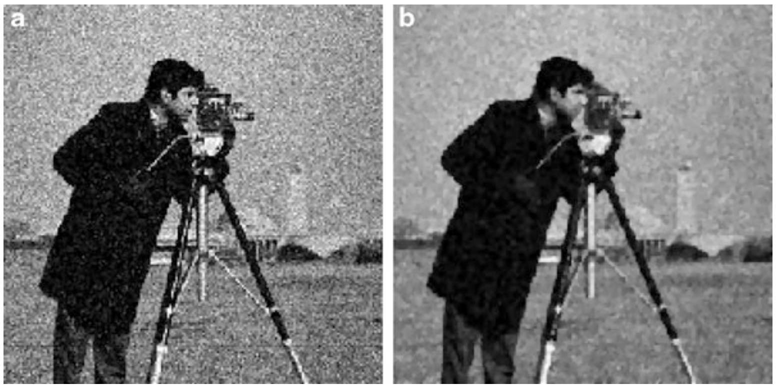


Fig. 4.48 (**a**) Noisy image (Gaussian noise), (**b**) filtered image

filtering by using the two-dimensional median filter realized by Matlab function medfilt2.

Solution:

```
I=imread('lena.jpg');
I=rgb2gray(I);
figure, imshow(I)
In=imnoise(I,'salt & pepper',0.1);
figure, imshow(In)
If=medfilt2(In);
figure, imshow(If)
```

4.10. Write your own code for median filtering in Matlab: the filtering should be applied to image "*cameraman.tif*" which is corrupted by the impulse noise with density 0.1. Use the window of size 3 × 3. It is necessary to include the image boundaries as well (Fig. 4.47).

Solution:

```
I=imread('cameraman.tif');
In=imnoise(I,'salt & pepper',0.03);
[m,n]=size(In);
IM=zeros(m,n);
In=double(In);
for i=1:m
     for j=1:n
     b=In(max(i,i-1):min(m,i+1),max(j,j-1):min(n,j+1));
     b=b(:);
    IM(i,j)=median(b);
     end
end
```

```
figure(1),imshow(uint8(I_n))
figure(2),imshow(uint8(I_M))
```

4.11. Write a Matlab code that filters an image corrupted by Gaussian noise with zero mean and variance equal to 0.01. The window of size 5 × 5 is used (Fig. 4.48).

Solution:

```
I=imread('cameraman.tif');
I_n =imnoise(I,'gaussian',0,0.01);
[m,n]=size(I_n);
I_M=zeros(m,n);
I_n=double(I_n);
for i=1:m
    for j=1:n
    b=I_n(max(i,i-2):min(m,i+2),max(j,j-2):min(n,j+2));
    b=b(:);
    I_M(i,j)=mean(b);
    end
end
figure(1),imshow(uint8(I_n))
figure(2),imshow(uint8(I_M))
```

4.12. For a given matrix of size 5 × 5, determine the corresponding cooccurrence matrix and the measure of contrast.

$$\begin{array}{ccccc} 12 & 11 & 11 & 11 & 14 \\ 12 & 11 & 11 & 11 & 14 \\ 12 & 11 & 11 & 14 & 14 \\ 13 & 11 & 11 & 14 & 14 \\ 13 & 13 & 12 & 12 & 12 \end{array}$$

Solution:

Cooccurence matrix c is obtained in the following form:

x/y	**11**	**12**	**13**	**14**
11	**6**	**0**	**0**	**4**
12	**3**	**2**	**0**	**0**
13	**1**	**1**	**1**	**0**
14	**0**	**0**	**0**	**2**

The measure of contrast is given by:

$$\text{Con} = \sum_{x=0}^{3} \sum_{y=0}^{3} (x-y)^2 c(x,y) = 44.$$

4.13. For a given block of 8 $\times$ 8 DCT coefficients and the given JPEG quantization matrix Q, perform the quantization, zigzag scanning and determine the coded sequence.

$$D=\begin{bmatrix}80 & 50 & 26 & 10 & 33 & 11 & 0 & 0\\ 22 & -28 & 34 & 10 & 0 & 0 & 0 & 0\\ 14 & 10 & 17 & 11 & 5 & 0 & 5 & 0\\ 56 & 17 & 20 & 12 & 0 & 12 & 8 & 0\\ 10 & 12 & 8 & 3 & 2 & 0 & 7 & 0\\ 10 & 13 & 17 & 3 & 0 & 2 & 2 & 0\\ 6 & 0 & 5 & 10 & 14 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix} \quad Q=\begin{bmatrix}3 & 5 & 7 & 9 & 11 & 13 & 15 & 17\\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19\\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21\\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23\\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25\\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27\\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29\\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31\end{bmatrix}$$

Solution:

DCT coefficients from the 8 $\times$ 8 block are divided by the quantization matrix and rounded to the integer values, as follows:

$$D_q = \text{round}(D/Q) = \begin{bmatrix}27 & 10 & 4 & 1 & 3 & 1 & 0 & 0\\ 4 & -4 & 4 & 1 & 0 & 0 & 0 & 0\\ 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0\\ 6 & 2 & 2 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

After performing the zigzag scanning of the matrix D_q, the sequence is obtained in the form:

27, 10, 4, 2, −4, 4, 1, 4, 1, 6, 1, 2, 2, 1, 3, 1, 0, 1, 2, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...

The intermediate symbol sequence is given by:

$(5)(27)$, $(0,4)(10)$, $(0,3)(4)$, $(0,2),(2)$, $(0,3)(-4)$, $(0,3)(4)$, $(0,1)(1)$, $(0,3)(4),(0,1)(1)$, $(0,3)(6)$, $(0,1)(1)$, $(0,2)(2)$, $(0,2)(2)$, $(0,1)(1)$, $(0,2)(3)$, $(0,1)(1),(1,1)(1)$, $(0,2)(2)$, $(0,1)(1)$, $(0,1)(1)$, $(1,1)(1)$, $(0,1)(1)$, $(0,1)(1)$, $(8,1)(1),(6,1)(1)$, $(9,1)(1)$, $(0,0)$.

The code words for the symbols (*a*,*b*) are given in the table:

Symbol (*a*,*b*)	Code word
(0,1)	00
(0,2)	01
(0,3)	100
(0,4)	1011
(1,1)	1100
(6,1)	1111011
(8,1)	111111000
(9,1)	111111001
(0,0) EOB	1010

Hence, the coded sequence is:

(101)(11011) (1011)(1010) (100)(100) (01)(10) (100)(011) (100) (100) (00)(1) (100)(100) (00)(1) (100)(110) (00)(1) (01)(10) (01) (10) (00)(1) (01)(11) (00)(1) (1100)(1) (01)(10) (00)(1) (00)(1) (1100)(1) (00)(1) (00)(1) (111111000)(1) (1111011)(1) (111111001) (1) (1010)

4.14. Consider four 8-bit coefficients A, B, C, and D with values $A = -1$, $B = -3$, $C = 11$, $D = 5$. Explain the encoding procedure using the bit planes concept as in JPEG2000 compression.

Solution:

Bit plane sign	*A*	*B*	*C*	*D*
8	1	1	0	0
7	0	0	0	0
6	0	0	0	0
5	0	0	0	0
4	0	0	0	0
3	0	0	1	0
2	0	0	0	1
1	0	1	1	0
0	1	1	1	1

The coding starts from the bit-plane 3 since it is the first bit-plane that contains the non-zero coefficients.

Bit-plane 3: This bit plane is encoded within a single encoding pass. C is declared significant, since its bit has value 1. The sign bit is encoded immediately after bit 1.

Bit-plane 2: The coefficients *B* and *D* are insignificant, but they are neighbors of the significant coefficient *C*. Hence, their bits are encoded in the first pass. The corresponding bit of coefficient *D* is 1, and thus *D* is declared significant (sign bit is encoded as well).

The bits of significant coefficients are encoded in the second pass, i.e., the bit of the coefficient *C*. Bit of the coefficient *A* is encoded in the third pass.

Bit-plane 1: The bit of the coefficient B is encoded in the first pass, since *B* is a neighbor of the significant coefficient *C* (the sign bit of *B* is encoded after its bit 1). The bits of significant coefficients *C* and *D* are encoded in the second pass. The bit of coefficient *A* is encoded in the third pass.

Bit-Plane 0: Bit of coefficient *A* is encoded in the first pass, and since the bit is 1, the sign bit is encoded as well. The bits of coefficients *B*, *C*, and *D* are encoded in the second pass.

Appendix – Matlab Codes for Some of the Considered Image Transforms

IMAGE CLIPPING

```
I=imread('lena512.bmp');
I=I(1:2:512,1:2:512);
I=double(I);
for i=1:256
    for j=1:256
       if I(i,j)<100
          I(i,j)=100;
       elseif I(i,j)>156
          I(i,j)=156;
       end
    end
end
I=uint8(I);
imshow(I)
```

TRANSFORMING IMAGE LENA TO IMAGE BABOON

```
Ia=imread('lena512.bmp');
Ia=Ia(1:2:512,1:2:512);
Ia=double(Ia);
Ib=imread('baboon.jpg');
Ib=rgb2gray(Ib);
Ib=double(Ib);
for i=1:10
   Ic=(1-i/10)*Ia+(i/10)*Ib;
imshow(uint8(Ic))
pause(0.5)
end
```

GEOMETRIC MEAN FILTER

```
clear all
I=imread('board.tif');
I=imnoise(I,'gaussian',0,0.025);
I=double(I);
[m,n]=size(I);
Im=zeros(size(I));
for i=1:m
        for j=1:n
              a=I(max(i,i-1):min(m,i+1),max(j,j-1):min(n,j+1));
              Im(i, j )=geomean(a(:));
        end
end
figure(1), imshow(uint8(I))
figure(2), imshow(uint8(Im))
```

CONSECUTIVE IMAGE ROTATIONS *(Image is rotated in steps of 5° up to 90°)*

```
I=imread('lena512.bmp');
I=I(1:2:512,1:2:512);
for k=5:5:90
     I1=imrotate(I,k,'nearest');
     imshow(I1)
     pause(1)
end
```

SOBEL EDGE DETECTOR version1

```
I=imread('cameraman.tif');
subplot(221),imshow(I)
edge_h=edge(I,'sobel','horizontal');
subplot(222),imshow(edge_h)
edge_v=edge(I,'sobel','vertical');
subplot(223),imshow(edge_v)
edge_b=edge(I,'sobel','both');
subplot(224),imshow(edge_b)
```

SOBEL EDGE DETECTOR version2
WITH AN ARBITRARY GLOBAL THRESHOLD

```
clear all
I=imread('lena512.bmp');
I=I(1:2:512,1:2:512);
[m,n]=size(I);
I=double(I);
H=[1 2 1; 0 0 0; -1 -2 -1];
V=[1 0 -1; 2 0 -2; 1 0 -1];
Edge_H=zeros(m,n);
Edge_V=zeros(m,n);
Edges=zeros(m,n);
```

```
thr=200;
for i=2:m-1
    for j=2:n-1
Lv=sum(sum(I(i-1:i+1,j-1:j+1).*V));
Lh=sum(sum(I(i-1:i+1,j-1:j+1).*H));
L=sqrt(Lv^2+Lh^2);
if Lv>thr
    Edge_V(i,j)=255;
end
if Lh>thr
    Edge_H(i,j)=255;
end
if L>thr
    Edges(i,j)=255;
end
    end
end
figure, imshow(uint8(Edge_H))
figure, imshow(uint8(Edge_V))
figure, imshow(uint8(Edges))
```

WAVELET IMAGE DECOMPOSITION

```
I=imread('lena512.bmp');
I=double(I);
n=max(max(I));
%First level decomposition
[S1,H1,V1,D1]=dwt2(I,'haar');
S1=wcodemat(S1,n);
H1=wcodemat(H1,n);
V1=wcodemat(V1,n);
D1=wcodemat(D1,n);
dec2d_1 = [S1 H1; V1 D1];
%Next level decomposition
I=S1;
[S2,H2,V2,D2]=dwt2(I,'haar');
S2=wcodemat(S2,n);
H2=wcodemat(H2,n);
V2=wcodemat(V2,n);
D2=wcodemat(D2,n);
dec2d_2 = [S2 H2; V2 D2];
dec2d_1 = [dec2d_2 H1; V1 D1];
imshow(uint8(dec2d_1))
```

JPEG IMAGE QUANTIZATION

```
I=imread('lena.jpg');
I=rgb2gray(I);
I=double(I(1:2:512,1:2:512));
```

```
Q50=[16 11 10 16 24 40 51 61;
     12 12 14 19 26 58 60 55;
     14 13 16 24 40 57 69 56;
     14 17 22 29 51 87 80 62;
     18 22 37 56 68 109 103 77;
     24 35 55 64 81 104 113 92;
     49 64 78 87 103 121 120 101;
     72 92 95 98 112 100 103 99];
```

Table 4.4 Symbols and corresponding code words for AC luminance components

(*a*,*b*)	Code word	(*a*, *b*)	Code word
(0,0)	1010	(3,9)	1111111110010100
(0,1)	00	(3,A)	1111111110010101
(0,2)	01	(4,1)	111011
(0,3)	100	(4,2)	1111111000
(0,4)	1011	(4,3)	1111111110010110
(0,5)	11010	(4,4)	1111111110010111
(0,6)	1111000	(4,5)	1111111110011000
(0,7)	11111000	(4,6)	1111111110011001
(0,8)	1111110110	(4,7)	1111111110011010
(0,9)	1111111110000010	(4,8)	1111111110011011
(0,A)	1111111110000011	(4,9)	1111111110011100
(1,1)	1100	(4,A)	1111111110011101
(1,2)	11011	(5,1)	1111010
(1,3)	1111001	(5,2)	11111110111
(1,4)	111110110	(5,3)	1111111110011110
(1,5)	11111110110	(5,4)	1111111110011111
(1,6)	1111111110000100	(5,5)	1111111110100000
(1,7)	1111111110000101	(5,6)	1111111110100001
(1,8)	1111111110000110	(5,7)	1111111110100010
(1,9)	1111111110000111	(5,8)	1111111110100011
(1,A)	1111111110001000	(5,9)	1111111110100100
(2,1)	11100	(5,A)	1111111110100101
(2,2)	11111001	(6,1)	1111011
(2,3)	1111110111	(6,2)	111111110110
(2,4)	111111110100	(6,3)	1111111110100110
(2,5)	1111111110001001	(6,4)	1111111110100111
(2,6)	1111111110001010	(6,5)	1111111110101000
(2,7)	1111111110001011	(6,6)	1111111110101001
(2,8)	1111111110001100	(6,7)	1111111110101010
(2,9)	1111111110001101	(6,8)	1111111110101011
(2,A)	1111111110001110	(6,9)	1111111110101100
(3,1)	111010	(6,A)	1111111110101101
(3/2)	111110111	(7,1)	11111010
(3,3)	111111110101	(7,2)	111111110111
(3,4)	1111111110001111	(7,3)	1111111110101110
(3,5)	1111111110010000	(7,4)	1111111110101111

(continued)

Table 4.4 (continued)

(*a*,*b*)	Code word	(*a*, *b*)	Code word
(3,6)	1111111110010001	(7,5)	1111111110110000
(3,7)	1111111110010010	(7,6)	1111111110110001
(3,8)	1111111110010011	(7,7)	1111111110110010
		(7,8)	1111111110110011
(7,9)	1111111110110100	(C,2)	1111111111011001
(7,A)	1111111110110101	(C,3)	1111111111011010
(8,1)	111111000	(C,4)	1111111111011011
(8,2)	111111111000000	(C,5)	1111111111011100
(8,3)	1111111110110110	(C,6)	1111111111011101
(8,4)	1111111110110111	(C,7)	1111111111011110
(8,5)	1111111110111000	(C,8)	1111111111011111
(8,6)	1111111110111001	(C,9)	1111111111100000
(8,7)	1111111110111010	(C,A)	1111111111100001
(8,8)	1111111110111011	(D,1)	11111111000
(8,9)	1111111110111100	(D,2)	1111111111100010
(8,A)	1111111110111101	(D,3)	1111111111100011
(9,1)	111111001	(D,4)	1111111111100100
(9,2)	1111111110111110	(D,5)	1111111111100101
(9,3)	1111111110111111	(D,6)	1111111111100110
(9,4)	1111111111000000	(D,7)	1111111111100111
(9,5)	1111111111000001	(D,8)	1111111111101000
(9,6)	1111111111000010	(D,9)	1111111111101001
(9,7)	1111111111000011	(D,A)	1111111111101010
(9,8)	1111111111000100	(E,1)	1111111111101011
(9,9)	1111111111000101	(E,2)	1111111111101100
(9,A)	1111111111000110	(E,3)	1111111111101101
(A,1)	111111010	(E,4)	1111111111101110
(A,2)	1111111111000111	(E,5)	1111111111101111
(A,3)	1111111111001000	(E,6)	1111111111110000
(A,4)	1111111111001001	(E,7)	1111111111110001
(A,5)	1111111111001010	(E,8)	1111111111110010
(A,6)	1111111111001011	(E,9)	1111111111110011
(A,7)	1111111111001100	(E,A)	1111111111110100
(A,8)	1111111111001101	(F,0)	11111111001
(A,9)	1111111111001110	(F,1)	1111111111110101
(A,A)	1111111111001111	(F,2)	1111111111110110
(B,1)	1111111001	(F,3)	1111111111110111
(B,2)	1111111111010000	(F,4)	1111111111111000
(B,3)	1111111111010001	(F,5)	1111111111111001
(B,4)	1111111111010010	(F,6)	1111111111111010
(B,5)	1111111111010011	(F,7)	1111111111111011
(B,6)	1111111111010100	(F,8)	1111111111111100
(B,7)	1111111111010101	(F,9)	1111111111111101
(B,8)	1111111111010110	(F,A)	1111111111111110
(B,9)	1111111111010111		
(B,A)	1111111111011000		
(C,1)	1111111010		

```
QF=70;
q=2-0.02*QF; %q=50/QF;
Q=round(Q50.*q);
I1=zeros(256,256);
for i=1:8:256-7
    for j=1:8:256-7
        A=I(i:i+7,j:j+7);
        dct_block=dct2(A);
        dct_Q=round(dct_block./Q).*Q;
        I1(i:i+7,j:j+7)=idct2(dct_Q);
    end
end
figure(1), imshow(uint8(I))
figure (2), imshow (uint8(I1))
```

Table 4.4

References

1. Askelof J, Larsson Carlander M, Christopoulos C (2002) Region of interest coding in JPEG2000. Signal Process Image Commun 17:105–111
2. Baret HH, Myers K (2004) Foundation of image science. Willey, Hoboken
3. Bednar J, Watt T (1984) Alpha-trimmed means and their relationship to median filters. IEEE Trans Acoust Speech Signal Process 32(1):145–153
4. Daubechies I (1992) Ten lectures on wavelets. Society for Industrial and Applied Mathematics, Philadelphia
5. Djurovic I (2006) Digitalna obrada slike. Univerzitet Crne gore, Elektrotehnicki fakultet Podgorica
6. Dyer M, Taubman D, Nooshabadi S, Kumar Gupta A (2006) Concurrency techniques for arithmetic coding in JPEG2000. IEEE Trans Circuit Syst I 53(6):1203–1213
7. Fisher Y (ed) (1995) Fractal image compression: theory and application to digital images. Springer, New York
8. Furth B, Smoliar S, Zhang H (1996) Video and image processing in multimedia systems. Kluwer Academic Publishers, Boston
9. González RC, Woods R (2008) Digital image processing. Prentice Hall, Englewood Cliffs
10. Kato T, Kurita T, Otsu N, Hirata KA (1992) Sketch retrieval method for full color image database-query by visual example. In: Proceedings of the 11th IAPR international conference on pattern recognition, vol I, computer vision and applications, pp 530–533
11. Khan MI, Jeoti V, Khan MA (2010) Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In: International conference on intelligent and advanced systems (ICIAS), pp 1–6
12. Man H, Docef A, Kossentini F (2005) Performance analysis of the JPEG2000 image coding standard. Multimedia Tools Appl J 26(1):27–57
13. Percival DB, Walden AT (2006) Wavelet methods for time series analysis. Cambridge University Press, Cambridge
14. Qi YL (2009) A relevance feedback retrieval method based on Tamura texture. In: Second international symposium on knowledge acquisition and modeling, Wuhan, China, pp 174–177
15. Rabbani M, Joshi B (2002) An overview of the JPEG2000 still image compression standard. Signal Process Image Commun 17(1):3–48

16. Salomon D, Motta G, Bryant D (2009) Handbook of data compression. Springer, London
17. Stanković LJ (1990) Digitalna Obrada Signala. Naučna knjiga, Beograd
18. Steinmetz R (2000) Multimedia systems. McGrawHill, New York
19. Stollnitz EJ, DeRose TD, Salesin DH (1995) Wavelets for computer graphics: a primer, part 1. IEEE Comput Graph Appl 15(3):76–84
20. Stollnitz EJ, DeRose TD, Salesin DH (1995) Wavelets for computer graphics: a primer, part 2. IEEE Comput Graph Appl 15(4):75–85
21. Strutz T (2009) Lifting parameterisation of the 9/7 wavelet filter bank and its application in lossless image compression. ISPRA'09, Cambridge, pp 161–166
22. Strutz T (2009) Wavelet filter design based on the lifting scheme and its application in lossless image compression. WSEAS Trans Signal Process 5(2):53–62
23. Tamura H, Shunji M, Takashi Y (1978) Textural features corresponding to visual perception. IEEE Trans Syst Man Cybern 8(6):460–473
24. Taubman D, Marcellin M (2002) JPEG2000: standard for interactive imaging. Proc IEEE 90:1336–1357
25. Thyagarajan KS (2006) Digital image processing with application to digital cinema. Elsevier Focal Press, Oxford
26. T.81: Information technology – digital compression and coding of continuous-tone still images – requirements and guidelines
27. Veterli M, Kovačević J (1995) Wavelets and subband coding. Prentice Hall, Englewood Cliffs
28. Wagh KH, Dakhole PK, Adhau VG (2008) Design and implementation of JPEG2000 encoder using VHDL. In: Proceedings of the world congress on engineering, vol I WCE 2008, London, UK
29. Young I, Gerbrands J, Vliet LV (2009) Fundamentals of image processing. Delft University of Technology

Chapter 5
Digital Video

Unlike digital audio signals that are sampled in time or digital images sampled in the spatial domain, a digital video signal is sampled in both space and time, as illustrated in Fig. 5.1.

Time sample is called frame. The sampling rate is 25 frames/s or 30 frames/s. However, it should be noted that instead of a frame, two fields are used, one containing even and the other odd lines. In this case, the sampling rate is 50 fields/s or 60 fields/s.

Digital video is based on the YCrCb color model given by:

$$\begin{aligned} \mathrm{Y} &= 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}, \\ \mathrm{Cb} &= 0.564(\mathrm{B} - \mathrm{Y}), \\ \mathrm{Cr} &= 0.713(\mathrm{R} - \mathrm{Y}). \end{aligned}$$

Different sampling schemes are available, depending on the resolution of the luminance Y and the chrominance components Cr and Cb. They have been known as: 4:4:4, 4:2:2, and 4:2:0.

The 4:4:4 scheme means that all components are used with the full resolution: each pixel contains Y, Cr, and Cb component, as shown in Fig. 5.2a. For 4:2:2 scheme, the Cr and Cb components are represented with a twice lower resolution compared to Y. Observing the four pixels, we see that for 4 Y samples, there are 2 Cr and 2 Cb samples, Fig. 5.2b. Lastly, the 4:2:0 sampling scheme has a four times lower resolution for Cr and Cb components compared to the Y component. So, among four pixels only one contains Cr and *Cb* component, Fig. 5.2c.

Now, let us compute a required number of bits per pixel for each of these sampling schemes. Again we consider the four neighboring pixels. In the 4:4:4 scheme, all pixels contain three components (YCrCb), and if 8 bits is required for each of them, we have:

$$4 \times 3 \times 8 = 96 \text{ b in total, i.e.,} 96/4 = 24 \text{ b/pixel}.$$

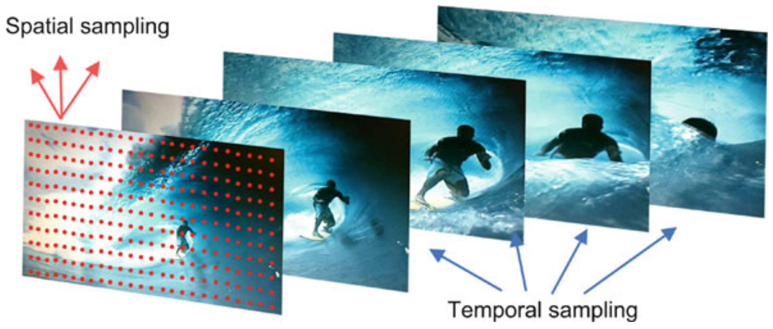


Fig. 5.1 An illustration of video signal sampling

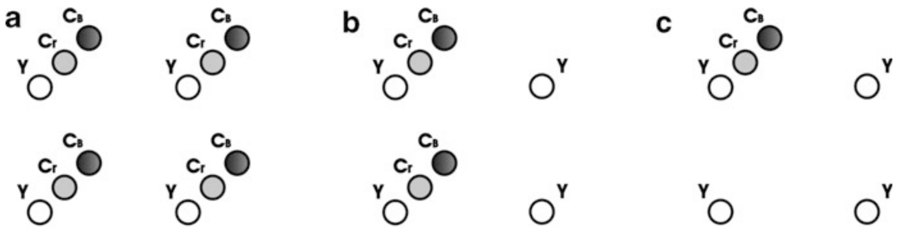


Fig. 5.2 Sampling schemes (**a**) 4:4:4, (**b**) 4:2:2, (**c**) 4:2:0

In the 4:2:2 scheme, two pixels contain three components and the other two pixels contain only one component. Hence, the average number of bits per pixel is calculated as:

$$2 \times 3 \times 8 + 2 \times 1 \times 8 = 64 \text{ b in total, i.e., } 64/4 = 16 \text{ b/pixel.}$$

In analogy to the previous case, for the 4:2:0 scheme we obtain:

$$1 \times 3 \times 8 + 3 \times 1 \times 8 = 48 \text{ b in total, i.e., } 48/4 = 12 \text{ b/pixel}$$

In the sequel, we consider one simple example to illustrate how the sampling schemes 4:4:4 and 4:2:0 influence the amount of data. The frame size is 352 × 288. In the first case (4:4:4), we have: 352 × 288 × 24 b/pixel = 2.433 Mb. In the second case (4:2:0): 352 × 288 × 12 b/pixel = 1.216 Mb.

5.1 Digital Video Standards

The standard for digital video broadcasting is ITU-R BT.601–5 (International Telecommunication Union, Radiocommunications Sector – ITU-R). This standard specifies:

- 60 fields/s for NTSC or 50 fields/s for PAL system
- NTSC requires 525 lines per frame, while PAL system requires 625 lines, with 8 b/sample in both systems.

The bit rate of video data is 216 Mb/s in both cases.

In addition to the considered sampling schemes, an important resolution parameter is video signal format. The most frequently used formats are:

- 4CIF with the resolution 704 × 576, and this format corresponds to broadcasting standards
- CIF 352 × 288
- QCIF 176 × 144
- SubQCIF 128 × 96

The video signal bit rate depends on the video frame format. The mentioned bit rate of 216 Mb/s corresponds to the quality used in standard television. Only 187 s of such a signal can be stored on a 4.7 GB DVD. For the CIF format with 25 frames/s and the 4:2:0 scheme, we achieve the bit rate of 30 Mb/s. Similarly, for the QCIF format with 25 frames/s, the bit rate is 7.6 Mb/s. Consider now these video bit rates in the context of the ADSL network capacity. For example, typical bit rates in the ADSL networks are 1–2 Mb/s. Hence, it is obvious that the signal must be compressed in order to be transmitted over the network.

Since we will deal with video compression later on, here we only mention that the compression algorithms belong to the International Organization for Standardization (ISO) and International Telecommunication Union (ITU) standards. The MPEG algorithms belong to the ISO standard, while the ITU standards cover VCEG algorithms. In order to improve the compression ratio and the quality of the compressed signal, compression algorithms have been improved over time, so today we have MPEG-1, MPEG-2, MPEG-4, MPEG-7. The VCEG standards include: H.261, H.263, H.264.

5.2 Motion Parameters Estimation in Video Sequences

Motion estimation is an important part of video compression algorithms. One of the simplest methods for motion estimation is a block matching technique. Namely, we consider a block of pixels from the current frame, and in order to estimate its position (motion vector), we compare it with the blocks within a predefined region in the reference frame. As a comparison parameter, it is possible to use the mean square error (MSE) or the sum of absolute errors (SAE):

$$\mathrm{MSE} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(C_{i,j} - R_{i,j}\right)^2, \tag{5.1}$$

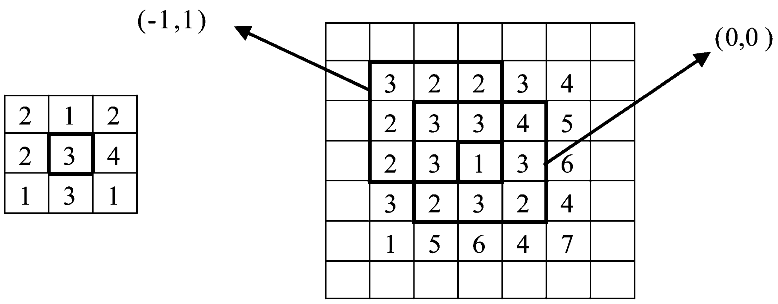


Fig. 5.3 Motion vector estimation for a 3 × 3 block

$$\mathrm{SAE} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left|C_{i,j} - R_{i,j}\right|, \tag{5.2}$$

where R_{ij} and C_{ij} are the pixels in the reference and current frame, respectively. Hence, MSE or SAE are calculated for a set of neighboring blocks in the reference frame. The minimal error is compared with a certain threshold. If the minimal error is below the threshold, the corresponding position represents the motion vector. This vector indicates the motion of the considered block within the two frames. The threshold plays an important role in determining motion vectors. Inappropriate threshold could cause spurious motion vectors, especially in the presence of noise.

Let us illustrate the block matching procedure on a simple example of a 3 × 3 block (larger blocks are used in practical applications), shown in Fig. 5.3.

Compute the MSE for the central position (0,0):

$$\begin{aligned}\mathrm{MSE}_{00} = \Big\{&(2-3)^2 + (1-3)^2 + (2-4)^2 + (2-3)^2 + (3-1)^2 + \\ &(4-3)^2 + (1-2)^2 + (3-3)^2 + (1-2)^2\Big\}/9 = 1.89.\end{aligned} \tag{5.3}$$

In analogy with (5.3), the MSEs for other positions are obtained as:

$$\begin{array}{ll} (-1,-1) \rightarrow 4.11 & (1,0) \quad \rightarrow 4.22 \\ (0,-1) \rightarrow 4.44 & (-1,1) \rightarrow 0.44 \\ (1,-1) \rightarrow 9.44 & (0,1) \quad \rightarrow 1.67 \\ (-1,0) \rightarrow 2.56 & (1,1) \quad \rightarrow 4 \end{array}$$

We see that $\min\{\mathrm{MSE}_{nk}\} = \mathrm{MSE}_{-1,1}$ and the vector (−1,1) is the candidate for motion vector. Assuming that the value 0.44 is below a threshold, the motion vector is determined by (−1,1).

A procedure for motion vectors estimation in the case of larger blocks is analyzed in the sequel, and it is known as the full search algorithm. It compares blocks of size 16×16, within the search area of 31 × 31 pixels. It means that the

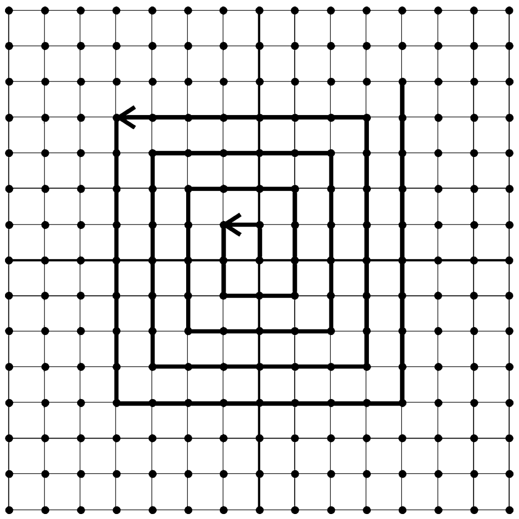


Fig. 5.4 Illustration of full search

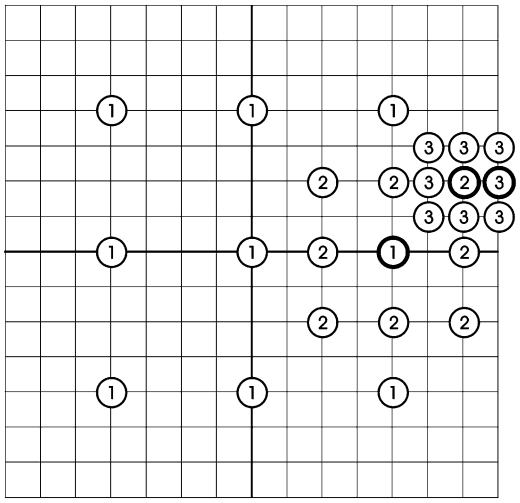


Fig. 5.5 Illustration of three-step search

search is done over 15 pixels on each side from the central position (0,0), Fig. 5.4. This method is computationally demanding, since we need to calculate 31 $\times$ 31 MSEs for 16 $\times$ 16 blocks.

Therefore, the fast search algorithms have been defined to reduce the number of calculations, still providing sufficient estimation accuracy. The search procedure based on the three-step algorithm is shown in Fig. 5.5. In the first step, we observe the eight positions at the distance of p pixels (e.g., $p = 4$) from the central point (0,0). The MSEs are calculated for all nine points (denoted by 1 in Fig. 5.5). The position that provides the lowest MSE becomes the central position for the next step.

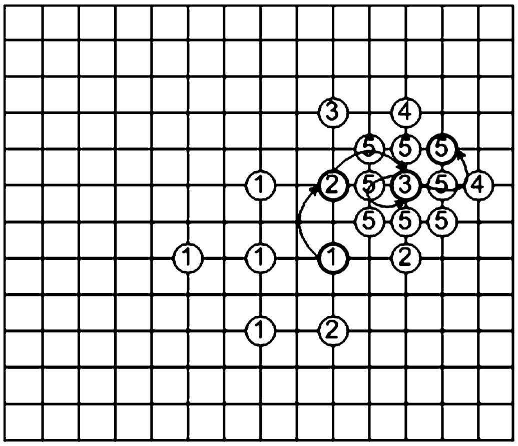


Fig. 5.6 Illustration of the logarithmic search

In the second step, we consider locations on a distance $p/2$ from the new central position. Again, the MSEs are calculated for eight surrounding locations (denoted by 2). The position related to the lowest MSE is a new central position. In the third step, we consider another eight points around the central position, with the step $p/4$. The position with minimal MSE in the third step determines the motion vector.

Another interesting search algorithm is called the logarithmic search (Fig. 5.6). In the first iteration, it considers the position that forms a "+" shape (denoted by 1 in Fig. 5.6). The position with the smallest MSE is chosen for the central point. Then, in the second iteration, the same formation is done around the new central point and the MSE is calculated. The procedure is repeated until the same position is chosen twice in two consecutive iterations. After that the search continues by using the closest eight points (denoted by 5). Finally, the position with the lowest MSE is declared as the motion vector.

The motion vectors search procedures can be combined with other motion parameters estimation algorithms to speed up the algorithms.

5.3 Digital Video Compression

Compression algorithms are of great importance for digital video signals. In fact, as previously demonstrated, the uncompressed video contains large amount of data that requires significant transmission and storage capacities. Hence, the powerful MPEG algorithms are developed and used.

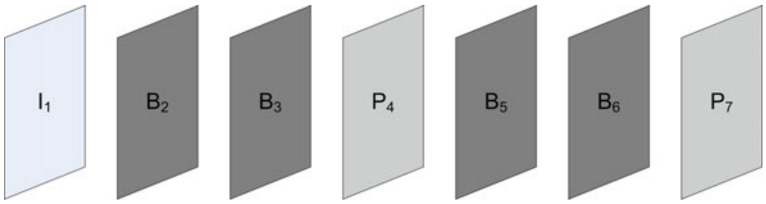


Fig. 5.7 Structure of frames in MPEG-1 algorithm

5.3.1 MPEG-1 Video Compression Algorithm

The primary purpose of the MPEG-1 algorithm was to store 74 min of digital video recording on a CD, with a bit rate 1.4 Mb/s. This bit rate is achieved by using the MPEG-1 algorithm, but with a VHS video quality. A low video quality obtained by the MPEG-1 algorithm was one of the main drawbacks that prevented a wide use of this algorithm. However, the MPEG-1 algorithm served as a basis for the development of the MPEG-2 and was used in some Internet applications, as well.

The main characteristics of the MPEG-1 algorithm are the CIF format (352×288) and the YCrCb 4:2:0 sampling scheme. The basic coding units are 16×16 macroblocks. Therefore, the 16×16 macroblocks are used for the Y component while, given the 4:2:0 scheme, the 8×8 macroblocks are used for the Cr and Cb components. The MPEG-1 algorithm consists of the I, B, and P frames. The I frame is first displayed, followed by B and then by P frames. The scheme continuously repeats as shown in Fig. 5.7.

The I frames are not coded by using motion estimation. Thus, the I frames use only the intracoding, where the blocks are compared within the same frame. P is an intercoded frame and it is based on the forward prediction. It means that this frame is coded by using motion prediction from the reference I frame. B frame is the intercoded frame as well, but unlike the P frame, the forward and backward motion predictions are used. Namely, the motion prediction can be done with respect to the I frame or to the P frame, depending on which gives more optimal results. Hence, the motion vectors are of particular importance in the MPEG algorithms. They are calculated for the blocks of DCT coefficients.

Let us consider the following example. Assume that we have a video scene in which there is a sudden change in the background at the position of the second B frame. In this case, it is much more efficient to code the first B frame with respect to I frame, while the second B frame is coded with respect to the P frame. Having in mind the role of individual frames in video decoding, the sequence of frames used for transmission is depicted in Fig. 5.8.

So, an I frame is transmitted first, followed by P and then by B frames. For the considered case, the frame transfer order is:

I1 P4 B2 B3 P7 B5…

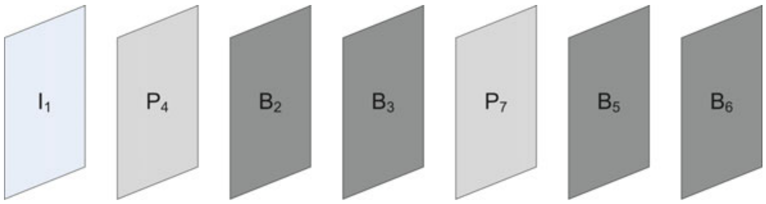


Fig. 5.8 The order of I, B and P frames during transmission

To reconstruct the video sequence, we use the following order:

I1 B2 B3 P4 B5...

The data structure of MPEG-1 algorithms
The data in MPEG-1 are structured in several levels.

1. *Sequence layer*. The level of sequence contains information about the image resolution and a number of frames per second.
2. *Group of pictures layer*. This level contains information about I, P, and B frames. For example, a scheme consisted of 12 frames can be: 1 I frame, 3 P frames, and 8 B frames.
3. *Picture layer*. It carries information on the type of images (e.g., I, P, or B frame), and defines when the picture should be displayed in relation to other pictures.
4. *Slice layer*. Pictures consist of slices, which are further composed of macroblocks. The slice layer provides information about slice position within the picture.
5. *Macroblock layer*. The macroblock level consists of six 8×8 blocks (four 8×8 blocks represent the information about luminance and two 8×8 blocks are used to represent colors).
6. *Block layer*. This level contains the quantized transform coefficients from 8×8 blocks.

5.3.2 MPEG-2 Compression Algorithm

The MPEG-2 is a part of the ITU-R 601 standard and it is still present in digital TV broadcasting. The standard consists of the MPEG-1 audio algorithm, MPEG-2 video algorithm, and a system for multiplexing and transmission of digital audio/video signals.

MPEG-2 is optimized for data transfer at a bit rate 3–5 Mb/s. Unlike the MPEG-1 algorithm, it is based on fields rather than the frames, i.e., the field pictures are coded separately. Recall that one frame consists of two fields: odd- and even-numbered frame lines are placed within two fields. If we observe a 16×16 block, the fields in DCT domain can be represented by using even and odd lines. On the other side, it is possible to split a DCT block into upper and lower blocks of size 16×8. This representation provides a separate motion estimation, which improves the performance of the algorithm, because a significant difference in motion may exist between fields with lower and higher frequencies.

5.3.3 MPEG-4 Compression Algorithm

The MPEG-4 compression algorithm is designed for low bit rates. The main difference in comparison to the MPEG-1 and MPEG-2 is reflected in the object-based coding and content-based coding. Hence, the algorithm uses an object as the basic unit instead of a frame (the entire scene is split into the objects and background). Equivalently to the frame, in the MPEG-4, we have a video object plane. This concept provides higher compression ratio, because the interaction between objects is much higher than among frames.

MPEG-4 video algorithm with very low bit rate (MPEG-4 VLBV) is basically identical to the H.263 protocol for video communications. The sampling scheme is 4:2:0 and it supports formats 16CIF, 4CIF, CIF, QCIF, SubQCIF with 30 frames/s. The motion parameters estimation is performed for 16×16 or 8×8 blocks. The DCT is used together with the entropy coding.

The data structure of MPEG-4 VLBV algorithm is:

1. *Picture layer*. It provides the information about the picture resolution, its relative temporal positions among other pictures, and the type of encoding (inter, intra).
2. *Group of blocks layer*. This layer contains a group of macroblocks (with a fixed size defined by the standard) and has a similar function as slices in the MPEG-1 and MPEG-2.
3. *Macroblock layer*. This consists of four blocks carrying information about luminance and two blocks with chrominance components. Therefore, its header contains information about the type of macroblock, about the motion vectors, etc.
4. *Block layer*. This consists of coded coefficients from the 8×8 blocks.

Shape coding is used to provide information about the shape of video object plane. In other words, it is used to determine whether a pixel belongs to an object or not, and thus, it defines the contours of the video object. The shape information can be coded as a binary (pixel either belongs to the object or not) or gray scale information (coded by 8 bits to provide more description about possible overlapping, pixel transparency, etc).

Objects are encoded by using the 16×16 blocks. Note that all pixels within the block can completely belong to an object, but can also be on the edge of the object. For blocks that are completely inside the object plane, the motion estimation is performed similarly to the MPEG-1 and MPEG-2 algorithms. For the blocks outside the object (blocks with transparent pixels) no motion estimation is performed. For the blocks on the boundaries of the video object plane, the motion estimation is done as follows. In the reference frame, the blocks (16×16 or 8×8) on the object boundary are padded by the pixels from the object edge, in order to fill the transparent pixels. Then the block in the current frame is compared with the blocks in the referent frame. The MSE (or SAE) is calculated only for pixels that are inside the video object plane.

Motion estimation is done for video object plane as follows:

- For the I frame, the motion estimation is not performed;
- For the P frame, the motion prediction is based on the I frame or the previous P frame;

– For the B frame, the video object plane is coded by using the motion prediction from the I and P frames (backward and forward).

The MPEG-4 in its structure contains spatial and temporal scalability. We can change the resolution with spatial scaling, while the time scaling can change the time resolution for objects and background (e.g., we can display objects with more frames/s, and the background with less frames/s). Also, at the beginning of the video sequence, we can transmit larger backgrounds than the one that is actually displayed at the moment. Hence, when zooming or moving the camera, the background information already exists. This makes the compression more efficient.

5.3.4 VCEG Algorithms

The VCEG algorithms are used for video coding and they belong to the ITU standards. Thus, they are more related to the communication applications. Some of the algorithms belonging to this group are: H.261, H.263, and H.264.

5.3.4.1 H.261

This standard was developed in the late 1980s and early 1990s. The main objective was to establish the standards for video conferencing via an ISDN network with a bit rate equal to $p \times 64$ Kb/s. Typical bit rates achieved with this standard are in the range 64–384 Kb/s. The CIF and QCIF formats are used with the 4:2:0 YCrCb scheme. The coding unit is a macroblock containing four luminance and two chrominance blocks (of size 8×8). This compression approach requires relatively simple hardware and software, but has a poor quality of video signals at bit rates below 100 Kb/s.

5.3.4.2 H.263

In order to improve compression performance, the H.263 standard is developed as an extension of H.261. It can support video communication at bit rates below 20 Kb/s with a quite limited video quality that may be used, for example, in video telephony. The functionality of H.263 is identical to the MPEG-4 algorithm. It uses 4:2:0 sampling scheme. The motion prediction can be done separately for each of the four 8×8 luminance blocks or for the entire 16×16 block. The novelty of this approach can be seen in introducing an extra frame, called a PB frame, whose macroblocks contain data from the P and B frames, which increase the efficiency of compression (H.263+ optional modes).

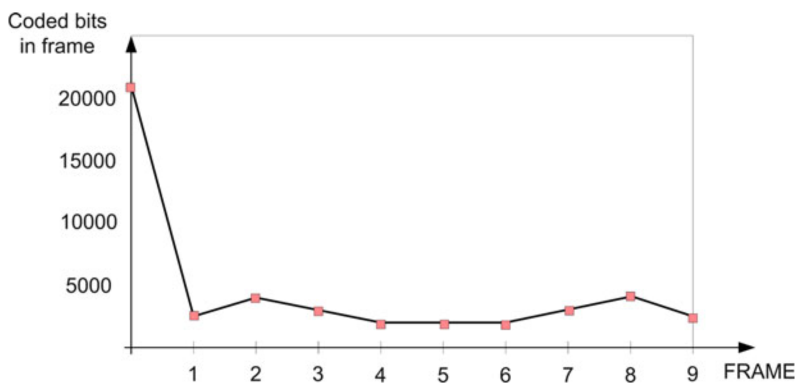


Fig. 5.9 Illustration of the bit rate (bit rate profile)

An illustration of bit rate variations, depicted as a function of frames, is given in Fig. 5.9. Note that the compression of the P frames is up to 10 times higher than that for I frames.

5.3.4.3 H.264/MPEG4-AVC

H.264/MPEG4-AVC is one of the latest standards for video encoding and has been introduced as a joint project of the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). This standard covers many current applications, including the applications for mobile phones (mobile TV), video conferencing, IPTV, HDTV, and HD video applications.

