Global and Regional Features

3

Measure twice, cut once.

—Carpenter's saying

This chapter covers the metrics of general feature description, often used for whole images and image regions, including textural, statistical, model based, and basis space methods. Texture, a key metric, is a well-known topic within image processing, and it is commonly divided into structural and statistical methods. Structural methods look for features such as edges and shapes, while statistical methods are concerned with pixel value relationships and statistical moments. Methods for modeling image texture also exist, primarily useful for image synthesis rather than for description. Basis spaces, such as the Fourier space, are also use for feature description.

It is difficult to develop clean partitions between the related topics in image processing and computer vision that pertain to global vs. regional vs. local feature metrics; there is considerable overlap in the applications of most metrics. However, for this chapter, we divide these topics along reasonable boundaries, though those borders may appear to be arbitrary. Similarly, there is some overlap between discussions here on global and regional features and topics that are covered in Chap. 2 on image processing and that are discussed in Chap. 6 on local features. In short, many methods are used for local, regional, and global feature description, as well as image processing, such as the Fourier transform and the LBP.

But we begin with a brief survey of some key ideas in the field of texture analysis and general vision metrics.

Historical Survey of Features

To compare and contrast global, regional, and local feature metrics, it is useful to survey and trace the development of the key ideas, approaches, and methods used to describe features for machine vision. This survey includes image processing (textures and statistics) and machine vision (local, regional, and global features). Historically, the choice of feature metrics was limited to those that were computable at the time, given the limitations in compute performance, memory, and sensor technology. As time passed and technology developed, the metrics have become more complex to compute, consuming larger memory footprints. The images are becoming *multimodal*, combining intensity, color, multiple spectrums, depth sensor information, multiple-exposure settings, high dynamic range imagery, faster frame rates, and more precision and accuracy in x, y, and Z depth. Increases in memory bandwidth and compute performance, therefore, have given rise to new ways to describe feature metrics and perform analysis.

S. Krig, *Computer Vision Metrics*, DOI 10.1007/978-3-319-33762-3_3

Many approaches to texture analysis have been tried; these fall into the following categories:

- **Structural**, describing texture via a set of micro-texture patterns known as texels. Examples include the numerical description of natural textures such as fabric, grass, and water. Edges, lines, and corners are also structural patterns, and the characteristics of edges within a region, such as edge direction, edge count, and edge gradient magnitude, are useful as texture metrics. Histograms of edge features can be made to define texture, similar to the methods used in local feature descriptors such as SIFT (described in Chap. 6).
- **Statistical**, based on gray level statistical moments describing point pixel area properties, and includes methods such as the co-occurrence matrix or SDM. For example, regions of an image with color intensity within a close range could be considered as having the same texture. Regions with the same histogram could be considered as having the same texture.
- **Model based**, including fractal models, stochastic models, and various semi-random fields. Typically, the models can be used to generate synthetic textures, but may not be effective in recognizing texture, and we do not cover texture generation.
- **Transform or basis based**, including methods such as Fourier, Wavelets, Gabor filters, Zernike, and other basis spaces, which are treated here as a subclass of the statistical methods (statistical moments); however, basis spaces are used in transforms for image processing and filtering as well.

Key Ideas: Global, Regional, and Local Metrics

Let us take a brief look at a few major trends and milestones in feature metrics research. While this brief outline is not intended to be a precise, inclusive look at all key events and research, it describes some general trends in mainstream industry thinking and academic activity.

1960s, 1970s, 1980s—Whole-Object Approaches

During this period, metrics describe mostly whole objects, larger regions, or images; pattern matching was performed on large targets via FFT spectral methods and correlation; recognition methods included object, shape, and texture metrics; and simple geometric primitives were used for object composition. Low-resolution images such as NTSC, PAL, and SECAM were common—primarily gray scale with some color when adequate memory was available. Some satellite images were available to the military with higher resolution, such as LANDSAT images from NASA and SPOT images from France.

Some early work on pattern recognition began to use local interest points and features: notably, Moravic [502] developed a local interest point detector in 1981, and in 1988 Harris and Stephens [148] developed local interest point detectors. Commercial systems began to appear, particularly the View PRB in the early 1980s, which used digital correlation and scale space super-pixels for coarse to fine matching, and real-time image processing and pattern recognition systems were introduced by Imaging Technology. Rack-mounted imaging and machine vision systems began to be replaced by workstations and high-end PCs with add-on imaging hardware, array processors, and software libraries and applications by companies such as Krig Research.

Early 1990s—Partial-Object Approaches

Compute power and memory were increasing, enabling more attention to local feature methods, such as developments from Shi and Tomasi [149] improving the Harris detector methods, Kitchen and Rosenfeld [200] developing gray level corner detection methods, and methods by Wang and Brady [205]. Image moments over polygon shapes were computed using Zernike polynomials in 1990 by

Khotanzad and Hong [268]. Scale space theory was applied to computer vision by Lindberg [502], and many other researchers followed this line of thinking into the future, such as Lowe [153] in 2004.

Metrics described smaller pieces of objects or object components and parts of images; there was increasing use of local features and interest points. Large sets of sub-patterns or basis vectors were used and corresponding metrics were developed. There was increased use of color information; more methods appeared to improve invariance for scale, rotational, or affine variations; and recognition methods were developed based on finding parts of an object with appropriate metrics. Higher image resolution, increased pixel depths, and color information were increasingly used in the public sector (especially in medical applications), along with of new affordable image sensors, such as the KODAK MEGA-PLUS, which provided a 1024×1024 image.

Mid-1990s—Local Feature Approaches

More focus was put on metrics that identify small local features surrounding interest points in images. Feature descriptors added more details from a window or patch surrounding each feature, and recognition was based on searching for sets of features and matching descriptors with more complex classifiers. Descriptor spectra included gradients, edges, and colors.

Late 1990s—Classified Invariant Local Feature Approaches

New feature descriptors were developed and refined to be invariant to changes in scale, lightness, rotation, and affine transformations. Work by Schmidt and Mohr [340] advanced and generalized the local feature description methods. Features acted as an alphabet for spelling out complex feature descriptors or vectors whereby the vectors were used for matching. The feature matching and classification stages were refined to increase speed and effectiveness using neural nets and other machine learning methods [134].

Early 2000s—Scene and Object Modeling Approaches

Scenes and objects were modeled as sets of feature components or patterns with well-formed descriptors; spatial relationships between features were measured and used for matching; and new complex classification and matching methods used boosting and related methods to combine strong and weak features for more effective recognition. The SIFT [153] algorithm from Lowe was published; SURF was also published by Bay et al. [152], taking a different approach using HAAR features rather than just gradients. The Viola–Jones method [486] was published, using HAAR features and a boosted learning approach to classification, accelerating matching. The OpenCV library for computer vision was developed by Bradski at INTEL™, and released as open source.

Mid-2000s—Finer-Grain Feature and Metric Composition Approaches

The number of researchers in this field began to mushroom; various combinations of features and metrics (bags of features) were developed by Czurka et al. [226] to describe scenes and objects using key points as described by Sivic [503]; new local feature descriptors were created and old ones refined; and there was increased interest in real-time feature extraction and matching methods for commercial applications. Better local metrics and feature descriptors were analyzed, measured, and used together for increased pattern match accuracy. Also, feature learning and sparse feature codebooks were developed to decrease pattern space, speed up search time, and increase accuracy.

Post-2010—Multimodal Feature Metrics Fusion

There has been increasing use of depth sensor information and depth maps to segment images and describe features and create VOXEL metrics for example see Rusu et al. [380], for example 2D texture metrics are expressed in 3-space. 3D depth sensing methods proliferate, increasing use of

high-resolution images and high dynamic range (HDR) images to enhance feature accuracy, and greater bit depth and accuracy of color images allows for valuable color-based metrics and computational imaging. Increased processing power and cheap, plentiful memory handle larger images on low-cost compute platforms. Faster and better feature descriptors using binary patterns have been developed and matched rapidly using Hamming distance, such as FREAK by Alahi et al. [122] and ORB by Rublee et al. [112]. Multimodal and multivariate descriptors [770, 771] are composed of image features with other sensor information, such as accelerometers and positional sensors.