Five Types of Frames

The H.264/MPEG4-AVC supports five types of frames: I, P, B, SP, and SI frames. SP and SI frames are used to provide transitions from one bit rate to another.

Intra 4×4 or Intra 16×16 blocks are used for intracoding. Intra 4×4 coding is based on the prediction of 4×4 blocks, and it is used to encode the parts of images that contain the details. Intra 16×16 coding is based on the 16×16 blocks that are used to encode uniform (smooth) parts of the frame.

SP and SI Frames

The SP and SI frames are specially encoded and they are introduced to provide a transition between different bit rates. These frames are also used to provide other operations such as frame skipping, fast forwarding, transition between two different video sequences, and so on. The SP and SI frames are added only if it is expected that some of these operations may be carried out. The application of these special

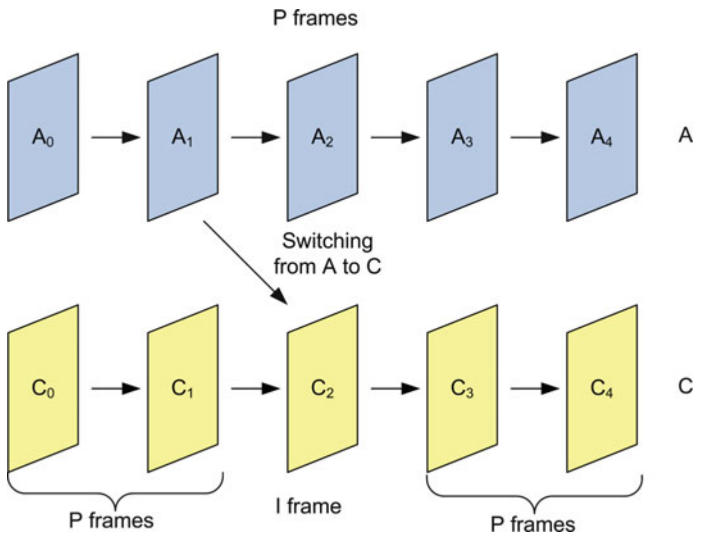


Fig. 5.10 Switching from one to another bit rate by using I frames

frames can be illustrated by the following example. During the transfer of signals over the Internet, the same video is encoded for different (multiple) bit rates. The decoder attempts to decode the video with the highest bit rate, but often there is a need to automatically switch to a lower bit rate, if the incoming data stream drops.

Let us assume that during the decoding of sequence with bit rate A, we have to switch automatically to the bit rate C (Fig. 5.10). Also, assume that the P frames are predicted from one reference I frame. After decoding P frames denoted by A_0 and A_1 (sequence A), the decoder needs to switch to the bit rate C and to decode frames C_2, C_3, etc. Now, since the frames in the sequence C are predicted from other I frames, the frame in A sequence is not an appropriate reference for decoding the frame in C sequence.

One solution is to determine a priori the transition points (e.g., the C_2 frame within the C sequence) and to insert an I frame, as shown in Fig. 5.10.

As a result of inserting I frames, the transitions between two video sequences would produce peaks in the bit rate. Therefore, the SP-frames are designed to support the transition from one bit rate to another, without increasing the number of I frames. Transition points are defined by the SP frames (in the example these are the frames A_2, C_2, and AC_2 that are shown in Fig. 5.11). We can distinguish two types of SP frames: the primary (A_2 and C_2, which are the parts of the video sequences A and C) and the switching SP frames. If there is no transition, the SP frame A_2 is decoded by using the frame A_1, while the SP frame C_2 is decoded using C_1. When switching from A to C sequence, the switching secondary frame (AC_2) is used. This frame should provide the same reconstruction as the primary SP frame C_2 in order to be the reference frame for C_3. Also, the switching frame needs to have characteristics that provide the smooth transition between the sequences. Unlike coding of the P frames, the SP-frames coding requires an additional requantization procedure with a quantization step that corresponds to the step used for the

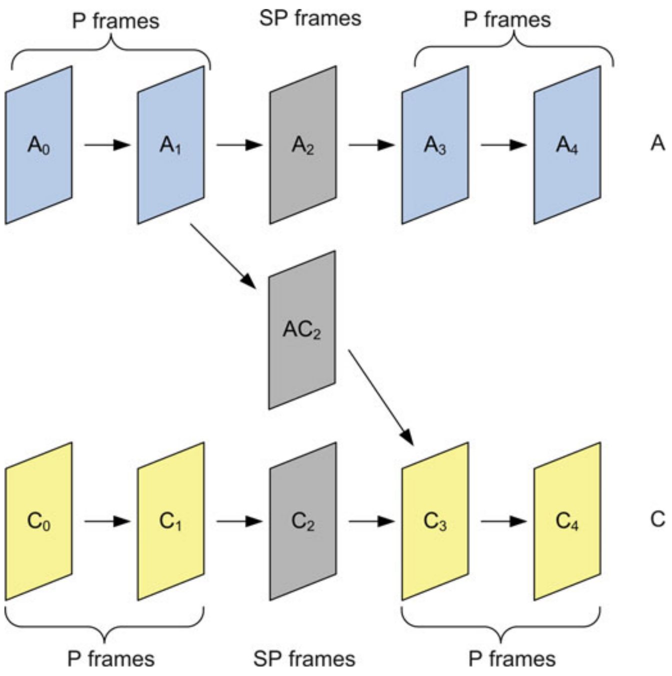


Fig. 5.11 Switching between different bit rates by using SP frames

switching SP frame. Obviously, the switching frame should also contain the information about the motion vectors corrections in order to provide an identical reconstruction in both cases: with and without the switching between the sequences.

In the case of switching from C to A bit rate, the switching frame CA_2 is needed.

Another application of SP frames is to provide arbitrary access to the frames of a video sequence, as shown in Fig. 5.12. For example, the SP frame (A_{10}) and the switching SP frame ($A_{0\text{–}10}$) are on the position of the tenth frame. The decoder performs a fast forward from the A_0 frame to the A_{11} frame, by first decoding A_0, then the switching SP frame $A_{0\text{–}10}$, which will use the motion prediction from A_0 to decode the frame A_{11}.

The second type of transition frames are the SI frames. They are used in a similar way as the SP frames. These frames can be used to switch between completely different video sequences.

Intracoding in the Spatial Domain

Unlike other video encoding standards where intracoding is performed in the transform domain, the H.264/MPEG-4 intracoding is performed in the spatial domain (i.e., the pixel domain).

For the intracoding, the prediction of each 4×4 block is based on the neighboring pixels. Sixteen pixels in the 4×4 block are denoted by a, b,. . ., p (Fig. 5.13a).

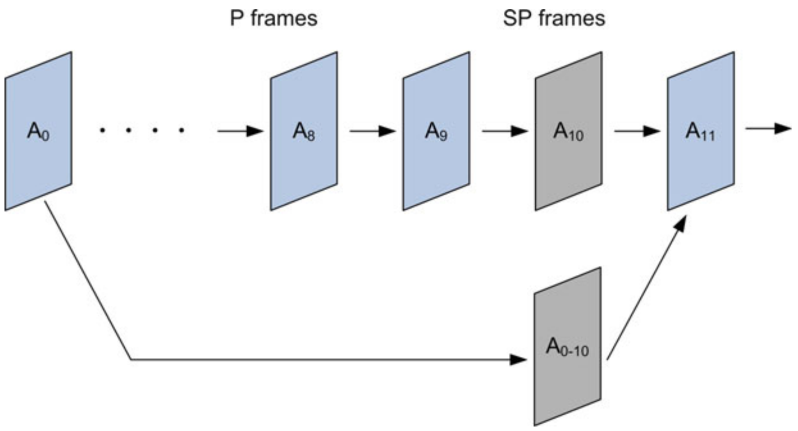


Fig. 5.12 Illustration of the fast-forward procedure using the SP frames

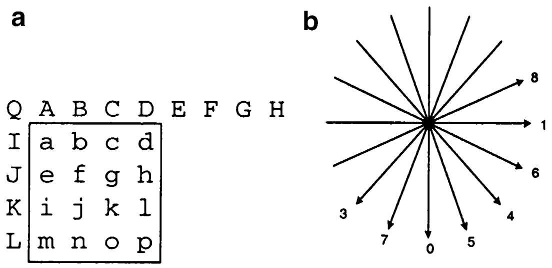


Fig. 5.13 (**a**) Intra 4 × 4 prediction of block a–p based on the pixels A–Q, (**b**) eight prediction directions for Intracoding

They are coded by using the pixels: A, B, C, D, E, F, G, H, I, J, K, L, Q, belonging to the neighboring blocks. Figure 5.13a shows the block that is used in the prediction and Fig. 5.13b depicts prediction directions. Figure 5.14 illustrates the way of using some directions for block prediction.

The vertical prediction, as shown in Fig. 5.14a, indicates that the pixels above the current 4 × 4 block are copied to the appropriate positions according to the illustrated direction. Horizontal prediction indicates that the pixels are copied to the marked positions on the left side. Figure 5.14c, d, and e also present some interesting prediction approaches.

Interframe Prediction with Increased Accuracy of Motion Parameters Estimation

This prediction method uses blocks of sizes 16 × 16, 16 × 8, 8 × 16, and 8 × 8. The 8 × 8 can be further divided into the subblocks of sizes 8 × 4, 4 × 8, or 4 × 4, as shown in Fig. 5.15.

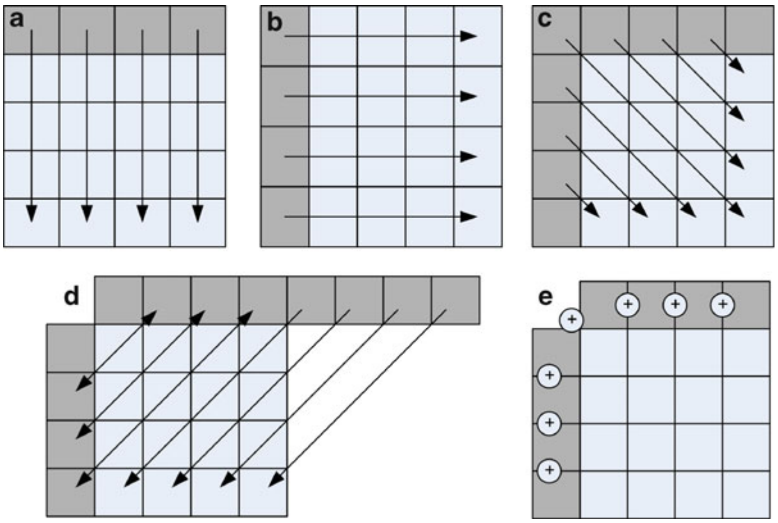


Fig. 5.14 Five (from nine) 4 × 4 intraprediction modes

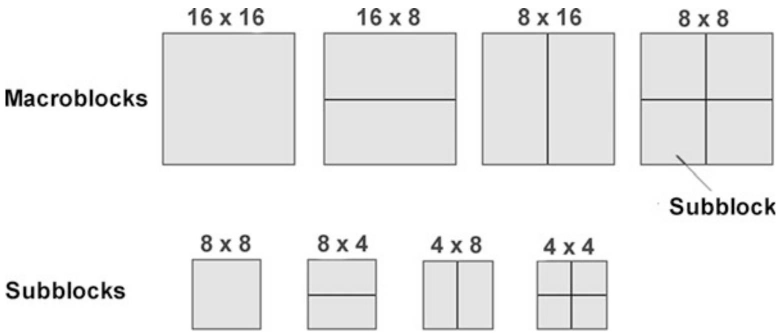


Fig. 5.15 Macroblocks and subblocks

In comparison to other algorithms, the H.264/MPEG-4 standard provides higher precision for the motion vectors estimation. Namely, its accuracy is equal to one quarter of pixels distance in the luminance component. For other algorithms, the precision is usually one-half of the distance. If the motion vector does not indicate an integer position (within the existing pixels grid), the corresponding pixel can be obtained by using the interpolation.

First, a six-tap FIR filter is used to obtain the interpolation accuracy equal to one half. Filter coefficients are (1, −5, 20, 20, −5, 1), and it can be considered as a low-pass filter. Then the bilinear filter is applied to obtain the precision equal to one quarter pixel.

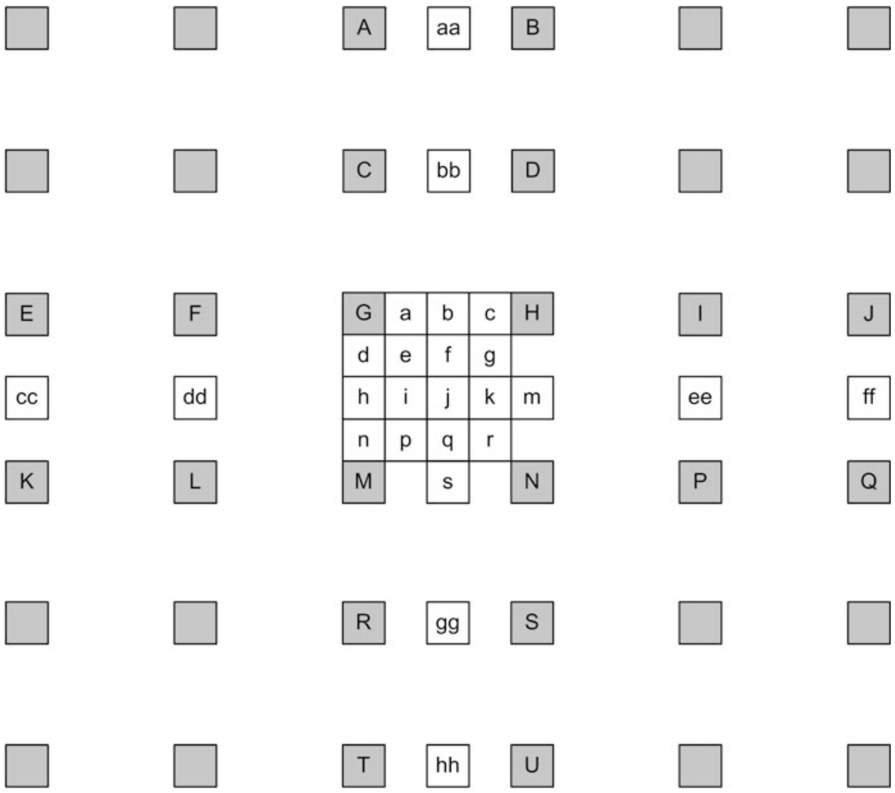


Fig. 5.16 Interpolation method for ¼ pixel precision (luminance component)

Pixels *b, h, j, m*, and *s* (Fig. 5.16) are obtained following the relations:

$$\begin{aligned} b &= ((E - 5F + 20G + 20H - 5I + J) + 16)/32 \\ h &= ((A - 5C + 20G + 20M - 5R + T) + 16)/32 \\ m &= ((B - 5D + 20H + 20N - 5S + U) + 16)/32 \\ s &= ((K - 5L + 20M + 20N - 5P + Q) + 16)/32 \\ j &= ((cc - 5dd + 20h + 20m - 5ee + ff) + 512)/1024 \\ or\ j &= ((aa - 5bb + 20b + 20s - 5gg + hh) + 512)/1024 \end{aligned} \tag{5.4}$$

To obtain a pixel *j*, it is necessary to calculate the values of pixels *cc, dd, ee,* and *ff,* or *aa, bb, gg,* and *hh*. Pixels placed at the quarter of the distance between the pixels *a, c, d, e, f, g, i, k, n, p, q* are obtained as:

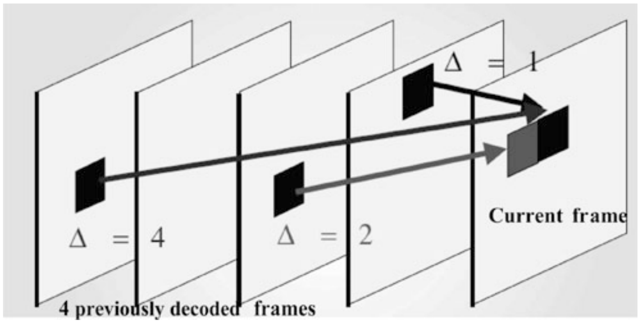


Fig. 5.17 Multiple reference frames

$$\begin{aligned}
a &= \frac{(G+b+1)}{2} & c &= \frac{(H+b+1)}{2} \\
d &= \frac{(G+h+1)}{2} & n &= \frac{(M+h+1)}{2} \\
f &= \frac{(b+j+1)}{2} & i &= \frac{(h+j+1)}{2} \\
k &= \frac{(j+m+1)}{2} & q &= \frac{(j+s+1)}{2} \\
e &= \frac{(b+h+1)}{2} & g &= \frac{(b+m+1)}{2} \\
p &= \frac{(h+s+1)}{2} & r &= \frac{(m+s+1)}{2}
\end{aligned} \tag{5.5}$$

Multiple Reference Frames

The H.264 introduces the concept of multiple reference frames. Specifically, the decoded reference frames are stored in the buffer. It allows finding the best possible references from the two sets of buffered frames (List 0 is a set of past frames, and List 1 is a set of future frames). Each buffer contains up to 16 frames. The prediction for the block is calculated as a weighted sum of blocks from different multiple reference frames. It is used in the scenes where there is a change in perspective, zoom, or the scene where new objects appear (Fig. 5.17).

Another novelty with the H.264 standard is a generalization of the B frames concept. B frames (bidirectional frames) can be encoded so that some macroblocks are obtained as the mean prediction, based on different frames from the List 0 and List 1. Hence, H.264/MPEG-4 allows three types of interprediction: List 0, List 1, and bidirectional prediction.

Coding in the Transform Domain Using the Integer Transform

The H.264/MPEG-4 and other coding standards encode the difference between the reference frame and the frame obtained by prediction. However, unlike the previous standards (such as MPEG-2 and H.263), based on the DCT coefficients, the H.264/MPEG-4 uses integer transform (based on the 4 × 4 or 8 × 8 transform matrices), which is simpler to implement and allows more accurate inverse transform. The commonly used 4 × 4 transform matrix is given by:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}.$$

The H.264 provides significantly better compression ratio than the existing standards. At the same time, it uses advanced entropy coding, such as Context Adaptive Variable Length Coding (CAVLC), and especially Context-based Adaptive Binary Arithmetic Coding (CABAC).

5.4 Data Rate and Distortion

The video sequences, in general, have variable bit rates that depend on several factors:

- Applied algorithms – intra- and intercoding techniques use different compression approaches. Hence, it is clear that different types of frames have different compression factors.
- Dynamics of videos sequence – compression will be higher in the sequences where there are fewer movements and moving objects.
- Encoding parameters – the choice of quantization steps will also influence the bit rate.

  It is obvious that the video compression ratio is closely related to the degree of quality distortion, which is in turn related to the degree of quantization Q. Therefore, an important issue is to provide a compromise between the data rate and quality distortion. This issue can be described by:

$$\min\{D\} \quad \text{for} \quad R \leq R_{\max}, \tag{5.6}$$

  where D is a distortion, while R is the data rate. The algorithm searches for an optimal combination of D and R. It can be summarized as follows:

- Encode a video signal for a certain set of compression parameters and measure the data rate and distortion of decoded signals.

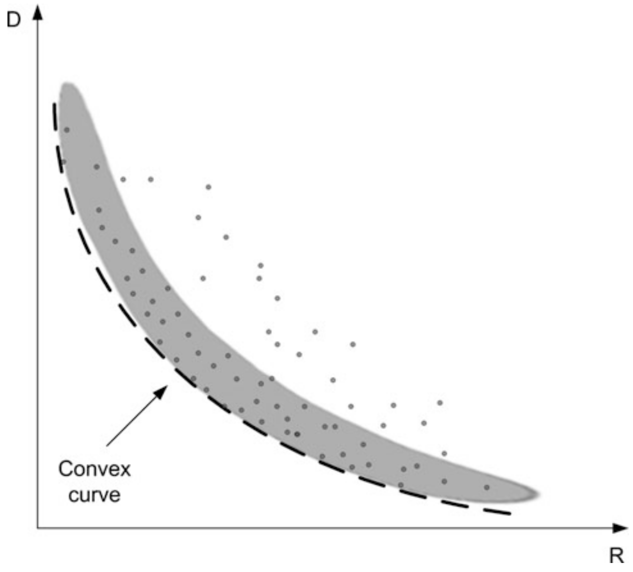


Fig. 5.18 Determining an optimal point for distortion-data rate compromise

- Repeat the coding procedure for different sets of compression parameters, which will produce different compression ratio. For each compression ratio, a measure of distortion is calculated.
- As a result, different points in the R-D (rate-distortion) plane are obtained.

The optimal point in the R-D plane is obtained by using the Lagrangian optimization:

$$\min\{J = D + \lambda R\}, \tag{5.7}$$

where λ is the Lagrange constant. It will find the nearest point on the convex optimization curve (Fig. 5.18).

In practice, most applications require the constant bit rate for the video signals. For this purpose, the bit rate control system, shown in Fig. 5.19, can be used.

R_V indicates the variable bit rates, while R_C denotes the constant ones. The system buffers a video signal with variable bit rate obtained at the output of coder, and then transmits the buffered signal with the constant bit rate. A buffer feedback controls the quantization step size Q and, consequently, the compression ratio. Namely, when the bit rate of the input signal to the buffer is high, the buffer may overflow. Then, the quantization step should be increased to increase compression and to reduce the bit rate at the output of the encoder. However, there are situations when the bit rate increases rapidly during one frame. In these cases, this system produces an image with significant quality variations, as it is shown in Fig. 5.20.

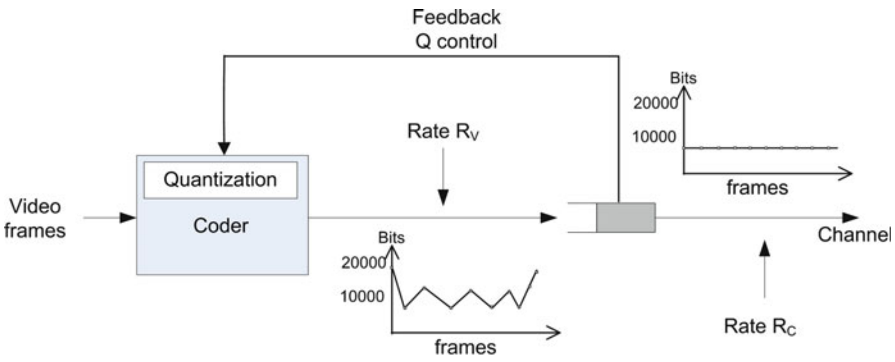


Fig. 5.19 Rate control for video signal

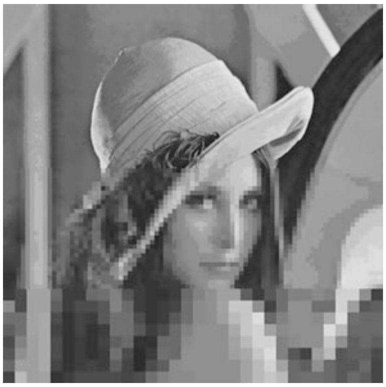


Fig. 5.20 Illustration of changes in quality due to reduced rate

A field dealing with matching the video quality and transmission capacity of the network is called the Quality of Service (QoS). On the one side, we have the QoS required by the application; and on the other side, the available QoS offered by the network. QoS differs for different video applications and transmission scenarios. For example, a one-sided simplex transmission (broadcasting) requires a different QoS compared to a two-sided duplex transmission (video conferencing). In simplex, it is important to have video and audio synchronization, because the synchronization loss greater than 0.1 s becomes obvious. In duplex, delays greater than 0.4 s cause difficulties and unnatural communication.

Digital data can be carried over networks with constant or variable bit rates. Networks with constant rates are the Public Switched Telephone Networks (PSTN) – circuit switched networks and Integrated Services Digital Networks (ISDN). Networks with variable bit rates are the Asynchronous Transfer Mode networks (ATM – packet switched networks), where the bit rate depends on the traffic within the network.

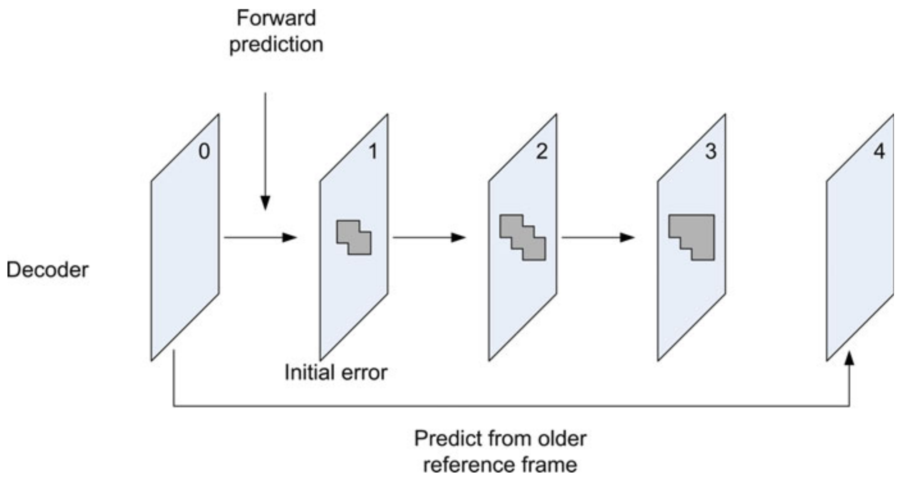


Fig. 5.21 Error reduction by using the older reference frame

Errors that can occur when transferring video material can generally be divided into spatial and temporal. The spatial error occurs in one of the macroblocks within the frame and it affects other blocks that are intracoded by using the erroneous block (Fig. 5.21).

Correcting these errors is done by interpolation, using the undamaged parts of the frame. The most prominent time errors are those that occur in the initial frame and they are transmitted through the motion vectors to B and P frames. When such an error is detected, then the motion prediction is done by using the previous error-free reference frame. An illustration of removing these errors is given in Fig. 5.21.

5.5 Communications Protocols for Multimedia Data

In this part, we provide an overview of some multimedia protocols used in different networks (other protocols are discussed in Chap. 9). In the PSTN (typically ISDN), H.324 and H.320 protocols are used, and both have a constant bit rate. For multimedia data over the IP and LAN, the H.323 protocol can be used, which has a variable delay and unreliable data transfer.

5.6 H.323 Multimedia Conference

The H.323 protocol provides the multimedia communication sessions (voice and videoconferencing in point-to-point and multipoint configurations). This standard involves call signaling, control protocol for multimedia communication, bandwidth

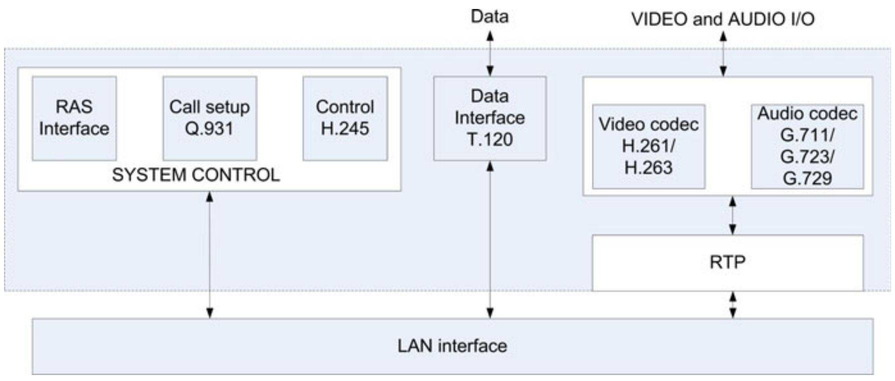


Fig. 5.22 H.323 terminal

control, etc. The H.323 network usually includes four components: the H.323 terminal, gatekeeper, gateway, and multipoint control units (MCU). The H.323 terminals are the endpoints on the LAN that provide real-time communications. The gateway provides communication between H.323 networks and other networks (PSTN or ISDN). The gatekeeper is used to translate IP addresses and to manage the bandwidth. The MCU allows communication between multiple conference units.

The structure of the H.323 terminal is given in Fig. 5.22.

This protocol requires the audio coding and control protocols, while the video coding and Real-Time Transport Protocol (RTP) are optional. The audio signals are encoded using the G.711, G.723, and G.729 standards, while the video signals are encoded using the H.261 and H.263 standards. The block Q.931 is used to set-up the calls, the H.245 block controls the operation of the network, and the RAS block is used to communicate with the gatekeeper.

A centralized conference assumes that all connections are routed through the MCU (unicast communication). Then, the MCU is loaded. In a decentralized conference (multicast communication), each terminal sends data to all other terminals. The basic transport protocol in the H.323 is the User Datagram Protocol (UDP) that will be explained in details in Chap. 9 – Multimedia Communications.

5.6.1 SIP Protocol

The Session Initiation Protocol (SIP) is a protocol designed for the session control in the multiservice networks. The software that provides real-time communications between the end users can use SIP to establish, maintain, and terminate the communication between two or more endpoints. These applications include the voice over IP (VoIP), video teleconferencing, virtual reality applications, multiplayer video games, etc. The SIP does not provide all the functions required for communication between these programs, but it is an important component that facilitates communication.

One of the major demands that the network should meet is the maintenance of QoS for the client application. The SIP is a client–server protocol, based on the protocols HTTP (HyperText Transfer Protocol) and SMTP (Simple Mail Transfer Protocol). The SIP can use either UDP or Transmission Control Protocol (TCP) as a transport protocol.

The SIP messages can be in a form of a *request* (from a client to a server) or a *response* (from a server to a client).

SIP performs five basic functions:

- Determines the location of endpoint
- Determines the availability of endpoint, i.e., whether the endpoint is able to participate in a session
- Determines the characteristics of users, i.e., the parameters of the medium that are essential for communication
- Establishes a session or performs the exchange of parameters for establishing a session
- Manages sessions

One of the main reasons for using the SIP is to increase flexibility in multimedia data exchange. Specifically, users of these applications can change the location and use different computers, with multiple user names and user accounts, or to communicate using a combination of voice, text and other media that require different protocols separately. The SIP uses various components of the network to identify and locate the users. The data go through a proxy server that is used to register and forward the user connection requests. Given that there are different protocols for voice, text, video, and other media, the SIP is positioned above any of these protocols.

The SIP architecture is illustrated in Fig. 5.23. The SIP is independent of network topology and can be used with different transport protocols such as the UDP, TCP, X.25, ATM AAL5, CLNP, TP4, IPX, and PPP. The SIP does not require a reliable transport protocol, and therefore, the client side can use the UDP. For servers, it is recommended to support both protocols, the UDP and TCP. The TCP connection is opened only when the UDP connection cannot be established.

The functionality of SIP is mainly based on signaling. This is its main difference in comparison to the H.323, which includes all necessary functions to carry out the conference. The SIP architecture is designed to be modular so that the different functions can be easily replaced. The SIP environment can implement some components of the H.323 protocol.

For a description of multimedia sessions, SIP uses the Session Description Protocol (SDP). To transfer in real time, the SIP architecture includes the RTP protocol. It also includes the Real-Time Streaming Protocol (RTSP), which is a control protocol for streaming multimedia data in real-time. This protocol is suitable for audio/video on-demand streaming.

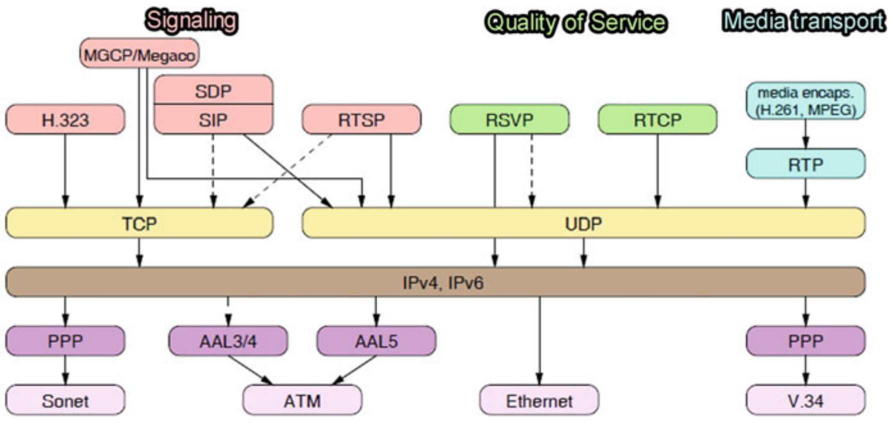


Fig. 5.23 SIP architecture

In the SIP protocol, the following methods are used:

INVITE – making the connection
BYE – end connection
OPTIONS – indicates information about the possibilities
ACK – is used for reliable messaging
CANCEL – cancels the last request
REGISTER – SIP server provides information about the location

5.7 Audio Within a TV Signal

Audio signal together with a video sequence is an integral part of the TV signal. Inserting audio in the video signal requires knowledge of many different disciplines like compression algorithms, multiplexing, standards for packetized data stream and algorithms for signal modulation. In the case of digital TV, the compression algorithms have the main influence to the received signal quality. A system for transmission of audio and video data is an iso-synchronized system. This means that both transmitter and receiver use the data buffering to avoid asynchronous data. Video and audio data from a channel form the elementary stream (ES). Multiple program channels are combined such that the variable-length elementary stream is packetized into the fixed length transport stream packets. A simplified block diagram of this system is shown in Fig. 5.24.

The metadata provides the synchronization of audio and video data (the timing reference). It should be noted that the stream of coded audio and video data is packetized by using the Packetized Elementary Stream (PES) blocks, which have a defined structure for both video and audio data. In video compression, the frames are not included in the same order as they are generated, so that the video block

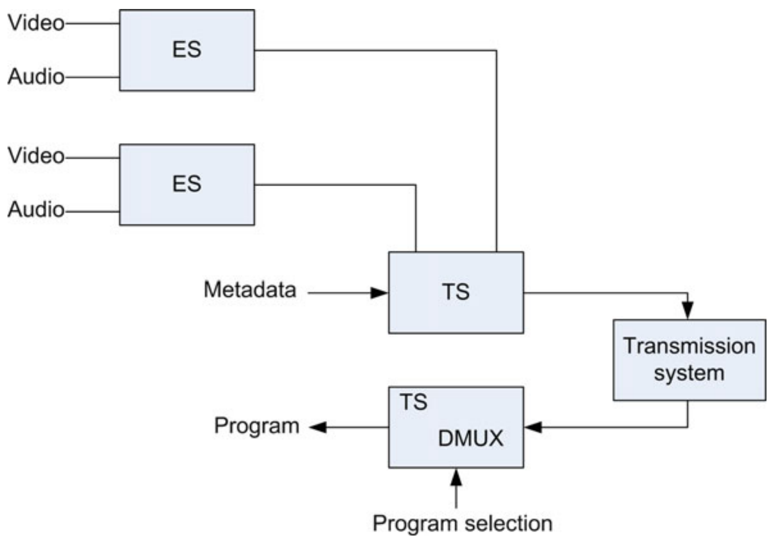


Fig. 5.24 Transport stream multiplexing and demultiplexing

must contain a part that takes care when the frame is played. The Transport Stream (TS) is composed of the fixed length packets (188 bytes). The TS packet is made of the header and the data. Multiplexer is an important part of the system, because it combines data from various channels.

5.8 Video Signal Processor

An example of a simple video processor (VCPex processor) is shown in Fig. 5.25.

The RISC is the major processor. The SRAM bus is used for lower bit rates, such as the compressed data (audio and video). The DRAM bus is used for higher bit

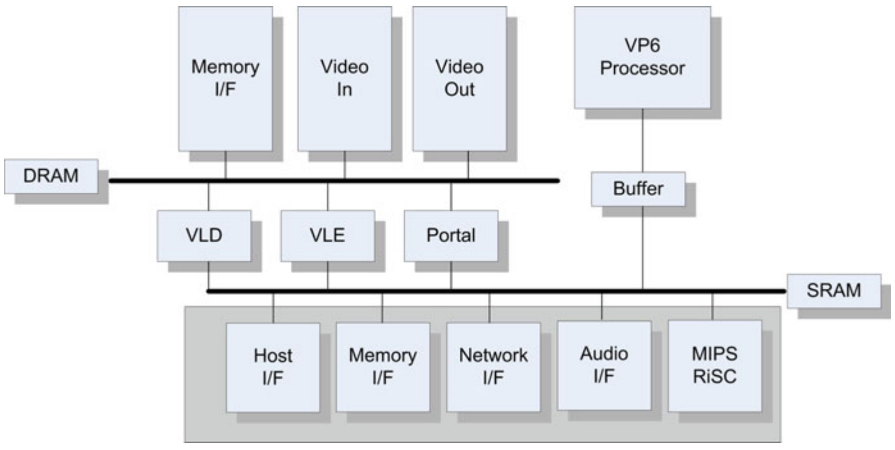


Fig. 5.25 Video processor

rates, as it is the case with the uncompressed material. The RISC and VP6 processor can be reprogrammed to support different coding standards. The variable-length encoding (VLE) and variable-length decoding (VLD) are used to encode and decode the signal.

5.9 Examples

5.1. Determine the number of bits used to represent 16 pixels, by using the following sampling schemes:

(a) 4:4:4
(b) 4:2:2
(c) 4:2:0

Solution:

(a) 4:4:4
We observe four blocks with four pixels, each having the three components (Y,Cr,Cb)

$$4 \times (4 \times 3 \times 8)\ \text{b} = 4 \times 96\ \text{b} = 384\ \text{b} \quad \text{or} \quad 16 \cdot 24\ \text{b/pixel} = 384\ \text{b/pixel}$$

(b) 4:2:2
According to this scheme, two out of four pixels within the observed block are represented by using three components (Y,Cr,Cb), while the remaining two pixels contain just the luminance Y.

$$4 \times (2 \times 3 \times 8\ \text{b}) + 4 \times (2 \times 1 \times 8\ \text{b}) = 256\ \text{b} \quad \text{or}$$
$$16 \cdot 16\ \text{b/pixel} = 256\ \text{b/pixel}$$

(c) 4:2:0
In this case, only one pixel is represented with a full resolution (Y,Cr,Cb), while the remaining three pixels contains the luminance components Y. Hence, for the observed 16 pixels, we have:

$$4 \times (1 \times 3 \times 8\ \text{b}) + 4 \times (3 \times 1 \times 8\ \text{b}) = 192\ \text{b} \quad \text{or}$$
$$16 \cdot 12\ \text{b/pixel} = 192\ \text{b/pixel}$$

5.2. Determine the bit rate of the PAL video sequence for the CIF format and sampling schemes:

(a) 4:4:4
(b) 4:2:2
(c) 4:2:0

Solution:

The CIF format resolution is 352 $\times$ 288. Hence, we obtain the following bit rates:

(a) $352 \times 288 \times 24 \times 25$ b/s $= 60825600$ b/s $= 60.8$ Mb/s
(b) $352 \times 288 \times 16 \times 25$ b/s $= 40550400$ b/s $= 40.5$ Mb/s
(c) $352 \times 288 \times 12 \times 25$ b/s $= 30412800$ b/s $= 30.4$ Mb/s

5.3. How many minutes of the uncompressed video data in the CIF format with sampling scheme 4:2:2 can be stored on a DVD (capacity 4.7 GB)? The PAL system is assumed.

Solution: $t = \frac{4.7 \cdot 1024^3 \cdot 8 \ \mathrm{b}}{25 \cdot 352 \cdot 288 \cdot 16 \cdot 60 \ \mathrm{b/s}} \approx 16.6 \ \ \mathrm{min}$

5.4. Consider a 3 $\times$ 3 block of pixels within a current frame and the 5 $\times$ 5 region centered at the same position in the reference frame. Determine the motion vector by using the block matching technique based on the MSE and assuming that the motion vector is within the given 5 $\times$ 5 block. The threshold value is 2.

1	**4**	**7**
9	**11**	**8**
4	**5**	**6**

2	**5**	**7**	**17**	**19**
9	**11**	**8**	**8**	**5**
4	**6**	**6**	**4**	**1**
0	**9**	**15**	**7**	**4**
4	**8**	**7**	**3**	**1**

Solution:

The observed 3 $\times$ 3 block is compared with the corresponding 3 $\times$ 3 block (within 5 $\times$ 5 block) in the reference frame, centered at (0,0). The MSE is calculated. Then, the procedure is repeated for eight positions around the central one.

$$\begin{matrix} 1 & 4 & 7 \\ 9 & 11 & 8 \\ 4 & 5 & 6 \end{matrix} \quad \overset{\text{MSE}_{00}}{\longleftrightarrow} \quad \begin{matrix} 11 & 8 & 8 \\ 6 & 6 & 4 \\ 9 & 15 & 7 \end{matrix}$$

$$MSE_{00} = \Big((1-11)^2 + (4-8)^2 + (7-8)^2 + (9-6)^2 + (11-6)^2 + (8-4)^2 \\ +(4-9)^2 + (515)^2 + (6-7)^2\Big)/9 = 32.55$$

$$\begin{matrix} 1 & 4 & 7 \\ 9 & 11 & 8 \\ 4 & 5 & 6 \end{matrix} \quad \overset{\text{MSE}_{-10}}{\longleftrightarrow} \quad \begin{matrix} 9 & 11 & 8 \\ 4 & 6 & 6 \\ 0 & 9 & 15 \end{matrix}$$

$$\text{MSE}_{-10} = \Big((1-9)^2 + (4-11)^2 + (7-8)^2 + (9-4)^2 + (11-6)^2 + (8-6)^2 \\ +(4-0)^2 + (5-9)^2 + (6-15)^2\Big)/9 = 31.22$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{-11}} \quad \begin{matrix}2&5&7\\9&11&8\\4&6&6\end{matrix}$$

$$\begin{aligned}\text{MSE}_{-11} = \Big(&(1-2)^2+(4-5)^2+(7-7)^2+(9-9)^2+(11-11)^2+(8-8)^2\\&+(4-4)^2+(5-6)^2+(6-6)^2)/9 = 0.33\end{aligned}$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{11}} \quad \begin{matrix}7&17&19\\8&8&5\\6&4&1\end{matrix}$$

$$\begin{aligned}\text{MSE}_{11} = \Big(&(1-7)^2+(4-17)^2+(7-19)^2+(9-8)^2+(11-8)^2+(8-5)^2\\&+(4-6)^2+(5-4)^2+(6-1)^2)/9 = 44.22\end{aligned}$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{-1-1}} \quad \begin{matrix}4&6&6\\0&9&15\\4&8&7\end{matrix}$$

$$\begin{aligned}\text{MSE}_{-1-1} = \Big(&(1-4)^2+(4-6)^2+(7-6)^2+(9-0)^2+(11-9)^2+(8-15)^2\\&+(4-4)^2+(5-8)^2+(6-7)^2)/9 = 17.55\end{aligned}$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{0-1}} \quad \begin{matrix}6&6&4\\9&15&7\\8&7&3\end{matrix}$$

$$\begin{aligned}\text{MSE}_{0-1} = \Big(&(1-6)^2+(4-6)^2+(7-4)^2+(9-9)^2+(11-15)^2+(8-7)^2\\&+(4-8)^2+(5-7)^2+(6-3)^2)/9 = 9.33\end{aligned}$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{1-1}} \quad \begin{matrix}6&4&1\\15&7&4\\7&3&1\end{matrix}$$

$$\begin{aligned}\text{MSE}_{1-1} = \Big(&(1-6)^2+(4-4)^2+(7-1)^2+(9-15)^2+(11-7)^2\\&+(8-4)^2+(4-7)^2+(5-3)^2+(6-1)^2)/9 = 18.55\end{aligned}$$

$$\begin{matrix}1&4&7\\9&11&8\\4&5&6\end{matrix} \quad \xleftrightarrow{\text{MSE}_{10}} \quad \begin{matrix}8&8&5\\6&4&1\\15&7&4\end{matrix}$$

$$\begin{aligned}\mathrm{MSE}_{10} = \Big((1-8)^2 + (4-8)^2 + (7-5)^2 + (9-6)^2 + (11-4)^2 + (8-1)^2 \\ +(4-15)^2 + (5-7)^2 + (6-4)^2\Big)/9 = 33.88\end{aligned}$$

$$\begin{matrix}1 & 4 & 7 \\ 9 & 11 & 8 \\ 4 & 5 & 6\end{matrix} \quad \overset{\mathrm{MSE}_{01}}{\longleftrightarrow} \quad \begin{matrix}5 & 7 & 17 \\ 11 & 8 & 8 \\ 6 & 6 & 4\end{matrix}$$

$$\begin{aligned}\mathrm{MSE}_{01} = \Big((1-5)^2 + (4-7)^2 + (7-17)^2 + (9-11)^2 + (11-8)^2 + (8-8)^2 \\ +(4-6)^2 + (5-6)^2 + (6-4)^2\Big)/9 = 16.33\end{aligned}$$

Since $\min(\mathrm{MSE}_{\mathrm{nm}}) = \mathrm{MSE}_{-11}$, and it is below the threshold ($\mathrm{MSE}_{-11} < 2$), we conclude that the position $(-1,1)$ represents the motion vector.

5.5. At the output of a video coder, the average bit rate is $R = 5.07$ Mb/s for the CIF video format in PAL system. The quantization is done by the matrix Q_1. The bit rate control system sends the information back to the coder to reduce the bit rate by increasing the quantization step. The coder switches to quantization Q_2 and increases the compression degree. Determine the new average bit rate at the coder output.

The quantization matrices Q_1 and Q_2, as well as a sample representative 8×8 block of the DCT coefficients (from video frames) are given below. In order to simplify the solution, one may assume that the ratio between the compression degrees achieved by Q_1 and Q_2 is proportional to the ratio between the number of nonzero DCT coefficients that remain within the representative 8×8 block after quantization.

$$\mathrm{DCT} = \begin{bmatrix} 96 & 35 & 82 & 41 & 11 & 0 & 0 & 0 \\ 70 & 70 & 40 & 21 & 5 & 0 & 0 & 0 \\ 45 & 40 & 20 & 29 & 13 & 19 & 0 & 0 \\ 27 & 44 & 42 & 15 & 0 & 20 & 0 & 0 \\ 34 & 23 & 0 & 35 & 11 & 0 & 10 & 0 \\ 68 & 34 & 32 & 34 & 10 & 10 & 0 & 0 \\ 38 & 25 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix} \quad Q_2 = \begin{bmatrix} 11 & 21 & 31 & 41 & 51 & 61 & 71 & 21 \\ 21 & 31 & 41 & 51 & 61 & 71 & 81 & 91 \\ 31 & 41 & 51 & 61 & 71 & 81 & 91 & 101 \\ 41 & 51 & 61 & 71 & 81 & 91 & 101 & 111 \\ 51 & 61 & 71 & 81 & 91 & 101 & 111 & 121 \\ 61 & 71 & 81 & 91 & 101 & 111 & 121 & 131 \\ 71 & 81 & 91 & 101 & 111 & 121 & 131 & 141 \\ 81 & 91 & 101 & 111 & 121 & 131 & 141 & 151 \end{bmatrix}$$

Solution:

First, we calculate the average number of bits per pixel for the given bit rate $R = 5.07$ Mb/s.

$$R = 25\ \text{frame/s} \cdot 352 \cdot 288\ \text{pixel} \cdot x_1\ \text{b/pixel}$$

$$x_1 = \frac{5.07 \cdot 10^6}{25 \cdot 352 \cdot 288} = 2\ \text{b/pixel}.$$

$$\mathrm{DCT}_{Q1}\begin{bmatrix} 32 & 7 & 12 & 5 & 1 & 0 & 0 & 0 \\ 14 & 10 & 4 & 2 & 0 & 0 & 0 & 0 \\ 6 & 4 & 2 & 2 & 1 & 1 & 0 & 0 \\ 3 & 4 & 3 & 1 & 0 & 1 & 0 & 0 \\ 3 & 2 & 0 & 2 & 1 & 0 & 0 & 0 \\ 5 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathrm{DCT}_{Q2} = \begin{bmatrix} 9 & 2 & 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The number of nonzero coefficients after Q_1 and Q_2 are respectively:

$$\mathrm{No}\{\mathrm{DCT}_{Q1} \neq 0\} = 30, \quad \mathrm{No}\{\mathrm{DCT}_{Q2} \neq 0\} = 15.$$

The average number of bits for the observed 8×8 block is: $Xb1 = x_1 \cdot 64$, when the quantization Q_1 is applied,

while in the case of $Q_2 : Xb2 = x_2 \cdot 64$. Since we assume that the ratio between compression degrees achieved by Q_1 and Q_2 is proportional to the ratio between the number of nonzero coefficients after Q_1 and Q_2, we may write:

$$\frac{\mathrm{No}\{\mathrm{DCT}_{Q_1} \neq 0\}}{\mathrm{No}\{\mathrm{DCT}_{Q_2} \neq 0\}} = \frac{Xb1}{Xb2} = k\frac{30}{15} \quad \Rightarrow \quad x_2 = \frac{x_1}{2k} = \frac{1}{k} \quad \text{b/pixel.}$$

The new average bit rate obtained at the coder output is:

$$R_{\text{new}} = 25\ \text{frame/s} \cdot 352 \cdot 288\ \text{pixels} \cdot x_2 \text{b/pixel} = \frac{2.53}{k}\ \text{Mb/s}$$

5.6. Consider a video sequence with $N = 1{,}200$ frames. The frames are divided into 8×8 blocks, in order to analyze the stationarity of the coefficients. We assume that the stationary blocks do not vary significantly over the sequence duration. The coefficients from the stationary blocks are transmitted only once (within the first frame). The coefficients from the nonstationary blocks change significantly over time. In order to reduce the amount of data that will be sent, the nonstationary coefficients are represented by using K Hermite coefficients, where $N/K = 1.4$.

Determine how many bits are required for encoding the considered sequence and what is the compression factor? The original video frames can be coded by using on average 256 bits per block.

Blocks statistics	
Total number of frames	1,200
Frame size	300×450
Stationary blocks	40%
Nonstationary blocks	60%

Solution:

The stationary blocks are transmitted only for the first frame. Thus, the total number of bits used to represent the coefficients from the stationary blocks is:

$$n_s = \frac{40}{100}\left(\frac{300 \cdot 450}{64} \cdot 256\right) = 216 \cdot 10^3 \text{b}.$$

In the case of nonstationary blocks, we observe the sequences of coefficients that are on the same position within different video frames. Hence, each sequence having $N = 1{,}200$ coefficients, is represented by using K Hermite coefficients, where $N/K = 1.4$ holds. The total number of bits used to encode the coefficients from the nonstationary blocks is:

$$n_n = 1{,}200 \cdot \frac{K}{N} \cdot \left(\frac{60}{100} \cdot \left(\frac{300 \cdot 450}{64} \cdot 256\right)\right) = 2.77 \cdot 10^8 \text{b}.$$

The number of bits that is required for sequence coding is:

$$p = 1{,}200 \cdot 300 \cdot 450 \cdot 4 = 6.4 \cdot 10^8 \text{b}.$$

The compression factor is: $\frac{6.4 \cdot 10^8}{216 \cdot 10^3 + 2.77 \cdot 10^8} = 2.33$.

5.7. A part of the video sequence contains 126 frames in the JPEG format (Motion JPEG – MJPEG format) and its total size is 1.38 MB. The frame resolution is 384×288, while an average number of bits per 8×8 block is $B = 51.2$. Starting from the original sequence, the DCT blocks are classified into stationary S and nonstationary NS blocks. The number of the blocks are $\text{No}\{\text{S}\} = 1{,}142$ and $\text{No}\{\text{NS}\} = 286$. The coefficients from the S blocks are almost constant over time and can be reconstructed from the first frame. The coefficients from the NS blocks are represented by using the Hermite coefficients. Namely, the each sequence of 126 coefficients is represented by 70 Hermite coefficients. Calculate the compression ratio between the algorithm based on the blocks classification and Hermite expansion, and the MJPEG algorithm.

Solution:

A set of 126 frames in the JPEG format requires $\mathrm{No}\{\mathrm{S}\} \cdot B \cdot 126$ bits for stationary and $\mathrm{No}\{\mathrm{NS}\} \cdot B \cdot 126$ bits for nonstationary blocks. In other words, the total number of bits for the original sequence in the MJPEG format is:

$$\mathrm{No}\{\mathrm{S}\} \cdot B \cdot 126 + \mathrm{No}\{\mathrm{NS}\} \cdot B \cdot 126 = \\ (1,142 + 286) \cdot 51.2 \cdot 126 = 9.21 \cdot 10^6\mathrm{b}$$

The algorithm based on the classification of blocks will encode the stationary blocks from the first frame only: $\mathrm{No}\{\mathrm{S}\} \cdot B$.

For nonstationary blocks, instead of 126 coefficients over time, it uses 70 Hermite coefficients, with the required number of bits equal to: $\mathrm{No}\{\mathrm{NS}\} \cdot N \cdot B$.

The total number of bits for stationary and nonstationary blocks is:

$$\mathrm{No}\{\mathrm{S}\} \cdot B + \mathrm{No}\{\mathrm{NS}\} \cdot N \cdot B = 1,142 \cdot 51.2 + 286 \cdot 70 \cdot 51.2 = 1.083 \cdot 10^6\ \mathrm{b}$$

In this example, the achieved compression factor is approximately 8.5 times.

References

1. Akbulut O, Urhan O, Ertürk S (2006) Fast sub-pixel motion estimation by means of one-bit transform. In: Proceedings of ISCIS. Springer, Berlin, pp 503–510
2. Djurović I, Stanković S (2003) Estimation of time-varying velocities of moving objects in video-sequences by using time-frequency representations. IEEE Trans Image Process 12(5):550–562
3. Djurović I, Stanković S, Oshumi A, Ijima H (2004) Motion parameters estimation by new propagation approach and time-frequency representations. Signal Process Image Commun 19(8):755–770
4. Furth B, Smoliar S, Zhang H (1996) Video and image processing in multimedia systems. Kluwer Academic Publishers, Boston
5. Grob B, Herdon C (1999) Basic television and video systems. McGraw-Hill, New York
6. Karczewicz M, Kurceren R (2001) A proposal for SP-frames. ITU-T Video Coding Experts Group meeting, Eibsee, Germany, Doc. VCEG-L-27
7. Karczewicz M, Kurceren R (2003) The SP- and SI-frames design for H.264/AVC. IEEE Trans Circuit Syst Video Technol 13(7):637–644
8. Kaup A (1999) Object-based texture coding of moving video in MPEG-4. IEEE Trans Circuit Syst Video Technol 9(1):5–15
9. Lie WN, Yeh HC, Lin TCI, Chen CF (2005) Hardware-efficient computing architecture for motion compensation interpolation in H.264 video coding. IEEE Int Symp Circuit Syst 3:2136–2139
10. Malvar HS, Hallapuro A, Karczevicz M, Kerofsky L (2003) Low-complexity transform and quantization in H.264/AVC. IEEE Trans Circuit Syst Video Technol 13(7):598–603
11. Marpe D, Wiegand T, Sullivan GJ (2006) The H.264/MPEG-4 advanced video coding standard and its applications. IEEE Commun Mag 44(8):134–143

12. Nisar H, Choi TS (2009) Fast and efficient fractional pixel motion estimation for H.264/AVC video coding. In: International conference on image processing (ICIP 2009), Cairo, Egypt, pp 1561–1564
13. Richardson I (2010) The H.264 Advanced video compression standard. Wiley, Chichester
14. Richardson I (2002) Video codec design. Wiley, Chichester
15. Salomon D, Motta G, Bryant D (2009) Handbook of data compression. Springer, London
16. Stanković S, Djurović I (2001) Motion parameter estimation by using time frequency representations. Electron Lett 37(24):1446–1448
17. Stanković S, Orović I, Krylov A (2010) Video frames reconstruction based on time-frequency analysis and Hermite projection method. EURASIP J Adv Signal Process, Special Issue Time-Freq Anal Appl Multimedia Signals, Article ID 970105, 11 pages
18. Steinmetz R, Nahrstedt K (2004) Multimedia systems. Springer-Verlag, Berlin Heidelberg
19. Sullivan GJ, Wiegand T (2005) Video compression – from concepts to the H.264/AVC standard. Proc IEEE 93(1):18–31
20. Sullivan GJ, Topiwala P, Luthra A (2004) The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions. In: SPIE conference on applications of digital image processing, vol 454. http://dx.doi.org/10.1117/12.564457
21. Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuit Syst Video Technol 13(7):560–576

Chapter 6
Compressive Sensing

According to the Shannon-Nyquist sampling theorem, a signal can be reconstructed from its samples only when the sampling frequency is at least twice higher than the maximal signal frequency ($2f_{\max}$). Obviously the sampling procedure results in a large number of samples for signals with considerably high maximal frequency. As discussed in the previous chapters, in order to store and transmit the signals over the communication channels with generally limited bit rates, it was necessary to develop efficient and sophisticated signal compression algorithms. We saw that the lossy compression algorithms are mainly based on the fact that signals actually contain a large amount of information that is not necessary for perceiving good signal quality. Namely, in a lossy compression, two basic assumptions are used: the first is related to the imperfection of human perception (sense), while the second is related to the specific properties of signals in a certain transform domain. For instance, in the case of images, a large energy compaction in the low-frequency region is achieved by using the DCT transform. Hence, a significant number of coefficients can be omitted without introducing visible image quality degradation.

Although compression algorithms can significantly reduce the total amount of data, the signal acquisition is performed with a large number of samples. So, one can ask if it would be possible to significantly reduce the amount of data (the number of samples) during the acquisition process, i.e., is it always necessary to sample the signals according to the Nyquist criterion? Thus, is it possible to take smaller number of samples and to sample data randomly? If so, in which case would this be possible?

Compressive sensing is a field dealing with the above-defined problem of interest and provides a solution that differs from the classical signal theory approach (Fig. 6.1). Namely, the compressive sensing represents an alternative method for signal acquisition. Based on the compressive sensing concepts, signal reconstruction can be performed by using a fewer number of randomly chosen signal samples. Hence, a certain signal f with N samples can be reconstructed by using a set of measurements obtained by the measurement matrix Φ, which randomly selects only M samples, with $M << N$.

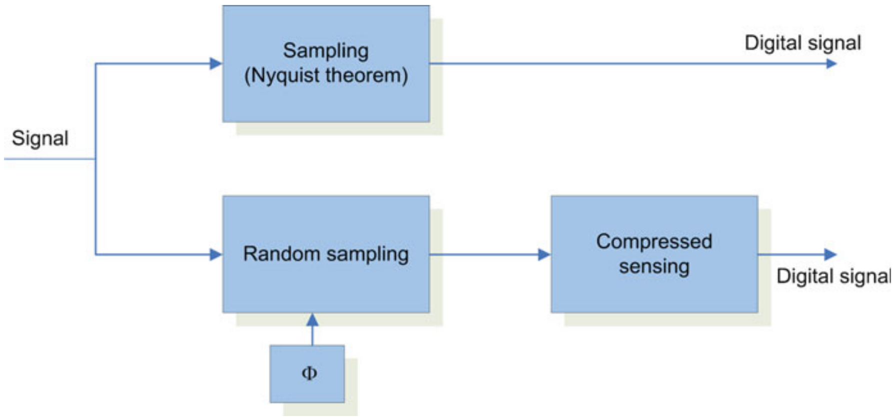


Fig. 6.1 Signal sampling: classical approach and compressive sensing alternative

Compressive sensing is based on powerful mathematical algorithms for error minimization. They are able to reconstruct the original signal by using a small set of signal samples. Sparsity is one of the main requirements that should be satisfied in order to efficiently apply the compressive sensing. Properly chosen basis can provide a sparse signal representation. If the signal is not sparse, then the compressive sensing cannot recover signal successfully.

6.1 The Compressive Sensing Requirements

6.1.1 Sparsity Property

Sparsity means that the signal in a certain transform domain contains just a small number of nonzero coefficients when compared to the signal length. Most of real signals can be considered as sparse or almost sparse if they are represented using the proper basis vectors. A signal f with N samples can be represented as a linear combination of the orthonormal basis vectors as:

$$f(t) = \sum_{i=1}^{N} x_i \Psi_i(t), \quad \text{or} \; : \quad f = \Psi x. \tag{6.1}$$

If the number of nonzero coefficients in x is $K << N$, then the signal:

$$f_K = \Psi x_K, \tag{6.2}$$

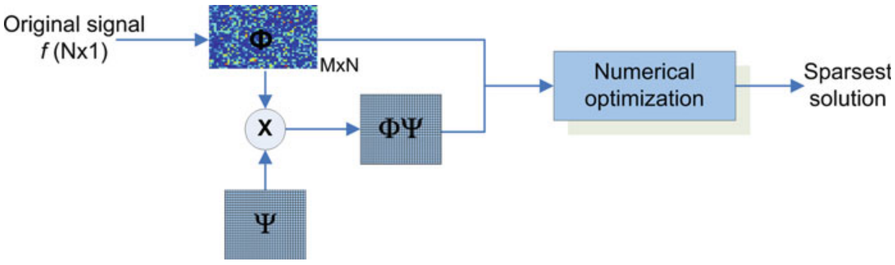


Fig. 6.2 Illustration of the compressive sensing procedure

is said to be K-sparse. Also, since the basis Ψ is orthonormal, we have:

$$\|f - f_K\|_{\ell 2} = \|x - x_K\|_{\ell 2}. \tag{6.3}$$

It means that, if x is well approximated by its K coefficients, the error $\|f - f_K\|_{\ell 2}$ will be sufficiently small.

We may observe that sparsity is a desirable property. The advantages of signal sparsity have been explored in compression algorithms: The coding is done for the K most significant coefficients in the transform domain, while the remaining N-K coefficients are discarded and set to zero. Note that different basis can be used, such as the Fourier basis, the DCT basis, the wavelet basis, etc.

Let us summarize the assumptions we have introduced so far (Fig. 6.2):

– A set of random measurements are selected from the signal f (N $\times$ 1), which can be defined by using the random measurement matrix Φ ($M \times N$) as follows:

$$y = \Phi f. \tag{6.4}$$

– In order to reconstruct f from y, f should be sparse in the transform domain (defined by the orthogonal basis matrix Ψ ($N \times N$)). Hence,

$$f = \Psi x. \tag{6.5}$$

Accordingly, (6.4) and (6.5) can be combined as:

$$y = \Phi\Psi x = Ax. \tag{6.6}$$

The measurement procedure and the measurement matrix should be properly created to provide the reconstruction of signal f (of length N) by using $M << N$ measurements. The reconstructed signal is obtained as a solution of M linear equations with N unknowns. Having in mind that this system is undetermined and can have infinitely many solutions, optimization-based mathematical algorithms should be used to search for the sparsest solution, consistent with the linear measurements.

Another important question is related to the conditions that matrices Ψ and Φ should satisfy in order to make compressive sensing applicable. In that sense, we consider the incoherence requirement in the sequel.

6.1.2 Incoherence

Incoherence is related to the property that signals having sparse representation in the transform domain Ψ should be dense in the domain where the acquisition is performed (e.g., the time domain). For instance, it is well known that the signal represented by the Dirac pulse in one domain is spread in the other (inverse) domain. Hence, the compressive sensing approach assumes that the signal is acquired in the domain where it is rich with samples, so that by using random sampling, we can collect enough information about the signal.

The relationship between the number of nonzero samples in the transform domain Ψ and the number of measurements (required to reconstruct the signal) depends on the coherence between the matrices Ψ and Φ. For example, if Ψ and Φ are maximally coherent, then all coefficients are required for the signal reconstruction. The matrices Φ and Ψ are incoherent if the rows of Φ are spread out in the domain Ψ (the rows of the first matrix cannot provide a sparse representation of the second matrix columns, and vice versa). The coherence between the two matrices Ψ and Φ can be measured as the maximal absolute value of correlation between their elements:

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{k \geq 1, j \leq N} \left| \langle \phi_k, \Psi_j \rangle \right|, \tag{6.7}$$

where N is the signal length, while ϕ_k and Ψ_j are the rows of Φ and columns of Ψ, respectively. Hence, each vector in Φ should be spread out in the transform domain Ψ. The coherence lies within the range:

$$1 \leq \mu \leq \sqrt{N}. \tag{6.8}$$

The minimum value of the coherence is 1, which is the maximum incoherence between the two matrices. If the number of measurements M (selected uniformly at random from Φ) is:

$$M \geq C \cdot K \cdot \mu(\Phi, \Psi) \cdot \log N, \tag{6.9}$$

then the sparsest solution is exact with high probability (C is a constant). It is assumed that the original signal $f \in \mathbb{R}^N$ is K-sparse in Ψ. The concept of incoherence is now clearer: the lower the coherence between Φ and Ψ is, the smaller number of random measurements is required for signal reconstruction.

6.1.3 Restricted Isometry Property

For each integer number K, the isometry constant δ_K of the matrix A is the smallest number for which the relation:

$$(1-\delta_K)\|x\|_{\ell 2}^2 \leq \|Ax\|_{\ell 2}^2 \leq (1+\delta_K)\|x\|_{\ell 2}^2, \tag{6.10}$$

holds for all K-sparse vectors, where $A = \Phi\Psi$. The restricted isometry property means that any subset of columns of the sensing matrix A is nearly orthogonal. We may say that a sensing matrix A satisfies the restricted isometry property with high probability if:

$$M \geq C \cdot K \cdot \log(N/K), \tag{6.11}$$

holds. Furthermore, if this property holds for $\delta_{2K} < 0.414$ or even for $\delta_{2K} < 0.465$, the algorithms for convex optimizations are stable and can be efficient in determining sparse vectors, based on their compressive measurements. Namely, the distances between K-sparse signals must be well preserved in the measurement space, i.e.:

$$(1-\delta_{2K})\|x_1 - x_2\|_{\ell 2}^2 \leq \|Ax_1 - Ax_2\|_{\ell 2}^2 \leq (1+\delta_{2K})\|x_1 - x_2\|_{\ell 2}^2, \tag{6.12}$$

holds for K-sparse vectors x_1 and x_2.

The method for solving the undetermined system of equations (6.6) by searching for the sparsest solution can be described as:

$$\min \|\tilde{x}\|_{\ell_0} \quad \text{subject to } y = A\tilde{x}, \tag{6.13}$$

where $\|x\|_{\ell_0}$ represents the ℓ_0 norm defined as the number of nonzero elements in x. This is a nonconvex combinatorial optimization problem and the solution requires exhaustive searches over subsets of columns of A with exponential complexity.

A more efficient approach uses the near-optimal solution based on the ℓ_1 norm, which is defined as:

$$\|x\|_{\ell_1} = \sum_{i=1}^{N} |x_i|. \tag{6.14}$$

The ℓ_1 norm-based minimization is given by:

$$\min \|\tilde{x}\|_{\ell_1} \quad \text{subject to } y = A\tilde{x}. \tag{6.15}$$

The ℓ_1 norm is convex and thus the linear programming can be used for solving the above optimization problem.

In real applications, we deal with noisy signals. Thus, the previous relation should be modified to include the influence of noise. Namely, it is assumed that in the presence of noise the observations contain an error:

$$y = \Phi\Psi x + e = Ax + e, \tag{6.16}$$

where e represents the error with its energy being limited to $\|e\|_{\ell_2} = \varepsilon$. The optimization problem (6.15) can now be reformulated as follows:

$$\min \|\tilde{x}\|_{\ell_1} \quad \text{subject to } \|y - A\tilde{x}\|_{\ell_2} \leq \varepsilon. \tag{6.17}$$

The ℓ_2 norm is defined as: $\|a\|_{\ell 2} = \sqrt{\sum\limits_{i=1}^{P} (a_i)^2}$, with P being the total number of samples in the vector a.

The reconstructed signal will be consistent with the original one in the sense that $y - A\tilde{x}$ will remain within the noise level.

6.1.4 Numerical Realizations

Most of the numerical methods or algorithms fall into three distinct categories: ℓ_1 minimization, greedy algorithms, and total variation (TV) minimization.

According to (6.15) and (6.17), the general system of equations that should be solved in compressive sensing approach is:

$$\begin{aligned} &\min \|x\|_{\ell_1} \quad \text{s.t. } Ax = y, \\ \text{or,}\quad &\min \|x\|_{\ell_1} \quad \text{s.t. } \|Ax - y\|_{\ell_2} < \xi, \end{aligned} \tag{6.18}$$

where s.t. stands for “subject to.” This approach is known as Basis Pursuit, which was introduced in computational harmonic analysis to extract a sparse signal representation from highly overcomplete dictionaries. The optimization problems can be solved by using some of the known solvers such as the simplex and interior point methods (e.g., primal-dual interior point method).

A modification of (6.18) can be defined as:

$$\min_x \frac{1}{2}\|y - Ax\|_{\ell_2}^2 \quad \text{s.t. } \|x\|_{\ell_1} < \tau, \tag{6.19}$$

which is known as Least Absolute Shrinkage and Selection Operator (LASSO).

Another frequently used approach is the Basis Pursuit denoising (BRDN), which considers solving this problem in Lagrangian form:

$$\min_x \frac{1}{2}\|y - Ax\|_{\ell_2}^2 + \lambda\|x\|_{\ell_1}, \tag{6.20}$$

where $\lambda > 0$ is a regularization parameter.

Commonly used greedy algorithms are Orthogonal Matching Pursuit (OMP) and Iterative Thresholding. The OMP algorithm provides a sparse solution by using an iterative procedure to approximate the vector y as a linear combination of a few columns of A. At each iteration, the algorithm selects the column of A that best correlates with the residual signal. The residual signal is obtained by subtracting the contribution of the partial signal estimate from the measurement vector.

Iterative hard thresholding algorithm starts from an initial signal estimate $\tilde{x} = 0$ and then iterates a gradient descent step followed by hard thresholding until a convergence criterion is achieved.

6.1.5 An Example of Using Compressive Sensing Principles

In order to provide better understanding of the compressive sensing, we will consider a simple example (sinusoidal signal), which aims only to demonstrate some of the concepts introduced in this chapter (such as the vector of measurements, sensing matrices, etc.).

For the purpose of signal visualization, it is a good idea to choose for sinusoid the sample rate, which is 10 (or more) times higher than it is required by the sampling theorem. Hence, a given signal x has $N = 21$ samples and it is defined as:

$$f_x = \sin(2 \cdot \pi \cdot (2/N) \cdot n) \quad \text{for } n = 0, \ldots, 20. \tag{6.21}$$

The values of the signal samples are given in the vector form:

$$\begin{aligned} f_x = [&0 \quad 0.5633 \quad 0.9309 \quad 0.9749 \quad 0.6802 \quad 0.149 \quad -0.4339 \quad -0.866 \\ &-0.9972 \quad -0.7818 \quad -0.2948 \quad 0.2948 \quad 0.7818 \quad 0.9972 \quad 0.866 \\ &0.4339 \quad -0.149 \quad -0.6802 \quad -0.9749 \quad -0.9309 \quad -0.5633] \end{aligned}$$

The Fourier transform of the observed signal consists of two frequency peaks: one belonging to positive and the other to the negative frequencies. The vector containing the Fourier transform coefficients of f_x, denoted as F_x (F_x corresponds to x from the previously presented theory), has the values:

$$F_x = [0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 10.5\,i \ \ 0 \ \ 0 \ \ 0 \ \ -10.5\,i \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0];$$

The signal and its Fourier transform are given in Fig. 6.3.

Note that f_x is sparse in the frequency domain. Hence, we may consider the signal reconstruction based on a small set of randomly selected signal samples. For this purpose we have to define the sensing matrix.

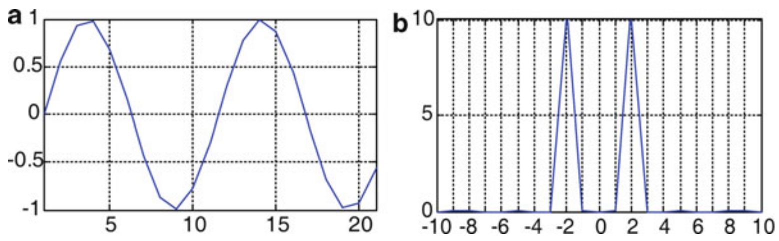


Fig. 6.3 (**a**) Signal f_x, (**b**) Fourier transform F_x

First, we calculate the elements of the inverse and direct Fourier transform matrices, denoted by Ψ and Ψ^{-1}, respectively:

$$\Psi = \frac{1}{21}\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{j\frac{2\pi}{21}} & e^{2j\frac{2\pi}{21}} & e^{3j\frac{2\pi}{21}} & \dots & e^{20j\frac{2\pi}{21}} \\ 1 & e^{2j\frac{2\pi}{21}} & e^{4j\frac{2\pi}{21}} & e^{6j\frac{2\pi}{21}} & \dots & e^{40j\frac{2\pi}{21}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & e^{19j\frac{2\pi}{21}} & e^{38j\frac{2\pi}{21}} & e^{57j\frac{2\pi}{21}} & \dots & e^{380j\frac{2\pi}{21}} \\ 1 & e^{20j\frac{2\pi}{21}} & e^{40j\frac{2\pi}{21}} & e^{60j\frac{2\pi}{21}} & \dots & e^{400j\frac{2\pi}{21}} \end{bmatrix}$$

$$\Psi^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-j\frac{2\pi}{21}} & e^{-2j\frac{2\pi}{21}} & e^{-3j\frac{2\pi}{21}} & \dots & e^{-20j\frac{2\pi}{21}} \\ 1 & e^{-2j\frac{2\pi}{21}} & e^{-4j\frac{2\pi}{21}} & e^{-6j\frac{2\pi}{21}} & \dots & e^{-40j\frac{2\pi}{21}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & e^{-19j\frac{2\pi}{21}} & e^{-38j\frac{2\pi}{21}} & e^{-57j\frac{2\pi}{21}} & \dots & e^{-380j\frac{2\pi}{21}} \\ 1 & e^{-20j\frac{2\pi}{21}} & e^{-40j\frac{2\pi}{21}} & e^{-60j\frac{2\pi}{21}} & \dots & e^{-400j\frac{2\pi}{21}} \end{bmatrix}$$

The matrices are of size $N \times N$. So, the relationship between f_x and F_x is:

$$f_x = \Psi F_x. \tag{6.22}$$

Now, we would like to select $M = 8$ random samples (measurements) in the time domain, which will be used to reconstruct the entire signal f_x by applying the compressive sensing approach. In other words, we should define the measurement matrix Φ of size $M \times N$, which is used to obtain the measurement vector:

$$y = \Phi f_x. \tag{6.23}$$

The measurement matrix Φ can be defined as a random permutation matrix, and thus y is obtained by taking the first M permuted elements of f_x. For instance, the vector y can be given by:

$$y = [0.7818 \quad -0.7818 \quad 0.8660 \quad 0.2948 \quad -0.9972 \quad -0.6802 \quad 0.6802 \quad -0.9309],$$

where the random permutation of $N = 21$ elements is done according to:

$$\text{perm} = [13 \quad 10 \quad 15 \quad 12 \quad 9 \quad 18 \quad 5 \quad 20 \quad 16 \quad 4 \quad 8 \quad 1 \quad 2 \quad 11 \quad 6 \quad 21 \quad 17 \quad 19 \quad 7 \quad 14 \quad 3]$$

or equivalently by taking only its $M = 8$ first elements:

$$\text{perm}(1:M) = [13 \ 10 \ 15 \ 12 \ 9 \ 18 \ 5 \ 20].$$

The N points Fourier transform of the vector y is obtained as:

$$F_y = \Psi^{-1} y, \tag{6.24}$$

resulting in:

$$\begin{aligned} F_y =& [(-0.0793 - 0.0997\,i) \ \ (0.0533 - 0.0739\,i) \ \ (-0.0642 - 0.0514\,i) \\ & (0.0183 + 0.0110\,i) \ (0.1005 + 0.0284\,i) \ \ (-0.0263 - 0.0178\,i) \\ & (0.0704 + 0.0174\,i) \ \ (0.0089 - 0.1007\,i) \ \ (-0.0270 + 0.2307\,i) \\ & (-0.0363 - 0.0171\,i) \ \ (-0.0365 + 0.0000\,i) \ \ (-0.0363 + 0.0171\,i) \\ & (-0.0270 - 0.2307\,i) \ \ (0.0089 + 0.1007\,i) \ \ (0.0704 - 0.0174\,i) \\ & (-0.0263 + 0.0178\,i) \ (0.1005 - 0.0284\,i) \ \ (0.0183 - 0.0110\,i) \\ & (-0.0642 + 0.0514\,i) \ \ (0.0533 + 0.0739\,i) \ (-0.0793 + 0.0997\,i)] \end{aligned}$$

The starting Fourier transform vector F_y significantly differs from F_x, which we aim to reconstruct (Fig. 6.4). Based on (6.22) and (6.23), y can be written as:

$$y = \Phi\Psi F_x.$$

In analogy with the random measurements vector y, the matrix $A = \Phi\Psi$ can be obtained by using the permutation of rows in Ψ and then selecting the first $M = 8$ permuted rows. The matrix $\mathbf{A}$ for this example is defined as:

$$A_{MxN} = \Phi\Psi = \frac{1}{21}\begin{bmatrix} 1 & e^{12j\frac{2\pi}{21}} & e^{24j\frac{2\pi}{21}} & \dots & e^{240j\frac{2\pi}{21}} \\ 1 & e^{9j\frac{2\pi}{21}} & e^{18j\frac{2\pi}{21}} & \dots & e^{180j\frac{2\pi}{21}} \\ 1 & e^{14j\frac{2\pi}{21}} & e^{28j\frac{2\pi}{21}} & \dots & e^{280j\frac{2\pi}{21}} \\ 1 & e^{11j\frac{2\pi}{21}} & e^{22j\frac{2\pi}{21}} & \dots & e^{220j\frac{2\pi}{21}} \\ 1 & e^{8j\frac{2\pi}{21}} & e^{16j\frac{2\pi}{21}} & \dots & e^{160j\frac{2\pi}{21}} \\ 1 & e^{17j\frac{2\pi}{21}} & e^{34j\frac{2\pi}{21}} & \dots & e^{340j\frac{2\pi}{21}} \\ 1 & e^{4j\frac{2\pi}{21}} & e^{8j\frac{2\pi}{21}} & \dots & e^{80j\frac{2\pi}{21}} \\ 1 & e^{19j\frac{2\pi}{21}} & e^{38j\frac{2\pi}{21}} & \dots & e^{380j\frac{2\pi}{21}} \end{bmatrix}$$

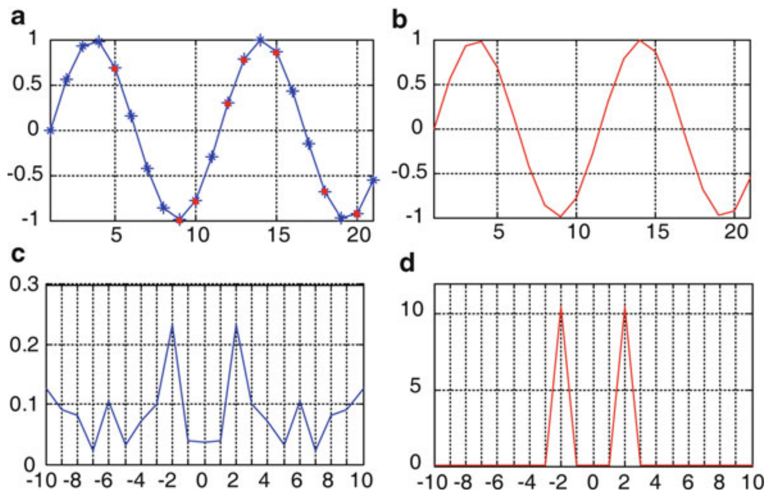


Fig. 6.4 (**a**) Original signal and randomly selected samples denoted by red points, (**b**) Reconstructed signal, (**c**) N point Fourier transform F_y, (**d**) Fourier transform of reconstructed signal

Finally, we obtain the system with 8 equations and 21 unknowns. The ℓ_1 norm minimization with equality constraint:

$$\min \|F_x\|_{\ell_1} \quad \text{s.t. } y = AF_x, \tag{6.25}$$

is recast as a linear program: $\min\limits_{F_x,u} \sum u \quad \text{s.t.} - u \leq F_x \leq u,\ y = AF_x.$

It is solved by applying the primal-dual interior point method. The reconstructed vector F_x is obtained, and then the signal f_x itself: $f_x = \Psi F_x$ (Fig. 6.4).

6.1.5.1 Primal-Dual Interior Point Method

Let us briefly consider the primal-dual interior point method, which has been used to solve most of the optimization problems discussed in this chapter. The optimization problem:

$$\min \|x\|_{\ell 1} \quad \text{s.t. } Ax = b,$$

can be recast as:

$$\min_u \sum u \quad \text{s.t. } Ax = b,\ f_{u_1} = x - u,\ f_{u_2} = -x - u$$

Generally, the minimization problem can be observed by forming the Lagrangian:

$$\Lambda(x, u, v, \lambda_{u_1}, \lambda_{u_1}) = f(u) + v(Ax - b) + \lambda_{u_1} f_{u_1} + \lambda_{u_2} f_{u_2}. \tag{6.26}$$

Finding its first derivatives in terms of x, u, v, λu_1, and λu_2, the following relations are obtained:

$$R_{\text{dual}}^{u} = \mathbf{1} - \lambda_{u_1} - \lambda_{u_2}, \quad R_{\text{dual}}^{\text{x}} = \lambda_{u_1} - \lambda_{u_2} + A^T v, \quad R_{\text{prim}}^{v} = Ax - b, \tag{6.27}$$

$$R_{\text{cent}}^{\lambda u_1} = \lambda_{u_1} f_{u_1} + \frac{1}{\tau}, \quad R_{\text{cent}}^{\lambda u_2} = \lambda_{u_2} f_{u_2} + \frac{1}{\tau}. \tag{6.28}$$

Note that besides A and $b_,$ which are known, we should initialize the following variables: $x = x_0$, $u = u_0$ (e.g., which is obtained by using x_0), λu_1 and λu_2, $v = -A(\lambda_{u_1} - \lambda_{u_2})$ and τ.

In order to compute Newton's steps, the following system of equations is solved:

$$\begin{aligned}
\frac{\partial R_{\text{dual}}^{u}}{\partial x}\Delta x + \frac{\partial R_{\text{dual}}^{u}}{\partial u}\Delta u + \frac{\partial R_{\text{dual}}^{u}}{\partial v}\Delta v &= -R_{\text{dual}}^{u}, \\
\frac{\partial R_{\text{dual}}^{x}}{\partial x}\Delta x + \frac{\partial R_{\text{dual}}^{x}}{\partial u}\Delta u + \frac{\partial R_{\text{dual}}^{x}}{\partial v}\Delta v &= -R_{\text{dual}}^{x}, \\
\frac{\partial R_{\text{prim}}^{v}}{\partial x}\Delta x + \frac{\partial R_{\text{prim}}^{v}}{\partial u}\Delta u + \frac{\partial R_{\text{prim}}^{v}}{\partial v}\Delta v &= -R_{\text{prim}}^{v},
\end{aligned} \tag{6.29}$$

$$\begin{aligned}
\frac{\partial R_{\text{cent}}^{\lambda u_1}}{\partial x}\Delta x + \frac{\partial R_{\text{cent}}^{\lambda u_1}}{\partial u}\Delta u + \frac{\partial R_{\text{cent}}^{\lambda u_1}}{\partial \lambda_{u_1}}\Delta \lambda_{u_1} &= -\partial R_{\text{cent}}^{\lambda u_1}, \\
\frac{\partial R_{\text{cent}}^{\lambda u_2}}{\partial x}\Delta x + \frac{\partial R_{\text{cent}}^{\lambda u_2}}{\partial u}\Delta u + \frac{\partial R_{\text{cent}}^{\lambda u_2}}{\partial \lambda_{u_2}}\Delta \lambda_{u_2} &= -\partial R_{\text{cent}}^{\lambda u_2}.
\end{aligned} \tag{6.30}$$

From (6.29), we have:

$$\begin{gathered}
\left(-\frac{1}{\tau f_{u_1}^2} + \frac{1}{\tau f_{u_2}^2}\right)\Delta x + \left(\frac{1}{\tau f_{u_1}^2} + \frac{1}{\tau f_{u_2}^2}\right)\Delta u = -\mathbf{1} - \frac{1}{\tau}\left(\frac{1}{f_{u_1}} + \frac{1}{f_{u_2}}\right) \\
\left(\frac{1}{\tau f_{u_1}^2} + \frac{1}{\tau f_{u_2}^2}\right)\Delta x + \left(-\frac{1}{\tau f_{u_1}^2} + \frac{1}{\tau f_{u_2}^2}\right)\Delta u + A^T \Delta v = \frac{1}{\tau}\left(\frac{1}{f_{u_1}} - \frac{1}{f_{u_2}}\right) - A^T v \\
A\Delta x = -Ax + b
\end{gathered}$$

After calculating Δx, Δu, and Δv (e.g. by using *linsolve* in Matlab), we compute $\Delta\lambda u_1$ and $\Delta\lambda u_2$ by using:

$$\begin{aligned}
\Delta\lambda_{u_1} &= \lambda_{u_1} f_{u_1}^{-1}(-\Delta x + \Delta u) - \lambda_{u_1} - \frac{1}{\tau f_{u_1}}, \\
\Delta\lambda_{u_2} &= \lambda_{u_2} f_{u_2}^{-1}(\Delta x + \Delta u) - \lambda_{u_2} - \frac{1}{\tau f_{u_2}},
\end{aligned}$$

which are derived from (6.30). The Newton step actually represents the step direction. In order to update the values of variables for the next iteration:

$$x = x + s\Delta x, \quad u = u + s\Delta u, \quad v = v + s\Delta v, \quad \lambda_{u_1} = \lambda_{u_1} + s\Delta\lambda_{u_1},$$
$$\lambda_{u_2} = \lambda_{u_2} + s\Delta\lambda_{u_2}$$

the step length s should be calculated. For this purpose, the backtracking line search can be applied. The general backtracking method is explained in the sequel.

6.1.5.2 Backtracking Method

Let us assume that $f(x)$ is the function that should be minimized. One solution would be to use the step length s ($s \geq 0$), which minimizes the function $f(x_k + s\Delta x_k)$:

$$\arg\min_{s\geq 0} f(x_k + s\Delta x_k).$$

This method can be computationally demanding, and thus the *backtracking line search* has been used:

```
for given $s (s>0)$
while $f(x_{k+1}) > f(x_k) + \alpha \cdot s \cdot f'(x_k) \cdot \Delta x_k$
$s = \beta \cdot s$
end
```

The constants α and β can take values in the range $\alpha \in (0, 0.5), \beta \in (0, 1)$. Since we have five variables that should be updated, the condition in *while* loop can be modified as follows:

$$\|r_{k+1}\|_2 > (1 - \alpha \cdot s)\|r_k\|_2,$$

where r is a vector that contains the elements of $R^u_{\text{dual}}, R^x_{\text{dual}}, R^v_{\text{dual}}, R^{\lambda u_1}_{\text{cent}}, R^{\lambda u_2}_{\text{cent}}$.

6.2 Applications of Compressive Sensing Approach

6.2.1 Multicomponent One-Dimensional Signal Reconstruction

In the sequel, we will apply the compressive sensing method to reconstruct a sparse signal composed of a few nonzero frequency components. Hence, let us consider a

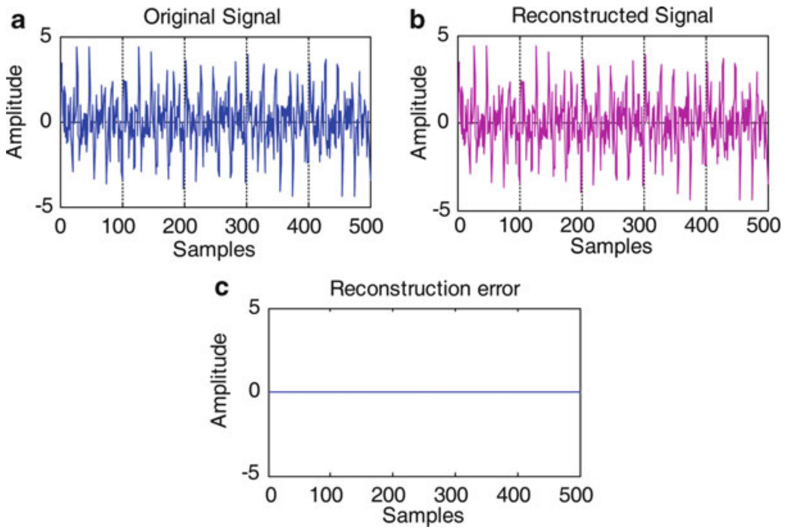


Fig. 6.5 (**a**) Original signal, (**b**) reconstructed signal, (**c**) reconstruction error

signal that is made of five different sinusoids. The signal length is $N = 500$ samples. The analytic form of signal can be written as:

$$f_x(n) = \sum_{i=1}^{5} \sin(2\pi f(i)n/N), \quad n = 0, \dots, N-1, \tag{6.31}$$

where the vector of frequencies is: $f = [25 \quad 45 \quad 80 \quad 100 \quad 176]$;

In the Fourier domain, the signal consists of five components. Consequently, the signal can be considered as sparse in the frequency domain. Thus, the signal reconstruction can be done by using a small set of samples (150) that are chosen randomly from 500 signal samples. The measurements are taken from the time domain, while the sparsifying matrix is obtained by taking the first $M = 150$ rows of the permuted inverse Fourier basis matrix.

The signal is reconstructed by using 30% of the total number of coefficients. The original and the reconstructed signal in the time domain are shown in Fig. 6.5. The original and the reconstructed Fourier transforms of the signal are shown in Fig. 6.6.

The reconstruction error is small and negligible when compared to the signal amplitudes (an average absolute error is $e \sim 10^{-4}$, for the average signal amplitude higher than 1). The maximal and mean absolute errors for different numbers of measurements M are plotted in Fig. 6.7.

Next, we consider an audio signal representing a flute tone, with a total length of 2,000 samples. In this application, we will use the DCT transform domain (for the matrix Ψ). The rows of Ψ are then randomly permuted and the first M rows are used for sensing. The signal is reconstructed by using $M = 700$ random measurements out of $N = 2{,}000$ signal samples. The results are shown in Fig. 6.8. By using the

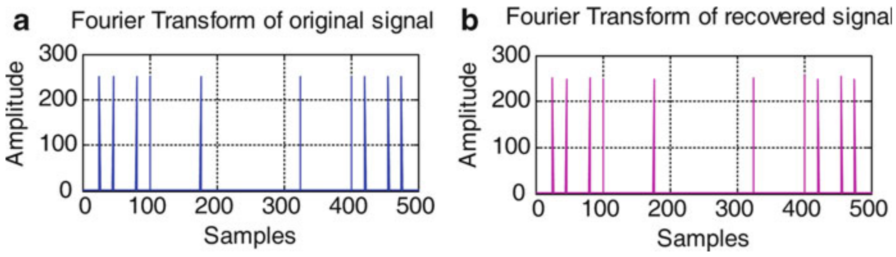


Fig. 6.6 (**a**) Fourier transform of the original signal, (**b**) Fourier transform of the reconstructed signal

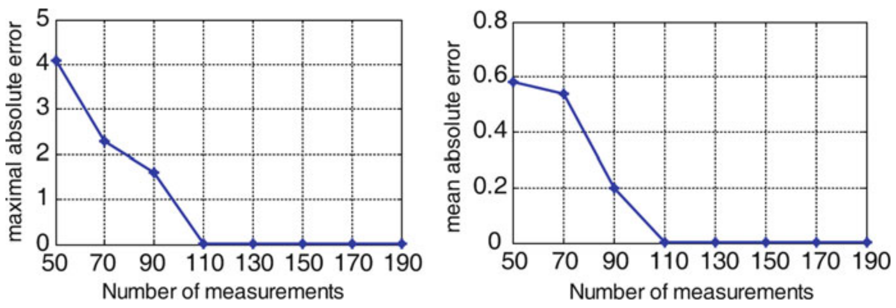


Fig. 6.7 Maximal and mean absolute errors of signal reconstruction

listening test it has been confirmed that the quality of the reconstructed audio file is preserved without introducing any audible distortions.

6.2.2 Compressive Sensing Applied to Image Reconstruction

In this section, we consider the compressive sensing applications to image processing. In order to illustrate the results of applying some of the basic compressive sensing concepts, let us consider the image of size 256 $\times$ 256. First, we split the image into blocks of size 64 $\times$ 64. The compressive sensing is performed as follows:

- Each block is represented as a vector f with $N = 4{,}096$ elements;
- As an observation set we select only $M = 1{,}500$ random measurements (within the vector $\mathbf{y}$) from the block elements;
- The DCT (of size 4,096 $\times$ 4,096) is used, while $A = \Phi\Psi$ is obtained by taking M rows of the randomly permuted transform matrix Ψ.

The original and the reconstructed version of the images "Lena" and "Baboon" are shown in Fig. 6.9. Note that, due to the fact that signal is not strictly sparse in the DCT domain, the reconstructed images would require further processing to enhance their quality.