Future computing research may even come full circle, when sufficient compute and memory capacity exist to perform the older methods, like correlation across multiple scales and geometric perspectives in real-time using parallel and fixed-function hardware methods. This would obviate some of the current focus on small invariant sets of local features and allow several methods to be used together, synergistically. Therefore, the history of development in this field is worth knowing, since it might repeat itself in a different technological embodiment.

Since there is no single solution for obtaining the right set of feature metrics, all the methods developed over time have applications today and are still in use.

Textural Analysis

One of the most basic metrics is *texture*, which is the description of the surface of an image channel, such as color intensity, like an elevation map or terrain map. Texture can be expressed globally or within local regions. Texture can be expressed *locally* by statistical relationships among neighboring pixels in a region, and it can be expressed *globally* by summary relationships of pixel values within an image or region. For a sampling of the literature covering a wide range of texture methods, see Refs. [13, 16–20, 52, 53, 302, 304, 305].

According to Gonzalez [4], there are three fundamental classes of texture in image analysis: statistical, structural, and spectral. *Statistical* measures include histograms, scatter plots, and SDMs. *Structural* techniques are more concerned with locating patterns or structural primitives in an image, such as parallel lines, regular patterns, and so on. These techniques are described in [1, 5, 8, 11]. *Spectral* texture is derived from analysis of the frequency domain representation of the data. That is, a fast Fourier transform is used to create a frequency domain image of the data, which can then be analyzed using Fourier techniques.

Histograms reveal overall pixel value distributions but say nothing about spatial relationships. Scatter plots are essentially two-dimensional histograms, and do not reveal any spatial relationships. A good survey is found in Ref. [307].

Texture has been used to achieve several goals:

- Texture-based segmentation (covered in Chap. 2).
- Texture analysis of image regions (covered in this chapter).
- Texture synthesis, creating images using synthetic textures (not covered in this book).

In computer vision, texture metrics are devised to describe the perceptual attributes of texture by using discrete methods. For instance, texture has been described *perceptually* with several properties, including:

- Contrast
- Color
- Coarseness

- Directionality
- Line-likeness
- Roughness
- Constancy
- Grouping
- Segmentation

If textures can be recognized, then image regions can be segmented based on texture and the corresponding regions can be measured using shape metrics such as area, perimeter, and centroid (as discussed in Chap. 6). Chapter 2 included a survey of segmentation methods, some of which are based on texture. Segmented texture regions can be recognized and compared for computer vision applications. Micro-textures of a local region, such as the LBP discussed in detail in Chap. 6, can be useful as a feature descriptor, and macro-textures can be used to describe a homogenous texture of a region such as a lake or field of grass, and therefore have natural applications to image segmentation. In summary, texture can be used to describe global image content, image region content, and local descriptor region content. The distinction between a feature descriptor and a texture metric may be small.

Sensor limitations combined with compute and memory capabilities of the past have limited the development of texture metrics to mainly 2D gray scale metrics. However, with the advances toward pervasive computational photography in every camera providing higher resolution images, higher frame rates, deeper pixels, depth imaging, more memory, and faster compute, we can expect that corresponding new advances in texture metrics will be made.

Here is a brief historical survey of texture metrics.

1950s Through 1970s—Global Uniform Texture Metrics

Auto-correlation or cross-correlation was developed by Kaizer [26] in 1955 as a method of looking for randomness and repeating pattern features in aerial photography, where auto-correlation is a statistical method of correlating a signal or image with a time-shifted version of itself, yielding a computationally simple method to analyze ground cover and structures.

Bajcsy [25] developed Fourier spectrum methods in 1973 using various types of filters in the frequency domain to isolate various types of repeating features as texture.

Gray level spatial dependency matrices, GLCMs, SDMs or co-occurrence matrices [6] were developed and used by Haralick in 1973, along with a set of summary statistical metrics from the SDMs to assist in numerical classification of texture. Some, but not all, of the summary metrics have proved useful; however, analysis of SDMs and development of new SDM metrics have continued, involving methods such as 2D visualization and filtering of the SDM data within spatial regions [23], as well as adding new SDM statistical metrics, some of which are discussed in this chapter.

1980s—Structural and Model-Based Approaches for Texture Classification

While early work focused on micro-textures describing statistical measures between small kernels of adjacent pixels, macro-textures developed to address the structure of textures within a larger region. K. Laws developed *texture energy-detection* methods in 1979 and 1980 [27–29], as well as *texture classifiers*, which may be considered the forerunners of some of the modern classifier concepts. The Laws method could be implemented as a texture classifier in a parallel pipeline with stages for taking gradients via of a set of convolution masks over Gaussian filtered images to isolate texture micro features, followed by a Gaussian smoothing stage to deal with noise, followed by the energy calculation from the combined gradients, followed by a classifier which matched texture descriptors.

Eigenfilters were developed by Ade [30] in 1983 as an alternative to the Laws gradient or energy methods and SDMs; eigenfilters are implemented using a covariance matrix representation of local 3×3 pixel region intensities, which allows texture analysis and aggregation into structure based on the variance within eigenvectors in the covariance matrix.

Structural approaches were developed by Davis [31] in 1979 to focus on gross structure of texture rather than primitives or micro-texture features. *Hough transforms* were invented in 1972 by Duda and Hart [220] as a method of finding lines and curves, and it was used by Eichmann and Kasparis [32] in 1988 to provide invariant texture description.

Fractal methods and *Markov random field* methods were developed into texture descriptors, and while these methods may be good for texture synthesis, they do not map well to texture classification, since both Fractal and Markov random field methods use random fields, thus there are limitations when applied to real-world textures that are not random.

1990s—Optimizations and Refinements to Texture Metrics

In 1993, Lam and Ip [33, 39] used pyramid segmentation methods to achieve spatial invariance, where an image is segmented into homogenous regions using Voronoi polygon tessellation and irregular pyramid segmentation techniques around Q points taken from a binary thresholded image; five shape descriptors are calculated for each polygon: area, perimeter, roundness, orientation, and major/minor axis ratio, combined into texture descriptors.

Local binary patterns (LBP) were developed in 1994 by Ojala et al. [165] as a novel method of encoding both pattern and contrast to define texture [15, 16, 35, 36]; since then, hundreds of researchers have added to the LBP literature in the areas of theoretical foundations, generalization into 2D and 3D, domain-specific interest point descriptors used in face detection, and spatiotemporal applications to motion analysis [34]. LBP research remains quite active at this time. LBPs are covered in detail in Chap. 6. There are many applications for the powerful LBP method as texture metric, a feature descriptor, and an image processing operator, the latter which was discussed in Chap. 2.

2000 to Today—More Robust Invariant Texture Metrics and 3D Texture

Feature metrics research is investigating texture metrics that are invariant to scale, rotation, lighting, perspective, and so on to approach the capabilities of human texture discrimination. In fact, texture is used interchangeably as a feature descriptor in some circles. The work by Pun and Lee [37] is an example of development of rotational invariant texture metrics, as well as scale invariance. Invariance attributes are discussed in the general taxonomy in Chap. 5.

The next wave of metrics being developed increasingly will take advantage of 3D depth information. One example is the surface shape metrics developed by Spence [38, 304] in 2003, which provide a bump-map type metric for affine invariant texture recognition and texture description with scale and perspective invariance. Chapter 6 also discusses some related 3D feature descriptors.

Statistical Methods

The topic of statistical methods is vast, and we can only refer the reader to selected literature as we go along. One useful and comprehensive resource is the online NIST National Institute of Science and Technology Engineering Statistics Handbook,[1] including examples and links to additional resources and tools.

[1] See the NIST online resource for engineering statistics: http://www.itl.nist.gov/div898/handbook/

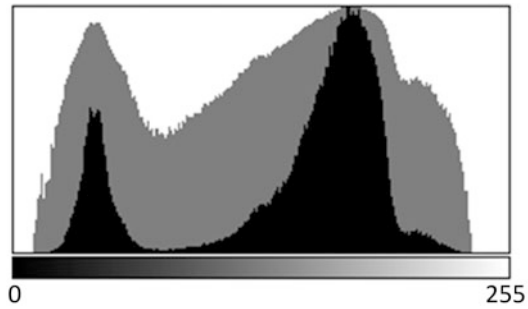


Figure 3.1 Histogram with linear scale values (*black*) and log scale values (*gray*), illustrating how the same data is interpreted differently based on the chart scale

Statistical methods may be drawn upon at any time to generate novel feature metrics. Any feature, such as pixel values or local region gradients, can be expressed statistically by any number of methods. Simple methods, such as the histogram shown in Fig. 3.1, are invaluable. Basic statistics such as minimum, maximum, and average values can be seen easily in the histogram shown in Chap. 2 in Fig. 2.21. We survey several applications of statistical methods to computer vision here.

Texture Region Metrics

Now we look in detail at the specific metrics for feature description based on texture. Texture is one of the most-studied classes of metrics. It can be thought of in terms of the surface—for example, a burlap bag compared to silk fabric. There are many possible textural relationships and signatures that can be devised in a range of domains, with new ones being developed all the time. In this section we survey some of the most common methods for calculating texture metrics:

- Edge metrics
- Cross-correlation
- Fourier spectrum signatures
- Co-occurrence matrix, Haralick features, extended SDM features
- Laws texture metrics
- Tessellation
- Local binary patterns (LBP)
- Dynamic textures

Within an image, each image region has a *texture signature*, where texture is defined as a common structure and pattern within that region. Texture signatures may be a function of position and intensity relationships, as in the spatial domain, or be based on comparisons in some other function basis and feature domain, such as frequency space using Fourier methods.

Texture metrics can be used to both segment and describe regions. Regions are differentiated based on texture homogeneousness, and as a result, texture works well as a method for region segmentation. Texture is also a good metric for feature description, and as a result it is useful for feature detection, matching, and tracking.

Appendix B contains several ground truth datasets with example images for computing texture metrics, including the CUReT reflectance and texture database from Columbia University. Several

key papers describe the metrics used against the CUReT dataset [21, 40–42] including the appearance of a surface as a bidirectional reflectance distribution function (BRDF) and a bidirectional texture function (BTF).

These metrics are intended to measure texture as a function of direction and illumination, to capture coarse details and fine details of each surface. If the surface texture contains significant sub-pixel detail not apparent in single pixels or groups of pixels, the BRDF reflectance metrics can capture the *coarse reflectance* details. If the surface contains pixel-by-pixel difference details, the BTF captures the *fine texture* details.

Edge Metrics

Edges, lines, contours, or ridges are basic textural features [308, 309]. A variety of simple metrics can be devised just by analyzing the edge structure of regions in an image. There are many edge metrics in the literature, and a few are illustrated here.

Computing edges can be considered on a continuum of methods from interest point to edges, where the interest point may be a single pixel at a gradient maxima or minima, with several connected gradient maxima pixels composed into corners, ridges line segments, or a contours. In summary, a *gradient point* is a degenerate edge, and an edge is a collection of connected gradient points.

The edge metrics can be computed locally or globally on image regions as follows:

- Compute the gradient **g(d)** at each pixel, selecting an appropriate gradient operator **g()** and select the appropriate kernel size or distance **d** to target either micro or macro edge features.
- The distance **d** or kernel size can be varied to achieve different metrics; many researchers have used 3×3 kernels.
- Compute edge orientation by binning gradient directions for each edge into a histogram; for example, use 45° angle increment bins for a total of 8 bins at 0°, 45°, 90°, 135°, 180°, 225°, 270°.

Several other methods can be used to compute edge statistics. The representative methods are shown here; see also Shapiro and Stockton [499] for a standard reference.

Edge Density

Edge density can be expressed as the average value of the gradient magnitudes g_{m} in a region.

$$E_{\mathrm{d}} = \frac{g_{\mathrm{m}}(d)}{\text{pixels in region}}$$

Edge Contrast

Edge contrast can be expressed as the ratio of the average value of gradient magnitudes to the maximum possible pixel value in the region.

$$E_{\mathrm{c}} = \frac{E_{\mathrm{d}}}{\text{max pixel value}}$$

Edge Entropy

Edge randomness can be expressed as a measure of the Shannon entropy of the gradient magnitudes.

$$E_{\mathrm{e}} = \sum_{i=0}^{n} g_{\mathrm{m}}(x_i) \log_{\mathrm{b}} g_{\mathrm{m}}(x_i)$$

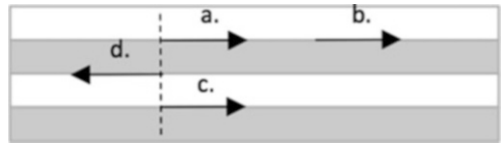


Figure 3.2 Gradient direction of edges a, b, c, d used to illustrate relationships for edge metrics

Edge Directivity

Edge directivity can be expressed as a measure of the Shannon entropy of the gradient directions.

$$E_{\mathrm{e}} = \sum_{i=0}^{n} g_{\mathrm{d}}(x_i) \log_{\mathrm{b}} g_{\mathrm{d}}(x_i)$$

Edge Linearity

Edge linearity measures the co-occurrence of collinear edge pairs using gradient direction, as shown by edges a–b in Fig. 3.2.

$$E_{\mathrm{l}} = \text{cooccurrence of colinear edge pairs}$$

Edge Periodicity

Edge periodicity measures the co-occurrence of identically oriented edge pairs using gradient direction, as shown by edges a–c in Fig. 3.2.

$$E_{\mathrm{p}} = \text{cooccurrence of identically oriented edge pairs}$$

Edge Size

Edge size measures the co-occurrence of opposite oriented edge pairs using gradient direction, as shown by edges a–d in Fig. 3.2.

$$E_{\mathrm{s}} = \text{cooccurrence of opposite oriented edge pairs}$$

Edge Primitive Length Total

Edge primitive length measures the total length of all gradient magnitudes along the same direction.

$$E_{\mathrm{t}} = \text{total length of gradient magnitudes with same direction}$$

Cross-Correlation and Auto-correlation

Cross-correlation [26] is a metric showing similarity between two signals with a time displacement between them. *Auto-correlation* is the cross-correlation of a signal with a time-displaced version of itself. In the literature on signal processing, cross-correlation is also referred to as a *sliding inner* product or *sliding dot* product. Typically, this method is used to search a large signal for a smaller pattern.

$$f * g = \overline{f}(-t) * g(t)$$

Using the Wiener–Khinchin theorem as a special case of the general cross-correlation theorem, cross-correlation can be written as simply the Fourier transform of the absolute square of the function f_{v}, as follows:

$$c(t) = \mathcal{F}_{\mathrm{v}}\left[|f_{\mathrm{v}}|^2\right](t)$$

In computer vision, the feature used for correlation may be a 1D line of pixels or gradient magnitudes, a 2D pixel region, or a 3D voxel volume region. By comparing the features from the current image frame and the previous image frame using cross-correlation derivatives, we obtain a useful texture change correlation metric.

By comparing displaced versions of an image with itself, we obtain a set of either local or global auto-correlation texture metrics. Auto-correlation can be used to detect repeating patterns or textures in an image, and also to describe the texture in terms of fine or coarse, where coarse textures show the auto-correlation function dropping of more slowly than fine textures. See also the discussion of correlation in Chap. 6 and Fig. 6.20.

Fourier Spectrum, Wavelets, and Basis Signatures

Basis transforms, such as the FFT, decompose a signal into a set of basis vectors from which the signal can be synthesized or reconstructed. Viewing the set of basis vectors as a spectrum is a valuable method for understanding image texture and for creating a signature. Several basis spaces are discussed in this chapter, including Fourier, HAAR, wavelets, and Zernike.

Although computationally expensive and memory intensive, the Fast Fourier Transform (FFT) is often used to produce a frequency spectrum signature. The FFT spectrum is useful for a wide range of problems. The computations typically are limited to rectangular regions of fixed sizes, depending on the radix of the transform (see Bracewell [219]).