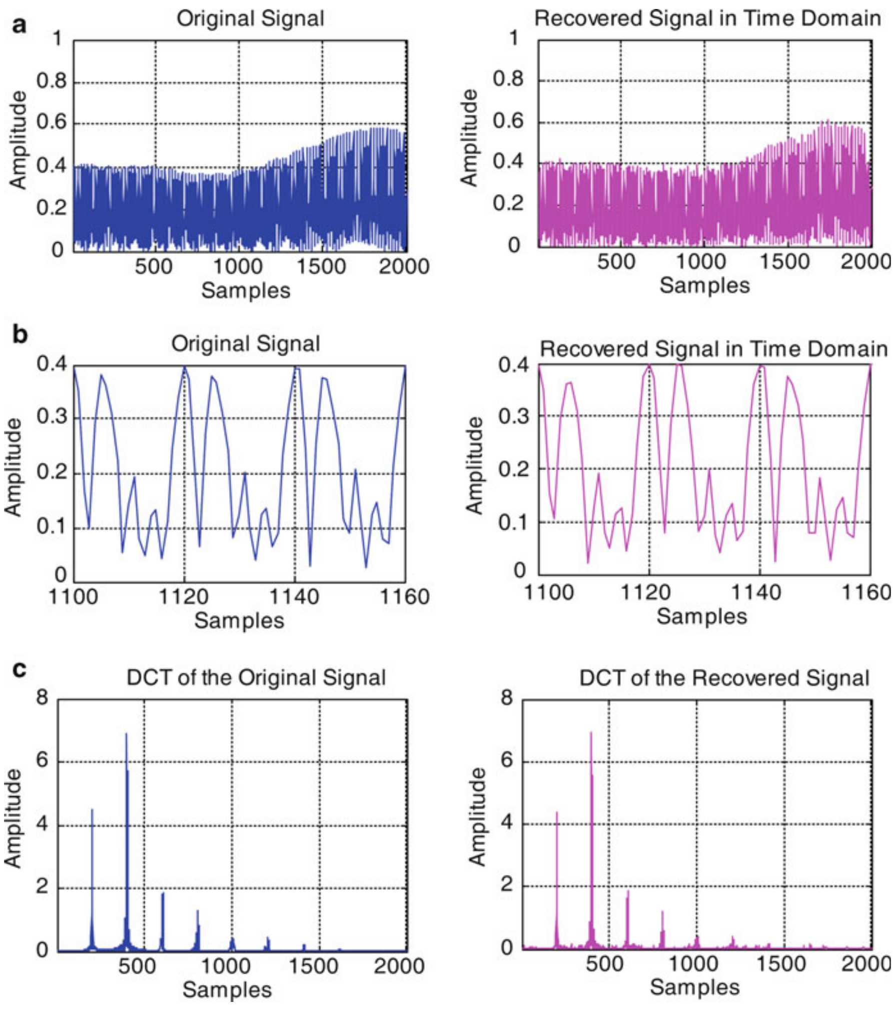


Fig. 6.8 (**a**) Original (*left*) and reconstructed (*right*) flute signal, (**b**) zoomed segment of the original (*left*) and reconstructed (*right*) signal, (**c**) Fourier transform of original (*left*) and reconstructed (*right*) signal

6.2.2.1 Total-Variation Method

One of the approaches used in various image processing applications is based on the variational parameters, i.e., on the total-variation of an image. An example of using the total-variation method is in denoising and restoring of noisy images. If $x_n = x_0 + e$ is a "noisy" observation of x_0, we can restore x_0 by solving the following minimization problem:

$$\min_x \quad \mathrm{TV}(x) \quad \text{s.t. } \|x_n - x\|_{\ell 2}^2 < \varepsilon^2, \tag{6.32}$$

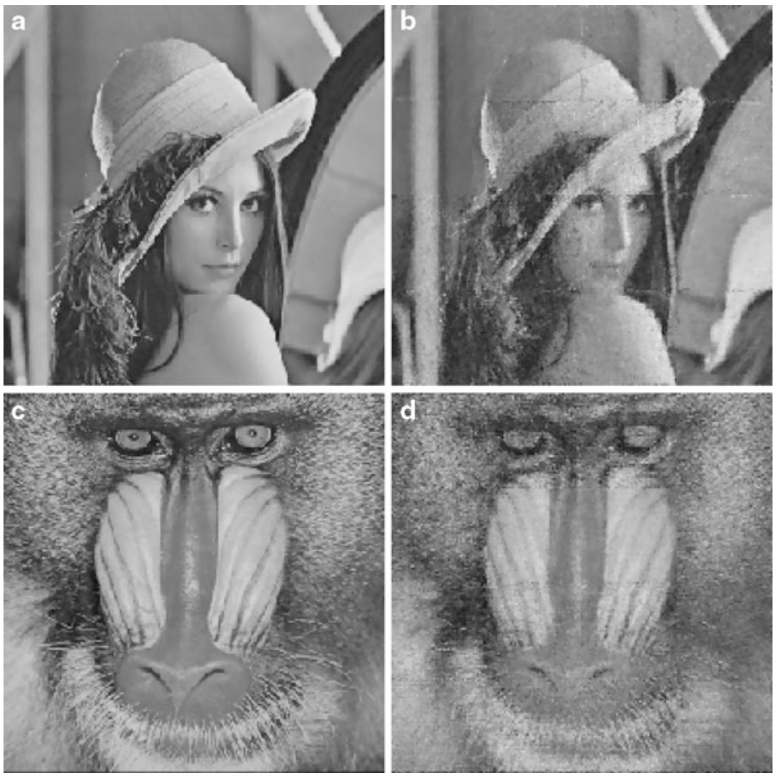


Fig. 6.9 L_1 minimization based reconstruction: (**a**) Original "Lena" image, (**b**) reconstructed "Lena" image, (**c**) original "Baboon" image, (**d**) reconstructed "Baboon" image

where $\varepsilon = \|e\|_{\ell_2}^2$ should hold and TV denotes the total-variation. The total-variation of x represents the sum of the gradient magnitudes at each point and can be approximated as:

$$\mathrm{TV}(x) = \sum_{i,j} \|D_{i,j}x\|_{\ell_2}, \quad D_{i,j}x = \begin{bmatrix} x(i+1,j) - x(i,j) \\ x(i,j+1) - x(i,j) \end{bmatrix}. \tag{6.33}$$

The TV based denoising methods tend to remove the noise while retaining the details and edges in an image. The TV approach could be applied in compressive sensing to define an efficient reconstruction method. Thus, in the light of compressive sensing we may write:

$$\min_x \quad \mathrm{TV}(x) \quad \text{s.t. } \|Ax - y\|_{\ell_2}^2 < \varepsilon^2. \tag{6.34}$$

The TV minimization provides a solution whose variations are concentrated on a small set (small number of edges). The results obtained by applying the TV

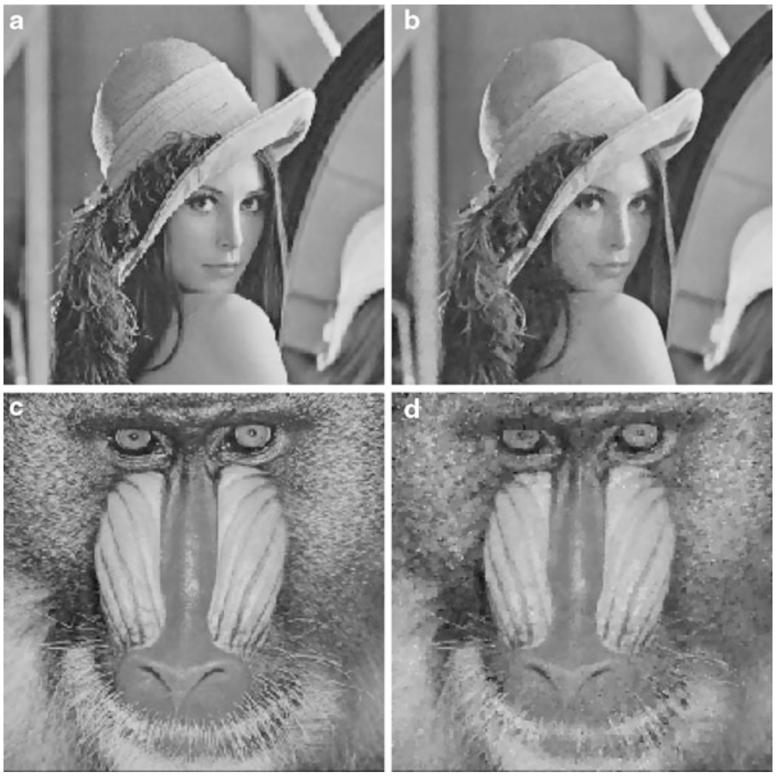


Fig. 6.10 TV reconstruction: (**a**) Original "Lena" image, (**b**) reconstructed "Lena" image, (**c**) Original "Baboon" image, (**d**) reconstructed "Baboon" image

minimization algorithm in image reconstruction are shown in Fig. 6.10. The algorithm is applied to 64×64 blocks. The total number of samples per block is $N = 4{,}096$, while the random $M = 1{,}500$ measurements are used for reconstruction (the DFT matrix is used). Note that the quality of results has been significantly improved when compared to the images in Fig. 6.9. The L1-magic toolbox (see References) is used for solving the minimization problems.

6.2.3 Compressive Sensing and Sparse Time-Frequency Analysis

Compressive sensing can be successfully applied in the area of time-frequency signal representation as well. It has been proven that a nonstationary signal, which is sparse neither in time nor in frequency, may become sparse in the joint time-frequency domain. One such example is a chirp signal whose power is concentrated along a linear time-frequency path, while the rest of time-frequency points can be considered to be zeros. This property can be generalized to a wide class of signals, which are uniquely characterized by their respective instantaneous frequency laws. The time-frequency sparsity would be certainly reduced in the case of multicomponent signals, since the occupancy of the time-frequency plane is increased.

As previously argued, in most cases the signal is typically sparse in one domain and nonsparse in others. However, it has been shown that for many signals, the joint-variable signal description yields sparse representation in the time-frequency and its dual ambiguity domain (related by the two-dimensional Fourier transform). For instance, the observations can be taken in the ambiguity domain and used to reconstruct the signal's instantaneous frequency.

6.2.3.1 Compressive Sensing Based on the Wigner Distribution and Ambiguity Domain Function

Let us consider the Wigner distribution, as one of the commonly used time-frequency distributions. The ambiguity domain counterpart of the Wigner distribution is called the ambiguity function. They are related by the two-dimensional Fourier transform as follows:

$$A_f(\tau,\theta) = \mathbf{F}_{2D}\{\mathrm{WD}(t,\omega)\}. \tag{6.35}$$

The advantage of using the ambiguity domain and its relation with the time-frequency domain has been widely used in the analysis of multicomponent signals. The Wigner distribution of multicomponent signal produces undesired components called cross-terms, which appear between the signal auto-terms, at the position of their arithmetic mean. On the other hand, the cross-terms are usually dislocated from the origin in the ambiguity domain and can be suppressed, or significantly attenuated, by the use of low-pass filtering. This is achieved by applying the kernel $c(\tau,\theta)$ in the ambiguity domain as:

$$A_f(\tau,\theta) = A(\tau,\theta)c(\tau,\theta). \tag{6.36}$$

Removing the cross-terms by the kernel function usually affects the concentration of the auto-terms. Hence, there is always a trade-off between the cross-terms reduction and auto-terms concentration, which may affect the accuracy of the instantaneous frequency estimation. An improved time-frequency signal power localization can be achieved by using the compressive sensing approach and exploiting the signal sparsity in the time-frequency domain. Namely, we can collect a set of samples from the ambiguity domain and solve the ℓ_1-norm minimization problem to obtain the sparsest time-frequency distribution.

By using the compressive sensing approach, the desired time-frequency distribution TFD_x can be obtained as:

$$\mathrm{TFD}_x = \arg\min_{\mathrm{TFD}} \|\mathrm{TFD}\|_{\ell_1}, \quad \mathbb{F}_{2D}^{-1}\{\mathrm{TFD}\} - A_f^M = 0\big|_{(\tau,\theta)\in\Omega}, \tag{6.37}$$

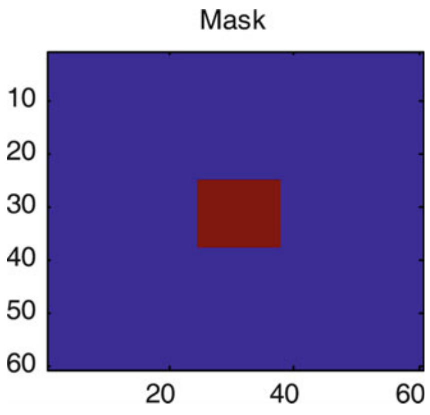


Fig. 6.11 Illustration of the ambiguity domain mask

where A_f^M denotes the set of samples from the ambiguity domain in the region defined by the mask $(\tau, \theta) \in \Omega$, while TFD denotes the time-frequency distribution.

In the presence of noise, we may use the approximation:

$$\mathrm{TFD}_x = \arg\min_{\mathrm{TFD}} \|\mathrm{TFD}\|_{\ell_1}, \quad \left\| \mathbb{F}_{2D}^{-1}\{\mathrm{TFD}\} - A_f^M \right\|_{\ell_2} \leq \varepsilon \big|_{(\tau,\theta)\in\Omega}. \tag{6.38}$$

Here, it is important to select a suitable set of ambiguity domain samples, which can be done by an appropriate ambiguity function masking. The mask (from which we obtain the measurement vector A_f^M) can be formed as a small area around the origin of the ambiguity plane, Fig. 6.11.

As an example, let us consider the monocomponent signal of the form:

$$f_1(t) = e^{j(32/3\cdot\cos(3/2\cdot\pi\cdot t)+3\cdot\cos(\pi\cdot t))} + \upsilon(t), \tag{6.39}$$

where $\upsilon(t)$ is Gaussian noise with variance equal to 1. In order to provide faster computations, we use the time-frequency representations of size 60×60 (3,600 points). The mask is of size 7×7 (1.4% of the total number of points) in the ambiguity domain.

The original time-frequency distribution and ambiguity function are shown in Fig. 6.12a, b, respectively. The resulting sparse representation is illustrated in Fig. 6.12c. Note that the compressive sensing approach provides improved results, by significantly reducing the noise influence. The number of nonzero points in the sparse time-frequency representation is approximately between 45 and 50 (estimated from different experiments), which is a small percentage of the total number of points in the time-frequency domain.

Next, we consider a noisy multicomponent signal, which consists of a chirp and a sine frequency-modulated component:

$$f_2 = e^{j(16/5\cos(3/2\pi t)+6\cos(\pi t)+12\pi t)} + e^{-j(5\pi t^2+20\pi t)} + \upsilon(t). \tag{6.40}$$

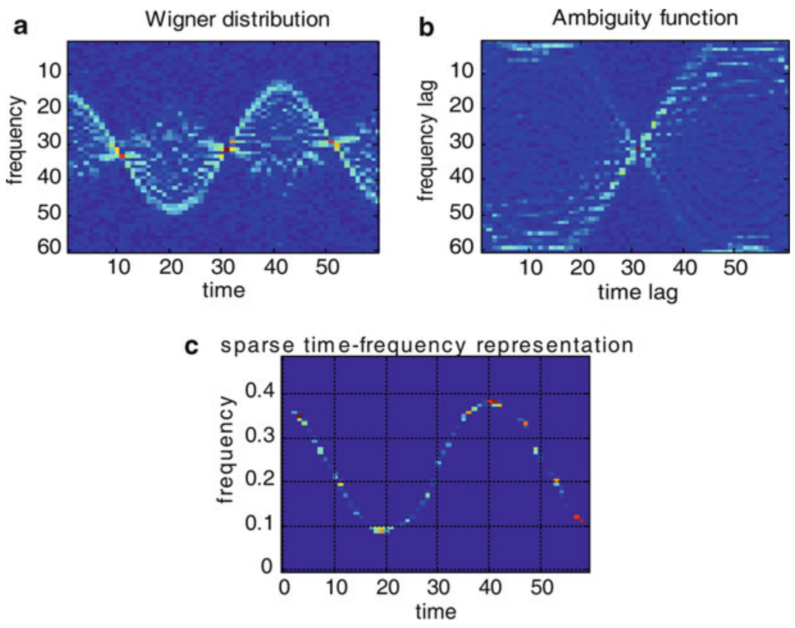


Fig. 6.12 (**a**) Wigner distributions, (**b**) ambiguity function, (**c**) resulting sparse time-frequency representation

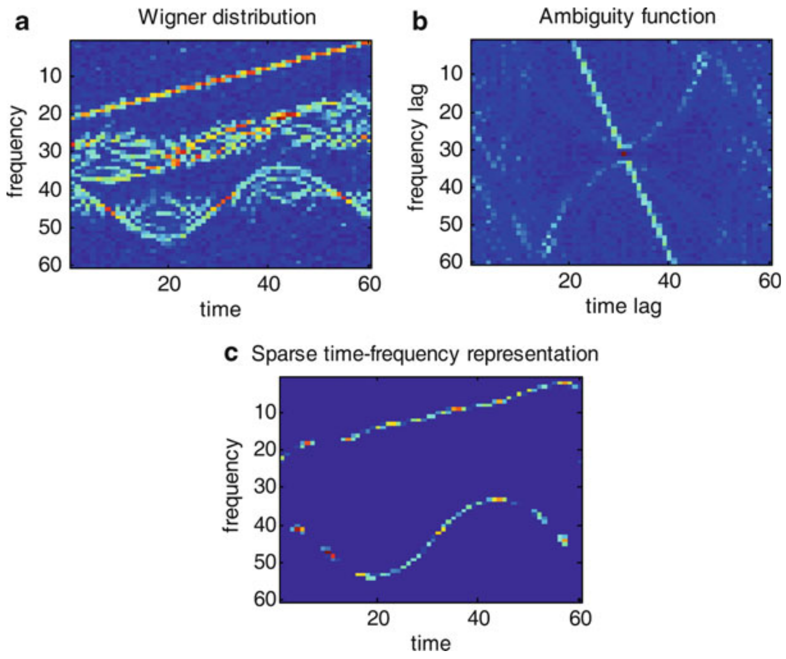


Fig. 6.13 (**a**) Wigner distribution, (**b**) ambiguity function, (**c**) resulting sparse time-frequency representation

The noise parameters are the same as in the previous example. The results are shown in Fig. 6.13. The mask of size 7×7 is used as in the previous example, while the resulting number of nonzero points in the sparse representation is approximately 130. It can be observed that the compressive sensing approach not only provides a good signal power localization, but it also improves the time-frequency resolution of signal components, while de-emphasizing cross-terms.

References

1. Ahmad F, Amin MG (2012) Sparsity-based change detection of short human motion for urban sensing, Seventh IEEE Workshop on Sensor Array and Multi-Channel Signal Processing, Hoboken, NJ
2. Baraniuk R (2007) Compressive sensing. IEEE Signal Process Mag 24(4):118–121
3. Candès E (2006) Compressive sampling. Int Congr Math 3:1433–1452
4. Candès E, Romberg J (2007) Sparsity and incoherence in compressive sampling. Inverse Probl 23(3):969–985
5. Candès E, Romberg J, Tao T (2006) Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans Inf Theory 52(2):489–509
6. Candès E, Wakin M (2008) An introduction to compressive sampling. IEEE Signal Process Mag 25(2):21–30
7. Chartrand R (2007) Exact reconstructions of sparse signals via nonconvex minimization. IEEE Signal Process Lett 14(10):707–710
8. Chen SS (1999) Donoho DL (1999) Saunders MA, atomic decomposition by basis pursuit. SIAM J Sci Comput 20(1):33–61
9. Donoho D (2006) Compressed sensing. IEEE Trans Inf Theory 52(4):1289–1306
10. Donoho DL, Tsaig Y, Drori I, Starck JL (2007) Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. IEEE Trans Inf Theory 58(2):1094–1121
11. Duarte M, Wakin M, Baraniuk R (2005) Fast reconstruction of piecewise smooth signals from random projections. In: SPARS Workshop, Rennes, France
12. Duarte M, Davenport M, Takhar D, Laska J, Sun T, Kelly K, Baraniuk R (2008) Single-pixel imaging via compressive sampling. IEEE Signal Process Mag 25(2):83–91
13. Flandrin P, Borgnat P (2010) Time-frequency energy distributions meet compressed sensing. IEEE Trans Signal Process 8(6):2974–2982
14. Fornasier M, Rauhut H (2011) Compressive sensing. In: Scherzer O (ed) Chapter in part 2 of the handbook of mathematical methods in imaging. Springer, New York
15. Gurbuz AC, McClellan JH, Scott WR Jr (2009) A compressive sensing data acquisition and imaging method for stepped frequency GPRs. IEEE Trans Geosci Remote Sens 57(7):2640–2650
16. Jokar S, Pfetsch ME (2007) Exact and approximate sparse solutions of underdetermined linear equations. SIAM J Sci Comput 31(1):23–44 (Preprint, 2007)
17. Laska J, Davenport M, Baraniuk R (2009) Exact signal recovery from sparsely corrupted measurements through the pursuit of justice. In: Asilomar conference on signals systems, and computers, Pacific Grove, CA
18. L1-MAGIC: http://users.ece.gatech.edu/~justin/l1magic/
19. Peyré G (2010) Best basis compressed sensing. IEEE Trans Signal Process 58(5):2613–2622
20. Romberg J (2008) Imaging via compressive sampling. IEEE Signal Process Mag 25(2):14–20
21. Saab R, Chartrand R, Yilmaz Ö (2008) Stable sparse approximation via nonconvex optimization. In: IEEE international conference on acoustics, speech, and signal processing (ICASSP), Las Vegas, NV

22. Saligrama V, Zhao M (2008) Thresholded basis pursuit: quantizing linear programming solutions for optimal support recovery and approximation in compressed sensing (Preprint, 2008). eprint arXiv:0809.4883
23. Stankovic LJ (1996) The auto-term representation by the reduced interference distributions; the procedure for a kernel design. IEEE Trans Signal Process 44(6):1557–1564
24. Stanković S, Zarić N, Orović I, Ioana C (2008) General form of time-frequency distribution with complex-lag argument. Electron Lett 44(11):699–701
25. Tropp J, Gilbert A (2007) Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans Inf Theory 53(12):4655–4666
26. Tropp J, Needell D (2008) CoSaMP: iterative signal recovery from incomplete and inaccurate samples. Appl Comput Harmon Anal 26(3):301–321
27. Yoon Y, Amin MG (2008) Compressed sensing technique for high-resolution radar imaging. Proc SPIE 6968:6968A–69681A-10

Chapter 7
Digital Watermarking

Advances in the development of digital data and the Internet have resulted in changes in the modern way of communication. A digital multimedia content, as opposed to an analog one, does not lose quality due to multiple copying processes. However, this advantage of digital media is also their major disadvantage in terms of copyright and the unauthorized use of data.

Cryptographic methods and digital watermarking techniques have been introduced in order to protect the digital multimedia content. Cryptography is used to protect the content during transmission from sender to recipient. On the other hand, digital watermarking techniques embed permanent information into a multimedia content. The digital signal embedded in the multimedia data is called digital watermark. A watermarking procedure can be used for the following purposes: ownership protection, protection and proof of copyrights, data authenticity protection, tracking of digital copies, copy and access controls.

The general scheme of watermarking is shown in Fig. 7.1. In general, a watermarking procedure consists of watermark embedding and watermark detection. Although the low watermark strength is preferable in order to meet the imperceptibility requirement, one must ensure that such a watermark is detectable as well. This can be achieved by using an appropriate watermark detector.

Watermark embedding can be based on additive or multiplicative procedures. In multiplicative procedures, the watermark is multiplied by the original content. Watermark detection can be blind (without using the original content) or nonblind (in the presence of the original content).

7.1 Classification of Digital Watermarking Techniques

A number of different watermarking techniques have been developed. Most of them can be classified into one of the categories given in Fig. 7.2. From the perceptual aspect, the watermark can be classified as either perceptible or imperceptible. Noticeable watermark visibly changes the original content. It is sometimes used

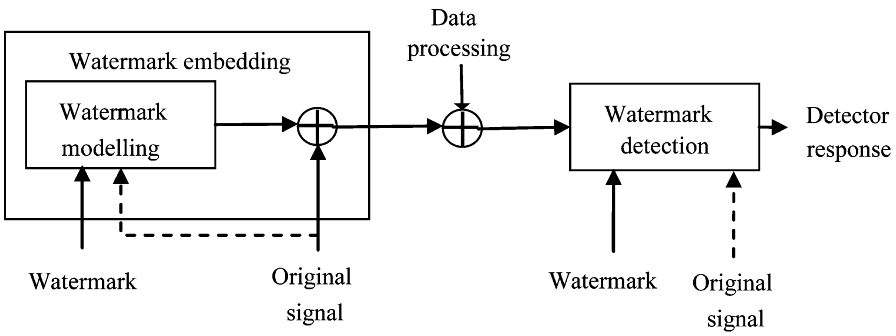


Fig. 7.1 A block scheme of a watermarking procedure

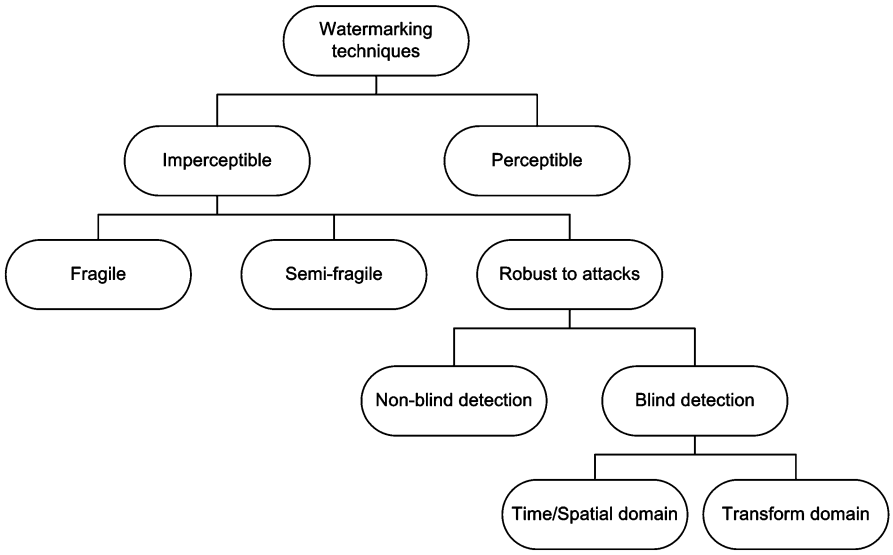


Fig. 7.2 Classification of digital watermarking techniques

to protect images and videos, but generally it is not very popular nowadays. This technique involves embedding characters that uniquely identify the owners of the content and appear as a background image, or as a visible sign. However, this type of watermark can be removed. Almost all techniques currently used fall into the class of imperceptible techniques.

Imperceptible techniques are further divided into robust techniques, semifragile and fragile watermarking techniques. Fragile watermarking assumes embedding of certain watermark that will be significantly damaged or removed in an attempt to modify the content. These techniques are useful in proving the data authenticity. In semifragile watermarking, the watermark should be resistant to certain signal processing techniques (e.g., compression), while it is fragile under any other attack. However, the most commonly used techniques are based on the robust watermarking and will be considered in the next sections.

Robust techniques involve embedding a watermark in the original signal, such that the watermark removal causes serious degradation of the signal quality. Watermark should be designed to be robust to the standard signal processing approaches (compression, filtering, etc.), as well as to intentional attempts to remove the watermark.

7.1.1 Classification in Terms of the Embedding Domain

Watermarking techniques are further divided by the domains in which the watermark is embedded. Namely, the watermark can be embedded directly in the signal domain, or in one of the transform domains. The choice of the watermarking domain depends on the type of multimedia data and the watermarking application. The most frequently used transform domains are based on the DFT, DCT, and DWT transforms. The transform domain watermarking is more convenient for modeling the spectral characteristics of watermark according to the human perceptual model. For highly nonstationary signals, the modeling can be achieved by using time-frequency transforms.

7.2 Common Requirements Considered in Watermarking

Depending on the application and the type of data to be watermarked, the watermarking procedure should fulfill a number of requirements. In the sequel, we discuss some general and very common watermarking requirements.

1. The watermark should be accessible only to the authorized users. This issue is referred as security of the watermarking procedure and it is generally achieved by using cryptographic keys.
2. The watermark detectability should be assured regardless of the conventional signal processing or malicious attacks that may be applied.
3. Generally, although one should provide an unremovable watermark, it should be imperceptible within the host data.
4. The watermark should convey a sufficient amount of information.

As stated above, the first requirement is related to the security of the watermark and watermarking procedure, in general. In some applications, the specific security keys (which can be encrypted) are used during the watermark embedding and extraction. If the watermark is created as a pseudorandom sequence, then the key used to generate a sequence can be considered as a watermarking key.

Table 7.1 Common attacks in audio and image watermarking procedures

Attacks	
Audio watermarking	Image watermarking
Resampling	Requantization
Wow and flutter	JPEG compression
Requantization	Darkening
mp3 with constant bit rate	Lightening
mp3 with variable bit rates	Mean filter (of size 3×3, 5×5, 7×7)
Pitch scaling	Median filter (of size 3×3, 5×5, 7×7)
Audio samples cropping	Image cropping
Echo and timescale modifications	Image resize
Filtering	Rotation
Amplitude normalization	Adding noise
	Gaussian or impulse

The next requirement is watermark robustness, which is one of the main challenges when designing the watermarking procedure. The watermark should be robust not only to the standard signal processing techniques, but also to the malicious attacks aiming to remove the watermark. All algorithms that may lead to the loss of the watermark information are simply called attacks. Some of the common examples are compression algorithms, filtering, change of the data format, noise, cropping signal samples, resampling, etc. The list of commonly present attacks for audio signals and images is given in Table 7.1.

Perceptual transparency is one of the most important requirements. Watermark should be adapted to the host content, and should not introduce any perceptible artifacts or signal quality degradations. However, the imperceptibility is usually confronted with the watermark robustness requirement. In order to be imperceptible, the watermark strength should be low, which directly affects its robustness. Hence, an efficient watermarking procedure should always provide a trade-off between the imperceptibility and robustness. In order to perform the watermark embedding just below the threshold of perception, various masking procedures can be employed.

In some applications it is desirable that the watermark convey a significant number of bits, which will be extracted by detector. Hence, it is sometimes required that the watermark data rate (payload) is high. The property that describes the ability to embed a certain amount of information is known as a capacity of the watermarking algorithm.

Besides the general watermarking requirements discussed above, there could be some specific requirements as well related to the following issues:

– Real-time implementation
– Complete extraction/reconstruction of the watermark at the decoder
– Absence of the original data during the watermark extraction (blind extraction)

7.3 Watermark Embedding

This section will consider the additive and multiplicative watermark embedding techniques.

Additive embedding techniques can be defined as:

$$I_{\mathrm{w}} = I + \alpha w, \tag{7.1}$$

where I represents the vector of signal samples or transform domain coefficients used for watermarking, I_w is the vector of watermarked coefficients, w is the watermark, while the parameter α controls the watermark strength. If the parameter α should be adjusted to the signal coefficients, then the watermark embedding can be written as:

$$I_{\mathrm{w}} = I + \alpha(I)w. \tag{7.2}$$

Another frequently used approach is multiplicative embedding, given by the relation:

$$I_{\mathrm{w}} = I + \alpha w I. \tag{7.3}$$

In order to provide that the watermark does not depend on the sign of selected watermarking coefficients, a modified version of (7.3) can be used:

$$I_{\mathrm{w}} = I + \alpha w|I|. \tag{7.4}$$

Multiplicative watermark is often used in the frequency domain to ensure that the watermark energy at a particular frequency is proportional to the image energy at that frequency. An additional advantage of multiplicative watermark embedding is that it is difficult to estimate and remove watermark by averaging a set of watermarked signals, which is one of the common attacks.

Let us consider an example of robust image watermarking in the transform domain. Two-dimensional DFT of an image is shown in Fig. 7.3a, while its two-dimensional DCT is illustrated in Fig. 7.3b. Note that the DCT is real and has only positive part of the spectrum, making it suitable for applications in watermarking.

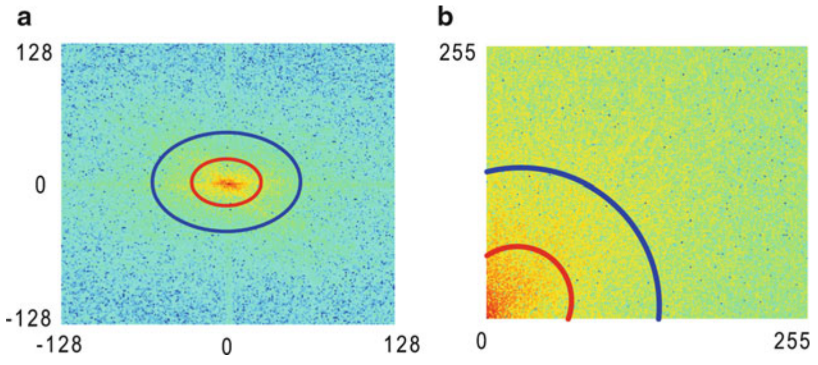


Fig. 7.3 (**a**) DFT of image “Baboon”, (**b**) DCT of image “Baboon”

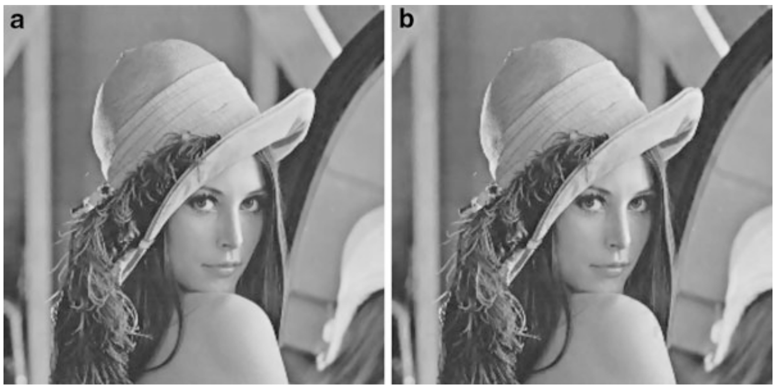


Fig. 7.4 (**a**) Original image "Lena", (**b**) watermarked image "Lena"

The region marked by the red line corresponds to low-frequency coefficients, which contain most of the image energy. Consequently, the modification of these coefficients can cause significant image quality degradation. Therefore, the low-frequency coefficients are usually avoided in watermarking. Outside the blue circle, we have high-frequency components. These components carry certain image details and can be filtered out, without significant image degradation. Therefore, the high-frequency components are also often omitted in watermarking. It follows that the watermarking should be done in the middle frequency part (between the blue and red circles in Fig. 7.3).

Consider the sorted DCT coefficients of an image. Given the nature of the DCT transform, it is necessary to omit the first L coefficients (which are mostly low-frequency components) and choose the next M coefficients (which mostly belong to middle frequencies). Watermarking is then performed as:

$$I_w(i) = I(i) + \alpha|I(i)|w(i) \quad \text{for } i = L+1, L+2, \ldots, L+M, \tag{7.5}$$

where I denotes the DCT coefficients of an image. The watermark w can be created as a pseudorandom sequence. The inverse DCT is then applied to obtain the watermarked image. The original and watermarked "Lena" images are shown in Fig. 7.4 (peak signal to noise ratio PSNR = 47 dB).

7.4 Watermark Detection

7.4.1 Hypothesis Testing Approach

The goal of each algorithm for watermark detection is to provide a reliable proof of the watermark presence within the signal. Denote by I_x a set of coefficients on which the watermark detection is performed (I_x can be either I_w or I, depending on whether the watermark is present or not), and the watermark is w. A general approach for watermark detection is based on *a hypothesis testing problem.* The assumptions are:

$$H_0 : I_x \text{ does not contain watermark } w,$$
$$H_1 : I_x \text{ contains watermark } w.$$

The problem of watermark detection is based on a reliable threshold, used to decide whether a watermark is present or not. The threshold is determined by defining a criterion that ensures a minimum probability of detection error. Since the watermark detection can be viewed as a detection of signal in noise, the *likelihood ratio* is used to minimize the error. Detection errors can occur in two cases: G_{10}—when the assumption of H_0 is accepted as true, although the correct hypothesis is H_1, G_{01}—when the assumption H_1 is accepted as true, but the correct hypothesis is H_0.

The criterion that determines the presence of the watermark is defined as follows:

$$\Phi(I_x) = \begin{cases} 1, & I_x \in R_1, \\ 0, & I_x \in R_0, \end{cases} \tag{7.6}$$

where R_1 and R_0 are regions in which the assumptions H_1 and H_0 are tested. In order to minimize error during detection, a likelihood ratio l is defined by using the conditional probability density functions $p(I_x|H_1)$ and $p(I_x|H_0)$:

$$l(I_x) = \frac{p(I_x|H_1)}{p(I_x|H_0)}. \tag{7.7}$$

The minimum probability of error will be achieved when the region R_1 is determined as:

$$R_1 = \left\{ I_x : l(I_x) > \frac{p_0 P_{01}}{p_1 P_{10}} \right\}, \tag{7.8}$$

where p_0 and p_1 are a priori known probabilities of the assumptions H_0 and H_1 occurrence, while P_{01} and P_{10} are decision weights associated with G_{01} and G_{10}, respectively. The criterion for the detection can be written as:

$$\Phi(I_x) = \begin{cases} 1, & l(I_x) > \frac{p_0 P_{01}}{p_1 P_{10}} \\ 0, & \text{otherwise.} \end{cases} \tag{7.9}$$

Therefore, the detection is done by comparing the likelihood ratio with:

$$\lambda = \frac{p_0 P_{01}}{p_1 P_{10}}. \tag{7.10}$$

The threshold λ can be set to minimize the total probability of error that occurs during detection:

$$P_e = p_0 P_f + p_1(1 - P_d), \tag{7.11}$$

where P_f is the probability that the watermark is detected, when in fact it is not present (false alarm), and $(1 - P_d)$ is the probability of watermark misdetection. The error minimization procedure is commonly performed under the assumption that $P_{01} = P_{10}$ and $p_0 = p_1$, or in other words for $\lambda = 1$. It means that the probabilities of false alarm P_f and misdetection $P_m = (1 - P_d)$ are the same. In practice, we usually have a predefined maximum false alarm probability from which the threshold λ is calculated as follows:

$$\int_{\lambda}^{\infty} p(l|H_0)\mathrm{d}l = P_f, \tag{7.12}$$

where $p(l|H_0)$ is the pdf of l under H_0. After the threshold λ is determined, the probability of misdetection is calculated as:

$$P_m = \int_{-\infty}^{\lambda} p(l|H_1)\mathrm{d}l. \tag{7.13}$$

7.4.1.1 Additive White Gaussian Model

Let us consider the procedure to minimize the detection error in the case of additive white Gaussian noise (AWGN) model, which is the simplest one encountered in practice. This model assumes that the coefficients are uncorrelated and have a Gaussian distribution. Note that the watermark is considered as a noisy signal:

$$I_x = I + w + n, \tag{7.14}$$

where I_x, I, and w are the coefficients of the watermarked content, the original content, and the watermark, respectively. The watermarked content can be modified in the presence of attack, which is modeled by noise n (white Gaussian noise). Under the assumption that the original coefficients, as well as the noise coefficients, are uncorrelated and follow the Gaussian distribution, (7.14) can be written as follows:

$$I_x = I_n + w. \tag{7.15}$$

I_n also has the Gaussian distribution with the modified mean value and the variance compared to the original content I. Now, the previously defined hypothesis can be written as:

$$\begin{aligned} H_0 &: I_x = I_n \\ H_1 &: I_x = I_n + w. \end{aligned}$$

In order to minimize the similarity measure $l(I_x) = \frac{p(I_x|H_1)}{p(I_x|H_0)}$, it is necessary to know the conditional probability density function, which in the case of the Gaussian distribution is defined as:

$$\begin{aligned} p(I_x|H_1) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(I_x(i)-\mu_x-w(i))^2}{2\sigma_x^2}} \\ p(I_x|H_0) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(I_x(i)-\mu_x)^2}{2\sigma_x^2}}, \end{aligned} \tag{7.16}$$

where μ_x is the mean value of signal coefficients used in watermark detection. Now the measure of similarity is calculated as:

$$l(I_x) = \frac{p(I_x|H_1)}{p(I_x|H_0)} = \frac{\prod_{i=1}^{n} e^{-\frac{(I_x(i)-\mu_x-w(i))^2}{2\sigma_x^2}}}{\prod_{i=1}^{n} e^{-\frac{(I_x(i)-\mu_x)^2}{2\sigma_x^2}}}. \tag{7.17}$$

Equation 7.17 can be written in a simplified form by applying the logarithmic function:

$$\begin{aligned} \ell(I_x) &= \sum_{i=1}^{n} \frac{1}{2\sigma_x^2} \left[(I_x(i)-\mu_x)^2 - (I_x(i)-\mu_x-w(i))^2 \right] = \\ &= \frac{1}{2\sigma_x^2} \left[\sum_{i=1}^{n} 2I_x(i)w(i) - \sum_{i=1}^{n} 2\mu_x w(i) - \sum_{i=1}^{n} w^2(i) \right], \end{aligned} \tag{7.18}$$

where $\ell(I_x)$indicates the natural logarithm function of $l(I_x)$. Note that the last two terms within the brackets do not depend on I_x. Therefore, the term representing linear correlation of I_x and w is used as a watermark detector:

$$D = \sum_{i=1}^{n} I_x(i)w(i), \tag{7.19}$$

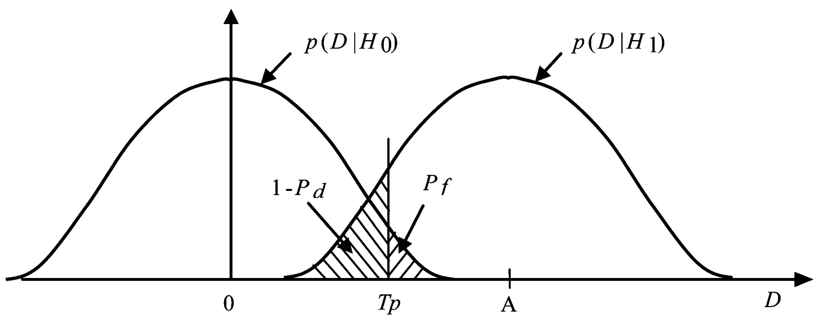


Fig. 7.5 Illustration of the errors that may occur in watermark detection

which is optimal under the considered assumptions and is called the standard correlator. In the case when the signal statistics is not distributed according to the Gaussian distribution, other detector forms can be used.

According to the procedure for determining the general detection threshold λ, we can now determine the threshold for the standard correlator as:

$$\int_{T_p}^{\infty} p(D|H_0)\mathrm{d}D = P_f, \tag{7.20}$$

where $p(D|H_0)$ is the pdf of detector responses D under H_0. The pdf of D under H_0 and H_1 are illustrated in Fig. 7.5. If the response of the detector is $D < T_p$, we conclude that the watermark is not present, and vice versa. In the case of equal probabilities $P_f = 1 - P_d$, the optimum threshold is $A/2$ (Fig. 7.5).

In order to determine the threshold and the probability of error, we need to know how the watermark is embedded, the statistical characteristics of the image coefficients, as well as the characteristics of attacks.

7.4.2 A Class of Locally Optimal Detectors

According to the signal detection theory, it is difficult to define a general test that maximizes the signal detection probability. Also, it is known that for detection of weak signals a locally optimal detector can be created (in our case a watermark signal is weak in comparison to the host signal). It is defined as follows:

$$D = g_{LO}(I_x) \cdot w, \tag{7.21}$$

where g_{LO} is the local optimum nonlinearity, defined by:

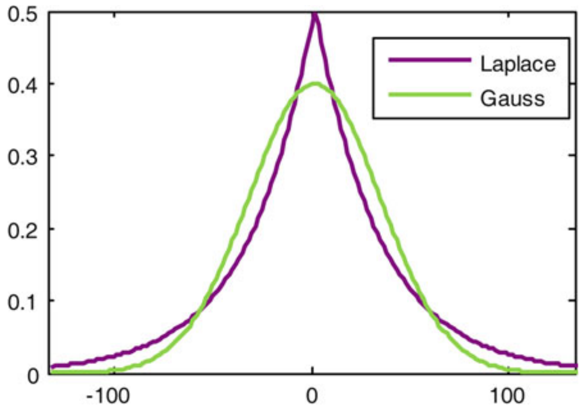


Fig. 7.6 Distribution of coefficients: Gaussian (*green line*) and Laplace distribution (*purple line*)

$$g_{\mathrm{LO}}(I_x) = -\frac{p'(I_x)}{p(I_x)}, \tag{7.22}$$

with $p(I_x)$ and $p'(I_x)$ indicating the coefficients probability density function and its derivative, respectively. Note that, the detector contains the nonlinear part g_{LO}, which is correlated with the watermark signal. If the coefficients have the Gaussian distribution, the proposed detector corresponds to the standard correlator.

7.4.2.1 The Most Commonly Used Distribution Functions and the Corresponding Detector Forms

The coefficients distribution for most images can be modeled by the Gaussian, Laplace, generalized Gaussian, or Cauchy distribution functions. For example, recall that the generalized Gaussian function can be defined as:

$$\mathrm{GGF} = \frac{\alpha\beta}{2\Gamma(1/\alpha)} e^{(-\beta|x-\mu|)^{\alpha}}, \;\; \alpha > 0, \;\; \beta = \frac{1}{\sigma}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2}. \tag{7.23}$$

For $\alpha = 1$, this function is equal to the Laplace distribution, and for $\alpha = 2$ it is equal to the Gaussian distribution. Figure 7.6 shows a coefficient distribution of an image. The form of the detector, which corresponds to the generalized Gaussian distribution, is given by:

$$D_1 = \sum_{i=1}^{L} \mathrm{sign}(I_x(i))|I_x(i)|^{\alpha-1} w(i), \tag{7.24}$$

while the detector form for Cauchy distribution, $\mathrm{CF} = \frac{\gamma}{\pi\left(\gamma^2+(x-\delta)^2\right)}$, is equal to:

$$D_2 = \sum_{i=1}^{L} \frac{2(I_x(i) - \delta)}{(I_x(i) - \delta)^2 + \gamma^2} w(i). \tag{7.25}$$

Note that x (in the pdf) corresponds to the watermarked coefficients I_x in the detector form.

It is important to emphasize that the locally optimum detector form can be quite sensitive to the pdf variations.