As shown in Fig. 3.3, Fourier spectrum plots reveal definite image features useful for texture and statistical analysis of images. For example, Fig. 3.10 shows an FFT spectrum of LBP pattern metrics. Note that the Fourier spectrum has many valuable attributes, such as rotational invariance, as shown

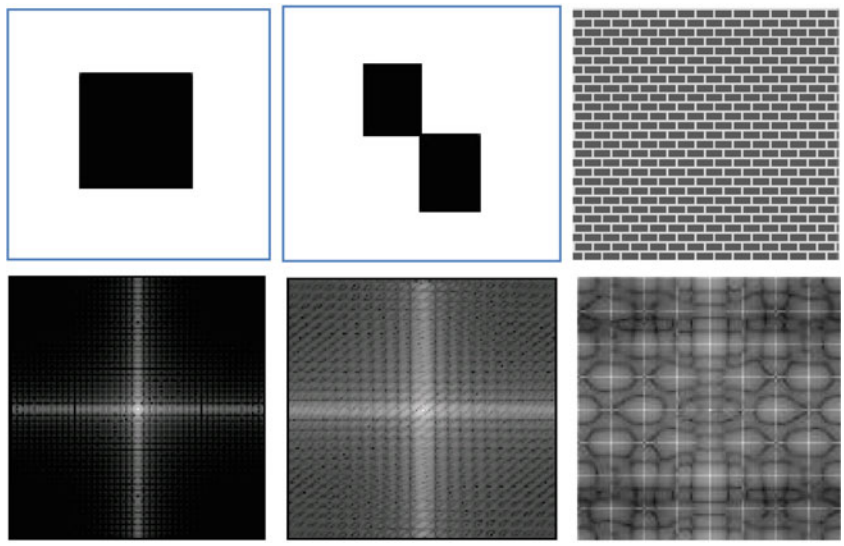


Figure 3.3 (*Top row*) Example images with texture. (*Bottom row*) Texture and shape information revealed in the corresponding FFT power spectrums

in Fig. 3.3, where a texture image is rotated 90° and the corresponding FFT spectrums exhibit the same attributes, only rotated 90°.

Wavelets [219] are similar to Fourier methods, and have become increasingly popular for texture analysis [303], discussed later in the section on basis spaces.

Note that the FFT spectrum as a texture metric or descriptor is rotational invariant, as shown in the bottom left image of Fig. 3.3. FFT spectra can be taken over rectangular 2D regions. Also, 1D arrays such as annuli or Cartesian coordinates of the shape taken around the perimeter of an object shape can be used as input to an FFT and as an FFT descriptor shape metric.

Co-occurrence Matrix, Haralick Features

Haralick [6] proposed a set of 2D texture metrics calculated from directional differences between adjacent pixels, referred to as *co-occurrence* matrices, *spatial dependency matrices* (SDM), or gray level co-occurrence matrices (GLCM). A complete set of four (4) matrices are calculated by evaluating the difference between adjacent pixels in the *x*, *y*, *diagonal x and diagonal y* directions, as shown in Fig. 3.4, and further illustrated with a 4×4 image and corresponding co-occurrence tables shown in Fig. 3.5.

One benefit of the SDM as a texture metric is that it is easy to calculate in a single pass over the image. The SDM is also fairly invariant to rotation, which is often a difficult robustness attribute to attain. Within a segmented region or around an interest point, the SDM plot can be a valuable texture metric all by itself, therefore useful for texture analysis, feature description, noise detection, and pattern matching.

For example, if a camera has digital-circuit readout noise, it will show up in the SDM for the *x* direction only if the lines are scanned out of the sensor one at a time in the *x* direction, so using the SDM information will enable intelligent sensor processing to remove the readout noise. However, it should be noted that SDM metrics are not always useful alone, and should be qualified with additional feature information. The SDM is primarily concerned with spatial relationships, with regard to spatial orientation and frequency of occurrence. So, it is primarily a statistical measure.

The SDM is calculated in four orientations, as shown in Fig. 3.4. Since the SDM is only concerned with adjacent pairs of pixels, these four calculations cover all possible spatial orientations. SDMs could be extended beyond 2×2 regions by using forming kernels extending into 5×5, 7×7, 9×9, and other dimensions.

A *spatial dependency matrix* is basically a count of how many times a given pixel value occurs next to another pixel value. Fig. 3.5 illustrates the concept. For example, assume we have an 8-bit image (0. 255). If an SDM shows that pixel value *x* frequently occurs adjacent to pixels within the range $x + 1$ to $x - 1$, then we would say that there is a "smooth" texture at that intensity. However, if

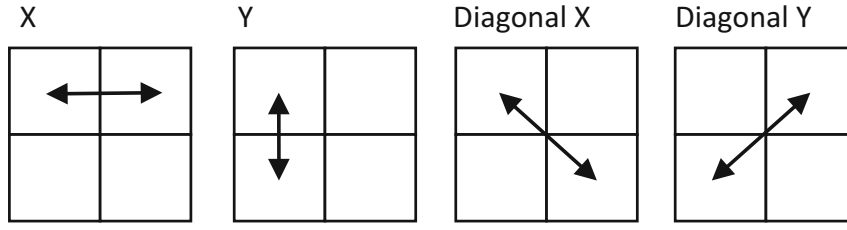


Figure 3.4 Four different vectors used for the Haralick texture features, where the difference of each pixel in the image is plotted to reveal the texture of the image

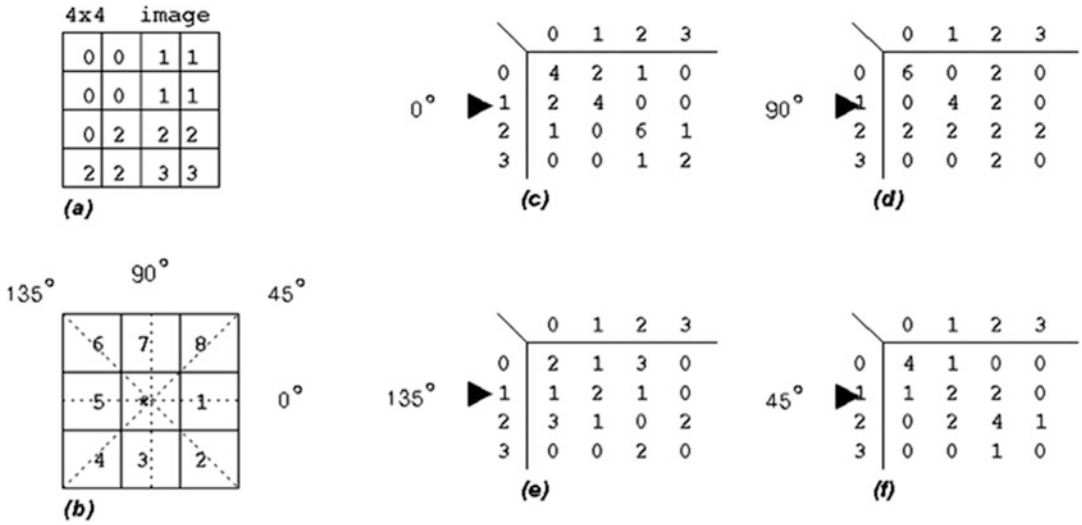


Figure 3.5 (**a**) 4 × 4 pixel image, with gray values in the range 0–3. (**b**) Nearest neighbor angles corresponding to SDM tables. (**c**–**f**) With neighborhood counts for each angle

pixel value x frequently occurs adjacent to pixels within the range $x + 70$ to $x - 70$, we would say that there is quite a bit of contrast at that intensity, if not noise.

A critical point in using SDMs is to be sensitive to the varied results achieved when sampling over small vs. large image areas. By sampling the SDM over a smaller area (say 64 × 64 pixels), details will be revealed in the SDMs that would otherwise be obscured. The larger the size of the sample image area, the more the SDM will be populated. And the more samples taken, the more likely that detail will be obscured in the SDM image plots. Actually, smaller areas (i.e., 64 × 64 pixels) are a good place to start when using SDMs, since smaller areas are faster to compute and will reveal a lot about local texture.

The Haralick metrics are shown in Fig. 3.6.

The statistical characteristics of the SDM have been extended by several researchers to add more useful metrics [23], and SDMs have been applied to 3D volumetric data by a number of researchers with good results [22].

Extended SDM Metrics (Krig SDM Metrics)

Extensions to the Haralick metrics have been developed by the author [23], primarily motivated by a visual study of SDM plots as shown in Fig. 3.7. Applications for the extended SDM metrics include texture analysis, data visualization, and image recognition. The visual plots of the SDMs alone are valuable indicators of pixel intensity relationships, and are worth using along with histograms to get to know the data.