A simple measure of detection quality can be defined as:

$$R = \frac{\overline{D}_{\mathrm{wr}} - \overline{D}_{\mathrm{ww}}}{\sqrt{\sigma_{\mathrm{wr}}{}^2 + \sigma_{\mathrm{ww}}{}^2}}, \tag{7.26}$$

where D and σ^2 are mean values and standard deviations of detector responses, while the indices *wr* and *ww* are used for right keys (watermarks) and wrong keys (wrong trials), respectively. The wrong trial is any sequence that is not the watermark, but is generated in the same way.

7.4.3 Correlation Coefficient and Similarity Measure

In order to determine the similarity between the original watermark w and the watermark w^* extracted from the protected data at the detection side, we can use the similarity measure defined as follows:

$$\mathrm{Sim}(w, w*) = \frac{w \cdot w*}{\sqrt{w \cdot w*}}. \tag{7.27}$$

The similarity measure is usually given in the form of the correlation coefficient, which can be calculated as:

$$\rho(w, w*) = \frac{\sum_{i=1}^{N} w(i) w^*(i)}{\sqrt{\sum_{i=1}^{N} (w(i))^2} \sqrt{\sum_{i=1}^{N} (w^*(i))^2}}. \tag{7.28}$$

7.5 Examples of Watermarking Procedures

7.5.1 Audio Watermarking Techniques

Audio watermarking procedures are mainly based on the specific audio signal characteristics and psychoacoustics. In the next subsections, a brief description of audio watermarking approaches such as the spread-spectrum audio watermarking, two-sets method, and echo embedding, is provided.

7.5.1.1 Spread-Spectrum Watermarking

Spread-spectrum watermarking is an example of correlation-based method that assumes a pseudorandom sequence embedding, where the standard correlator is used for detection. This is a commonly used watermarking approach. The pseudorandom sequence, $r(n)$, i.e., the wideband noise sequence, can be embedded in the time or in the transform domain. This sequence is used to modulate the binary message $\upsilon = \{0, 1\}$, or equivalently $b = \{-1, 1\}$. The watermarked sequence $w(n) = br(n)$ obtained in this way is scaled according to the energy of the host signal $s(n)$ to provide a compromise between the watermark imperceptibility and robustness. The watermark embedding can be done, for example, by an additive procedure: $s_w(n) = s(n) + \alpha w(n)$. A suitable pseudorandom sequence should have good correlation properties, in such a way that it should be orthogonal to the other pseudorandom sequences. The commonly used sequence is called the M sequence (maximum length sequence), and its autocorrelation is given by:

$$\frac{1}{N}\sum_{i=0}^{N-1} w(i)w(i-k) = \begin{cases} 1, & \text{for } k = 0, \\ \frac{1}{N}, & \text{for } k \neq 0. \end{cases} \tag{7.29}$$

7.5.1.2 Two-Sets Method

This blind audio watermarking procedure is based on the two sets A and B of audio samples. A value d (watermark) is added to the samples within the set A, while it is subtracted from the samples in B:

$$a_i^* = a_i + d, \quad b_i^* = b_i - d,$$

where a_i and b_i are samples from A and B, respectively. When making decision about watermark presence, the expected value $E\left[\bar{a}^* - \bar{b}^*\right]$ is employed, where $\bar{a}^*$and $\bar{b}^*$are mean values of samples a_i^* and b_i^*. This method is based on the assumption that the mean values of the samples from different signal blocks are the same, i.e., that $E[\bar{a} - \bar{b}] = 0$ holds (which may not be always the case in the practice). Only in this case, the watermark can be detected as:

$$E\left[\bar{a}^* - \bar{b}^*\right] = E[(\bar{a} + d) - (\bar{b} - d)] = E(\bar{a} - \bar{b}) + 2d = 2d. \tag{7.30}$$

7.5.1.3 Echo Embedding

The echo embedding procedure can be realized according to:

$$x(n) = s(n) + \alpha s(n - d), \tag{7.31}$$

where d represents a certain delay of the echo signal. The extraction of the embedded echo requires the detection of delay d. The signal copy is usually delayed for approximately 1 ms. The echo amplitude is significantly lower than the original signal amplitude, and hence, the signal quality is not degraded. On the contrary, the sound is enriched. There is also a variant of this procedure, where two delays are considered: one is related to the logical value "1," while the other is related to "0." The double echo embedding operation can be written as:

$$x(n) = s(n) + \alpha s(n-d) - \alpha s(n-d-\Delta), \tag{7.32}$$

where the difference between delays corresponding to "1" and "0" is denoted by Δ, and its value does not exceed four samples. The delay detection is done by using the cepstrum autocorrelation, which is the inverse Fourier transform of the log-magnitude spectrum. The complexity of cepstrum calculation is one of the main disadvantages of this method.

7.5.1.4 Watermarking Based on the Timescale Modifications

Timescale modifications are related to compressing and expanding of the time axis. The basic idea of timescale watermarking is to change the timescale between two successive extremes (maximum and minimum). The interval between two extremes is divided into N segments with equal amplitudes. The signal slope is changed within a certain amplitudes range according to the bits that should be embedded. Namely, the steep slope corresponds to "0,"while the mild slope corresponds to bit "1."

7.5.2 Image Watermarking Techniques

A simple watermarking algorithm for digital image protection is based on the additive watermark embedding procedure in the 8×8 DCT domain. First, an image is divided into 8×8 blocks of pixels as in the case of JPEG algorithm. Then, the two-dimensional DCT transform is applied to each block separately. The watermark is embedded into the set of selected coefficients. In order to provide a good compromise between the watermark imperceptibility and robustness, the coefficients are selected from the middle frequency region, as illustrated in Fig. 7.7.

Watermark embedding is based on the standard additive procedure: $I_w = I + \alpha w$, where I denotes the original middle-frequency DCT coefficients (from 8×8 block), while I_w are the watermarked DCT coefficients. Next, we perform the inverse DCT transform that results in watermarked 8×8 block. This is repeated for each block.

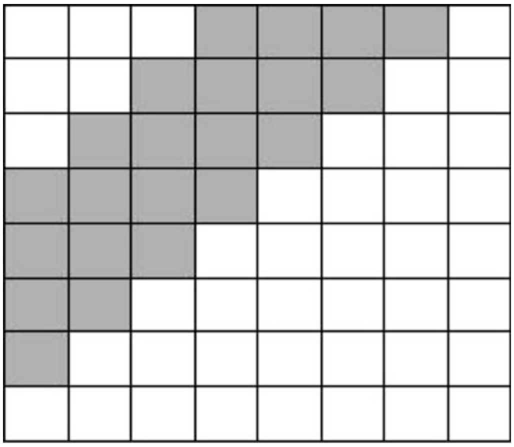


Fig. 7.7 A region of middle frequency DCT coefficients within 8 × 8 block (shaded in *gray*)

Watermark detection can be performed by using a standard correlation detector (assuming that the distribution of selected coefficients can be modeled by the Gaussian function). However, more accurate modeling can be obtained by using generalized Gaussian and Cauchy function, where the corresponding detectors D_1 and D_2 (defined by (7.24) and (7.25)) are used for detection.

7.5.3 The Procedure for Watermarking of Color Images

Unlike the previous procedure, where the block-based DCT is performed, here we will use the two-dimensional DCT transform of the entire image. The procedure is described in the sequel.

(a) The selection of coefficients for watermark embedding is done through the following steps:

   1. The color channels are separated (e.g., R,G, B), Fig. 7.8.
   2. Two-dimensional DCT is computed for each color matrix.
   3. The matrices of DCT coefficients are transformed into vectors and sorting operation is performed.
   4. The largest L coefficients are omitted and the next M coefficients are selected for watermarking.

(b) Watermark embedding
   Let us denote the sorted DCT coefficients by I, while w is the watermark created as a pseudorandom sequence. The watermarked DCT coefficients are calculated as:

$$I_w(i) = I(i) + \alpha \cdot |I(i)| \cdot w(i), \quad i = L + 1, \ldots, M + L,$$

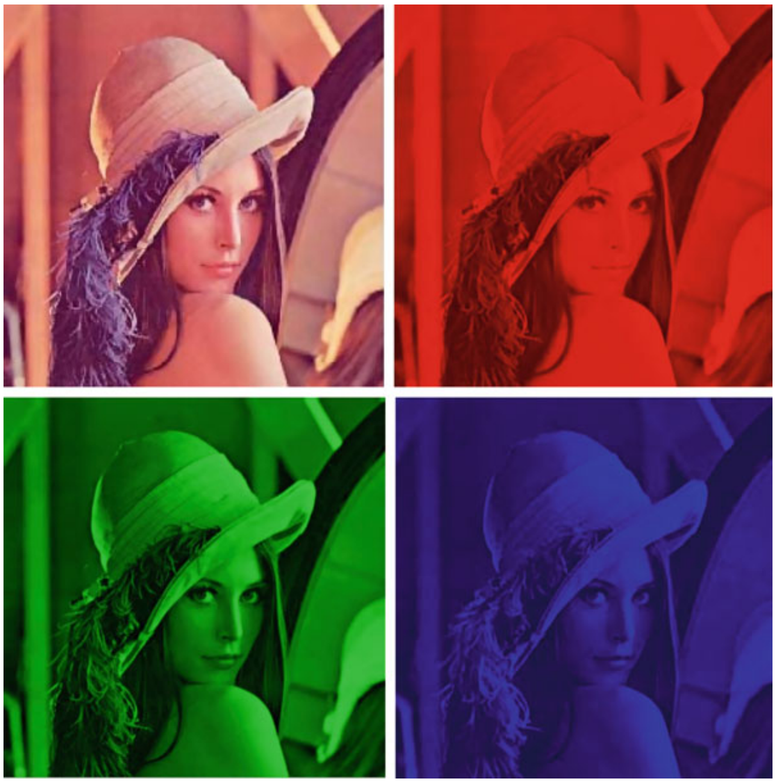


Fig. 7.8 Color image and the separated color channels

where i denotes the coefficient position in the sorted sequence.

(c) Reorder the sequence into matrix form.

(d) Calculate the two-dimensional inverse DCT (with rounding to integer values).

7.5.4 An Overview of Some Time-Frequency-Based Watermarking Techniques

The time-frequency-based watermarking can be used for different types of multimedia data: audio signals, images, and video signals. The time-frequency domain can be efficient regarding the watermark imperceptibility and robustness. Namely, the watermark with specific time-frequency characteristics can be designed and adapted to the host signal components, which enhances the efficiency of the watermarking procedure. Note that the time-frequency representations defined for one-dimensional signals can be extended to two-dimensional cases in order to be applied to images. In this case, they are usually referred as the space/spatial-frequency representations.

1. The watermark can be created with specific space/spatial-frequency characteristics, while its embedding can be done even in the space domain. This approach is among the first time-frequency-based image watermarking procedures. Namely, a two-dimensional chirp signal is used as watermark:

$$W(x,y) = 2A\cos\left(ax^2 + by^2\right) = A\left(e^{j\left(ax^2+by^2\right)} + e^{-j\left(ax^2+by^2\right)}\right). \tag{7.33}$$

   The watermark is embedded within the entire image:

$$I_w(x,y) = I(x,y) + W(x,y). \tag{7.34}$$

   It is interesting to observe that multiple different chirps with small amplitudes can be used for watermarking. The parameters of the chirp signals and the random sequence that define the amplitudes of chirps serve as the watermark key. Since the watermark is embedded within the entire image in the spatial domain, a proper masking that provides imperceptibility should be applied. Note that the Wigner distribution provides an ideal representation for the chirp signal. Hence, the watermark detection is performed by using a form of the Radon-Wigner distribution:

$$P\left(\omega_x,\omega_y;W_v\right) = \left|FT_{2D}\{I_w(x,y)W_v(x,y)\}\right|^2 = \left|\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_w(x,y)W_v(x,y)e^{-j\left(x\omega_x+y\omega_y\right)}\mathrm{d}x\mathrm{d}y\right|^2, \tag{7.35}$$

   where:

$$W_v(x,y) = e^{-j\left(a_vx^2+b_vy^2+c_vxy\right)}. \tag{7.36}$$

   Different values of parameters a_v, b_v, and c_v define a set of projection planes. The additional term c_vxy is used to detect some geometrical transformations, as well. In order to make a decision about the watermark presence within the image, the maxima of the Radon-Wigner distribution are calculated:

$$M(a_v,b_v,c_v) = \max_{\omega_x,\omega_y} P\left(\omega_x,\omega_y;W_v\right), \tag{7.37}$$

   and compared with a reference threshold. This procedure provides robustness to various attacks, some being a median filter, geometrical transformations (translation, rotation, and cropping simultaneously applied), a high-pass filter, local notch filter, and Gaussian noise.
2. Digital audio watermarking can be done by using time-frequency expansion and compression. The audio signal is first divided into frames of size 1,024 samples, where the successive frames have 512 samples overlapping. If the original frame

is lengthened or shortened, the logical value 1 is assigned, otherwise the "normal frames" corresponds to the logical value 0. The watermark is a sequence obtained as a binary code of the alphabet letters, converted to the ASCII code. The frames with signal energy level above a certain threshold are selected. The signal is transformed to frequency domain and a psychoacoustic model is used to determine the masking threshold for each selected frame. The length of the frames is changed in frequency domain by adding or removing four samples with amplitudes that do not exceed the masking threshold. It prevents a perceptual distortion. In order to preserve the total length of the signal, the same number of expanded and compressed frames are used (usually an expanded frame is followed by a compressed frame). The detection procedure is nonblind, i.e., the original signal is required. The difference between the original and watermarked samples in time domain will have diamond shape for the pair expanded–compressed frame (Diamond frames), while the difference is flat and close to zero for unaltered frames. The pair of Diamond frames is used to represent the binary 1, while the logical values 0 are assigned to the unaltered frames. Hence, it is possible to detect binary values, and, consequently, the corresponding alphabetical letters.

3. *A spread spectrum-based watermarking in the time-frequency domain.*
The watermark is created as:

$$w_i(n) = a(n)m_i(n)p_i(n)\cos(\omega_0(n)n), \tag{7.38}$$

where $m_i(n)$ is the watermark before spreading, $p_i(n)$ is the spreading code or the pseudonoise sequence (bipolar sequence taking the values +1 and −1 with equal probabilities), while ω_0 is the time-varying carrier frequency that represents the instantaneous mean frequency of the signal. The parameter $a(n)$ controls the watermark strength. The masking properties of the human auditory system are used to shape an imperceptible watermark. The pseudonoise sequence is low-pass filtered according to the signal characteristics. Two different scenarios of masking have been considered. The tone- or noise-like characteristics are determined by using the entropy:

$$H(X) = -\sum_{i=1}^{\omega_{\max}} P(x_i)\log_2 P(x_i). \tag{7.39}$$

The probability of energy for each frequency (within a window used for the spectrogram calculation) is denoted by $P(x_i)$, while $\omega_{\max}$ is the maximum frequency. A half of the maximum entropy $H_{\max}(x) = \log_2\omega_{\max}$ is taken as a threshold between noise-like and tone-like characteristics. If the entropy is lower than $H_{\max}$, it is considered as a tone-like, otherwise it is a noise-like characteristic.

The time-varying carrier frequency is obtained as the instantaneous mean frequency of the host signal, calculated by:

$$\omega_i(n) = \frac{\sum\limits_{\omega=0}^{\omega_{\max}} \omega \mathrm{TFD}(n,\omega)}{\sum\limits_{\omega=0}^{\omega_{\max}} \mathrm{TFD}(n,\omega)}. \tag{7.40}$$

The instantaneous mean frequency is computed over each time window of the STFT, and the TFD(n,ω) is the energy of the signal at a given time and frequency.

Finally, after the watermark is modulated and shaped, it is embedded in the time domain as: $s_{w_i}(n) = s_i(n) + w_i(n)$. During the detection, the demodulation is done by using the time-varying carrier and then the watermark is detected by using the standard correlation procedure.

4. *Watermarking approach based on the time-frequency-shaped watermark.*
In order to ensure imperceptibility constraints, the watermark can be modeled according to the time-frequency characteristics of the signal components. For this purpose, the concept of nonstationary filtering is adapted and used to create a watermark with specific time-frequency characteristics. The algorithm includes the following steps:

(a) Selection of signal regions suitable for watermark embedding;
(b) Watermark modeling according to the time-frequency characteristics of the host signal;
(c) Watermark embedding and watermark detection in the time-frequency domain.

Due to the multicomponent nature of multimedia signals (e.g., speech signals), the cross-terms free time-frequency distributions (TFD) should be used, such as the spectrogram and the S-method. If a region selected from the TFD is:

$$D = \{(t,\omega) : t \in (t_1, t_2), \omega \in (\omega_1, \omega_2)\}, \tag{7.41}$$

we can define a time-frequency mask as follows:

$$L_M(t,\omega) = \begin{cases} 1 \text{ for } (t,\omega) \in D \text{ and } |\mathrm{TFD}(t,\omega)| > \xi, \\ 0 \text{ for } (t,\omega) \notin D \text{ or } |\mathrm{TFD}(t,\omega)| < \xi. \end{cases} \tag{7.42}$$

The parameter ξ is a threshold that can be calculated as a portion of the TFD maximum: $\xi = \lambda 10^{\lambda \log_{10}(\max(|\mathrm{TFD}(t,\omega)|))}$ (λ is a constant). The mask L_{M} contains the information about the significant components within the region D. Hence, if we start with an arbitrary random sequence p, the modeled watermark is obtained at the output of the nonstationary (time-varying) filter:

$$w(t) = \sum_{\omega} L_{\mathrm{M}}(t,\omega) \mathrm{STFT}_p(t,\omega), \tag{7.43}$$

where STFT_p stands for the short-time Fourier transform of p. The watermark embedding is done according to:

$$\mathrm{STFT}_{I_\mathrm{w}}(t,\omega) = \mathrm{STFT}_I(t,\omega) + \mathrm{STFT}_{w_\mathrm{key}}(t,\omega), \tag{7.44}$$

where I_w, I, and w are related to the watermarked signal, original signal, and watermark, respectively.

The watermark detector can be made by using the correlation in the time-frequency domain:

$$D = \sum_{i=1}^{N} \mathrm{STFT}^i_{w_\mathrm{key}} \mathrm{STFT}^i_{I_\mathrm{w}}. \tag{7.45}$$

Note that the time-frequency domain provides a larger number of coefficients for correlation (compared to time or frequency domains), which enhances the detection performance.

7.6 Examples

7.1. Consider a vector with a few image DFT coefficients chosen for watermarking. DFT = [117 120 112 145 136 115].
The watermarking procedure should be done in the following way:

(a) Sort the vector of DFT coefficients.
(b) Add a watermark given by $w = [-3.5\ -2\ 4\ 5\ 9\ -7]$.
(c) Assume that the sequence $wrong = [3\ 2\ -5\ -7\ 2\ 4]$ provides the highest response of the correlation-based detector among large number of wrong trials (wrong keys) used for testing.
(d) Prove that the watermark can be successfully detected by using the standard correlator.

Solution:

$$\mathrm{DFT}_\mathrm{sort} = [112 \quad 115 \quad 117 \quad 120 \quad 136 \quad 145].$$

$$\mathrm{DFT}_w = \mathrm{DFT}_\mathrm{sort} + w = [108.5 \quad 113 \quad 121 \quad 125 \quad 145 \quad 138].$$

In order to ensure a reliable watermark detection using the standard correlator, the detector response for the watermark should be higher than the maximal detector response when using wrong trials:

$$\Sigma \mathrm{DFT}_w \cdot w > \Sigma \mathrm{DFT}_w \cdot \mathrm{wrong}$$

$$\Sigma \mathrm{DFT}_w \cdot w = 842.25$$

$$\Sigma \mathrm{DFT}_w \cdot \mathrm{wrong} = -86.5$$

Having in mind the results, we may conclude that the watermark detection is successful.

7.2. Write a program in Matlab that performs the image watermarking as follows:

(a) Calculate and sort the DCT coefficients of the considered image;
(b) Create a watermark as a pseudorandom sequence, e.g., watermark = 0.5rand (1500,1);
(c) After omitting the strongest 1,500 coefficients, embed the watermark into the next 1,500 coefficients by using the multiplicative procedure with $\alpha = 0.8$;
(d) Check if the watermark is imperceptible within the protected image;
(e) Perform the watermark detection in the DCT domain by using the standard correlator. It is necessary to demonstrate that the detector response for the watermark is higher than the detector response for any of the 100 wrong trials (Fig. 7.9).

Solution:

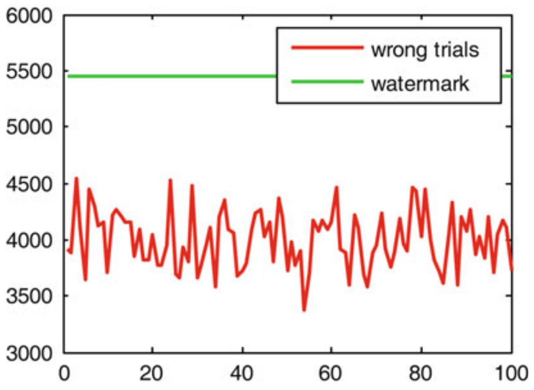
```
alfa=0.8;
Det=zeros(2,100);
image=imread('lena512.bmp');
image=image(1:2:512,1:2:512);
N=256;

DCT1=dct2(image);
Vector=DCT1(:);
[g,v]=sort(abs(Vector));
watermark=0.5*rand(1500,1);
Vectorwat=Vector;
Vectorwat(v(N*N-1500-1500+1:N*N-1500))=Vector(v(N*N-
1500-1500+1:N*N-1500))+alfa*abs(Vector(v(N*N-1500-1500
+1:N*N-1500))).*watermark;
DCTwat=DCT1;

DCTwat(:)=Vectorwat;
imagewat=idct2(DCTwat);
figure,imshow(uint8(imagewat))

DCTwat1=dct2(imagewat);
DCTwat1=DCTwat1(:);
x=DCTwat1(v(N*N-1500-1500+1:N*N-1500));
for k=1:100
wrong=0.5*rand(1500,1);
Det(1,k)=sum(x.*watermark);
```

Fig. 7.9 Results of watermark detection



```
Det(2,k)=sum(x.*wrong);
end
figure,
plot(1:100,Det(2,1:100),'r',1:100,Det(1,1:100),'g')
```

7.3. Consider the watermarking procedure described in the sequel. A block of the 8×8 DCT coefficients is selected. The watermark is added to the block coefficients: $I_w = I + w$. The watermarked image is exposed to the quantization attack defined by the quantization matrix Q. Determine which watermark samples will contribute to the difference between the watermarked and the original coefficient after quantization attack.

$$\mathrm{DCT} = \begin{matrix} 45 & 20 & 54 & 81 & 0 & 0 & 0 & 0 \\ 15 & 77 & 0 & 11 & 0 & 0 & 0 & 0 \\ 21 & 0 & 0 & 39 & 0 & 0 & 0 & 0 \\ 27 & 44 & 52 & 75 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad w = \begin{matrix} 5 & 4 & 1 & -3 & 0 & 0 & 0 & 0 \\ 3.5 & 5 & 0 & 5 & 0 & 0 & 0 & 0 \\ 3 & 3 & 2 & -5 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 6.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$Q = \begin{matrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{matrix}$$

Solution:

Approach I: It is possible to perform the quantization of the original and the watermarked coefficients, to compare them, and to select the coefficients that are different after quantization.

$$\mathrm{DCT}_q = \begin{matrix} 15 & 4 & 8 & 9 & 0 & 0 & 0 & 0 \\ 3 & 11 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 3 & 4 & 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \quad \mathrm{DCT}_{wq} = \begin{matrix} \boxed{17} & \boxed{5} & 8 & 9 & 0 & 0 & 0 & 0 \\ \boxed{4} & \boxed{12} & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 3 & 4 & 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Hence, the positions of the selected coefficients are: (1,1), (1,2), (2,1), (2,2).

Approach II: Select the watermark samples higher than $Q/2$.

$$\frac{Q}{2} = \begin{matrix} 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 \\ 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5 \\ 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5 & 10.5 \\ 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5 & 10.5 & 11.5 \\ 5.5 & 6.5 & 7.5 & 8.5 & 9.5 & 10.5 & 11.5 & 12.5 \\ 6.5 & 7.5 & 8.5 & 9.5 & 10.5 & 11.5 & 12.5 & 13.5 \\ 7.5 & 8.5 & 9.5 & 10.5 & 11.5 & 12.5 & 13.5 & 14.5 \\ 8.5 & 9.5 & 10.5 & 11.5 & 12.5 & 13.5 & 14.5 & 15.5 \end{matrix}$$

$$w = \begin{matrix} \boxed{5} & \boxed{4} & 1 & -3 & 0 & 0 & 0 & 0 \\ \boxed{3.5} & \boxed{5} & 0 & 5 & 0 & 0 & 0 & 0 \\ 3 & 3 & 2 & -5 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 6.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

The selected watermark samples will produce a difference between the quantized original and watermarked image coefficients.

7.4. Based on the principles introduced in the previous example, the image watermarking procedure is implemented as follows:

(a) DCT is calculated for the 8 × 8 image blocks.
(b) The watermark w is added to the quantized coefficients: $I_q(i,j) = K(i,j)Q(i,j)$, ($K(i,j)$ are integers), but the coefficients quantized to zero value are immediately omitted.
(c) The selection of coefficients suitable for watermarking is done according to the constraints:
   - The watermarked DCT coefficients after quantization with Q should have nonzero values.
   - The watermarked coefficients should not be rounded to the same value as the original coefficients.

Analyze and define the values of $K(i,j)$ and watermark w that satisfy the above constraints.

Solution and explanation:

To ensure that the watermarked DCT coefficients after quantization with Q have nonzero values, the following relation should hold:

$$|K(i,j)Q(i,j)| - |w| \geq \frac{Q(i,j)}{2}, \tag{7.46}$$

or equivalently, the watermark should satisfy the condition:

Condition 1

$$|w| \leq |K(i,j)Q(i,j)| - \frac{Q(i,j)}{2}. \tag{7.47}$$

The watermarked coefficients will not be quantized to the same value as the original one if the following condition is satisfied:

Condition 2

$$\begin{aligned} &K(i,j)Q(i,j) + w < K(i,j)Q(i,j) - Q/2 \\ &\text{or} \\ &K(i,j)Q(i,j) + w \geq K(i,j)Q(i,j) + Q/2 \end{aligned} \tag{7.48}$$

From (7.48), we have: $|w| > Q(i,j)/2$. Combining conditions 1 and 2, we get:

$$w \subset \left\{ -[|K(i,j)| - 1/2]Q(i,j), -\frac{Q(i,j)}{2}) \cup (\frac{Q(i,j)}{2}, [|K(i,j)| - 1/2]Q(i,j) \right\},$$

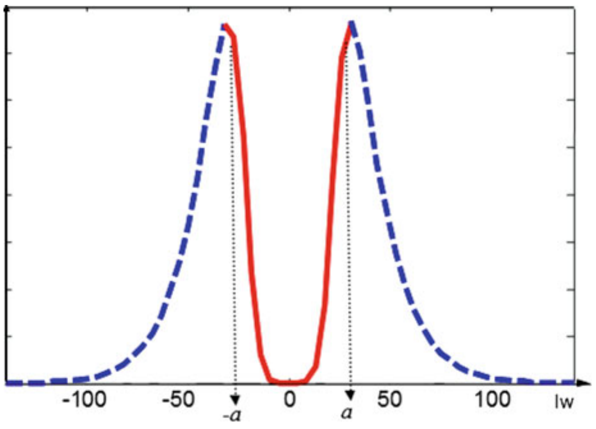


Fig. 7.10 Distribution of coefficients after omitting low-frequency components

Note that $|K(i,j)| \geq 2$ should hold.

7.5. Define the form of a locally optimal watermark detector, that corresponds to the watermarked coefficients pdf, assuming that they are selected by using the criterion $|K(i,j)| \geq 2$. The coefficients pdf can be modeled by a function illustrated in Fig. 7.10.

Solution:

In the considered case, the coefficients pdf from Fig. 7.10 can be approximately described by using the following function:

$$p(I_x) = \frac{\left(\frac{I_x}{a}\right)^{2n}}{1 + \left(\frac{I_x}{a}\right)^{2n}} e^{-\left|\frac{I_x}{a}\right|^{2\gamma}}, \tag{7.49}$$

where parameter a defines the positions of the pdf maxima, while n controls the pdf decay between the maximum and the origin. The parameter γ is usually equal to 1/2, 1, or 2.

A locally optimal detector can be defined as:

$$D_{\text{opt}} = -\frac{p'(I_x)}{p(I_x)} \cdot w, \tag{7.50}$$

which in the case of the specified function p becomes:

$$D_{\text{opt}} = \sum_{i=1}^{K} w_i \left(\frac{\gamma}{a^{2\gamma}} I_{x_i}^{2\gamma-1} \operatorname{sgn}\left(\frac{I_{x_i}}{a}\right)^{2\gamma} - \frac{n}{I_{x_i}\left(1 + \left(\frac{I_{x_i}}{a}\right)^{2n}\right)} \right). \tag{7.51}$$

7.6. By using the results obtained in the Example 7.4, derive the condition for watermarked coefficients selection which would provide the robustness to a certain JPEG quantization degree defined by Q'. By robustness we assume that the coefficients pdf is preserved even after quantization Q', in order to provide successful detection by using locally optimal detector. Assume that the watermark embedding is done according to:

$$I_w(i,j) = \text{round}\left(\frac{I(i,j)}{Q(i,j)}\right)Q(i,j) + Q(i,j)w, \tag{7.52}$$

where Q is the quantization with high quality factor QF (i.e., a low compression ratio).

Solution:

In order to provide robustness to the quantization Q', the criterion for coefficients selection should be modified. The watermarked coefficients will be robust after applying Q' if they are not rounded to zero, i.e., if the following condition is satisfied:

$$|K(i,j)Q(i,j)| - |Q(i,j)w| > \frac{Q'(i,j)}{2}. \tag{7.53}$$

Note that the worst case is used in (7.53): the coefficient and the watermark have opposite signs. Hence, we may observe that for efficient watermark detection, the coefficients should be selected for watermarking if:

$$|K(i,j)| \geq w + \frac{Q'(i,j)}{2Q(i,j)}. \tag{7.54}$$

Therefore, if Q with an arbitrary high QF is used for watermark embedding, the robustness is satisfied even for Q' as long as the criterion (7.54) is satisfied. In this way, the procedure provides the full control over the robustness to any JPEG quantization degree.

Note that if the criterion is satisfied for QF$'$ $<$ QF, then the watermark detection will certainly be successful for any Q_x defined by QF$_x$ for which QF$_x$ $>$ QF$'$ holds.

7.7. A speech watermarking procedure in the time-frequency domain can be designed according to the following instructions:

1. Voiced speech regions are used for watermarking.
2. Watermark is modeled to follow the time-frequency characteristics of speech components in the selected region.
3. Watermark embedding and detection is done in the time-frequency domain by using the S-method and time-varying filtering procedure.

   (a) Design a time-frequency mask for watermark modeling and define the modeled watermark form.
   (b) Define a watermark detection procedure in the time-frequency domain which includes the cross-terms.

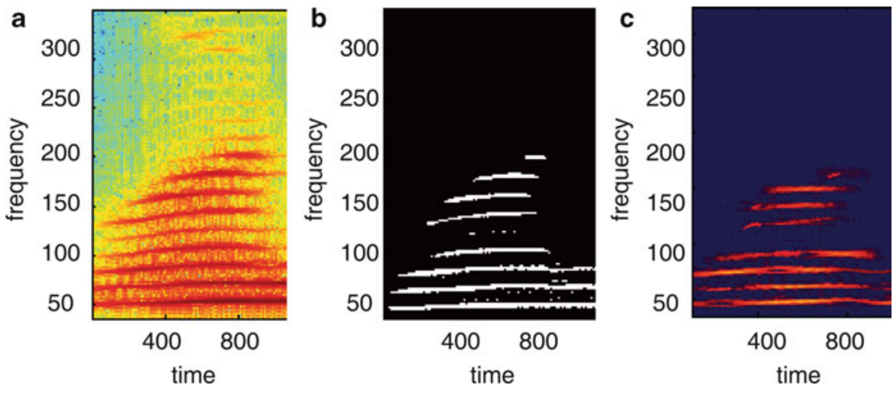


Fig. 7.11 (**a**) Speech region selected for watermarking, (**b**) mask function, (**c**) time-frequency representation of modeled watermark

Solution:

(a) By using the S-method (with $L = 3$), a voiced speech region is selected:

$$D = \{(t, \omega) : \ t \in (t_1, t_2), \ \omega \in (\omega_1, \omega_2)\},$$

where t_1 and t_2 are the start and endpoints in the time domain, while frequency range is $\omega \in (\omega_1, \omega_2)$. According to (7.42), the time-frequency mask can be defined as:

$$L_M(t, \omega) = \begin{cases} 1 \, for \ (t, \omega) \in D \text{ and } |\mathrm{SM}(t, \omega)| > \xi \\ 0 for \ (t, \omega) \notin D \text{ or } |\mathrm{SM}(t, \omega)| \ \leq \ \xi \end{cases},$$

where parameter λ within the energy floor ξ can be set to 0.7. The illustration of speech region is given in Fig. 7.11a, the corresponding mask is shown in Fig. 7.11b, while the time-frequency representation of the modeled watermark is shown in Fig. 7.11c. The modeled version of the watermark is obtained by using (7.43).

Note that the time-frequency characteristics of watermark correspond to the speech components. Hence, it would be difficult to remove the watermark without introducing serious signal quality degradation.

(b) The watermark detection can be performed by using the S-method with $L = 32$ to intentionally produce the cross-terms:

$$D = \sum_{i=1}^{N} \mathrm{SM}_{w_{\mathrm{key}}}^{i} \mathrm{SM}_{x_{\mathrm{w}}}^{i} + \sum_{\substack{i,j=1 \\ i \neq j}}^{N} \mathrm{SM}_{w_{\mathrm{key}}}^{i,j} \mathrm{SM}_{x_{\mathrm{w}}}^{i,j}, \tag{7.55}$$

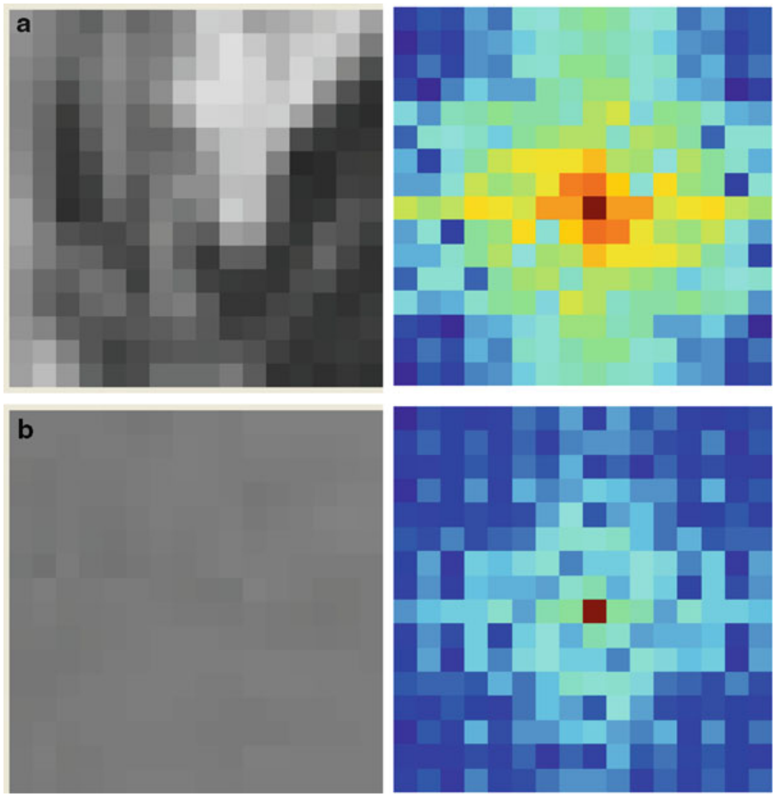


Fig. 7.12 (**a**) Busy region and its spectrogram, (**b**) flat region and its spectrogram

where the index w is related to watermark and x_{w} to watermarked coefficients. Although the cross-terms are usually undesirable in the time-frequency analysis, they may increase performance of watermark detector.

7.8. In analogy with one-dimensional case described in the previous example, design a space/spatial-frequency-based image watermarking procedure.
Note: Space/spatial-frequency representation is calculated for each pixel and it reflects the two-dimensional local frequency content around the pixel. The two-dimensional form of the STFT for the window of size $N \times N$ is extended from the one-dimensional version as:

$$\mathrm{STFT}(n_1, n_2, k_1, k_2) = \sum_{i_1=-N/2}^{N/2-1} \sum_{i_2=-N/2}^{N/2-1} I(n_1 + i_1, n_2 + i_2) w(i_1, i_2) e^{-j\frac{2\pi}{N}(k_1 i_1 + k_2 i_2)}.$$

Solution:

Space/spatial-frequency representation can be used for classification between the flat and busy image regions. Namely, busy image regions are preferred in watermarking, because it is easier to provide watermark imperceptibility. The examples of busy and flat image regions are shown in Fig. 7.12a, b, respectively. Note that, unlike the busy regions, the flat regions contain small number of significant components in the space/spatial-frequency domain, which can be used as a criterion for regions classification.

Following analogy with the procedure for speech signals, watermark can be modeled according to the local frequency characteristics defined by the mask L:

$$w_{\text{key}}(n_1,n_2)=\sum_{\omega_1}\sum_{\omega_2}\text{STFT}_p(n_1,n_2,\omega_1,\omega_2)L(n_1,n_2,\omega_1,\omega_2), \tag{7.56}$$

where STFT_p is a short-time Fourier transform of the two-dimensional pseudorandom sequence. The mask is obtained as:

$$L(n_1,n_2,\omega_1,\omega_2)=\begin{cases}1\,\text{for}\ (\omega_1,\omega_2):|\text{STFT}(n_1,n_2,\omega_1,\omega_2)|^2>\xi\\ 0\,\text{for}\ (\omega_1,\omega_2):|\text{STFT}(n_1,n_2,\omega_1,\omega_2)|^2\le\xi.\end{cases}$$

Watermark embedding and detection can be done in the space/spatial-frequency domain in the same way as in the case of speech signals.

References

1. Al-khassaweneh M, Aviyente S (2005) A time-frequency based perceptual and robust watermarking scheme. In: Proceedings of the EUSIPCO 2005, Antalya, Turkey
2. Barni M, Bartolini F (2004) Watermarking systems engineering. Marcel Dekker Inc, New York
3. Briassouli A, Strintzis MG (2004) Locally optimum nonlinearities for DCT watermark detection. IEEE Trans Image Process 13(12):1604–1618
4. Cox IJ, Miller ML, Bloom JA (2002) Digital watermarking. Academic, San Diego
5. Djurović I, Stanković S, Pitas I (2001) Digital watermarking in the fractional Fourier transformation domain. J Netw Comput Appl Academic 24(2):167–173
6. Esmaili S, Krishnan S, Raahemifar K (2003) Audio watermarking time-frequency characteristics. Canadian J Electr Comput Eng 28(2):57–61
7. Foo SW, Ho SM, Ng LM (2004) Audio watermarking using time-frequency compression expansion. In: Proceedings of the international symposium on circuits and systems, ISCAS 04, vol 3, pp III-201–III-4
8. Hernandez JR, Amado M, Perez Gonzales F (2000) DCT-domain watermarking techniques for still images: detector performance analysis and a new structure. IEEE Trans Image Process 9:55–68
9. Kirovski D, Malvar HS (2003) Spread-spectrum watermarking of audio signals. IEEE Trans Signal Process 51(4):1020–1033

10. Mobasseri BG, Zhang Y, Amin MG, Dogahe BM (2005) Designing robust watermarks using polynomial phase exponentials. In: Proceedings of acoustics, speech, and signal processing (ICASSP'05), vol 2, pp ii/833–ii/836
11. Muharemagić E, Furht B (2006) Survey of watermarking techniques and applications (-Chapter 3). In: Furht B, Kirovski D (eds) Multimedia watermarking techniques and applications. Auerbach Publication, Boca Raton, pp 91–130
12. Nikolaidis A, Pitas I (2003) Asymptotically optimal detection for additive watermarking in the DCT and DWT domains. IEEE Trans Image Process 12(5):563–571
13. Proceedings of the IEEE: special issue on identification and protection of multimedia information, vol 87, July 1999
14. Stankovic LJ, Stankovic S, Djurovic I (2000) Space/spatial-frequency based filtering. IEEE Trans Signal Process 48(8):2343–2352
15. Stanković S, Djurović I, Herpers R, Stanković LJ (2003) An approach to the optimal watermark detection. AEUE Int J Electron Commun 57(5):355–357
16. Stanković S, Djurović I, Pitas I (2001) Watermarking in the space/spatial-frequency domain using two-dimensional Radon-Wigner distribution. IEEE Trans Image Process 10:650–658
17. Stanković S, Orović I, Žarić N (2008) Robust watermarking procedure based on JPEG-DCT image compression. J Electron Imaging 17(4):043001
18. Stanković S, Orović I, Žarić N (2010) An application of multidimensional time-frequency analysis as a base for the unified watermarking approach. IEEE Trans Image Process 19 (2):736–745
19. Stanković S (2000) About time-variant filtering of speech signals with time-frequency distributions for hands-free telephone systems. Signal Process 80(9):1777–1785
20. Stanković S, Orović I, Žarić N (2008) Robust speech watermarking in the time-frequency domain. EURASIP J Adv Signal Process, Issue ID 519206
21. Steinebach M, Dittmann J (2003) Watermarking-based digital audio data authentication. EURASIP J Appl Signal Process 2003(10):1001–1015
22. Wickens TD (2002) Elementary signal detection theory. Oxford University Press, Oxford

Chapter 8
Multimedia Signals and Systems in Telemedicine

Due to the advances in technology and medicine, humans tend to live longer. This has increased the pressure on healthcare systems worldwide to provide higher quality health care for a greater number of patients. A greater demand for healthcare services prompted researchers to seek new ways of organizing and delivering healthcare services. Telemedicine, as a new research area, promotes the use of multimedia systems as a way of increasing the availability of care for patients in addition to cost and time-saving strategies. In other words, telemedicine provides a way for patients to be examined and treated, while the healthcare provider and the patient are at different physical locations. Using telemedicine technologies, future hospitals will provide healthcare services to patients all over the world using multimedia systems and signals that can be acquired over distances. Signal and image transmission, storage, and processing are the major components of telemedicine.

8.1 General Health Care

8.1.1 Telenursing

Telenursing requires the use of multimedia systems and signals to provide nursing practice over the distance. It was developed as a need to alter the current nursing practices and provide home care to older adults and/or other patient groups, which preferred to stay in the comfort of their own homes. Multimedia technologies (e.g., video-telephony) allow patients to maintain their autonomy by enhancing their emotional, relational, and social abilities. Generally, the patients welcome the use of multimedia systems to communicate with a nurse about their physical and psychological conditions. So far, the use of advanced technology did not have any significant effects on healthcare providers and patients, as well as on their abilities to communicate.

DOI 10.1007/978-1-4614-4208-0_8, © Springer Science+Business Media, LLC 2012

Multimedia systems, when used for patient care, provide various advantages. As an example, let us mention that the multimedia signals and systems have been used to significantly reduce congestive heart failure readmission charges. Also, these systems can reduce the amount of needed time to care for a patient, while providing the same level of health care as in-patient visits. Similarly, an analysis of telenursing in the case of metered dose inhalers in a geriatric population have shown that multimedia systems can provide most of services, and only small percentage require on-site visits. It should be mentioned that some of the telenursing systems can reduce the number of visits to emergency departments or doctors in private practice.

It should be mentioned that other potential applications of telenursing also include, but are not limited to:

- Training nurses remotely
- Caring for patients in war zones
- Global collaboration between nurses

8.1.2 Telepharmacy

Particularly, in rural and remote areas the pharmacy services to patients are often limited. This has led to the creation of service called telepharmacy, which assumes providing pharmaceutical care to patients and medication dispensing from distance. In this way, the multimedia systems and technology preserve pharmacy services in remote rural communities. The telepharmacy services adhere to all official regulations and services as traditional pharmacies, including verification of drugs before dispensing and patient counseling. In other words, telepharmacy services maintain the same services as the traditional pharmacies and provide additional value-added features. Additional services provided by telepharmacies can also include point-of-care refill authorization and medication assistance referrals. Specifically, in a recent study analyzing the utility of telepharmacy services for education on a metered-dose inhaler technique, it has been shown that patient education provided by pharmacists via video was superior to education provided via written instructions on an inhaler package insert.

8.1.3 Telerehabilitation

Rehabilitation is based on the idea that therapeutic interventions can enhance patient outcomes, since human physiological system can dynamically alter as a function of inputs (e.g., exercise). Therefore, telerehabilitation tools enable us to decrease the distance between patients and clinicians/researchers, which opens up new possibilities for discovering and implementing optimized intervention strategies.

Telerehabilitation was established in 1997 when the National Institute on Disability and Rehabilitation Research (U.S. Department of Education) ranked it as one of the top priorities for a newly established Rehabilitation Engineering

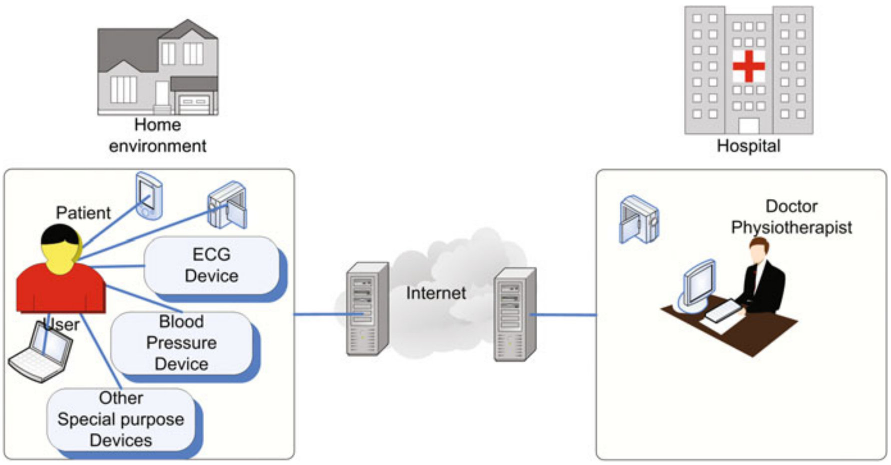


Fig. 8.1 A typical telerehabilitation system

Research Center (RERC). While telerehabilitation covers diverse fields of investigations (e.g., intelligent therapeutic robots and other health gadgets), it also addresses societal challenges in the delivery of rehabilitative services. The main efforts have been made to provide telecommunication techniques that are capable of supporting rehabilitation services at a distance, then to provide technology for monitoring and evaluating the rehabilitation progress, and, finally, to provide technology for therapeutic intervention at a distance, Fig. 8.1.

Having these comprehensive objectives, telerehabilitation may have far-reaching effects on patients. One such example is based on using a video consulting system in a community-based poststroke program that involves educational talks, exercise, and psychosocial support, proving significant improvements in the health status of the patients after the intervention.

Furthermore, the feasibility of telerehabilitation has been applied for functional electrical stimulation of affected arm after stroke or for evaluating the speech and swallowing status of laryngectomy patients following discharge from acute care. Telerehabilitation tools have also been used to address the fall risks. Nevertheless, we still need to acquire strong evidence regarding the impact of telerehabilitation on resources and associated costs to support clinical and policy decision making.

8.2 Specialist Health Care

8.2.1 Telecardiology

Telecardiology encompasses merging technology with cardiology in order to provide a patient with a proper medical care without disturbing the patient's daily

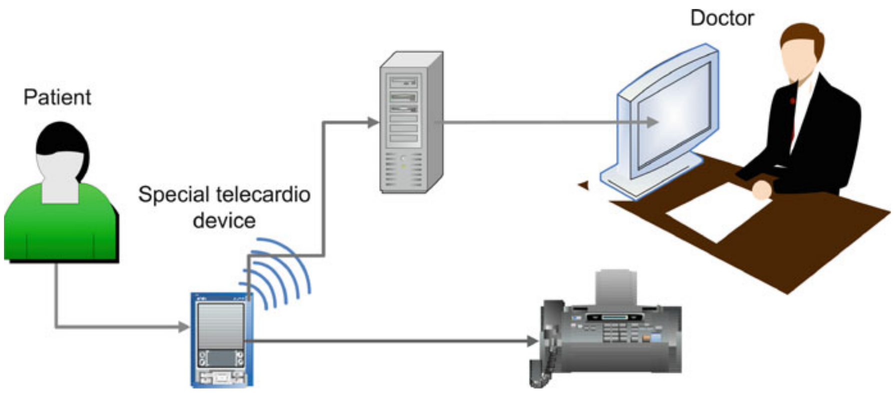


Fig. 8.2 An example of multimedia systems in telecardiology

routines. Due to the advances in multimedia systems, telecardiology is currently a well-developed medical discipline involving many different aspects of cardiology (e.g., acute coronary syndromes, congestive heart failure, sudden cardiac arrest, arrhythmias). It is safe to state that telecardiology has become an essential tool for cardiologists in either a hospital-based or community-based practices. Patient consultations with cardiologists via multimedia systems are becoming extremely common. For example, a consulting cardiologist receives many signals and images in real time to assess the patient condition (Fig. 8.2).

Further technological advances will be fueled by the development of novel sensors and multimedia systems. This will result in a move from device-centered to patient-oriented telemonitoring. By focusing on patient-oriented monitoring, a comprehensive approach of disease management, based on coordinating healthcare interventions, is provided. Such a possibility will not only help us with early diagnosis and quick interventions, but will also prove to be cost effective.

Therefore, telecardiology has the two major aims. The first aim is to reduce the healthcare cost. The second aim is to evaluate the efficiency of telecardiac tools (e.g., wireless ECG) at variable distances. By accomplishing these two aims, telecardiology will enhance the psychological well-being of patients in addition to bridging the gap between rural areas and hospitals. Note that, the search for new telecardiac technologies is a big challenge. However, various constraints such as institutional and financial factors may play a significant role in the further development of these multimedia systems needed in telecardiology before we see an increase of these tools in clinical practices.

In order to fulfill the aims of telecardiology, multimedia-based technologies have been increasingly applied to patients in small rural communities needing distance monitoring of their chronic heart conditions. These multimedia systems provided an alternative modality for effective cardiac care from a variable distances by utilizing information and communication technology.

8.2.2 Teleradiology

Teleradiology defines a field in which multimedia systems are used to acquire and interpret multimedia signals (e.g., radiological images) at different geographical locations. Teleradiology has especially flourished in recent years due to the increased development of digital imaging systems and the Internet. Nevertheless, it should be mentioned that teleradiology is not only acquisition and transmission of digital images between different locations, but also involves sharing of expertise between radiologists across the globe, providing radiological services to remote areas, around the clock coverage, etc.

Initially, teleradiology was developed to provide health care to wounded soldiers. In the 1980s, the first commercial teleradiology system was developed with the ability to capture and transfer radiological videos. However, further development of teleradiological systems was slowed down to lack of systems for inexpensive transfer of radiological videos. Due to the advances in multimedia systems for acquisition and transfer of video data (e.g., wavelet compression algorithms for images) and the development of the Internet telecommunication systems, we have witnessed a significant growth in teleradiological services. An illustration of teleradiological communication system is shown in Fig. 8.3. A picture archive and communication system is used to store, transfer, and display digital images acquired in diagnostic imaging. The archive server is used to provide a long-term backup of the data. The radiology information server is used to connect different aspects of the radiology information system.

Given the current state-of-the-art, when it comes to multimedia systems, most of the current efforts in teleradiology are geared toward medico-legal issues. Teleradiology is one of the first fields where the development of technologies in the recent years has sparked intense professional and even legal debates regarding the role of radiologists in patient care. For example, teleradiology can provide great benefits in emergency departments, when used correctly. However, poorly implemented teleradiological services can degrade the quality of patient care. Hence, it has been urged that during the design, management, and performance of teleradiology services, radiologists should play a significant role.

8.2.3 Telesurgery

Dissemination of new surgical skills and techniques across the wide spectrum of practicing surgeons is often difficult and time consuming, especially because the practicing surgeons can be located very far from large teaching centers. Therefore, telesurgery provides multiple advantages to practicing surgeons, including but not limited to dissemination of expertise, widespread patient care, cost savings, and improved community care (Fig. 8.4). It is expected that more widespread

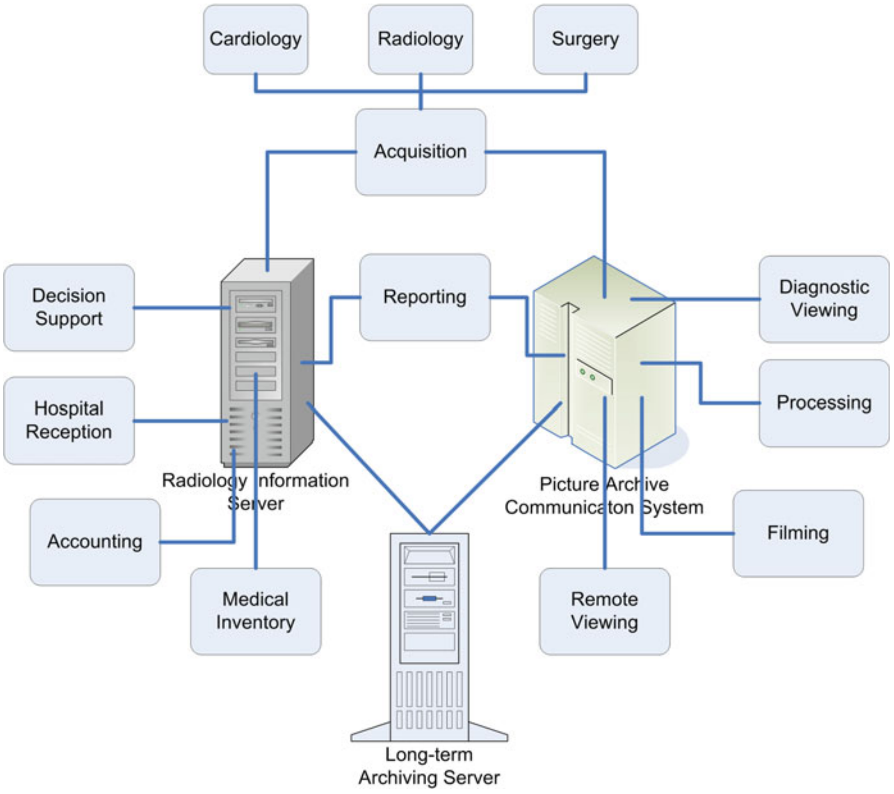


Fig. 8.3 A teleradiology system

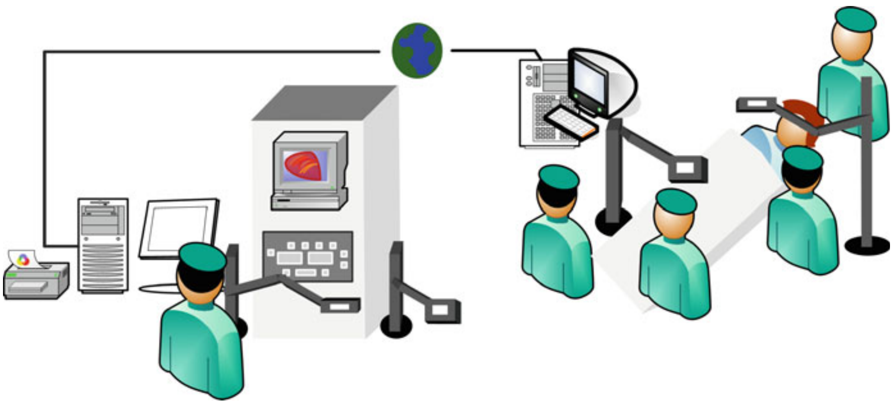


Fig. 8.4 A typical use of a telesurgery system

multimedia systems and technologies may exist to launch everyday telesurgery procedures within a few years.

Specifically, telesurgery has already shown to be a powerful method for performing minimally invasive surgery (MIS) because patients recover more rapidly when small MIS incisions are made in comparison to conventional methods. To examine the practicality of telesurgery over long distances, a recent study showed that operators using a telesurgery platform can complete maneuvers with delays up to 500 ms and no additional increase in error rates. Also, the emulated surgery in animals can be effectively executed using either ground or satellite, while keeping the satellite bandwidth above 5 Mb/s.

References

1. Anvari M (2007) Telesurgery: remote knowledge translation in clinical surgery. World J Surg 31(8):1545–1550
2. Arnaert A, Delesie L (2001) Telenursing for the elderly. The case for care via video-telephony. J Telemed Telecare 7(6):311–316
3. Barneveld Binkhuysena FH, Ranschaert ER (2011) Teleradiology: evolution and concepts. Eur J Radiol 78(2):205–209
4. Brunetti ND, Amodio G, De Gennaro L, Dellegrottaglie G, Pellegrino PL, Di Biase M, Antonelli G (2009) Telecardiology applied to a region-wide public emergency health-care service. J Thromb Thrombolysis 28(1):23–30
5. Bynum A, Hopkins D, Thomas A, Copeland N, Irwin C (2001) The effect of telepharmacy counseling on metered-dose inhaler technique among adolescents with asthma in rural Arkansas. Telemed J e-Health 7(3):207–217
6. Chan WM, Hjelm NM (2001) The role of telenursing in the provision of geriatric outreach services to residential homes in Hong Kong. J Telemed Telecare 7(1):38–46
7. Friesner DL, Scott DM, Rathke AM, Peterson CD, Anderson HC (2011) Do remote community telepharmacies have higher medication error rates than traditional community pharmacies? Evidence from the North Dakota telepharmacy project. J Am Pharm Assoc 51 (5):580–590
8. Garrelts JC, Gagnon M, Eisenberg C, Moerer J, Carrithers J (2010) Impact of telepharmacy in a multihospital health system. Am J Health Syst Pharm 67(17):1456–1462
9. Giansanti D, Morelli S, Maccioni G, Costantini G (2009) Toward the design of a wearable system for fall-risk detection in telerehabilitation. Telemed e-Health 15(3):296–299
10. Hagan L, Morin D, Lepine R (2000) Evaluation of telenursing outcomes: satisfaction, self-care practices, and cost savings. Public Health Nurs 17(4):305–313
11. Hermann VH, Herzog M, Jordan R, Hofherr K, Levine P, Page SJ (2010) Telerehabilitation and electrical stimulation: an occupation-based, client-centered stroke intervention. Am J Occup Ther 64(1):73–81
12. Jerant AF, Azari R, Martinez C, Nesbitt TS (2003) A randomized trial of telenursing to reduce hospitalization for heart failure: patient-centered outcomes and nursing indicators. Home Health Care Serv Q 22(1):1–20
13. Jönsson AM, Willman A (2009) Telenursing in home care services: experiences of registered nurses. Electron J Health Inform 4(1):e9–1–7
14. Kairy D, Lehoux P, Vincent C, Visintin M (2009) A systematic review of clinical outcomes, clinical process, healthcare utilization and costs associated with telerehabilitation. Disabil Rehabil 31(6):427–447
15. Katz ME (2010) Pediatric teleradiology: the benefits. Pediatr Radiol 40(8):1345–1348

16. Lai JCK, Woo J, Hui E, Chan WM (2004) Telerehabilitation – a new model for community-based stroke rehabilitation. J Telemed Telecare 10(4):199–205
17. Lam AY, Rose D (2009) Telepharmacy services in an urban community health clinic system. J Am Pharm Assoc 49(5):652–659
18. Mitchell JR, Sharma P, Modi J, Simpson M, Thomas M, Hill MD, Goyal M (2011) A smartphone client-server teleradiology system for primary diagnosis of acute stroke. J Med Internet Res 2:e31
19. Nikus K, Lähteenmäkib J, Lehto P, Eskola M (2009) The role of continuous monitoring in a 24/7 telecardiology consultation service – a feasibility study. J Electrocardiol 42(6):473–480
20. Pappasa Y, Sealeb C (2010) The physical examination in telecardiology and televascular consultations: a study using conversation analysis. Patient Educ Couns 81(1):113–118
21. Peterson CD, Anderson HC (2004) The North Dakota telepharmacy project: restoring and retaining pharmacy services in rural communities. J Pharm Technol 20(1):28–39
22. Wakefield DS, Ward MM, Loes JL, O'Brien J, Sperry L (2010) Implementation of a telepharmacy service to provide round-the-clock medication order review by pharmacists. Am J Health Syst Pharm 67(23):2052–2057
23. Ward E, Crombie J, Trickey M, Hill A, Theodoros D, Russell T (2009) Assessment of communication and swallowing post-laryngectomy: a telerehabilitation trial. J Telemed Telecare 15(5):232–237
24. Whitten P, Mair F, Collins B (2001) Home telenursing in Kansas: patients' perceptions of uses and benefits. J Telemed Telecare 3(1):67–69
25. Winters JM (2002) Telerehabilitation research: emerging opportunities. Annu Rev Biomed Eng 4:287–320

Chapter 9
Multimedia Communications

Multimedia communications are related to the transmission of multimedia data. Different types of networks were initially used for transmission depending on the data type. Nowadays, given the high degree of interaction among different types of networks, all of them together would be called a global multimedia network. When it comes to multimedia communications, it is necessary to look back on some widely used networks, such as telephone communication network, Public Switched Telephone Network (PSTN), data network, television broadcasting network, broadcast TV network (BTVN), Integrated Services Digital Network (ISDN), and broadband multipurpose networks.

9.1 An Overview of Different Networks Types

9.1.1 Telephone Networks

Telephone networks were originally designed for transmission of voice, although nowadays they can be used for transmission of multimedia signals. These networks are based on the local exchange (LE) offices which connect different types of users, from households and small businesses to the various local networks of large corporations. National PSTN are interconnected. In fact, international calls are realized using the International gateway exchange (IGE), by which data are directed to a specified PSTN abroad. Furthermore, these networks are connected to mobile operators via Mobile Switching Center (MSC). During each communication between the users, a connection is created, i.e., a circuit is set up through the network. Hence, these networks operate in a circuit mode. The PSTN modems are used for transmission of digital signals over analog networks. In this way, it is ensured that the digital data is routed within the PSTN network in the same way as voice, for which it was primarily designed (Fig. 9.1).

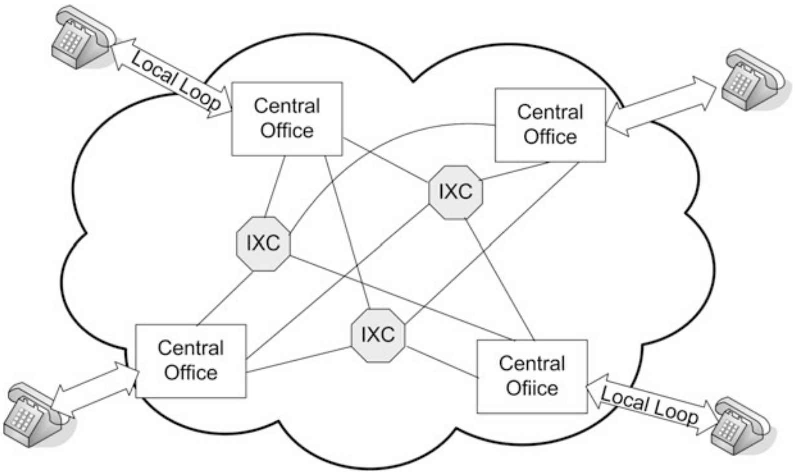


Fig. 9.1 An illustration of PSTN as a system of Central Offices that provide access to subscribers and Inter eXchange Carriers providing long-distance services

Due to the fact that modems usually have two channels, one for telephone communications (it requires a small bandwidth), and others with a high data rate for audio and video connections, the PSTNs today can be treated as multimedia networks.

9.1.2 Data Network

Data networks are basically developed to provide file transfer and email communication. Initially, the **X.25** network and the *Internet* were the two types of networks developed for this purpose. Over the time, the Internet has become the primary network used to realize the multimedia communication. Internet is a global network composed of many interconnected networks that use the same rules of communication, i.e., the same communication protocols. The communication protocol should provide the same syntax for all data to be transferred.

The basic structure of the Internet network is quite complex and includes the global Internet backbone network (IBN). Different types of networks are connected to the backbone network: computer networks connected to the PSTN through the same intermediary called Internet Service Provider (ISP), then the local area networks (LAN), computer networks of large companies, and so on. All these types of networks are connected to the IBN over the network units called gateways (GW). Therefore, it can be said that the Internet is a global network comprising a huge number of interconnected, smaller networks.

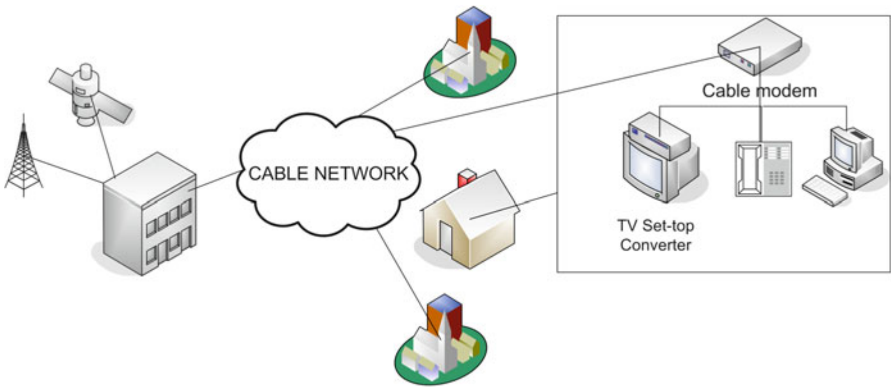


Fig. 9.2 An illustration of broadcast cable network

9.1.3 Broadcast TV Networks

Broadcast TV networks were initially designed specifically for the transmission of TV signals. The cable networks are used in urban environment (towns), while satellite networks are more suitable for the large areas. These networks are also upgraded to enable multiuser and service interaction. A special device called set-top box in cable networks provides the control of TV channels, but for interaction purposes it can also provide other multimedia communications services. A user can connect to the PSTN and the Internet using channels with either low or high data rates. Therefore, the ability to link with other networks is the basis of what we call the interactive television (Fig. 9.2).

9.1.4 Integrated Services Digital Network (ISDN)

Unlike the previous networks, the ISDNs are designed for simultaneous transmission of multiple data types. Hence, they were created to extend types of services available within the PSTN. In other words, the ISDNs are designed to integrate voice and nonvoice services together in digital form. The ISDN supports both switched and nonswitched connections. Also, switched connection includes both packet-switched and circuit-switched connections. A user has access to the ISDN through the local interface to a digital "pipe" with a certain bit rate. It may require a capacity that is sufficient to handle a phone and a personal computer. However, larger business user may connect a digital Private Branch eXchange (PBX—switch station for telephone systems used by the companies to connect all internal phones to external line) and LAN to the ISDN, which will require higher capacity pipe (Fig. 9.3).

There are two basic types of ISDN service: Basic Rate Interface (BRI) and Primary Rate Interface (PRI). The BRI consists of two 64 Kb/s B channels (Bearer channel) and

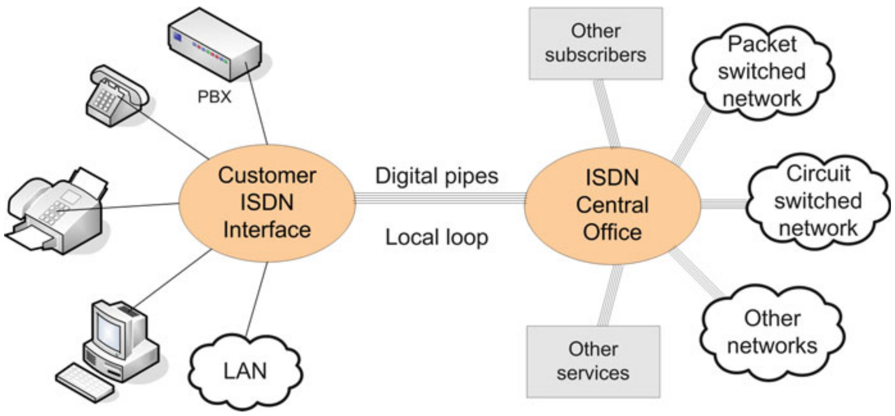


Fig. 9.3 Illustration of ISDN communication concepts

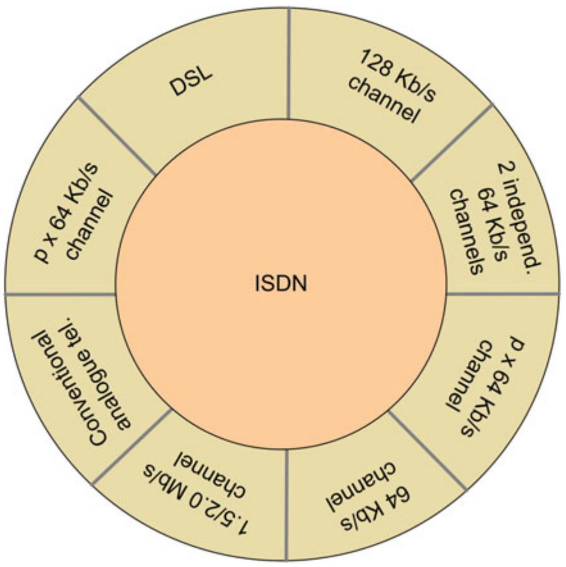


Fig. 9.4 Services provided by an ISDN

one 16 Kb/s D channel (Data channel), which provide in total 144 Kb/s. B channel transmits digital voice, data, or compressed video. This service is intended to meet the needs of most individual users. D channel is used for signaling, i.e., it controls B channels.

The PRI is intended for users with higher capacity requirements. Typically, the channel structure is 23 B channels plus one D channel of 64 Kb/s, having a total of 1,536 Kb/s. Generally, the PRI consists of 30 B channels: $n \times 64$ Kb/s, for $n = 1,2,\ldots,30$, and one additional 64 Kb/s D channel (1,984 Kb/s in total) (Fig. 9.4).

9.1.5 Multiservice Broadband Networks

Broadband networks are designed as the enhanced ISDNs, with a data rate that corresponds to the maximum flow of the ISDN: 30×64 Kb/s = 1,920 Kb/s ≈ 2 Mb/s. Hence, these networks are called the B-ISDNs (Broadband ISDN), while the ordinary ISDNs are often referred to as the N-ISDN (Narrowband ISDN). The B-ISDNs are used as the basis for the development of other broadband multiservice networks.

Since the purpose of multiservice networks is to provide multiple services, assuming different multimedia applications with different bit rates, switching, and transmission methods should be more efficient. Thus, the data belonging to different multimedia types are first divided into cells of constant length (fixed-size packets) that are transmitted over the network. Switching the fixed-size cells is much faster than the variable-length packets.

It is interesting that depending on the type of multimedia data, the cells transfer rate can differ. This method is called the ATM transfer mode, while the networks are called the cell-switching networks. Depending on the size and application, these can be ATM LAN, or larger ATM MAN (metropolitan area network).

9.2 Multimedia Applications

Applications that involve different types of multimedia data can be divided into several categories, such as interpersonal communications, interactive Internet applications, and entertainment.

9.2.1 Communications

Communication may include transmission of voice, images, text, and/or video data.

Voice transmission can be done by using the PSTN, but a more interesting example of communication is by using a computer. In such cases, it is necessary that a computer has an appropriate extra hardware (phone cards and related software). Also, by using special servers, additional voice-mail service can be provided.

Teleconferencing is also an interesting form of communication, which is used to establish a conference call using a device called audio bridge.

An especially important role plays the *telephony over the Internet*. In this case, the speech signal is transmitted in the packets form (i.e., as in the case of data transmission network), Fig. 9.5. This type of telephony is called packet voice or voice over IP (VoIP), due to the transmission using the Internet Protocol (IP). The communication between computers (on the Internet) and phones (connected to the

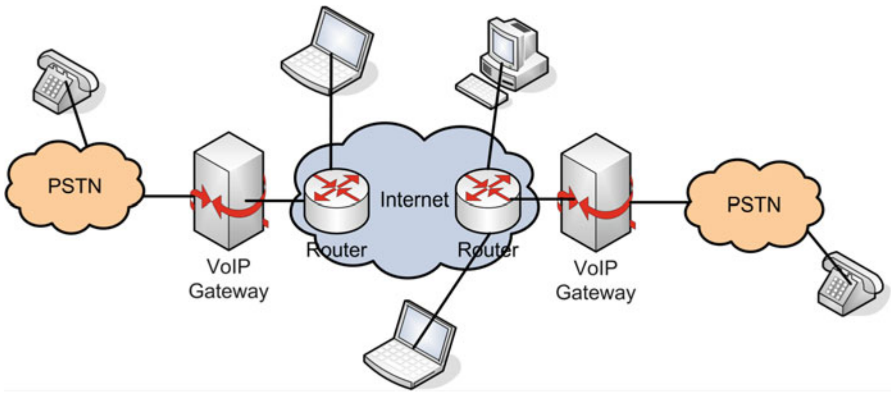


Fig. 9.5 A scheme of an VoIP system

PSTN or ISDN) is carried out through a special device called a telephone gateway. Specifically, the computer communicates with the associated phone gateway device and requires a connection to one of the registered users of this service. The gateway based on the phone number initiates the call with a user connected to the PSTN or ISDN. Hence, the telephony gateway represents a kind of adapter between the packet and circuit modes.

Text transmission is now carried out in a number of applications by using email. For this type of communication, it is necessary to have an email server, where each user will have a space to store emails (i.e., mailboxes). Each network should have email servers connected to the Internet over a gateway.

Particularly interesting applications are those involving *voice and video*. Nowadays, they can be transmitted over each of the networks described earlier. Communication can be established between two parties (person to person) or can be realized in the form of *videoconferencing*, when all conference participants can communicate with each other. To achieve the communication without a large number of channels (if n is the number of participants, we should have $n - 1$ outgoing and incoming channels), the multicontrol units (MCUs) are used. In this case, only a singe two-way channel is required between each participant and the MCU. Therefore, the MCU transmits the signal from active participants (those who currently speak) to the other participants. The conference with multiple participants is based on *multicasting,* which provides transmission from any participant to all other participants belonging to the predefined multicast group. This mode of communication does not require a separate MCU. A multicast participant broadcasts the material using the appropriate application (e.g., the VLC). This form of communication can be used even in monitoring systems, when a large number of cameras transmit the video material to different multiple locations within the network (tunnels, railways, subways, etc.).

9.2.2 Interactive Internet Applications

Besides already described applications, where direct communication is established between two or more users, the *interactive Internet applications* are nowadays very popular. They use interaction with Web servers, which stores a large number of multimedia files. These files on Web servers can be in the form of a hypertext, text, or hypermedia documents. The main feature of the Web is that there is no central database, but each server on the network is a separate source of information. Hence, using all the available space on the server, information can be shared with all users on the Internet. In addition, any term from a given document can be hyperlinked with the desired address on the Internet. Each document has a first page called a home page with a unique **U**niform **R**esource **L**ocator (URL). It is not necessary to explain that such a broad exchange of information requires strict standards that enable a unique representation of data for all users. The standard format for the Web pages is **H**yper **T**ext **M**arkup **L**anguage (HTML).

9.2.3 Entertainment Applications

The entertainment applications can be generally divided into video on demand and interactive television.

Video on demand means that users access a server containing preferred video materials. Since many customers may require the same movie in short time intervals, the movies are split into sequences that are broadcast to all users. This solution is called near video on demand.

Interactive television is based on the communication between TV users and service providers. Namely, we said that in cable broadcasting networks, a dual-channel device can establish a connection to the PSTN (low bit rate) and the Internet network (high bit rate). Hence, the connection to the PSTN can be used for voting, games, home shopping, etc., which is called the interactive television.

9.3 Multiplexing

The goal of each network is to connect users and to provide communication between them. The connection can be accomplished via routers, hubs, or switches. The simplest connection can be achieved via a switch, Fig. 9.6. Note that, for establishing the connection, we need one selector at each side, which will connect one of the available ports. However, there is a serious limitation, because the connection can be established only between two users at the same time. On the other hand, a realization in which each possible pair of ports can communicate simultaneously can make the system very complex and impractical.

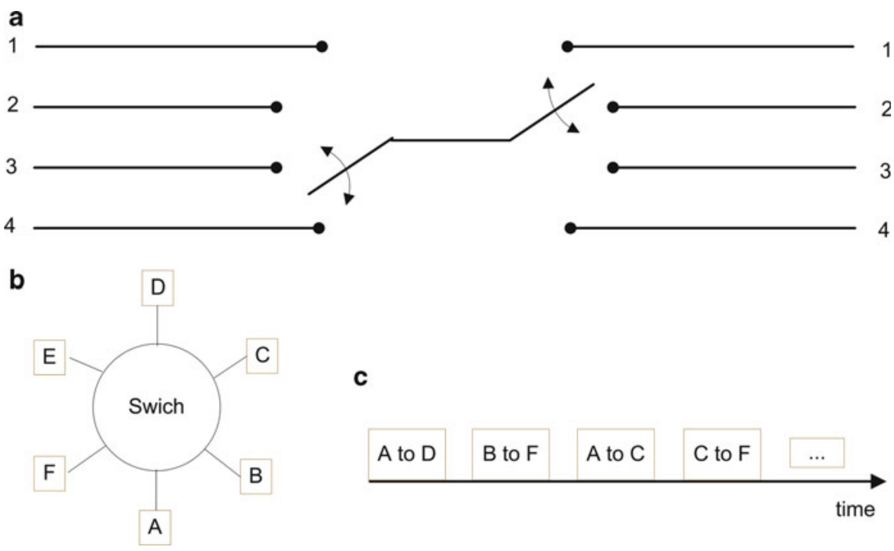


Fig. 9.6 (**a**) Switch connects a pair of users, (**b**) each user is connected by a separate link, thus forming a radial network, (**c**) time-division multiplexing

This problem is solved by using time or frequency multiplexing, and in some applications, we use a combination of time and frequency multiplexing. In a time division multiplexing (Fig. 9.6c), one pair of connections is established during a certain interval. Frequency multiplexing is based on the use of different frequency bands for simultaneous communication.

The networks can be divided into asynchronous, synchronous and isochronous systems. In asynchronous network, the time required to deliver the data is unknown (Ethernet, storage systems buses, etc.). The synchronous networks guarantee a constant data rate and small fixed delay. Hence, it is used in real-time applications (broadcasting). Finally, the isochronous system can be described as a strictly controlled asynchronous network (e.g., ATM network).

9.4 Quality of Service: QoS

Important features of the network are the QoS (Quality of Service) parameters. In the circuit-switched networks, the QoS parameters are the bit rate, the average error, and the transmission delay. In the packet-switched networks, the QoS parameters are the maximum packet size, the average error, the mean data rate, the mean packet delay, and the total delay.

A large number of networks do not provide a reliable service. It means that during the data transfer, erroneously transmitted blocks are dismissed. These networks are called networks with the best-effort service. In the case of reliable service, if it happens that a block of data is transferred with an error, the source is asked to retransmit the same block, which introduces additional delay.

To calculate the probability that a bit error occurs in one block with N bits is:

$$P_{\mathrm{B}} = 1 - (1 - P)^N,$$

where P is the probability of error for a single bit. This can be approximated by:

$$P_{\mathrm{B}} = NP \text{ for } NP < 1.$$

If we assume $P_{\mathrm{B}} = 0.05$ and $P = 10^{-4}$, we get the approximate length of the block: $N = 500$ bits.

9.5 Internet

The Internet is a global network of interconnected computers or "the network of all networks". The most important feature of the Internet is the facility of information sharing. On the Internet, one can find a company presentation, personal presentations, books, encyclopedias, magazines, archives, and many other types of information and multimedia content. All components of the Internet are connected by a communication medium that may be of different types: fiber optics, radio communication links, satellite communication links, etc.

The data are divided into packets that are transmitted across routers in the network (Fig. 9.7).

A router checks the destination address and forwards the package to the next router on the path to the destination. If there is more than one router in the direction of destination, the forwarding router chooses the least loaded path. Thus, the packets originating from the same data set do not have to travel along the same path within the network. In these networks, the problems often occur in the form of packet delay or even packet loss due to network congestion.

Internet works on the client-server principle (Fig. 9.8). Clients are users who require information from the server. Servers are devices used to store the data and send them to clients if required. The connection between client and server exists only when there is a need for providing information.

9.6 IP Address

The IP address is used to uniquely identify devices (computers, servers, gateways, routers) within the network. Namely, in order to provide the transfer of information between clients and servers, it is essential that every computer in the network has a corresponding address. In this way, for each data packet, we know the originating

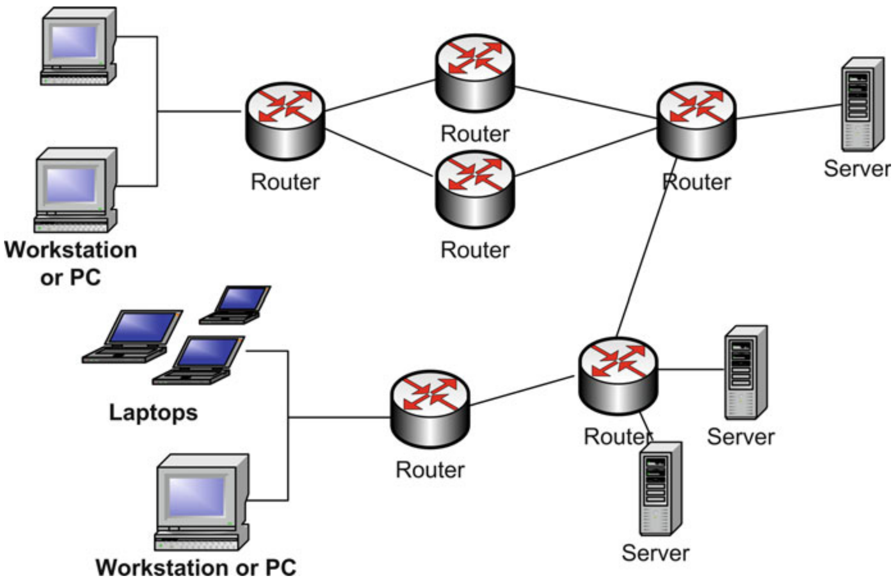


Fig. 9.7 Network as a system of routers, servers, and workstations

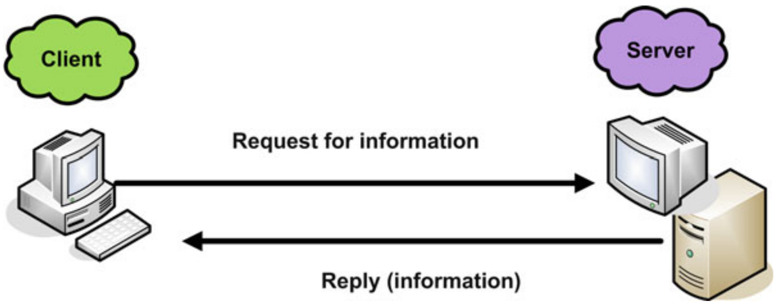


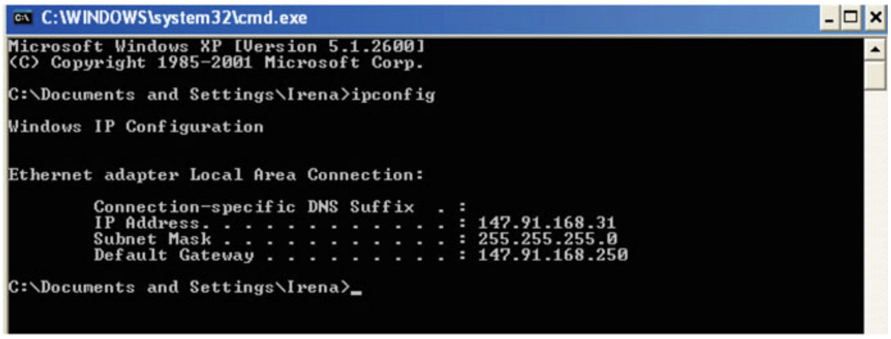
Fig. 9.8 Communication between server and client

and destination addresses. An IP address can be permanent and it is called static, or it can change whenever we connect to the Internet, which is called dynamic. An example of IP address displayed in the Command prompt is given in Fig. 9.9.

IP Address 147.91.168.31

9.6.1 Format of IPv4 Address

An IP address according to the IPv4 standard consists of four numbers separated by dots:



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Irena>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 147.91.168.31
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 147.91.168.250