The extended SDM metrics include centroid, total coverage, low-frequency coverage, total power, relative power, locus length, locus mean density, bin mean density, containment, linearity, and linearity strength. The extended SDM metrics capture key information that is best observed by looking at the SDM plots. In many cases the extended SDM metric are be computed four times, once for each SDM direction of 0°, 45°, 90°, and 135°, as shown in Fig. 3.5.

The SDMs are interesting and useful all by themselves when viewed as an image. Many of the texture metrics suggested are obvious after viewing and understanding the SDMs; others are neither

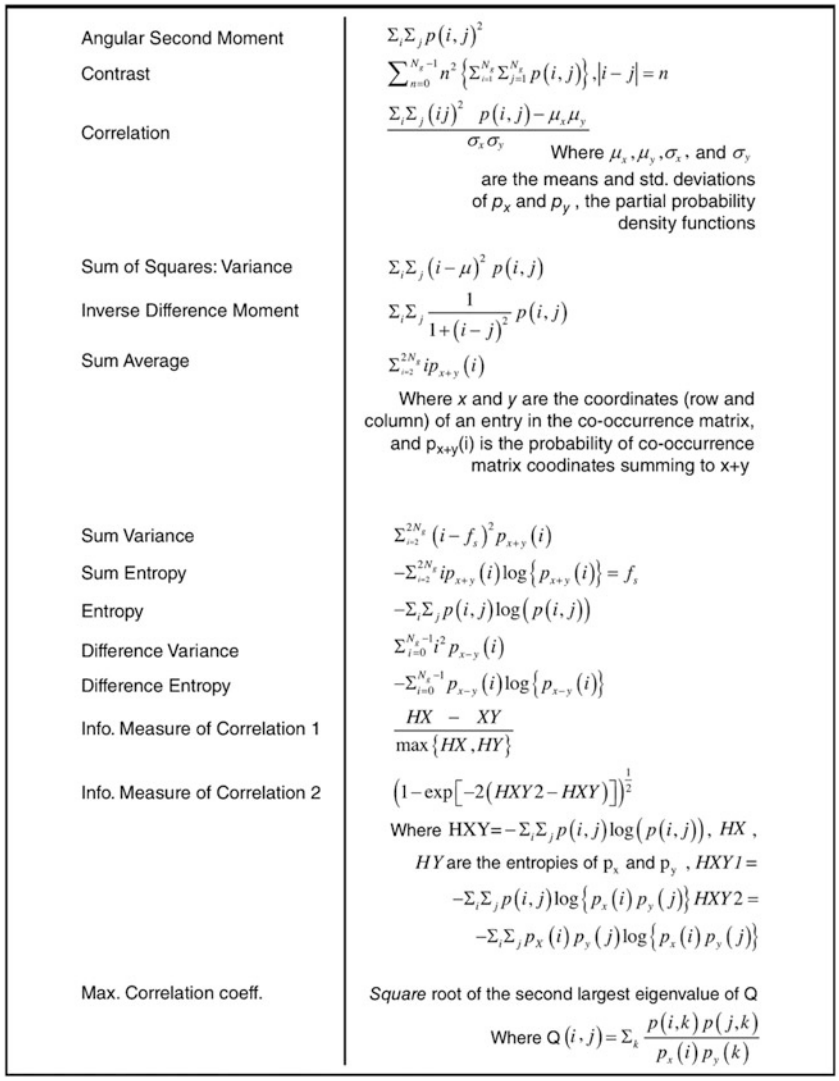


Figure 3.6 Haralick texture metrics. (Image used by permission, © Intel Press, from Building Intelligent Systems)

obvious nor apparently useful until developing a basic familiarity with the visual interpretation of SDM image plots. Next, we survey the following:

- **Example SDMs showing four directional SDM maps**: A complete set of SDMs would contain four different plots, one for each orientation. Interpreting the SDM plots visually reveals useful information. For example, an image with a smooth texture will yield a narrow diagonal band of co-occurrence values; an image with wide texture variation will yield a larger spread of values; a noisy image will yield a co-occurrence matrix with outlier values at the extrema. In some cases, noise may only be distributed along one axis of the image—perhaps, across rows or the *x* axis, which could indicated sensor readout noise as each line is read out of the sensor, suggesting a row- or line-oriented image preparation stage in the vision pipeline to compensate for the camera.

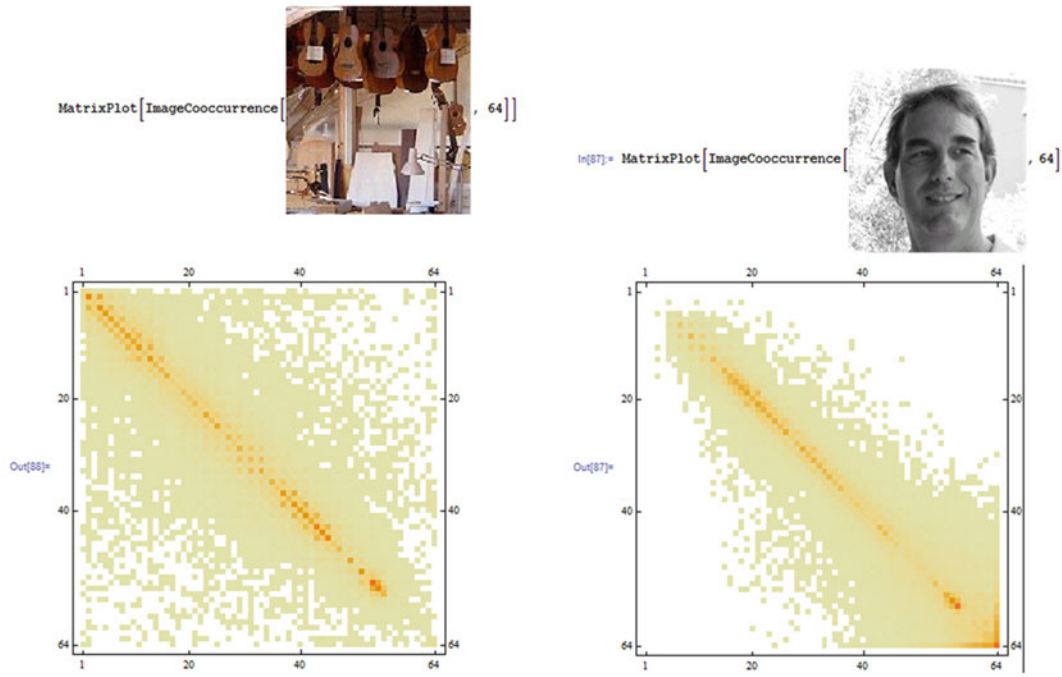


Figure 3.7 Pair of image co-occurrence matrix plots (x-axis plots) computed over 64 bins in the *bottom row* corresponding to the images in the *top row*

- **Extended SDM texture metrics**: The addition of 12 other useful statistical measures to those proposed by Haralick.
- **Some code snippets**: These illustrate the extended SDM computations, full source code is shown in Appendix D.

In Fig. 3.7, several of the extended SDM metrics can be easily seen, including containment and locus mean density. Note that the right image does not have a lot of outliner intensity points or noise (good containment); most of the energy is centered along the diagonal (tight locus), showing a rather smooth set of image pixel transitions and texture, while the left image shows a wider range of intensity values. For some images, wider range may be noise spread across the spectrum (poor containment), revealing a wider band of energy and contrast between adjacent pixels.

Metric 1: Centroid

To compute the centroid, for each SDM bin $p(i,j)$, the count of the bin is multiplied by the bin coordinate for x,y and also the total bin count is summed. The centroid calculation is weighted to compute the centroid based on the actual bin counts, rather than an unweighted "binary" approach of determining the center of the binning region based on only bin data presence. The result is the weighted center of mass over the SDM bins.

$$\text{centroid} = \sum_{i=0}^{n}\sum_{j=0}^{m}\begin{pmatrix} x = jp(i,j) \\ y = ip(i,j) \\ z = p(i,j) \end{pmatrix}$$

$$\text{centroid}_y = \frac{y}{z}$$

$$\text{centroid}_x = \frac{x}{z}$$

Metric 2: Total Coverage

This is a measure of the spread, or range of distribution, of the binning. A small coverage percentage would be indicative of an image with few gray levels, which corresponds in some cases to image smoothness. For example, a random image would have a very large coverage number, since all or most of the SDM bins would be hit. The coverage feature metrics (2, 3, 4), taken together with the linearity features suggested below (11, 12), can give an indication of image smoothness.

$$\text{coverage}_c = \sum_{i=0}^{n}\sum_{j=0}^{m}\begin{pmatrix} 1 & \text{if } 0 < p(i,j), \\ 0 & \text{otherwise} \end{pmatrix}$$

$$\text{coverage}_t = \frac{\text{coverage}_c}{(n * m)}$$

Metric 3: Low-Frequency Coverage

For many images, any bins in the SDM with bin counts less than a threshold value, such as 3, may be considered as noise. The low-frequency coverage metric, or noise metric, provides an idea how much of the binning is in this range. This may be especially true as the sample area of the image area increases. For whole images, a threshold of 3 has proved to be useful for determining if a bin contains noise for a data range of 0-255, and using the SDM over smaller local kernel regions may use all the values with no thresholding needed.

$$\text{coverage}_c = \sum_{i=0}^{n}\sum_{j=0}^{m}\text{if } 0 < p\,(i,j) < 3\begin{pmatrix} 1, \\ \text{else } 0 \end{pmatrix}$$

$$\text{coverage}_l = \frac{\text{coverage}_c}{(n*m)}$$

Metric 4: Corrected Coverage

Corrected coverage is the total coverage with noise removed.

$$\text{coverage}_n = \text{coverage}_t - \text{coverage}_l$$

Metric 5: Total Power

The power metric provides a measure of the swing in value between adjacent pixels in an image, and is computed in four directions. A smooth image will have a low power number because the differences between pixels are smaller. Total power and relative power are inter-related, and relative power is computed using the total populated bins (z) and total difference power (t).

$$\text{power}_c = \sum_{i=0}^{n}\sum_{j=0}^{m} \text{if } p(i,j) \neq 0 \begin{pmatrix} z+ = 1, \\ t+ = |i-j| \end{pmatrix}$$

$$\text{power}_t = t$$

Metric 6: Relative Power

The relative power is calculated based on the scaled total power using nonempty SDM bins t, while the total power uses all bins.

$$\text{power}_r = \frac{t}{z}$$

Metric 7: Locus Mean Density

For many images, there is a "locus" area of high-intensity binning surrounding the bin axis (locus axis is where adjacent pixels are of the same value $x = y$) corresponding to a diagonal line drawn from the upper left corner of the SDM plot. The degree of clustering around the locus area indicates the amount of smoothness in the image. Binning from a noisy image will be scattered with little relation to the locus area, while a cleaner image will show a pattern centered about the locus.

$$\text{locus}_c = \sum_{i=0}^{n}\sum_{j=0}^{m} \text{if } \; 0 < |i-j| < 7 \begin{pmatrix} z+ = 1, \\ d+ = p(i,j) \end{pmatrix}$$

$$\text{locus}_d = \frac{d}{z}$$

The locus mean density is an average of the bin values within the locus area. The locus is the area around the center diagonal line, within a band of 7 pixels on either side of the identity line ($x = y$) that passes down the center of each SDM. However, the number 7 is not particularly special, but based upon experience, it just gives a good indication of the desired feature over whole images. This feature is good for indicating smoothness.

Metric 8: Locus Length

The locus length measures the range of the locus concentration about the diagonal. The algorithm for locus length is a simple count of bins populated in the locus area; a threshold band of 7 pixels about the locus has been found useful.

```
y = length = 0;
while (y < 256) {
    x = count = 0;
    while (x < 256) {
       n = |y-x|;
       if (p[i,j] == 0) && (n < 7) count++;
       x++;
    }
    if (!count) length++;
    y++;
}
```

Metric 9: Bin Mean Density

This is simply the average bin count from nonempty bins.

$$\text{density}_c = \sum_{i=0}^{n} \sum_{j=0}^{m} \text{if } p(i,j) \neq 0 \; (v = p(i,j),\; z+ = 1)$$

$$\text{density}_b = \frac{v}{z}$$

Metric 10: Containment

Containment is a measure of how well the binning in the SDM is contained within the boundaries or edges of the SDM, and there are four edges or boundaries, for example assuming a data range [0. . .255], there are containment boundaries along rows 0 and 255, and along columns 0 and 255. Typically, the bin count m is 256 bins, or possibly less such as 64. To measure containment, basically the perimeters of the SDM bins are checked to see if any binning has occurred, where the perimeter region bins of the SDM represent extrema values next to some other value. The left image in Fig. 3.7 has lower containment than the right image, especially for the low values.

$$\text{containment}_1 = \sum_{i=0}^{m} \text{if } p(i,0) \neq 0 \; (c_1+ = 1)$$

$$\text{containment}_2 = \sum_{i=0}^{m} \text{if } p(i,m) \neq 0 \; (c_2+ = 1)$$

$$\text{containment}_3 = \sum_{i=0}^{m} \text{if } p(0,i) \neq 0 \; (c_3+ = 1)$$

$$\text{containment}_4 = \sum_{i=0}^{m} \text{if } p(m,i) \neq 0 \; (c_4+ = 1)$$

$$\text{containment}_t = 1.0 - \frac{(c_1 + c_2 + c_3 + c_4)}{4m}$$

If extrema are hit frequently, this probably indicates some sort of overflow condition such as numerical overflow, sensor saturation, or noise. The binning is treated unweighted. A high containment number indicates that all the binning took place within the boundaries of the SDM. A lower number indicates some bleeding. This feature appears visually very well in the SDM plots.

Metric 11: Linearity

The linearity characteristic may only be visible in a single orientation of the SDM, or by comparing SDMs. For example, the image in Fig. 3.8 reveals some linearity variations across the set of SDMs. This is consistent with the image sensor used (older tube camera).

$$\text{linearity}_c = \sum_{j=0}^{m} \text{if } p(jm,j) > 1 \begin{pmatrix} z+ = 1, \\ l+ = p(256j,j) \end{pmatrix}$$