C:\Documents and Settings\Irena>_
```

Fig. 9.9 Information about the IP address using the Command Prompt

Number1. Number2. Number3. Number4

where the numbers range from 0 to 255. An IP address contains 32 bits, which means that 8 bits are used for each of the four numbers. Each IP address can be viewed as a two-part sequence of bits: one part is related to the network identifier (Network ID) and the other to the user identifier (User ID). There are different classes of IP addresses: A, B, C, etc.

The position of the first zero bit in the sequence determines the class to which the corresponding IP address belongs: zero bit on the first position—Class A, zero on the second position—Class B, zero on the third position—Class C, and so on. Class A addresses use 7 bits for the network ID and 24 bits for the user ID. An address from class B uses 14 bits for the network and 16 bits for the user ID, while the class C addresses have 21 bit for the Network ID and 8 bits for the User ID.

Number of bits	1	7	24
Class A	0	Network ID	User ID

Number of bits	2		14	16
Class B	1	0	Network ID	User ID

Number of bits	3			21	8
Class C	1	1	0	Network ID	User ID

Each of these classes is designed to be used within the networks of corresponding sizes. Consider first the addresses from class A.

Class A:
Networks ID: from 1 to 126
User ID: from 0.0.1 to 255.255.254
The total number of networks with IP addresses that belong to class A is 126, while the total number of users that may be within such a network is 16,777,214.

We can similarly determine the total number of networks and users for the addresses from classes B and C.

Class B:
Network ID: from 128.0 to 191.255 (total of 16,382 networks).
User ID: from 0.1 to 255.254 (a total of 65,534 users in the network).

Class C:
Network ID: from 192.0.0 to 223.255.255 (can support up to 2,097,152 networks).
User ID: from 1 to 254 (total of 254 users in the network).

Given the number of bits reserved for network and user IDs, it can be concluded that class A addresses are used in the networks with a large number of users such as national networks, while class C addresses are used within the networks with a few users (a small LAN).

Some IP addresses for special purposes are:

- The address whose User ID is made of all zeros (but not the Network ID), represents the IP address of the network.
- The address that in the binary form contains all bits equal to 1 within the User ID is used for broadcasting within the network.

9.6.2 *Classless Addressing*

This method implies the existence of *subnet masks*. It does not require the existence of classes. The subnet mask defines which part of the IP address indicates the network and which part indicates the computers within the network. We will consider the three most common types of subnet mask:

Subnet mask: 255.0.0.0
Subnet mask: 255.255.0.0
Subnet mask: 255.255.255.0

If the subnet mask is equal to 255.0.0.0, then the first number of IP address refers to the network, and the remaining three numbers indicate the computers in the network. Assume that the first number of IP address is denoted as **Num 1**. Then, knowing that the subnet mask is 255.0.0.0, we define the following terms:

IP address of the network:	Num1 . 0 . 0 . 0
IP addresses of computers:	Num1 . 0 . 0 . 1 to Num1 . 255 . 255 . 254
Broadcast address:	Num1 . 255 . 255 . 255

For the networks with the subnet mask equal to 255.255.0.0, the first two numbers indicates the network, while the remaining two numbers are used to denote computers within the network. In this case, the first two numbers of IP addresses will be equal for all computers within the network. If the first two numbers are denoted as **Num 1** and **Num 2,** then:

IP address of the network:	Num1.Num2.0.0
IP addresses of computers:	Num1.Num2.0.1 to Num1.Num2.255.254
Broadcast address:	Num1.Num2.255.255

When using the network mask 255.255.255.0, the first three numbers of IP addresses indicate the network, and the last number refers to the computers in the network. The first three numbers of IP addresses will be common to all computers on the network.

IP address of the network:	Num1.Num2.Num3.0
IP addresses of computers:	Num1.Num2.Num3. 1 to Num1.Num2.Num3.254
Broadcast address:	Num1.Num2.Num3.255

There are other types of subnet masks given in the binary form: they consist of the sequence of "1" followed by a sequence of "0." The length of the sequence with values "1" determines the network ID. For example, a mask with 9 ones (denoted by /9): 11111111 10000000 00000000 00000000 (or 255.127.0.0) means that the first nine bits determine the network ID and they are common to all computers in the network.

9.6.3 IPv6 Address Format

The IPv6 addressing is introduced in order to provide a larger address space than the IPv4. The IPv6 addresses are hierarchical and they can be used in a number of alternative formats. For example, to facilitate the use of the existing IPv4 routers, there is a format that allows the IPv4 addresses to be embedded in the IPv6 addresses. The address format is defined by the first set of bits in the address, called the prefix format. The IPv6 addresses are 128 bits long, unlike the 32 bits long IPv4 addresses. The IPv4 address space contains about $4.3 \cdot 10^9$ addresses, while the IPv6 supports approximately $3.4 \cdot 10^{38}$ addresses. IPv6 addresses consist of two parts: a 64-bit network prefix and a 64-bit identifier of computers/devices within the network. These addresses are classified into the following types:

- Unicast addresses identify individual network interfaces.
- Anycast addresses identify a group of network interfaces, usually at different locations.
- Multicast addresses are used to deliver data packets to multiple interfaces.
- Loopback addresses are used by an interface to send an IPv6 packet to itself.
- Unspecified address indicates the absence of an IPv6 address. For example, the new interface can be initialized using unspecified address (as the source address when sending packets) until it receives an IPv6 address.

Table 9.1 Examples of long and short formats of IPv6 addresses

IPv6 addresses	Long format	Short format
Unicast (an example)	10FB:0:0:0:C:ABC:1F0C:44DA	10FB::C:ABC:1F0C:44DA
Multicast	FF01:0:0:0:0:0:0:1F	FF01::1F
Loopback	0:0:0:0:0:0:0:1	::1

The IPv6 addresses are composed of eight groups with four hexadecimal numbers. The symbol ":" is used between groups (e.g., 4FDE: 0000:0000:0002:0022: F376: FF3B: AB3F).

Each digit is a 4-bit hexadecimal value. A continuous sequence of zeros within the IPv6 address can be denoted as "::". An example is given in Table 9.1.

9.6.3.1 Specific IPv6 Addresses

The unicast address has the following format:

Network ID	Interface ID
N bits	128 – N bits

There are several types of unicast addresses: global unicast address, site-local unicast address, link-local unicast address, IPv6 address with IPv4 address inserted, and so on. Link local addresses are used for communication of neighboring devices on the same link, and cannot be used outside the area. Site-local address is used for routing within a private network of the company. It can be used within the public IPv6 networks.

Global Unicast Address

Global routing prefix	Network ID	Interface ID
N	M	$128 - N - M$

For example, according to the standard RFC 3587 and RFC 3177, $N = 48$ bits, and $M = 16$ bits.

The format of link-local addresses

10	54-bit	64 bits
1111111010	0	Interface ID

The format of site-local addresses

10 bit	38-bit	16 bits	64 bits
1111111011	0	Network ID	Interface ID

IPv4-Mapped Embedded IPv6 Addresses

80-bit	16 bits	32-bit
0000 . . . 0000	FFFF	IPv4 address

This address designates an IPv4 device that is not compatible with IPv6 and whose IPv4 address has been mapped into the IPv6 format.

IPv4-compatible IPv6 address

80-bit	16 bits	32-bit
0000 . . .0000	0000	IPv4 address

The IPv4-compatible IPv6 address is used for devices that are compatible with both IPv4 and IPv6.

Multicast addresses

80-bit	4 bits	4 bits	112 bits
11111111. . .	Flags	Scope	Group ID

Four bits denoted as *flags* in a multicast address are: 000X. The case X = 0 indicates that the multicast has been permanently assigned by the Internet Assigned Numbers Authority (IANA). When X = 1, the multicast address is a transient multicast address.

A scope is a 4-bit field that defines the area of multicast groups. For example, interface-local area is the interface itself. For the link-local address, used for communication between neighboring interfaces on the same link, the area is actually the local link.

The Group ID is the identifier of a multicast group.

9.7 A Protocol Set for Data Transmission over the Internet (TCP/IP Environment)

Communication among different computers and smaller LANs within the Internet is possible by using the same protocol. Primarily, we should mention the *IP network layer protocol*, **T**ransmission **C**ontrol **P**rotocol (**TCP**), **U**ser **D**atagram **P**rotocol (**UDP**), **R**eal-time **T**ransport **P**rotocol (**RTP**) and its associated protocol **R**eal-time **T**ransport **C**ontrol **P**rotocol (**RTCP**). The TCP provides a reliable data transfer, because it requires the acknowledgement for the received packets. The UDP and RTP protocols represent the "best try" methods, since they do not require feedback on the transmitted data. Their application depends on the specific requirement. For example, the TCP can be used for the transfer of text, while the UDP can be used for VoIP. The RTP protocol can be used to transmit video signals when time-synchronization of video stream is required.

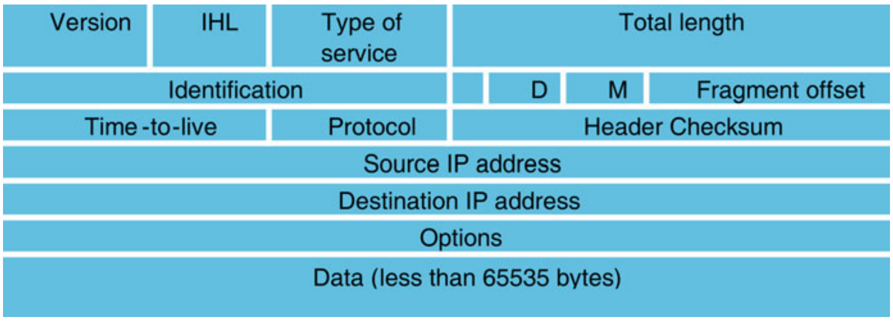


Fig. 9.10 Format of IP datagram

Let us observe the hierarchy of the considered protocols within the TCP/IP suite. The first level protocol is the IP protocol as a part of the operating system kernel and it belongs to the network layer. The IP protocol is in charge for IP addressing. It allows routers within the network to check the destination address and thus to route the data. Also, it determines the size of data packets. Thus, IP is a basic level protocol that is used by the higher level protocols.

The TCP and UDP belong to the transport level (higher than the network level), while the RTP and RTCP belong to the application level and they are located above the UDP protocol. In fact, both protocols (TCP and UDP) are always available, and the choice depends on the application requirements.

9.7.1 IP Protocol and IP Datagram

As we have pointed out, the IP protocol is a network layer protocol. It is responsible for routing data packets to different users through a system of gateways and routers. The IP protocol defines a unique IP address for each device in the network and identifies the device. When sending a data block to a specific address, the IP adds a header containing both a source and a destination IP address. Also, depending on the application, the IP specifies the requested transport protocol (TCP or UDP). This block of data with a specific header is referred to as IP datagram (Fig. 9.10). The datagram is forwarded to the first gateway on the path to the destination.

Field *Version* refers to the IP version (e.g., IPv4 or IPv6).

The header can be of different length and the intermediate header length (*IHL*) field is used to specify the length of the packet header.

The *Type of service* field specifies the priority for data to be transferred. It is used to transfer first the packages with higher priority.

The *total length* field refers to the length of the datagram including the header and data. It is sometimes necessary to transmit the data divided into several smaller packets called fragments. Then the value of the total length is used at the destination when collecting fragments of the original datagram.

In doing so, the value of the *Identification* field must be the same for all fragments, in order to connect all fragments at the destination into the original datagram.

The *D* bit (do not fragment) indicates that a packet should be transferred in whole without fragmenting. *M* bit (may fragment) is used when a packet is divided into several fragments. It has a value of 1 for all fragments except the last one, when it is equal to 0.

The *fragment offset* is used to indicate the position of the first byte of the fragment from the original data packet.

The *Time-to-live* field defines the maximum time for which a packet can be routed over the Internet.

A value in the field *Protocol* refers to a type of protocol that accepts data at the destination. For example, it may be higher level protocol such as TCP or UDP.

Header checksum is calculated only for the data in the header and is used to detect an error in the packet header.

Field *Options* contains some additional information about security of data (if the data are encrypted), the specification of the route (route can be specified as a list of IP addresses of gateways and routers), memorizing the route (storing the IP addresses of devices used to route one packet, so that the same devices could be used for the next packet), etc.

9.7.2 TCP Protocol and Connection-Oriented Service

The TCP protocol organizes data in packets that networks can effectively transmit. In order to provide reliable service, a logical connection is established between two TCP objects that communicate. After the completion of data transfer in both directions, the logical connection is closed. Because of the connection between client and server, this protocol is called the connection-oriented.

The TCP takes care that all data arrive at the proper destination, and then reorganizes the data in the original form. The TCP is reliable service. It detects the erroneous blocks discarded at the destination, and assures that they are resent again from the source. Applications such as file transfer or email require transmission without possible errors and expect to receive data in the same order in which they are sent. Therefore, this type of application requires the reliable service, which is made possible by the TCP protocol. In order to detect errors in transmission, the data are divided into blocks, called *segments*. The rule that defines the maximum size of data segments is called the **M**aximum **S**egment **S**ize (**MSS**), which is typically 536 bytes.

The TCP protocol includes a special control:

– *Flow control*—the sender cannot send data faster than the receiver can receive them,
– *Congestion control*—if it detects congestion in the traffic data, the sender must reduce the speed of sending.

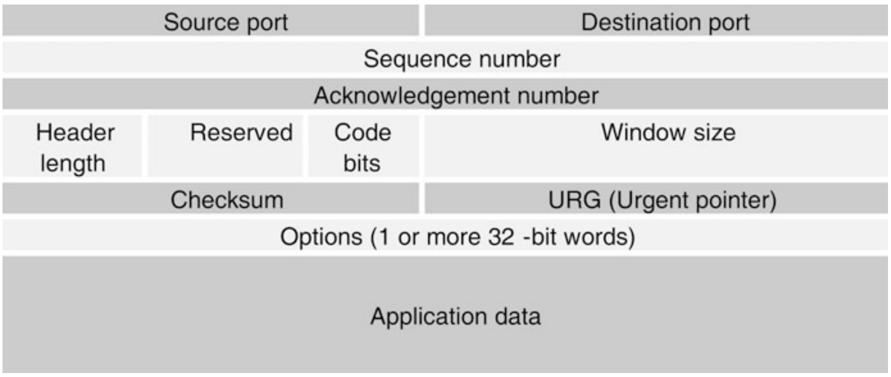


Fig. 9.11 Format of TCP segment

9.7.2.1 Format of TCP Segments

Each segment begins with a 20-bit header. In the case of segments that carry acknowledgment and flow rate control segments, there is only a header (Fig. 9.11).

As illustrated above, each segment, besides the data related to application, contains a header that defines all the necessary parameters such as the source code (16-bit data), the number of segments in a sequence that is sent, a confirmation number for the received segment checksum (to verify the errors in the segment), and so on.

The fields that contain 16 bits for source and 16 bits for destination port along with 32 bits for source and 32 bits for destination address form a 96 bit identifier of the connection. Typically, the port number on client side specifies the client application, and server port number is one of the well-known ports (numbers less than 1024 are called well-known ports), which are reserved for standard services:

Port	Protocol	Description
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial file transfer protocol
80	HTTP	World Wide Web
110	POP-3	Remote email access

The header length indicates the number of 32-bit words in the header. The *Reserved* field consists of bits reserved for a later use. One bit code (a total of 6 code bits) is associated with each of the following fields:

URG: Urgent Pointer field significant
ACK: Acknowledgment field significant
PSH: Push Function
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender

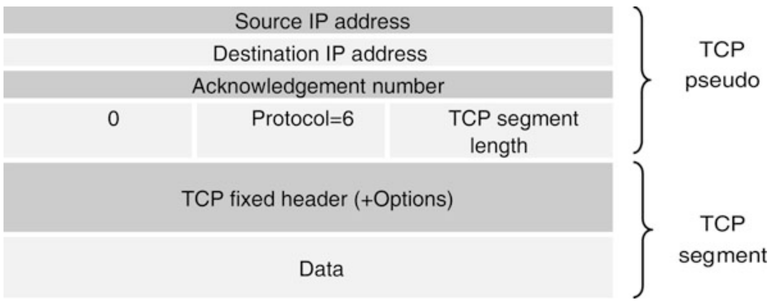


Fig. 9.12 TCP pseudoheader and TCP segment

The URG (urgent) bit is set to 1 when the urgent pointer is used to indicate the position of urgent data (if any) in the segment.

The ACK bit indicates that the segment carries an acknowledgement.

The PUSH bit indicates the data that must be transferred immediately, i.e., we do not wait to fill buffers (important for interactive work, e.g., Telnet)

The RST will reset the connection because of possible problems in communication. It is also used to reject invalid segment or to refuse a connection.

The SYN bit is used to establish a connection. The request for connection is determined by the combination SYN = 1 and ACK = 0, while the response to the request is determined by SYN = 1 and ACK = 1.

The FIN bit is used to terminate a connection.

The checksum field is a 16-bit one's complement, of the one's complement of the total 16-bit words sum in the header and the text. If the segment contains an odd number of octets, which should be considered within the checksum, then the zero octet is added to form a 16-bit word. However, the additional zero octet is not transmitted as a part of the segment.

Checksum also includes a 96-bit pseudoheader, which contains 32-bit source and destination addresses, protocol, and TCP header, Fig. 9.12. It provides correction of errors that appear during the routing process. The value 6 in field *Protocol* indicates the TCP.

9.7.2.2 Establishing a Connection

To establish a TCP connection we use the three-way handshaking procedure (Fig. 9.13):

1. *ACTIVE OPEN* – The client sends a segment with:
   - SYN bit set to 1
   - The client port number
   - Initial sequence number (ISN) of the client

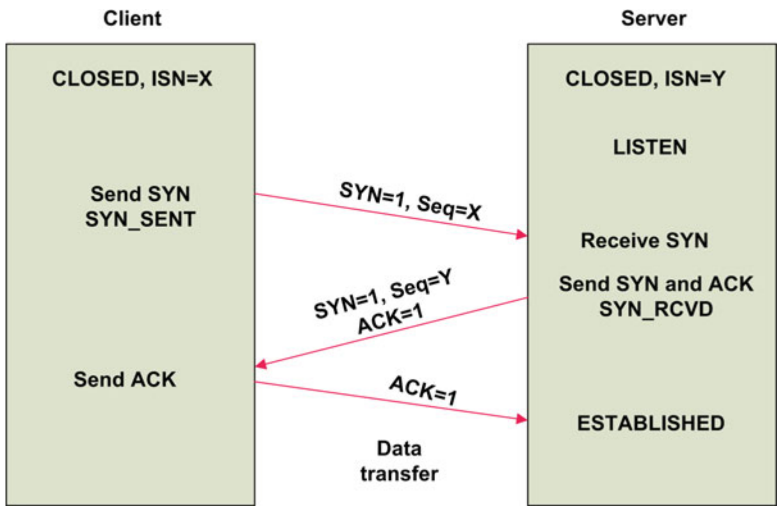


Fig. 9.13 Illustration of procedure for establishing TCP connection

2. *PASSIVE OPEN* – If the server application is in the LISTEN state, the server responds by sending a segment to the client (otherwise the connection is refused):
   – SYN bit set to 1
   – ISN of the server
   – Confirmation (ACK) for the client's ISN
3. *The client sends an acknowledgment (ACK)*

9.7.3 UDP and Connectionless Services

In the case of applications that include sending, receiving, and broadcasting of audio/video in real time (video conferencing, Internet telephony, etc.), resending corrupted data is excessive and unnecessary. Therefore, for these types of applications, the best efforts protocols such as the *UDP* are sufficient.

Some important features of this protocol are:

- UDP segments may be lost or may arrive at the destination in a different order than the order of sending.
- Flow or congestion controls are not provided.

Therefore, it is used only in applications that are not too sensitive to loss of segments but, on the other hand, require a certain speed.

The UDP service is called a connectionless one, because there is no connection established before sending data. This reduces the delay.

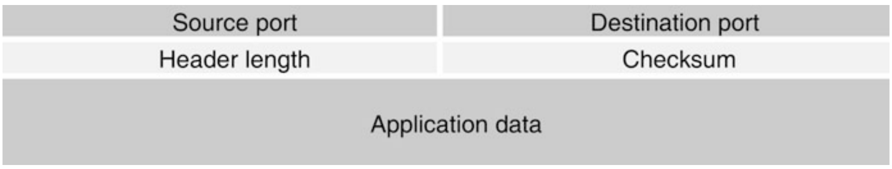


Fig. 9.14 Format of UDP segment

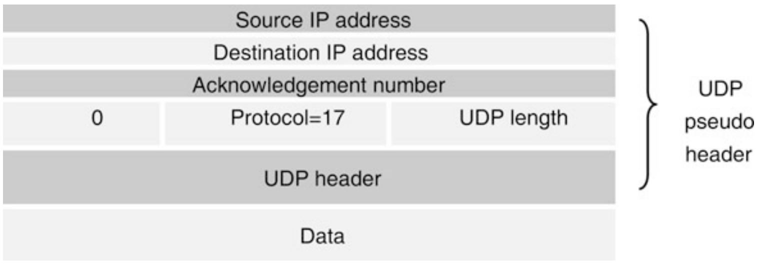


Fig. 9.15 UDP pseudoheader

In the case of UDP each block of data is transmitted directly within the IP datagram. The UDP simply adds a header to form a UDP datagram, which is forwarded to the IP layer for transmission over the Internet. At the destination, the IP protocol uses the information within the field *Protocol* (in the datagram header) to forward the content to the UDP protocol.

A UDP segment header is much simpler than the TCP header due to considerably simpler implementation of the protocol (Fig. 9.14).

As with the TCP protocol, some additional fields in the IP header are included to calculate the UDP checksum. These fields form the UDP pseudoheader. Each protocol has its corresponding number. The value of 17 in the *Protocol* field indicates the UDP protocol (Fig. 9.15).

9.8 Higher-Order Protocols

9.8.1 HTTP Protocol

The Hypertext Transfer Protocol (*HTTP*) is a set of rules that allows a client to connect to the server, to create and send a request for information, and also allows the server to accept connections and send the feedback information. As its name suggests, this protocol provides hypertext transfer between computers. Hypertext is a special kind of text that is encoded by using the Hypertext Markup Language (HTML). So, one of the most important services of the Internet, the *World Wide*

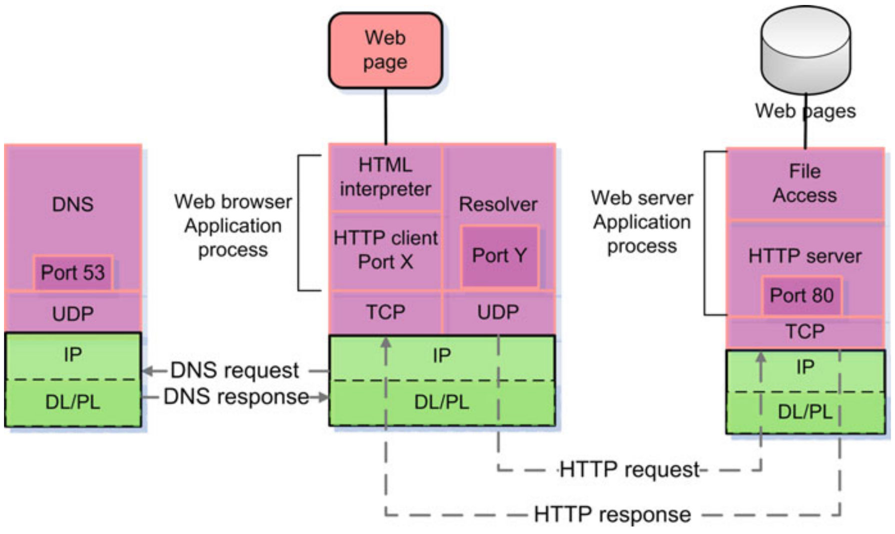


Fig. 9.16 HTTP communication

Web (WWW) is based on the HTTP protocol. A set of documents that can be found on the Web are called Web pages. Web pages can contain different data types: text, images, audio, video, etc. In addition, very often Web pages contain links (connections) to other Web pages. To open a web page, its Web address (or URL) should be entered in the address line of a browser. In a general case, the URL can be expressed as:

protocol://server_name.domain_name.domain:PortNumber/folder/file_name

Example: http://www.tfsa.ac.me/tfsa_members.html

Fig. 9.16 shows the basic operations performed by the HTTP when opening a Web page.

The *Domain Name System (DNS)* is a system based on the databases. Specifically, the DNS translates domain names that are suitable for users to numerical identifiers and IP addresses. For example, the domain name www.tfsa.ac.me is mapped to the address *89.188.43.17* (IPv4). A user specifies an URL of the page or an email address. The *DNS resolver* is responsible for initiating and forwarding queries to translate URLs to IP addresses.

9.8.1.1 Persistent and Nonpersistent HTTP Protocol

There are two types of HTTP protocol: *persistent* and *nonpersistent*. In the case of nonpersistent HTTP, the client first initiates a connection to the server. The server accepts the connection and then sends feedback on the established connection.

After receiving the confirmation of an established connection, the client sends a request for a specific object. When the server receives the request, it creates and sends a response (i.e., the object). At the same time, the server closes the TCP connection and the client receives the object. Thus, for each of the objects, it is necessary to establish a separate TCP connection.

In case of persistent HTTP, the server does not close the connection after sending the response and allows for all subsequent HTTP messages to be exchanged over the same connection.

9.8.1.2 HTTP Format

HTTP requests and responses have specific formats that are reviewed in the sequel.

Request format

A request is formed as a set of ASCII characters. Version HTTP 1.0 request contains the commands GET, POST, and HEAD, while the HTTP 1.1 version also includes the commands PUT and DELETE.

Request line	GET, POST, HEAD methods
Header	Host:
	User-agent:
	Connection:
	Accept:
Empty line – denotes the end of request	

The lines within the header are:

Host – the URL of the requested content,
User-agent – specifies the type of browser (program that allows viewing of Web pages),
Connection – defines the type of connection (if the connection is *close* it means that no persistent connection is required)
Accept – defines the types of objects that can be displayed by the browser.

Example:	***GET***	*/tutorial/index.html HTTP*1.1
	Host:	*www.tfsa.ac.me*
	User – agent:	*InternetExplorer*
	Connection:	*close*
	Accept:	*text/html, image/giff, image/jpeg*

Response format

Response is also made of ASCII characters. The response format is illustrated in the sequel.

Status line	Version HTTP and status code
Header	Date:
	Server:
	Location:
	Last modified:
	Content-Type:
	Content-Length:
Required data	

Beside the HTTP version (HTTP 1.0 or HTTP 1.1), within the status line there is a status code which can be:

200 OK – indicates a successful request,
304 Not Modified – the required page is not modified,
400 Bad Request – the request is not understood by the server,
404 Not found – the requested page was not found on this server.

The fields *Content-Type* and *Content-Length* refer to the type and the length of the content.

Example:	***HTTP/1.1 200 OK***
	Date: *Mon*, 12 *Jan* 2004.
	Server: *Aname*
	Location: *www.w3school.edu*
	LastModified: *Fri*, 16 *Oct* 2005.
	Content – Type:*text/html*
	Content – Length:76234

9.8.2 FTP Protocol

The File Transfer Protocol (*FTP*) is a set of rules that allows transferring files between the computers (clients and servers). Devices used to store the files are called the FTP servers. Computer programs that access these files are called the FTP client programs. Computers that communicate using the FTP protocol may have different operating systems and file systems. A schematic diagram that includes the most important components to transfer files using the FTP protocol is shown in Fig. 9.17.

Note that each FTP client and server consists of two parts: Control part and Data part. The Control part is used for transferring control messages (commands and replies), while the Data transfer part is used to transfer the file content. The users communicate with their local FTP via user interfaces, which converts each user command to the format of FTP control. After receiving the command from the user, the FTP client establishes a connection with the Control part on the server side. This connection is called the control connection and it lasts until the file transfer is finished.

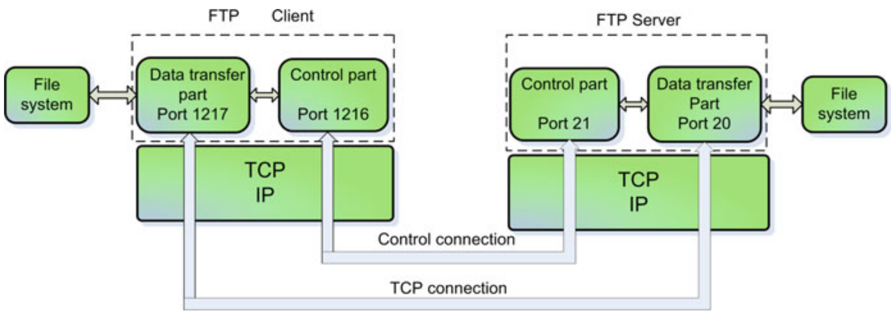


Fig. 9.17 System components for FTP file transfers

The port on the client side (1216 in the example shown in Fig. 9.17) is a currently available port, while the port on the server side is reserved for the FTP control connection. When the server side receives a command from the client side, it sends the reply message by using the control connection.

The second connection is a TCP connection for data transfer and is used to transfer the file content. The control of the client-side server also sends a message about the number of port to be used for data transfer, and the server then establishes a TCP connection for data transfer, using port 20, which is reserved for this type of FTP connection.

An example of an FTP client-server communication

Command	Command description
USER user name	User name on the FTP server
PASS password	Password on the server
TYPE type	File type to be transferred
GET filename.type	Get a file
PUT filename.type	Store a file
LS (or DIR)	List files and directories
QUIT	Log off from the server

Reply 3 digits xyz	Description
1yz	Positive reply. Waiting for another reply before sending a new command
2yz	Positive reply and a new command can be sent
3yz	Positive reply and wait for another command
4yz	Negative reply, try again
5yz	Negative reply, do not try again
x0z	Syntax
x1z	Information
x2z	Control or data connection
x3z	Authentication
x4z	Unspecified
x5z	File status

Any reply message is composed of three digits: the first specifies type of response (positive or negative), the second indicates the nature of the answer, and the third digit provides additional information on error messages. A few typical responses are given below.

220 FTP server is ready
331 Password required
425 Data connections can be established
530 User access is rejected

9.8.3 Other Higher-Order Protocols

The *TELNET* is a protocol that allows the user (computer) to connect to another computer, which may be at a great distance, and to use data and programs from that computer.

The Simple Mail Transfer Protocol (*SMTP*) is a protocol governing email transfers between PCs. The POP3 or IMAP protocols are used for receiving emails.

9.9 HTML

The *HyperText Markup Language* (*HTML)* is a standard language for creating Web pages. All page elements: text, images, graphics, tables, etc. are added to Web pages using commands that are called tags. Tags are written in square brackets <>. HTML documents have the .html extension.

Each HTML page begins with the tag <html> and ends with the tag </html>. <head> marks the beginning of the header, while the page title can be specified by using tag < title>. The end of tag is specified by using "/", for example </head> and </title>.

Consider the following example.

```
<html><head><title> Multimedia signals and systems </title></head>
  <! - - HERE YOU CAN ADD CODE FOR HTML PAGE BODY- - !>
  </html>
```

The title of our web page will be *Multimedia signals and systems*. Note that the page is blank except for the title and still does not contain any other text.

The comment is written within the tag:

```
<!-- HERE YOU CAN ADD CODE FOR HTML PAGE BODY-- !>
```

and it is not visible on the page.

Note that the tag that was opened last, must be closed first. For example, we first end the tag </title>, and then </head>.

To begin creating content on the page, we use the command <body>. For example, let us add text "*Multimedia signal processing and Multimedia systems*."

```
<html><head><title> Multimedia signals and systems </title></head>
<body>
              Multimedia signal processing and Multimedia systems
</body>
</html>
```

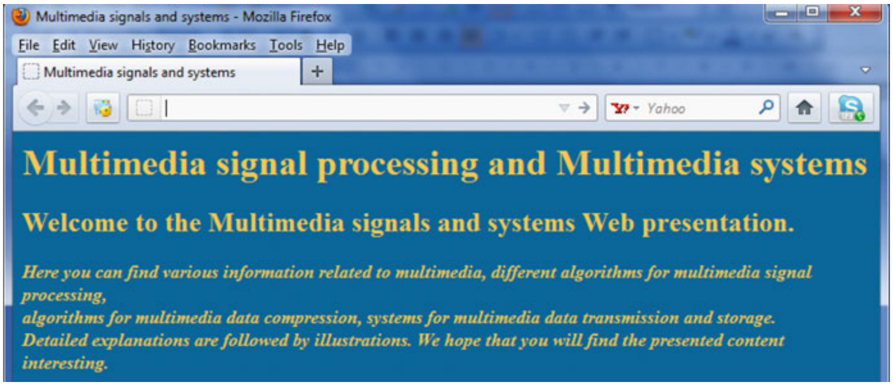
9.9.1 Text Formatting

To format the text, we use the following tags:

<h1> defines the font size for the main title (Heading 1)
<h2> defines the font size for subheading (Heading 2), etc.
<p> is used to start a new paragraph
<b> denotes bold letters
<i> denotes italics
<u> underlines letters
<br> denotes a new line

These tags are used to format the text in the example:

```
<html><head><title>Multimedia signals and systems </title>
</head>
<body>
<h1>Multimedia signal processing and Multimedia systems</h1>
<h2> Welcome to the Multimedia signals and systems Web presentation.</h2>
<p><b><i> Here you can find various information related to multimedia,
different algorithms for multimedia signal processing, <br> algorithms for
multimedia data compression, systems for multimedia data transmission and
storage. <br>
Detailed explanations are followed by illustrations. We hope that you will find
the presented content interesting. </b></i></p>
</body></html>
```


Multimedia signals and systems - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Multimedia signals and systems
Yahoo

Multimedia signal processing and Multimedia systems

Welcome to the Multimedia signals and systems Web presentation.

Here you can find various information related to multimedia, different algorithms for multimedia signal processing,
algorithms for multimedia data compression, systems for multimedia data transmission and storage.
Detailed explanations are followed by illustrations. We hope that you will find the presented content interesting.

Fig. 9.18 Web page after setting the background color and text color

The code is written in the text document, and then the file is saved as *Multimedia signals and systems.html*. To access the code of the Web page (in order to modify or add content on a page), choose the *Source* option from the *View* menu.

9.9.2 Background Color and Text Color

The attributes defining the color of background and text need to be added inside the tag < body>. Otherwise, the color of pages will be white, and the text color will be black. Setting the background color and text is done as follows:

<body bgcolor="Page_Color" text="Text_Color">

The colors are usually given in a hexadecimal form #RRGGBB, where the numbers R, G, and B can have values 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F. RR refers to the red color, GG to the green, and BB to the blue color in the RGB color system.

In addition, the value of FF means that the color is present in full intensity, and 00 is the absence of color. All colors can be obtained from a combination of the three color channels.

To define the background color and text in the given example, the tag <body> should be extended as follows:

```
<html><head><title> Multimedia signals and systems </title></head>
<body bgcolor="#006699" text="#FFC852">
 . . .
</body></html>
```

The page with the specified color is shown in Fig. 9.18.

9.9.3 Adding an Image to HTML Page

Adding an image to a HTML page is possible by using the tag <**img**>. Also, the location from which the image is loaded has to be specified.

Tag for image embedding:
<img src=”IMAGE LOCATION/Image_name.format”>

If we want to add an image at the top of previously designed page, we need to include a line of code:

```
<img src="C:/Folder/multimedia.jpg" width="800" height="150">
```

In addition, the image height and width are set by using the *width* and *height* attributes.

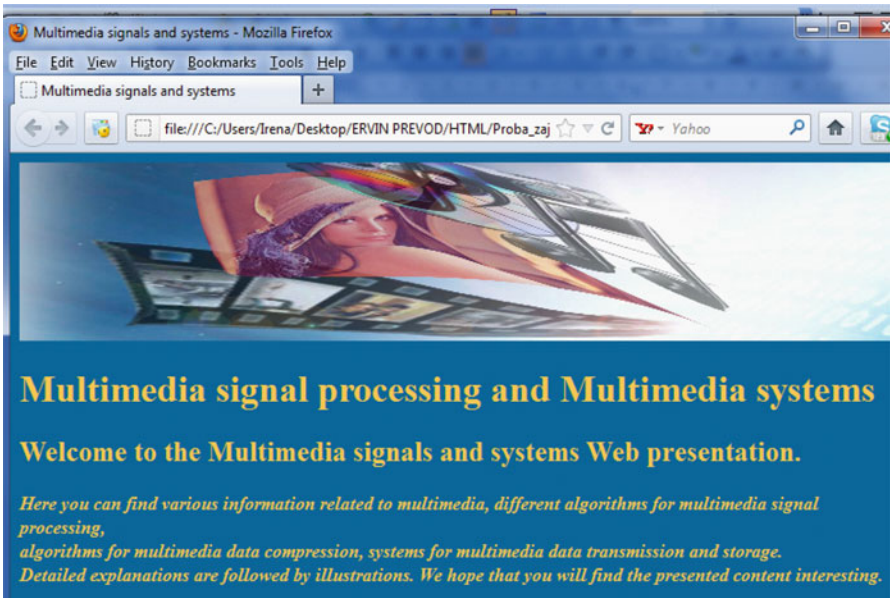
For example, width = 800 height = 150, means that the image width is set to 800 pixels and its height is set to 150 pixels. The images, in our example, are located in some folder “*Folder*” on drive C, so the path is: C:/Folder/

```
<html><head><title>Multimedia signals and systems</title></head>
<body bgcolor="#006699" text="#FFC852" link="#FFFFFF">
<img src="C:/Folder/multimedia.jpg" width="800" height="150">
<h1>Multimedia signal processing and Multimedia systems</h1>
<h2> Welcome to the Multimedia signals and systems Web presentation.</h2>
<p><b><i> Here you can find various information related to multimedia,
different algorithms for multimedia signal processing,<br> algorithms for
multimedia data compression, systems for multimedia data transmission and
storage. <br>
Detailed explanations are followed by illustrations. We hope that you will find
the presented content interesting. </b></i></p>
</body></html>
```

The page is shown in Fig. 9.19.

9.9.4 Unordered and Ordered Lists

If there is a need that the content of a web page is displayed as a list of items, then we can use the unordered (items are marked with bullets) or ordered (items are marked with numbers) lists. The tag that marks the beginning of the unordered list is <**ul**>, while the tag for an ordered list is <**ol**>. Each element of the list is indicated by the tag <**li**>.


Multimedia signals and systems - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Multimedia signals and systems
file:///C:/Users/Irena/Desktop/ERVIN PREVOD/HTML/Proba_zaj
Yahoo

Multimedia signal processing and Multimedia systems

Welcome to the Multimedia signals and systems Web presentation.

Here you can find various information related to multimedia, different algorithms for multimedia signal processing,
algorithms for multimedia data compression, systems for multimedia data transmission and storage.
Detailed explanations are followed by illustrations. We hope that you will find the presented content interesting.

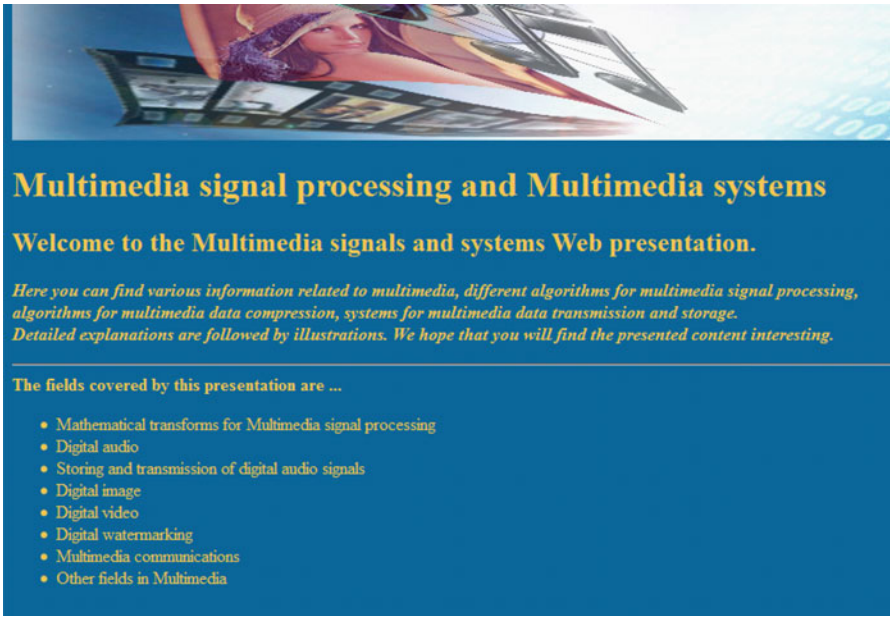
Fig. 9.19 A web page with image included

Suppose that we would like to add a list of a few members. Therefore, we add new elements (marked in red) to the existing code.

Note that before the unordered list, we added a horizontal line by using tag <hr>. The Web page is illustrated in Fig. 9.20.

```
<html><head><title>Multimedia signals and systems</title></head>
<body bgcolor="#006699" text="#FFC852">
<img src="C:/Folder/multimedia.jpg" width="800" height="150">
<h1>Multimedia signal processing and Multimedia systems</h1>
<h2> Welcome to the Multimedia signals and systems Web presentation.</h2>
<p><b><i> Here you can find various information related to multimedia,
different algorithms for multimedia signal processing,<br> algorithms for
multimedia data compression, systems for multimedia data transmission and
storage. <br>
Detailed explanations are followed by illustrations. We hope that you will find
the presented content interesting. </b></i></p>
<b>The fields covered by this presentation are ...</b>
<ul>
<li>Mathematical transforms for Multimedia signal processing</li>
<li>Digital audio</li>
<li>Storing and transmission of digital audio signals</li>
<li>Digital image</li>
<li>Digital video</li>
```

(continued)



Multimedia signal processing and Multimedia systems

Welcome to the Multimedia signals and systems Web presentation.

Here you can find various information related to multimedia, different algorithms for multimedia signal processing, algorithms for multimedia data compression, systems for multimedia data transmission and storage.
Detailed explanations are followed by illustrations. We hope that you will find the presented content interesting.

The fields covered by this presentation are ...

- Mathematical transforms for Multimedia signal processing
- Digital audio
- Storing and transmission of digital audio signals
- Digital image
- Digital video
- Digital watermarking
- Multimedia communications
- Other fields in Multimedia

Fig. 9.20 Unordered list is added to the web page

```
<li>Digital watermarking</li>
<li>Multimedia communications</li>
<li>Other fields in Multimedia</li></ul>
</body></html>
```

9.9.5 Links to Other Websites

Link creates a connection to another page. It is usually placed on a single word or line of text. By clicking on the link, the Web browser opens the page that link refers to.

To create a link we use the tag:
<a href=”http://www.page_name”> Text </a>

We can change the color of the displayed link by setting the following attributes within the <body> tag:

Color of the link: link = "#ffffff" (for example)
Color of the visited link: vlink = "#ffffff"
Color of the active link: alink = "#ff0000"

9.9.6 Setting an Anchor Within a Web Page

A specific kind of link is used to create an anchor or bookmark inside the document.

The link to an anchor can be created as:
<a href="#**Anchor_name**"> Text that denotes link</a>

Symbol # gives an instruction to the Web browser to look for an anchor with the specified name. In other words, when a user activates the anchor link, a part of the page containing the anchor name will be focused.

Anchor name is set as:
<a name="**Anchor_name**"> Any text </a>

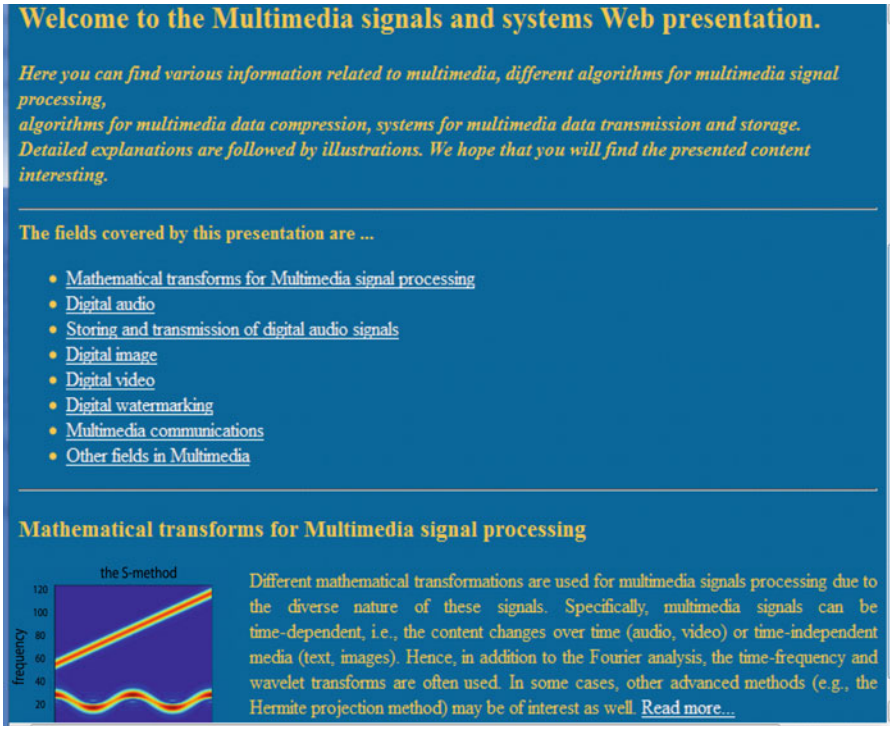
Hence, in one part of the page we set the anchor link and in the other part we set the anchor.

Every anchor link <a href="Anchor_name"> should refer to the corresponding anchor name <a name="Anchor_name">

The use of anchors and links will be illustrated by modifying our web page. The new lines of code are marked in red. The appearance of added elements is illustrated in Fig. 9.21.

```
<html><head><title>Multimedia signals and systems</title></head>
<body bgcolor="#006699" text="#FFC852" link="#ffffff" vlink="#ffffff"
alink="#ff0000">
<img src="C:/Folder/multimedia.jpg" width="800" height="150">
<h1>Multimedia signal processing and Multimedia systems</h1>
<h2> Welcome to the Multimedia signals and systems Web presentation.</
h2>
<p><b><i> Here you can find various information related to multimedia,
different algorithms for multimedia signal processing,<br> algorithms for
multimedia data compression, systems for multimedia data transmission and
storage. <br>
Detailed explanations are followed by illustrations. We hope that you will find
the presented content interesting. </b></i></p>
<b>The fields covered by this presentation are . . .</b>
<ul><li><a href="#Mathematical transforms">Mathematical transforms for
Multimedia signal processing</a></li>
<li><a href="#Digital audio">Digital audio</a></li>
<li><a href="#Storing and transmission">Storing and transmission of digital
audio signals</a></li>
```

(continued)


Welcome to the Multimedia signals and systems Web presentation.

Here you can find various information related to multimedia, different algorithms for multimedia signal processing,
algorithms for multimedia data compression, systems for multimedia data transmission and storage.
Detailed explanations are followed by illustrations. We hope that you will find the presented content interesting.

The fields covered by this presentation are ...

- Mathematical transforms for Multimedia signal processing
- Digital audio
- Storing and transmission of digital audio signals
- Digital image
- Digital video
- Digital watermarking
- Multimedia communications
- Other fields in Multimedia

Mathematical transforms for Multimedia signal processing

Different mathematical transformations are used for multimedia signals processing due to the diverse nature of these signals. Specifically, multimedia signals can be time-dependent, i.e., the content changes over time (audio, video) or time-independent media (text, images). Hence, in addition to the Fourier analysis, the time-frequency and wavelet transforms are often used. In some cases, other advanced methods (e.g., the Hermite projection method) may be of interest as well. Read more...

Fig. 9.21 Web page after adding anchors to text paragraphs

```
<li><a href="#Digital image">Digital image</a></li>
<li><a href="#Digital video">Digital video</a></li>
<li><a href="#Digital watermarking">Digital watermarking</a>
</li>
<li><a href="#Multimedia communications">Multimedia communications</
a></li>
<li><a href="#Other fields in Multimedia">Other fields in Multimedia</
a></li></ul>
<hr>
<h3><a name="Mathematical transforms">Mathematical transforms for
Multimedia signal processing</a></h3>
<img style="float: left; margin-right: 15px;" src="C:/Folder/SM.jpg"
width="180" height="150">
<p align=justify>Different mathematical transformations are used for multi-
media signals processing due to the diverse nature of these signals. Specifically,
multimedia signals can be time-dependent, i.e., the content changes over time
(audio, video) or time-independent media (text, images). Hence, in addition to
the Fourier analysis, the time-frequency and wavelet transforms are often used.
```

(continued)

```
In some cases, other advanced methods (e.g., the Hermite projection method)
may be of interest as well.

 <a href="http://Mathematical_transforms.html">Readmore...</a>
 </p>
 </body></html>
```

As an example, let us observe the anchor link:

<a href = "#Mathematical transforms"> Mathematical transforms for Multimedia signal processing</a>

It looks for an anchor name, i.e., the string “Mathematical transform” and focuses the page view to its location.

The same holds for the links:

<a href = "#Digital audio"> Digital audio</a>
<a href = "#Storing and transmission"> Storing and transmission of digital audio signals</a>
<a href = "#Digital image"> Digital image</a>
<a href = "#Digital video">Digital video</a></li>
<a href = "#Digital watermarking"> Digital watermarking</a>
<a href = "#Multimedia communications"> Multimedia communications</a>
<a href = "#Other fields in Multimedia"> Other fields in Multimedia</a>

Now we should set the names for the anchors. For instance, the name for the first anchor is set as follows:

<a name = "Mathematical transforms" > Mathematical transforms for Multimedia signal processing</a>

Together with the anchor links and names, we added an image, text paragraph (the attribute align is set to justify), and finally a standard link to another html page (Mathematical_transform.html):

<a href = "http://Mathematical_transforms.html">Readmore...</a>

Placing images in the same line with text is achieved by defining the following attributes in <img>:

<img style = "FLOAT: margin-right:15px;" src = "name.format" width = 180 height = 150>

The *Float* option provides a flexibility to place the image anywhere with respect to the text (not just above or below) while the *left* means that the image is placed on the left. The option margin-right: 15px; means that the image is shifted by 15 pixels to the right from the edge of the page.

Note that each anchor should provide a link to a subsection on the web page. Hence, we should add other subsections as we did for the subsection "*Mathematical transforms for Multimedia signal processing*."

Link to an email

In addition to linking a web page, it is possible to link an email address. By clicking on the text that indicates the link, it is possible to open an email client (e.g., Outlook Express). In the field **To:** we should enter email address, while in the field **Subject**, we enter the message subject.

Link to an email address:
<a href="mailto:user_email@domain?subject=Hello">
Text</a>

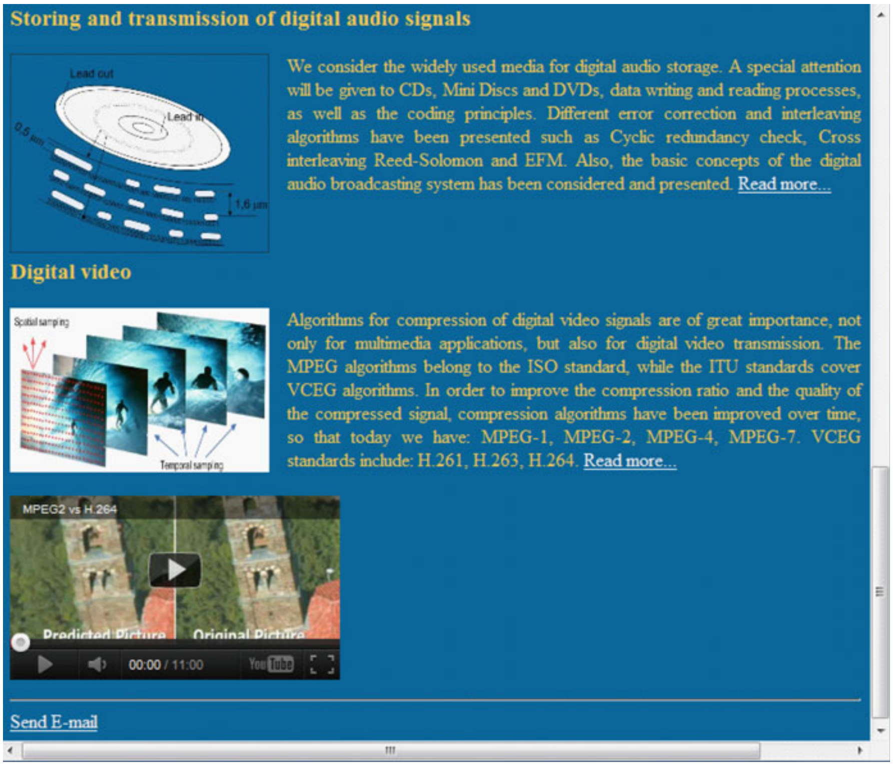
The corresponding commands for linking to an email address are indicated in red in the code below. Hence, we set the link (as text "Send Email") to the email address: WebT@gmail.com, while the title of the message is "Hello."