$$\text{linearity}_{\text{normalized}} = \frac{z}{m}$$

$$\text{linearity}_{\text{strength}} = \frac{l}{z} * m^{-1}$$

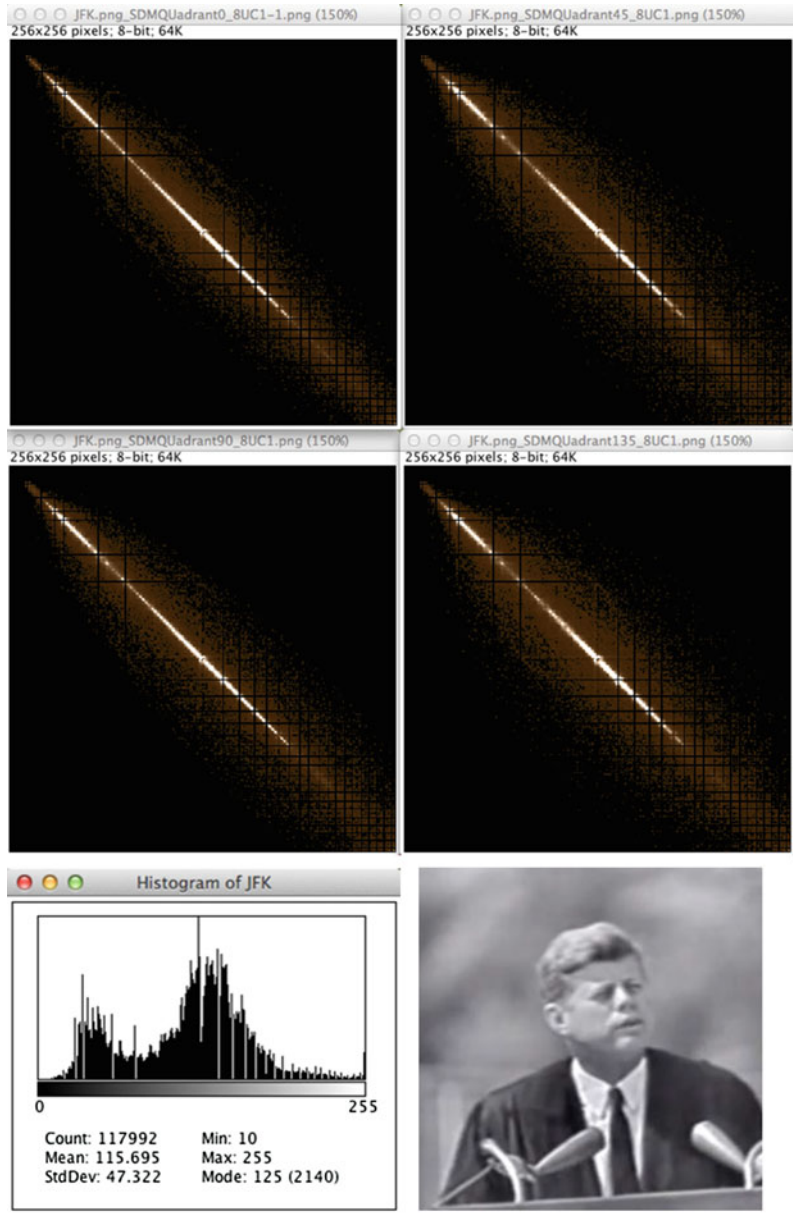


Figure 3.8 SDMs from old tube camera showing linearity variations in the sensor, includes full set of 0°, 45°, 90°, and 135° SDMs. (Public domain image from National Archives)

Metric 12: Linearity Strength

The algorithm for linearity strength is shown in Metric 11. If there is any linearity present in a given angle of SDM, both linearity strength and linearity will be comparatively higher at this angle than the other SDM angles (Table 3.1).

Table 3.1 Extended SDM metrics from Fig. 3.8

METRIC	0 Deg	45 Deg	90 Deg.	135 Deg.	Ave.
xcentroid	115	115	115	115	115
ycentroid	115	115	115	115	115
low_frequency_coverage	0.075	0.092	0.103	0.108	0.095
total_coverage	0.831	0.818	0.781	0.780	0.803
corrected_coverage	0.755	0.726	0.678	0.672	0.708
total_power	2.000	3.000	5.000	5.000	3.750
relative_power	17.000	19.000	23.000	23.000	20.500
locus_length	71	72	71	70	71
locus_mean_density	79	80	74	76	77
bin_mean_density	21	19	16	16	18
containment	0.961	0.932	0.926	0.912	0.933
linearity	0.867	0.848	0.848	0.848	0.853
linearity_strength	1.526	1.557	0.973	1.046	1.276

Laws Texture Metrics

The Laws metrics [24, 27–29] provide a structural approach to texture analysis, using a set of masking kernels to measure texture energy or variation within fixed sized local regions, similar to the 2 × 2 region SDM approach but using larger pixel areas to achieve different metrics.

The basic Laws algorithm involves classifying each pixel in the image into texture based on local energy, using a few basic steps:

1. The mean average intensity from each kernel neighborhood is subtracted from each pixel to compensate for illumination variations.
2. The image is convolved at each pixel using a set of kernels, each of which sums to zero, followed by summing the results to obtain the absolute average value over each kernel window.
3. The difference between the convolved image and the original image is measured, revealing the Laws energy metrics.

Laws defines a set of nine separable kernels to produce a set of texture region energy metrics, and some of the kernels work better than others in practice. The kernels are composed via matrix multiplication from a set of four vector masks L5, E5, S5, and R5, described below. The kernels were originally defined as 5 × 5 masks, but 3 × 3 approximations have been used also, as shown below.

5 × 5 form

L5	Level Detector	[1	4	6	4	1]
E5	Edge Detector	[-1	-2	0	2	1]
S5	Spot Detector	[-1	0	2	0	1]
R5	Ripple Detector	[1	-4	6	-4	1]

3×3 approximations of 5×5 form

L3	Level Detector	[1 2 1]
E3	Edge Detector	[-1 0 1]
S3	Spot Detector	[-1 2 -1]
R3	Ripple Detector	[*NOTE: cannot be reproduced in 3x3 form]

$$\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} * [1,2,1] = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

E3L3 $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$	E3S3 $\begin{pmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{pmatrix}$	L3S3 $\begin{pmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{pmatrix}$
E5L5 $\begin{pmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{pmatrix}$	E5S5 $\begin{pmatrix} 1 & 2 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -4 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & -2 & -1 \end{pmatrix}$	L5S5 $\begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -4 & -6 & -4 & -1 \end{pmatrix}$

Figure 3.9 L3E3 kernel composition example

To create 2D masks, vectors *Ln*, *En*, *Sn*, *and Rn* (as shown above) are convolved together as separable pairs into kernels; a few examples are shown in Fig. 3.9.

Note that Laws texture metrics have been extended into 3D for volumetric texture analysis [43, 44].

LBP Local Binary Patterns

In contrast to the various structural and statistical methods of texture analysis, the LBP operator [18, 50] computes the local texture around each region as an LBP binary code, or *micro-texture*, allowing simple micro-texture comparisons to segment regions based on like micro-texture. (See the very detailed discussion on LBP in Chap. 6 for details and references to the literature, and especially Fig. 6.6.) The LBP operator [165] is quite versatile, easy to compute, consumes a low amount of memory, and can be used for texture analysis, interest points, and feature description. As a result, the LBP operator is discussed is several places in this book.

As shown in Fig. 3.10, the uniform set of LBP operators, composed of a subset of the possible LBPs that are by themselves rotation invariant, can be binned into a histogram, and the corresponding bin values are run through an FFT as a 1D array to create an FFT spectrum, which yields a robust metric with strong rotational invariance.

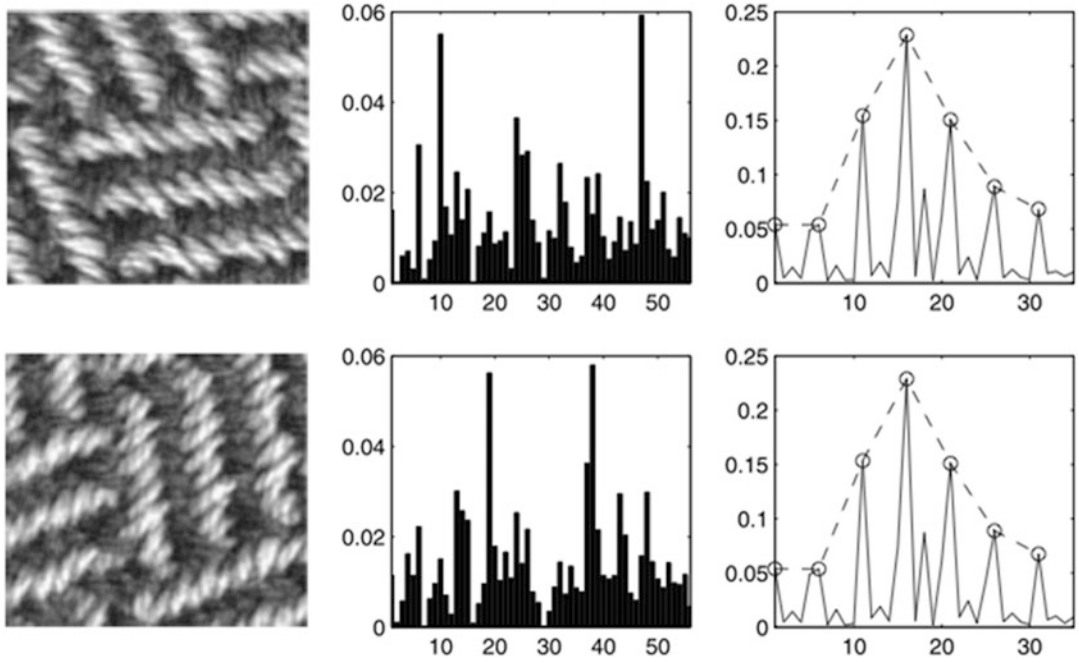


Figure 3.10 (*Left*) texture images. (*Center*) LBP histograms. (*Right*) FFT spectrum plots of the histograms which reveal the rotational invariance property of the LBP histograms. Note that while the histogram binning looks different for the rotated images, the FFT spectrums look almost identical. (Image © Springer-Verlag London Limited from Computer Vision Using Local Binary Patterns)

Dynamic Textures

Dynamic textures are a concept used to describe and track textured regions as they change and morph dynamically from frame to frame [13–15, 45] For example, dynamic textures may be textures in motion, like sea waves, smoke, foliage blowing in the wind, fire, facial expressions, gestures, and poses. The changes are typically tracked in spatiotemporal sets of image frames, where the consecutive frames are stacked into volumes for analysis as a group. The three dimensions are the *XY* frame sizes, and the *Z* dimension is derived from the stack of consecutive frames $n - 2$, $n - 1$, n.