```
<html><head><title>Multimedia signals and systems </title></head>
<body bgcolor="#006699" text="#FFC852" link="#ffffff" vlink="#ffffff"
alink="#ff0000">
   . . .
 <hr>
 <a href="mailto:WebT@gmail.com?subject=Hello">
 Send E-mail</a><br>
 </body></html>
```

By activating the link "Send Email" at the bottom of the page (Fig. 9.22), we open an email client window as in Fig. 9.23.

9.9.7 Adding Video Content to a Web Page

Video content can be embedded within the web page. For example, we can add a video file from the existing websites (e.g., www.youtube.com). When we select a video file on the mentioned website, there will be the option *Share* just below the video playing window. By clicking on *Embed*, the code appears as shown in Fig. 9.24.



Storing and transmission of digital audio signals

We consider the widely used media for digital audio storage. A special attention will be given to CDs, Mini Discs and DVDs, data writing and reading processes, as well as the coding principles. Different error correction and interleaving algorithms have been presented such as Cyclic redundancy check, Cross interleaving Reed-Solomon and EFM. Also, the basic concepts of the digital audio broadcasting system has been considered and presented. Read more...

Digital video

Algorithms for compression of digital video signals are of great importance, not only for multimedia applications, but also for digital video transmission. The MPEG algorithms belong to the ISO standard, while the ITU standards cover VCEG algorithms. In order to improve the compression ratio and the quality of the compressed signal, compression algorithms have been improved over time, so that today we have: MPEG-1, MPEG-2, MPEG-4, MPEG-7. VCEG standards include: H.261, H.263, H.264. Read more...

Send E-mail

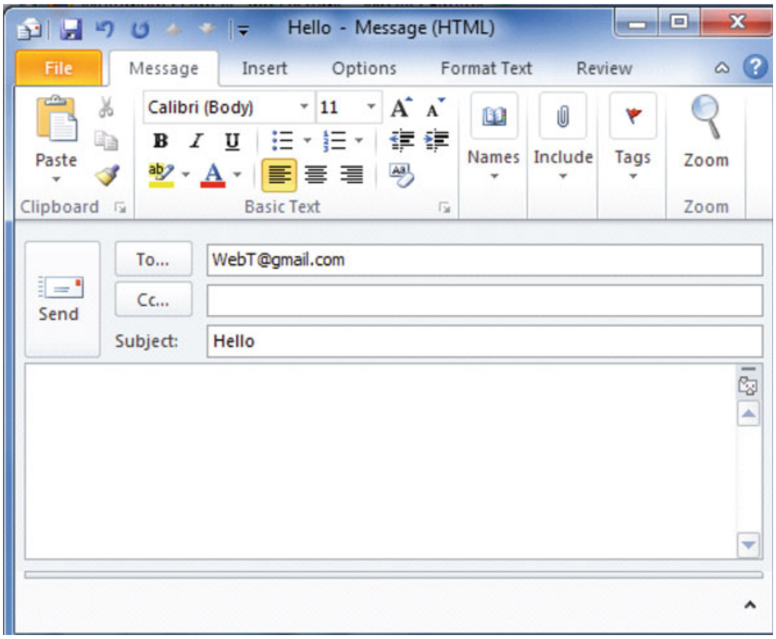
Fig. 9.22 The layout of a created web page

```
<iframe width="280" height="157"
 src="http://www.youtube.com/embed/Uyxy9DCrq1U"          frameborder="0"
 allowfullscreen></iframe>
```

Among different options, we can choose the color and size of the video playing frame. Then, it is necessary to copy the code and paste it in the desired location within the HTML page, as we did at the end of the web page (Fig. 9.25).

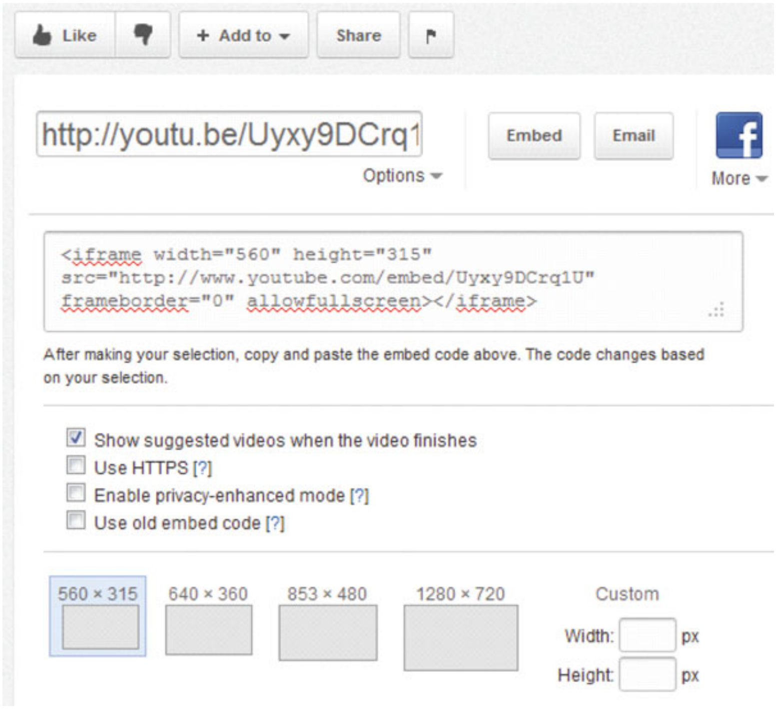
9.9.8 Creating and Formatting Tables

Tables are created using the tag <table>. A row is created by the command <tr>, while <td> is used for each cell in the row. The table header with column names is set by using the tag <th>.



To...: WebT@gmail.com
Subject: Hello

Fig. 9.23 Message window in Microsoft Outlook



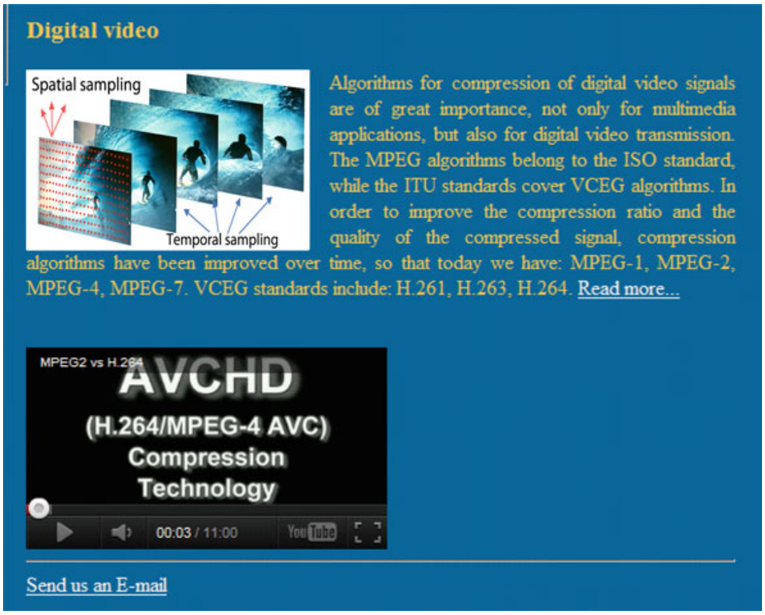
http://youtu.be/Uyxy9DCrq1

```
<iframe width="560" height="315"
src="http://www.youtube.com/embed/Uyxy9DCrq1U"
frameborder="0" allowfullscreen></iframe>
```

After making your selection, copy and paste the embed code above. The code changes based on your selection.

- [✓] Show suggested videos when the video finishes
- [ ] Use HTTPS [?]
- [ ] Enable privacy-enhanced mode [?]
- [ ] Use old embed code [?]

Fig. 9.24 The window with the code to embed video



Digital video

Algorithms for compression of digital video signals are of great importance, not only for multimedia applications, but also for digital video transmission. The MPEG algorithms belong to the ISO standard, while the ITU standards cover VCEG algorithms. In order to improve the compression ratio and the quality of the compressed signal, compression algorithms have been improved over time, so that today we have: MPEG-1, MPEG-2, MPEG-4, MPEG-7. VCEG standards include: H.261, H.263, H.264. Read more...

Send us an E-mail

Fig. 9.25 Embedded video on the Web page

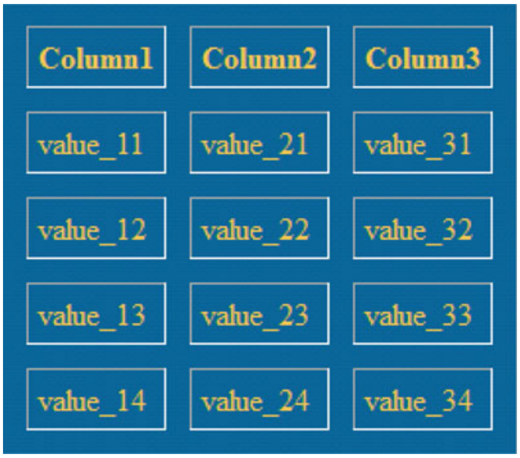
An example of the table is shown in Fig. 9.26, and the code is given below.

```
<html><head><title>Example</title></head>
<body text="#FFC852">
   <table    width=”400”    height=”250”    border=3    cellspacing="12"
cellpadding="5" bgcolor="006699" bordercolor="white">
<tr><th>Column1</th><th>Column2</th><th>Column3</th></tr>
<tr><td>value_11 </td><td>value_21</td><td>value_31</td></tr>
<tr><td>value_12</td><td>value_22</td><td>value_32</td></tr>
<tr><td>value_13</td><td>value_23</td><td>value_33</td></tr>
<tr><td>value_14</td><td>value_24</td><td>value_34</td></tr>
</table>
</body></html>
```

Inside the tag <table> we add attributes to format table cells (boundary lines width, line colors, background colors of cells, spaces between the cells, etc.).

In our example, the attributes have the following values:

width = "400" – the table width is 400 pixels
height = "250" – the height table is 250 pixels
border = "3" – the width of the boundary lines is 3



Column1	Column2	Column3
value_11	value_21	value_31
value_12	value_22	value_32
value_13	value_23	value_33
value_14	value_24	value_34

Fig. 9.26 An example of the table

bgcolor = **"006699" –** the background color is set to dark blue
bordercolor = "white" – the color of cell borders
cellspacing = "12" – the distance between the cells is 12
cellpadding = "5" – the text distance from the edge of the cell is 5

In the first row of the table, we have the header cells:

<tr><th>Column1</th><th>Column2</th><th>Column3</th></tr>

Other table rows are filled by the values:

<tr><td>value_11 </td><td>value_21</td><td>value_31</td></tr>
<tr><td>value_12</td><td>value_22</td><td>value_32</td></tr>
<tr><td>value_13</td><td>value_23</td><td>value_33</td></tr>
<tr><td>value_14</td><td>value_24</td><td>value_34</td></tr>

Tables could be used to efficiently organize the content of the page. For instance, the text paragraphs and images on the Web page could be organized within the table cells.

9.9.9 Forms for Data Entry

Data entry forms can be created in HTML. However, to send data to a server, the HTML code can be used in combination with the Hypertext Preprocessor (PHP) code.

Form is indicated by : <form> and </form>

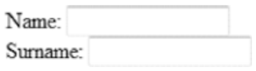
Some of the most commonly used form elements are discussed in the sequel. For the sake of simplicity, we first use some very basic examples and at the end we create the entire form for the considered Multimedia signals and systems web page.

Text field

<input type = "text" > defines the field as a single line in which the user enters text.

Example:

```
<form>
Name: <input type="text" name="NAME" size=15> <br>
Surname: <input type="text" name="SNAME" size=15>
</form>
```



Name: and **Surname:** are plain text.
name = "NAME" specifies name of the location on the server where the name data will be stored.
name = "SNAME" specifies name of the location on the server where the surname data will be placed.
size defines the number of characters that can be entered.

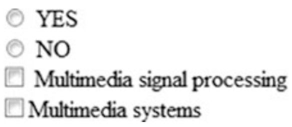
Radio Button and Checkbox

The fields that are used to select an option are known as the radio button and checkbox. They are defined with the following tags:

<input type = "radio">
<input type = "checkbox">

Example:

```
<form>
<input type="radio" name="answer" value="Yes" > YES <br>
<input type="radio" name="answer" value="No" > NO <br>
<input type="checkbox" name="Multimedia" value="MSP"> Multimedia
signal processing<br><input type="checkbox" name="Multimedia"
value="MS">Multimedia systems
</form>
```



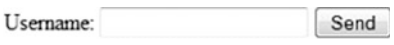
Note that the **name** field has the same name for both “radio” options (for **Yes** and **No** options). The field **value** must have different values. The same holds for the checkbox.

Setting options for sending data

The option to send data can be achieved by using the Submit button. The command <input type = "submit"> defines the data sent to the server. In this case, the data are sent to the page specified in the **action** attribute.

Example:

```
<form name="input" action="html_form.asp" method="post">
Username: <input type="text" name="user">
<input type="submit" value="Send">
</form>
```



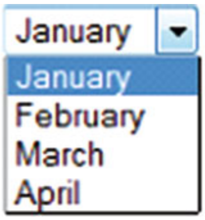
Parameter *method* can be "get" or "post"

In our example, the entered data should be sent to a page titled "html_form.asp".

Drop-down menu

A drop-down menu is created using the tag < select>. The elements are specified as:
<option value = " . . . "> . . . </option>

```
<select>
   <option value="Jan">January</option>
   <option value="Feb">February</option>
   <option value=Mar">March</option>
   <option value="Apr">April</option>
 </select>
```



Now let us create a simple form for log-in, which will be later included in the web page for Multimedia signals and systems (Fig. 9.27).

```
<html><head>
<title>Log in</title></head>
<body bgcolor="006699" text="#ffc852">

<H4 style="color:#FFFFFF">log in to download codes</H4>
<form action="www.ac.me" method=POST>
Provide the following info:
```

(continued)

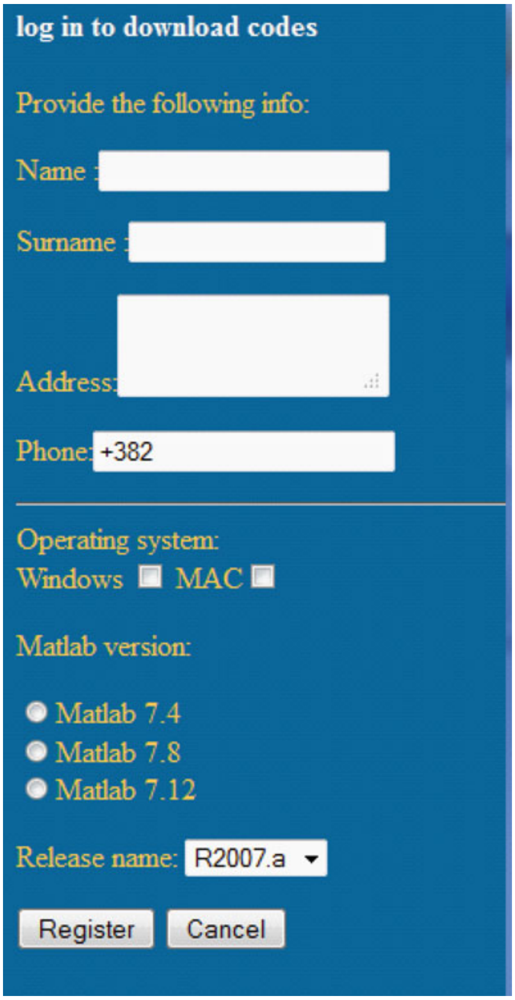


Fig. 9.27 A form for data entry

log in to download codes

Provide the following info:

Name :

Surname :

Address:

Phone: +382

Operating system:
Windows ☐ MAC ☐

Matlab version:

○ Matlab 7.4
○ Matlab 7.8
○ Matlab 7.12

Release name: R2007.a

```
<P>Name :<input name="user_name" size=21><P>
<P>Surname :<input name="user_surname" size=18><P>

Address:<textarea name="address" cols=15 rows=2></textarea><P>
Phone:<input name="phone" size=22 value="+382"><P>
<hr>
Operating system:<br>
Windows <input type=checkbox name="OS" value="win">
MAC<input type=checkbox name="OS" value="mac">
<P>
Matlab version:<P>
```

(continued)

```
<input type=radio name="matlab" value="R2007">Matlab 7.4<br>
<input type=radio name="matlab" value="R2009">Matlab 7.8<br>
<input type=radio name="matlab" value="R2011">Matlab 7.12<P>
Release name:
<select>
   <option value="R2007.a">R2007.a</option>
   <option value="R2009.a">R2009.a</option>
   <option value="R2011.b">R2010.b</option>
</select>
<P>
<input type=submit value="Register">
<input type=reset value="Cancel">
</form>
</body></html>
```

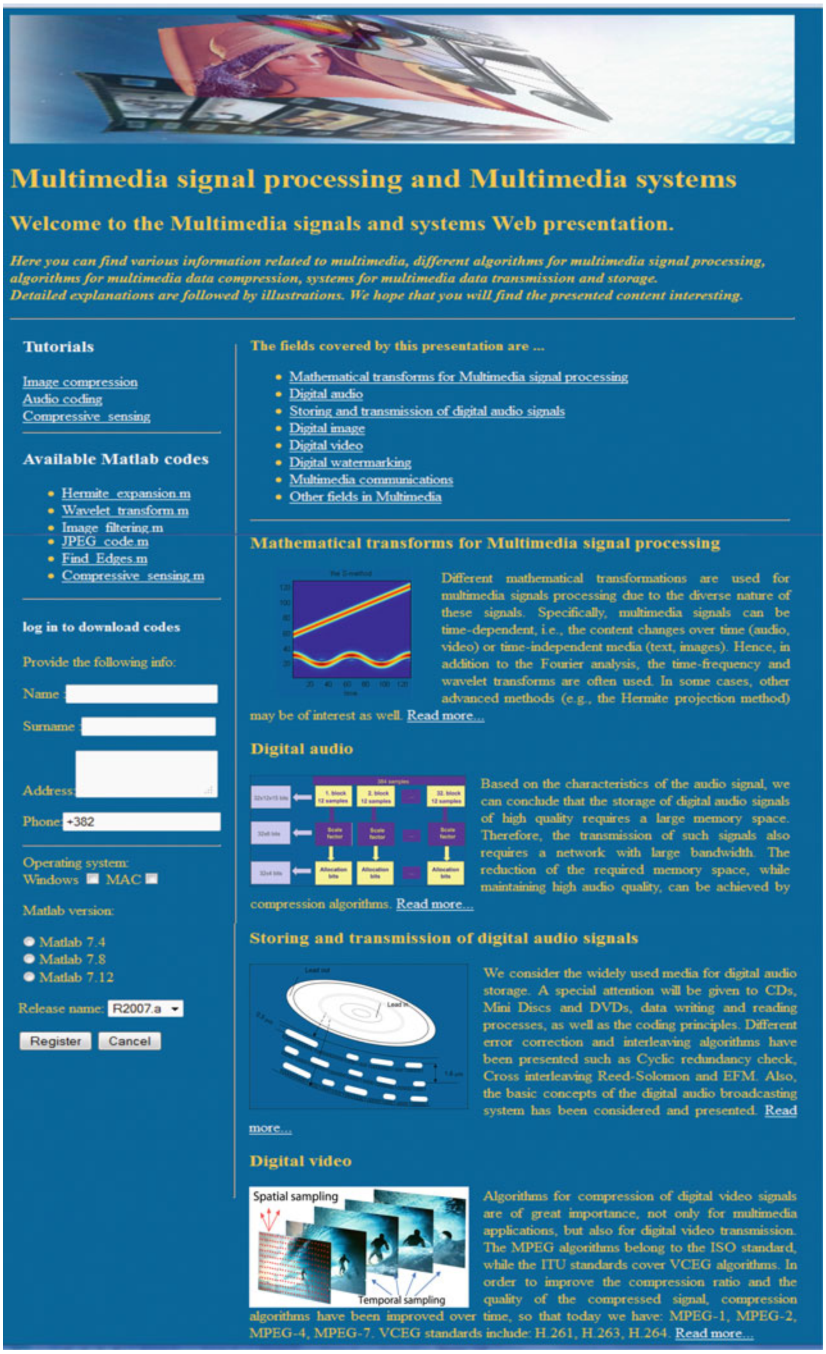
In the sequel, we integrate all the commands discussed so far, in order to get a complex page that contains all of the analyzed options. The appearance of the central part of the page is shown in Fig. 9.28.

Code for the page

```
<html><head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-
1"><title>Multimedia signals and systems</title></head><body bgcolor=
"006699" text="#ffc852" link="#ffffff" vlink="#ffffff" alink="#ff0000">
<img src=" C:/Folder/multimedia.jpg" width="800" height="150">

<h1>Multimedia signal processing and Multimedia systems</h1>
<h2>Welcome to the Multimedia signals and systems Web presentation.</h2>
<p><b><i>Here you can find various information related to multimedia, differ-
ent algorithms for multimedia signal processing,<br>
algorithms for multimedia data compression, systems for multimedia data trans-
mission and storage. <br>
Detailed explanations are followed by illustrations. We hope that you will find the
presented content interesting.
</i></b><i></i></p>
<hr align=left width=800>
<table align=left cellpading=5 cellspacing=12><tr><td VALIGN="top">

<H3 style="color:#FFFFFF">Tutorials</H4>
<a href="Hermite_expansion.pdf">Image compression</a><br>
<a href="Digital_audio.pdf">Audio coding</a><br>
<a href="Compressive_sensing.pdf">Compressive_sensing
</a><br>
```



Multimedia signal processing and Multimedia systems

Welcome to the Multimedia signals and systems Web presentation.

Here you can find various information related to multimedia, different algorithms for multimedia signal processing, algorithms for multimedia data compression, systems for multimedia data transmission and storage.
Detailed explanations are followed by illustrations. We hope that you will find the presented content interesting.

Tutorials

Image compression
Audio coding
Compressive sensing

Available Matlab codes

- Hermite_expansion.m
- Wavelet_transform.m
- Image_filtering.m
- JPEG_code.m
- Find_Edges.m
- Compressive_sensing.m

log in to download codes

Provide the following info:

Name:
Surname:
Address:
Phone: +382

Operating system:
Windows MAC

Matlab version:

- Matlab 7.4
- Matlab 7.8
- Matlab 7.12

Release name: R2007.a

Register Cancel

The fields covered by this presentation are ...

- Mathematical transforms for Multimedia signal processing
- Digital audio
- Storing and transmission of digital audio signals
- Digital image
- Digital video
- Digital watermarking
- Multimedia communications
- Other fields in Multimedia

Mathematical transforms for Multimedia signal processing

Different mathematical transformations are used for multimedia signals processing due to the diverse nature of these signals. Specifically, multimedia signals can be time-dependent, i.e., the content changes over time (audio, video) or time-independent media (text, images). Hence, in addition to the Fourier analysis, the time-frequency and wavelet transforms are often used. In some cases, other advanced methods (e.g., the Hermite projection method) may be of interest as well. Read more...

Digital audio

Based on the characteristics of the audio signal, we can conclude that the storage of digital audio signals of high quality requires a large memory space. Therefore, the transmission of such signals also requires a network with large bandwidth. The reduction of the required memory space, while maintaining high audio quality, can be achieved by compression algorithms. Read more...

Storing and transmission of digital audio signals

We consider the widely used media for digital audio storage. A special attention will be given to CDs, Mini Discs and DVDs, data writing and reading processes, as well as the coding principles. Different error correction and interleaving algorithms have been presented such as Cyclic redundancy check, Cross interleaving Reed-Solomon and EFM. Also, the basic concepts of the digital audio broadcasting system has been considered and presented. Read more...

Digital video

Algorithms for compression of digital video signals are of great importance, not only for multimedia applications, but also for digital video transmission. The MPEG algorithms belong to the ISO standard, while the ITU standards cover VCEG algorithms. In order to improve the compression ratio and the quality of the compressed signal, compression algorithms have been improved over time, so that today we have: MPEG-1, MPEG-2, MPEG-4, MPEG-7. VCEG standards include: H.261, H.263, H.264. Read more...

Fig. 9.28 Final Web page design

```
<!DOCUMENTS SHOULD BE IN THE SAME FOLDER AS THE WEB
PAGE. OTHERWISE THE ENTIRE PATH TO THE DOCUMENT
SHOULD BE SPECIFIED!>

<hr>
<H3 style="color:#FFFFFF">Available Matlab codes</H3>
<ul>

<li><a href="Hermite_expansion.m">Hermite_expansion.m</a>
</li><li><a href="Wavelets.m">Wavelet_transform.m</a>
</li><li><a href="Imafilt.m">Image_filtering.m</a>
</li><li><a href="jpeg.m">JPEG_code.m</a>
</li><li><a href="findedges.m">Find_Edges.m</a>
</li><li><a href="compressive_sens.m">Compressive_sensing.m</a>
</li></ul>
<!M-FILES SHOULD BE IN THE SAME FOLDER AS THE WEB PAGE!>

<hr>
<H4 style="color:#FFFFFF">log in to download codes</H4>
<form action="www.ac.me" method=POST>
Provide the following info:
<P>Name :<input name="user_name" size=21><P>
<P>Surname :<input name="user_surname" size=18><P>

Address:<textarea name="address" cols=15 rows=2></textarea><P>
Phone:<input name="phone" size=22 value="+382"><P>
<hr>
Operating system:<br>
Windows <input type=checkbox name="OS" value="win">
MAC<input type=checkbox name="OS" value="mac">
<P> Matlab version:<P>
<input type=radio name="matlab" value="R2007">Matlab 7.4<br>
<input type=radio name="matlab" value="R2009">Matlab 7.8<br>
<input type=radio name="matlab" value="R2011">Matlab 7.12<P>
Release name:
<select>
<option value="R2007.a">R2007.a</option>
<option value="R2009.a">R2009.a</option>
<option value="R2011.b">R2010.b</option>
</select>
<P>
<input type=submit value="Register">
<input type=reset value="Cancel">
</form></td>
<td VALIGN="top"><hr width=1, size=2500></td>
```

```
<td VALIGN="top" width=550>
<b>The fields covered by this presentation are ...</b>
<ul>
<li><a href="#Mathematical transforms">
Mathematical transforms for Multimedia signal processing</a></li>
<li><a href="#Digital audio">Digital audio</a></li>
<li><a href="#Storing and transmission">Storing and transmission of digital
audio signals</a></li>
<li><a href="#Digital image">Digital image</a></li>
<li><a href="#Digital video">Digital video</a></li>
<li><a href="#Digital watermarking">Digital watermarking</a></li>
<li><a href="#Multimedia communications">Multimedia communications
</a></li>
<li><a href="#Other fields in Multimedia">Other fields in Multimedia
</a></li></ul>

<hr><h3><a name="Mathematical transforms">Mathematical transforms for
Multimedia signal processing</a></h3>
<img style="float: left; margin-right: 15px;" src="C:/Folder/SM.jpg"
width="180" height="150">
<p align=justify>Different mathematical transformations are used for multimedia
signals processing due to the diverse nature of these signals. Specifically, multime-
dia signals can be time-dependent, i.e., the content changes over time (audio, video)
or time-independent media (text, images). Hence, in addition to the Fourier
analysis, the time-frequency and wavelet transforms are often used. In some
cases, other advanced methods (e.g., the Hermite projection method) may be of
interest as well.

<a href="Mathematical_transforms.html">Read more... </a> </p>
<h3><a name="Digital audio">Digital audio</a></h3>
<img style="float: left; margin-right: 15px;" src="C:/Folder/mpeg1.jpg"
width="220" height="130">
<p align=justify> Based on the characteristics of the audio signal, we can con-
clude that the storage of digital audio signals of high quality requires a large
memory space. Therefore, the transmission of such signals also requires a network
with large bandwidth. The reduction of the required memory space, while
maintaining high audio quality, can be achieved by compression algorithms.

<a href="Digital_audio.html">Read more... </a> </p>
<h3><a name="Storing and transmission">Storing and transmission of digital
audio signals</a></h3>
<img style="float: left; margin-right: 15px;" src="C:/Folder/CD1.jpg"
width="220" height="170">
```

```
<p align=justify> We consider the widely used media for digital audio storage.
A special attention will be given to CDs, Mini Discs and DVDs, data writing and
reading processes, as well as the coding principles. Different error correction and
interleaving algorithms have been presented such as Cyclic redundancy check,
Cross interleaving Reed-Solomon and EFM. Also, the basic concepts of the digital
audio broadcasting system has been considered and presented.

<a href="Storing_and_transmission.html">Read more… </a> </p>
<h3><a name="Digital video">Digital video</a></h3>
<img style="float: left; margin-right: 15px;" src="C:/Folder/video.jpg"
width="220" height="140">
<p align=justify> Algorithms for compression of digital video signals are of great
importance, not only for multimedia applications, but also for digital video trans-
mission. The MPEG algorithms belong to the ISO standard, while the ITU
standards cover VCEG algorithms. In order to improve the compression ratio and
the quality of the compressed signal, compression algorithms have been improved
over time, so that today we have: MPEG-1, MPEG-2, MPEG-4, MPEG-7. VCEG
standards include: H.261, H.263, H.264.
<a href="Digital_video.html">Read more… </a></p>
<br>
<iframe width="280" height="157"
src="http://www.youtube.com/embed/Uyxy9DCrq1U" frameborder="0" allow-
fullscreen></iframe>

<!ADDITIONAL PARAGRAPHS ON THE WEB PAGE SHOULD BE
PLACED HERE !>
    …

<hr>
<a href="mailto:WebT@gmail.com?subject=Hello">Send us an E-mail</a><br>
</td></tr></table></body></html>
```

9.10 Email

Electronic mail or email is another useful Internet service that allows the exchange of information between the computer users. An electronic address (email address) is needed to send and receive messages. Email can be sent to one or more email addresses (one or more users). It works through the SMTP protocol, which is based on TCP/IP protocol.

In addition to text, messages can also contain other data types, which need to be encoded and decoded. For encoding and decoding of nontext files the Multipurpose Internet Mail Extensions (MIME) standard is used.

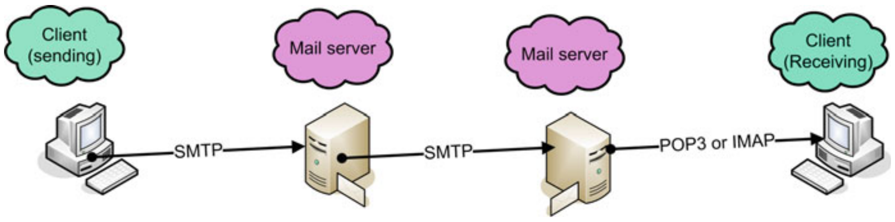


Fig. 9.29 Client–server model for sending/receiving email

As noted above, email operates on the principle of client and server, which also means that there must be appropriate software for clients and servers that allow sending and retrieving email. Namely, when sending an email, the message is sent from the client to the appropriate server. A client needs the access (by using either POP3 or IMAP protocols) to the appropriate server to download or read emails (Fig. 9.29).

Client programs (email clients) are the programs that are used for creating, sending, and reading emails. Examples of the email clients are *Outlook Express* and *Mozilla Thunderbird.*

E-mail Address
The general form of an email address is:
username@domain.top_domain

9.10.1 MIME: Multipurpose Internet Mail Extensions

The SMTP protocol is used to exchange email messages. Messages are formatted using the RFC 822 standard. Specifically, each message consists of header and message body. The format of each header field is defined by this standard. The standard header (defined in the RFC 822 standard) is extended to the new fields in the MIME standard. In addition to fields:

To:
From:
Subject:

the header also contains:
MIME version (a version of **MIME** used)
Content-Description: (description of message)
Content-ID (unique identifier of the content)
Content-Type: (type of content contained in the message body)
Content-Transfer-Encoding: (method for encoding data)
Content-Length: (size in bytes of the message body)

MIME content types

Content type	Subtype
Text	Plain (ASCII), Richtext (HTML)
Image	JPEG, GIFF
Audio	Basic
Video	MPEG
Applications	MSWord, Octet-stream
Multipart	Different content types

An example of multipart message type

```
From:   Karen@yahoo.com
To:   Jack@gmail.com
Subject:   Out of sight
MIME-version:   1.0
Content-Type:   Multiport/Alternative; boundary=”StartOfNextPart”;

–StartOfNextPart
Content-Type: Message/External-body;
                  name = “out_of_sight.mpeg”;
                  directory = “Out_of_Sight”;
                  access-type=”anon-ftp”;
                  site=”universalpictures.com”;
     Content-Type: Video/MPEG;
     Content-Transfer-Encoding: Base64
     – StartOfNextPart
     Content-Type: Text/Plain;
                  ***Have you watched this movie?***
     –StartOfNextPart
```

9.11 Examples

9.1. Consider the networks denoted as X, Y, and Z. The IP addresses within the network X begin with 110, within the network Y with 133, while the addresses within the network Z begin with 192 (Fig. 9.30).

(a) Determine IP addresses of networks X, Y, and Z.
(b) Determine the class of addresses for each of the networks X, Y, and Z.

Solution:

(a) IP address of X is: 110.0.0.0,
IP address of Y is: 133.12.0.0,
IP address of Z is: 192.3.12.0.

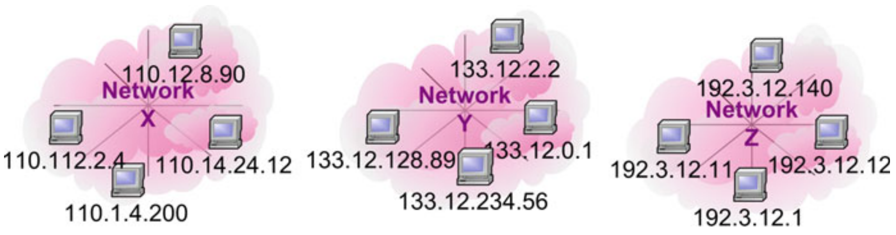


Fig. 9.30 Networks X, Y and Z

(b) Addresses from X belong to class A (the first number is 110),
Addresses from Y belong to class B (the first number is 133),
Addresses from Z belong to class C (the first number is 192).

9.2. For a given IP addresses of a PC, determine whether it belongs to one of the networks from the previous example. The IP addresses are:

(a) 110.3.14.55
(b) 11.2.14.55
(c) 133.12.12.12
(d) 192.12.2.2
(e) 192.3.12.31

Solution:

(a) PC belongs to network X.
(b) PC does not belong to any of the networks X, Y, and Z.
(c) PC belongs to Y.
(d) PC does not belong to any of the networks X, Y, and Z.
(e) PC belongs to the network Z.

9.3. Consider the following data:

```
Physical Address.....................: 00 - 54 - 99 - B4 - BA - 92
Dhcp Enabled..........................: Yes
Autoconfiguration Enabled.....: Yes
Subnet Mask...........................: 255.255.0.0
Default Gateway.....................: 38.189.1.2
DHCP Server..........................: 38.189.1.3
```

(a) Determine the local network address.
(b) What could be the IP addresses of the PCs within the network?

Solution:

(a) The local network address is : 38.189.0.0.

(b) The PCs within the observed network could have addresses from 38.189.0.4 to 38.189.255.254.

9.4. What type of IPv6 addresses starts with FEC0 (in hexadecimal format)?

Solution:

The address starting with FEC0 or in the binary form with 1111 1110 1100 0000 has the first 10 bits that correspond to the site-local address.

9.5. Determine the type of IPv6 address:

(a) ::FFFF:129.144.52.38
(b) ::129.144.52.38
(c) FE80:0000:0000:0000:0000:0800:0212:3456
(d) FEC0:0000:0000:0000:0011:0000:0C12:3456
(e) 0000:0000:0000:0000:0000:0000:0000:0000
(f) 0000:0000:0000:0000:0000:0000:0000:0001
(g) FF01:0000:0000:0000:0000:0000:0000:0002

Solution:

(a) IPv4-Mapped Embedded IPv6 address
(b) IPv4 compatible Ipv6 address
(c) link-local address
(d) site-local address
(e) unspecified
(f) loopback address
(g) multicast address

9.6. Write the following addresses in the short format:

(a) 4E80:00D2:0156:0000:0000:0000:0000:3456
(b) 4E80:00D2:0156:0000:0000:0800:0000:3456

Solution:

(a) 4E80:D2:156::3456
(b) 4E80:D2:156::800:0:3456

9.7. Discuss the following commands:

```
ftp> open domain.name
Connected to domain.name
220 FTP server ready.
User (domain.name:(none)): User-Name
331 Password required for user-name
Password: password
230 User user-name logged in.
  ftp>
```

Solution:

The command *ftp* is entered in the command line. Then using the command *open*, the connection is initiated with the server. We get the information that the connection is established (*Connected to domain.name*) and the acknowledgement that the server is ready for communication (*220 FTP server ready*). The user enters its *User-name*, and then the server requires the password (*331 Password required for user-name*). The user enters the password as well. The server sends the acknowledgement that the user is logged in (*230 User user-name logged in*).

References

1. Black UD (2000) Internet architectures: an introduction to IP protocols. Prentice Hall, Upper Saddle River
2. Doyle J (1998) Routing TCP/IP volume I (CCIE professional development). Cisco Press, Indianapolis
3. Halsall F (2001) Multimedia communications: applications, networks, protocols, and standards. Addison-Wesley, Harlow
4. Forouzan BA, Fegan SC (2003) Data communications and networking. McGraw-Hill, Boston
5. Johnston AB, Sinnreich H (2006) Internet communications using SIP: delivering VoIP and multimedia services with session initiation protocol. Wiley, Indianapolis
6. Keshav S (1997) An engineering approach to computer networking: ATM networks, the internet, and the telephone network. Addison-Wesley, Reading
7. Korpi M, Kumar V, Sengodan S (2001) IP telephony with H.323: architectures for unified networks and integrated services. Wiley, New York
8. Kraig G (2007) The essential guide to CSS and HTML web design. Springer, New York
9. Niederst J, Niederst Robbins J (2001) Web design in a nutshell: a desktop quick reference. O'Reilly Media, Inc, Beijing
10. Ohm JR (2004) Multimedia communication technology representation, transmission and identification of multimedia signals. Springer, Berlin
11. Sulkin A (2002) PBX systems for IP telephony. McGraw-Hill Professional, New York
12. Wright DJ (2001) Voice over packet networks. Wiley, Chichester

Index

A
AC-3 Dolby, 90
Adaptive Spectral Perceptual Entropy Coding (ASPEC), 88
Adaptive Transform Acoustic Coder (ATRAC), 121
Affine transformation, 157, 179
A-law, 60
Aliasing, 58
Ambiguity function, 250
Analysis filters, 25
Antialiasing filter, 58
Antishock, 121
Approximation space, 23
Arithmetic coding, 178
Arithmetic mean filter, 144
ASPEC. *See* Adaptive Spectral Perceptual Entropy Coding (ASPEC)
Asynchronous Transfer Mode (ATM), 218, 221, 297
ATRAC. *See* Adaptive Transform Acoustic Coder (ATRAC)
Attacks, 258
Audio signal, 55, 57, 86
Auto-correlation, 72

B
Bark, 87
Basic Rate Interface (BRI), 295
Best-effort service, 300
Binary image, 133
Bit allocation, 91
Bit context, 175
Bit planes, 173, 176
BRI. *See* Basic Rate Interface (BRI)
Broadband ISDN, 297

C
CABAC. *See* Context-based Adaptive Binary Arithmetic Coding (CABAC)
Cauchy distribution, 265–266
CAVLC. *See* Context Adaptive Variable Length Coding (CAVLC)
CD encoding, 111
Cell-switching networks, 297
CIF, 201
4CIF, 201
CIRC. *See* Cross-Interleaved Reed-Solomon Coding (CIRC)
Classes of IP addresses, 303
Classless addressing, 304
Cleanup pass, 176
CMY, 139
CMYK, 139
Colorfulness, 152
Compact disc (CD), 57, 109
Compression ratio, 79
Consonants, 63, 65
Context, 178, 179
Context Adaptive Variable Length Coding (CAVLC), 216
Context-based Adaptive Binary Arithmetic Coding (CABAC), 216
Continuous wavelet transform (CWT), 18
Contrast, 153, 189
Co-occurrence matrix, 154
Correlation, 155, 236, 263
Correlation coefficient, 266
CRC. *See* Cyclic redundancy check (CRC)
Critical bands, 78, 86
Cross-Interleaved Reed-Solomon Coding (CIRC), 113–117
Cross-terms, 13

Curie point, 120
CWT. *See* Continuous wavelet transform (CWT)
Cyclic redundancy check (CRC), 111

D
DAB. *See* Digital audio broadcasting (DAB)
Daubechies filters, 26
Daubechies wavelets, 19
Diagnostic imaging, 289
Dictionary, 81
Digital audio broadcasting (DAB), 57
Dilation equation, 23
Directions for block prediction, 212
Discrete cosine transform, 5
DNS. *See* Domain Name System (DNS)
DNS resolver, 314
Domain Name System (DNS), 314
DVD-audio, 123
Dyadic sampling, 19

E
Edison phonograph, 57
EFM. *See* Eight to Fourteen Modulation (EFM)
Eigenvalues, 74
Eigenvectors, 74
Eight to Fourteen Modulation (EFM), 111, 119
E-mail address, 327, 339
Energy-entropy feature, 71
Entropy, 71, 79
Entropy coding, 161, 175

F
False alarm, 262
File Transfer Protocol (FTP), 316
Filter bank, 26, 87
Fletcher curve, 56
Floating-point conversion, 61
Floating-point wavelet transform (9, 7), 170
Formant bandwidth, 65
Formants, 65
Fourier transform, 1, 3
Fractal compression, 179
Fractional Image Format (FIF), 181
Fragile watermarking, 256
Frame, 69, 71, 116, 199
Frequency resolution, 9
FTP. *See* File Transfer Protocol (FTP)

G
Galois fields, 116
Gatekeeper, 220
Gateway, 220
Gauss-Hermite quadrature, 32
Gaussian distribution, 262–265
Gaussian noise, 142
Generalized Gaussian function, 265
Geometric mean filter, 144
Global edge, 157
Glottal tract, 63, 65
Grayscale image, 133, 138

H
H.261, 208
H.263, 208
H.264/MPEG4-AVC, 209
H.323 protocol, 219
Haar wavelet, 19
Hermite functions, 31, 33
Hermite polynomials, 32
High-pass filter, 5, 149
HSV color system, 140
HTML. *See* Hyper text markup anguage (HTML)
HTTP. *See* Hypertext transfer protocol (HTTP)
Huffman coding, 79, 84, 123, 161
Human auditory system, 55, 56
Hyper text markup language (HTML), 299, 313, 318
Hypertext transfer protocol (HTTP), 313
Hypothesis testing problem, 261

I
Impulse noise, 142
Incoherence, 236
Instantaneous frequency, 8
Instantaneous power, 12
Integer wavelet transform (5,3), 170
Integrated Services Digital Networks (ISDN), 208, 218, 293, 295
Interactive internet applications, 297
Interleaving, 112
Internet, 294, 301
Inverse discrete Fourier transform, 4
IP address, 301, 302
IP datagram, 308
IP protocol, 308
IPv4, 302
Irreversible color transform, 168
ISDN. *See* Integrated Services Digital Networks (ISDN)

J
JPEG, 158, 164, 166
JPEG2000, 168
JPEG encoder, 158

L
Land, 109
Lead-in, 109
Lead-out, 109
Likelihood ratio, 261
L_1 minimization, 238
L_1 norm, 237
Logarithmic search, 204
Low-pass filter, 5, 26, 149

M
Macroblock, 205, 206, 208
Magnetic head, 120
Marginal conditions, 11
Masking curve, 78, 88
Masking-pattern Universal Subband Integrated Coding And Multiplexing (MUSICAM), 88
Maximum segment size (MSS), 309
Measure of detection quality, 266
Median filter, 145
Mini Disc, 120
Minimally invasive surgery, 291
Misdetection, 262
Motion estimation, 201
Motion vector, 201, 202, 213
MPEG audio, 88
MPEG Layer I audio, 91
MPEG-1 video, 205
MPEG-2 video, 206
MPEG-4 video, 207
MSS. *See* Maximum segment size (MSS)
Multicasting, 298
Multimedia, 1
Multimedia signals, 1
Multiplexing, 223, 300
Multipurpose Internet Mail Extensions (MIME) standard, 339
Multiresolution analysis, 23
MUSICAM. *See* Masking-pattern Universal Subband Integrated Coding And Multiplexing (MUSICAM)

N
National Television System Committee (NTSC), 201
Nonlinear quantization, 60
Nonpersistent HTTP, 314
NTSC. *See* National Television System Committee (NTSC)

O
Orthogonal frequency-division multiplexing (OFDM), 125

P
PAL system, 201
PBX. *See* Private branch exchange (PBX)
Persistent HTTP, 314
Phonemes, 65
Pit, 109, 119
Pitch frequency, 67
Prediction gain, 69
Primal-dual interior point method, 242
Primary rate interface, 295
Private branch exchange (PBX), 295
Probability of error, 261
PSTN. *See* Public Switched Telephone Networks (PSTN)
Psychoacoustical model, 93
Public Switched Telephone Networks (PSTN), 218, 293
Pyramidal decomposition, 29

Q
QCIF. *See* Quarter common intermediate format (QCIF)
Quadrature phase-shift keying (QPSK), 125
Quality factor, 160
Quality of service, 218, 300
Quantization, 59, 158, 160, 172
Quantization matrix, 160
Quantization noise, 59, 88
Quarter common intermediate format (QCIF), 201
Quarter pixel precision, 214
Quincunx decomposition, 28

R
Random measurements, 235
Rayleigh distribution, 143
Real-time Transport Control Protocol (RTCP), 307
Real-time Transport Protocol (RTP), 220, 307
Reed-Solomon code, 116
Reference frame, 201, 207, 215
Regions of interest (ROI), 173

Restricted isometry property, 237
Reversible color transform, 168
RGB color system, 138
ROI. *See* Regions of interest (ROI)
RTCP. *See* Real-time Transport Control Protocol (RTCP)
RTP. *See* Real-Time Transport Protocol (RTP)

S
SACD. *See* Super Audio CD (SACD)
Sample and hold circuit, 59
Sampling theorem, 36, 37
Scale factor, 91
Scaling parameter, 18
Semi-fragile watermarking, 256
Sensing matrix, 237, 239
Session Initiation Protocol (SIP), 220, 222
Short-time Fourier transform, 8
SI frames, 211
Signal to noise ratio, 59, 62
Simple Mail Transfer Protocol (SMTP), 318, 339, 340
Singular value decomposition (SVD), 72
Singular values, 72
Singular vectors, 72, 76
SIP. *See* Session Initiation Protocol (SIP)
S-method, 13–14, 73, 273
SMTP. *See* Simple Mail Transfer Protocol (SMTP)
Sobel matrices, 155
Sound pressure level, 55
Sparsity, 234
Sparsity property, 234
Spectral energy density, 12
Spectrogram, 9
SP frames, 210
Spread-spectrum watermarking, 267
Standard wavelet decomposition, 28
SubQCIF, 201
Super Audio CD (SACD), 122
SVD. *See* Singular value decomposition (SVD)
Switch, 299
Synchronization word, 111, 120
Synthesis filters, 25

T
Tag, 318
TCP. *See* Transmission Control Protocol (TCP)
Telecardiology, 287
Teleconferencing, 297
Telemedicine technologies, 285
Telenursing, 285
Telepharmacy, 286
Teleradiology, 289
Telerehabilitation, 286
Telesurgery, 289
Texture, 153
Three steps algorithm, 203
Three-way handshaking, 311
Threshold, 70, 72, 77, 86, 141, 155, 202, 261, 262, 264
Threshold in quiet, 88
Time-frequency analysis, 6
Time resolution, 9
Time-varying filtering, 16
Total-variation minimization, 238, 247
Translation parameter, 18, 20
Transmission Control Protocol (TCP), 307
  segment, 310
  segment header, 310

U
UDP. *See* User Datagram Protocol (UDP)
Uncertainty principle, 10
Uniform distribution, 143
Uniform wavelet decomposition, 30
User Datagram Protocol (UDP), 307
  segment header, 313
  segments, 312

V
Video Coding Experts Group (VCEG) algorithms, 208
Videoconferencing, 298
Video formats, 201
Video on demand, 299
Vocal folds, 63
Vocal tract, 63, 65
Voice activity indicators, 68
Voice over IP, 220, 297
Vowels, 63

W
Watermark, 255, 257
Watermark detection, 255, 260, 261
Watermark robustness, 258
Wavelets, 17, 19

Wavelet space, 23
Web page, 314
Web server, 299
Wiener filter, 150
Wigner distribution, 7–8
Wireless ECG, 288
World Wide Web (WWW), 313–314

Y
YCrCb color model, 166, 169

Z
Zero-crossing rate, 68
Zig-zag reordering, 161