A close cousin to dynamic texture research is the field of *activity recognition* (discussed in Chap. 6), where features are parts of moving objects that compose an activity—for example, features on arms and legs that are tracked frame to frame to determine the type of motion or activity, such as walking or running. One similarity between activity recognition and dynamic textures is that the features or textures change from frame to frame over time, so for both activity recognition and dynamic texture analysis, tracking features and textures often requires a spatiotemporal approach involving a data structure with a history buffer of past and current frames, which provides a volumetric representation to the data.

For example, VLBP and LBP-TOP (discussed in Chap. 6) provide methods for dynamic texture analysis by using the LBP constructed to operate over three dimensions in a volumetric structure, where the volume contains image frames $n - 2$, $n - 1$, *and* n stacked into the volume.

Statistical Region Metrics

Describing texture in terms of statistical metrics of the pixels is a common and intuitive method. Often a simple histogram of a region will be sufficient to describe the texture well enough for many applications. There are also many variations of the histogram, which lend themselves to a wide range of texture analysis. So this is a good point at which to examine histogram methods. Since statistical mathematics is a vast field, we can only introduce the topic here, dividing the discussion into image moment features and point metric features.

Image Moment Features

Image moments [4, 500] are scalar quantities, analogous to the familiar statistical measures such as mean, variance, skew, and kurtosis. Moments are well suited to describe polygon shape features and general feature metric information such as gradient distributions. Image moments can be based on either scalar point values or basis functions such as Fourier or Zernike methods discussed later in the section on basis space.

Moments can describe the projection of a function onto a *basis space*—for example, the Fourier transform projects a function onto a basis of harmonic functions. Note that there is a conceptual relationship between 1D and 2D moments in the context of shape description. For example, the 1D mean corresponds to the 2D centroid, and the 1D minimum and maximum correspond to the 2D major and minor axis. The 1D minimum and maximum also correspond to the 2D bounding box around the 2D polygon shape (also see Fig. 6.29).

In this work, we classify image moments under the term *polygon shape descriptors* in the taxonomy (see Chap. 5). Details on several image moments used for 2D shape description are covered in Chap. 6, under "Object Shape Metrics for Blobs and Objects."

Common properties of moments in the context of 1D distributions and 2D images include:

- Zeroth order moment is the mean or 2D centroid.
- Central moments describe variation around the mean or 2D centroid.
- First order central moments contain information about 2D area, centroid, and size.
- Second order central moments are related to variance and measure 2D elliptical shape.
- Third order central moments provide symmetry information about the 2D shape, or skewness.
- Fourth order central moments measure 2D distribution as tall, short, thin, short, or fat.
- Higher-level moments may be devised and composed of moment ratios, such as co-variance.

Moments can be used to create feature descriptors that are invariant to several robustness criteria, such as scale, rotation, and affine variations. The taxonomy of robustness and invariance criteria is provided in Chap. 5. For 2D shape description, in 1961 Hu developed a theoretical set of seven 2D planar moments for character recognition work, derived using invariant algebra, that are invariant under scale, translation, and rotation [7]. Several researchers have extended Hu's work. An excellent resource for this topic is *Moments and Moment Invariants in Pattern Recognition*, by Jan Flusser et al. [500].

Point Metric Features

Point metrics can be used for the following: (1) feature description, (2) analysis and visualization, (3) thresholding and segmentation, and (4) image processing via programmable LUT functions (discussed in Chap. 2). Point metrics are often overlooked. Using point metrics to understand the structure of the image data is one of the first necessary steps toward devising the image preprocessing pipeline to prepare images for feature analysis. Again, the place to start is by analysis of the histogram, as shown in Figs. 3.1 and 3.11. The basic point metrics can be determined visually, such as minima, maxima, peaks, and valleys. False coloring of the histogram regions for data visualization is simple using color lookup tables to color the histogram regions in the images.

Here is a summary of statistical point metrics:

- **Quantiles, median, rescale**: By sorting the pixel values into an ordered list, as during the histogram process, the various quartiles can be found, including the median value. Also, the pixels can be rescaled from the list and used for pixel remap functions (as described in Chap. 2).

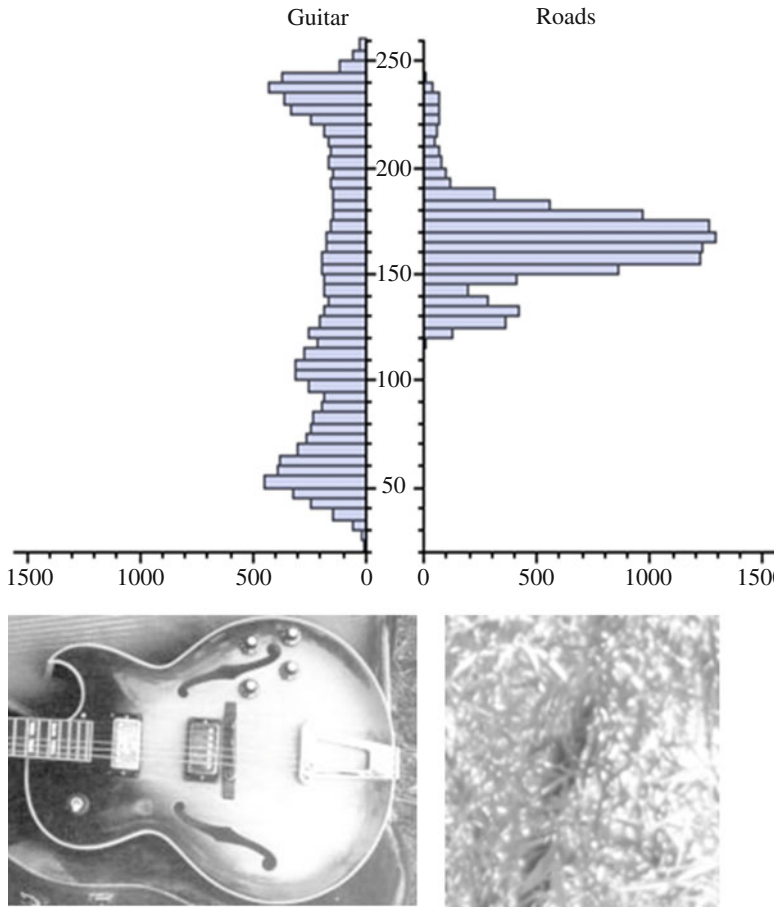


Figure 3.11 Two image histograms side by side, for analysis

- **Mix, max, mode**: The minimum and maximum values, together with histogram analysis, can be used to guide image preprocessing to devise a threshold method to remove outliers from the data. The mode is the most common pixel value in the sorted list of pixels.
- **Mean, harmonic mean, and geometric mean**: Various formulations of the mean are useful to learn the predominant illumination levels, dark or light, to guide image preprocessing to enhance the image for further analysis.
- **Standard deviation, skewness, and kurtosis**: These moments can be visualized by looking at the SDM plots.
- **Correlation**: Topic was covered earlier in this chapter under cross-correlation and auto-correlation.
- **Variance, covariance**: The variance metric provides information on pixel distribution, and covariance can be used to compare variance between two images. Variance can be visualized to a degree in the SDM, also as shown in this chapter.
- **Ratios and multivariate metrics**: Point metrics by themselves may be useful, but multivariate combinations or ratios using simple point metrics can be very useful as well. Depending on the application, the ratios themselves form key attributes of feature descriptors (as described in Chap. 6). For example, mean: min, mean: max, median: mean, area: perimeter.

Global Histograms

Global histograms treat the entire image. In many cases, image matching via global histograms is simple and effective, using a distance function such as SSD. As shown in Fig. 3.12, histograms reveal quantitative information on pixel intensity, but not structural information. All the pixels in the region

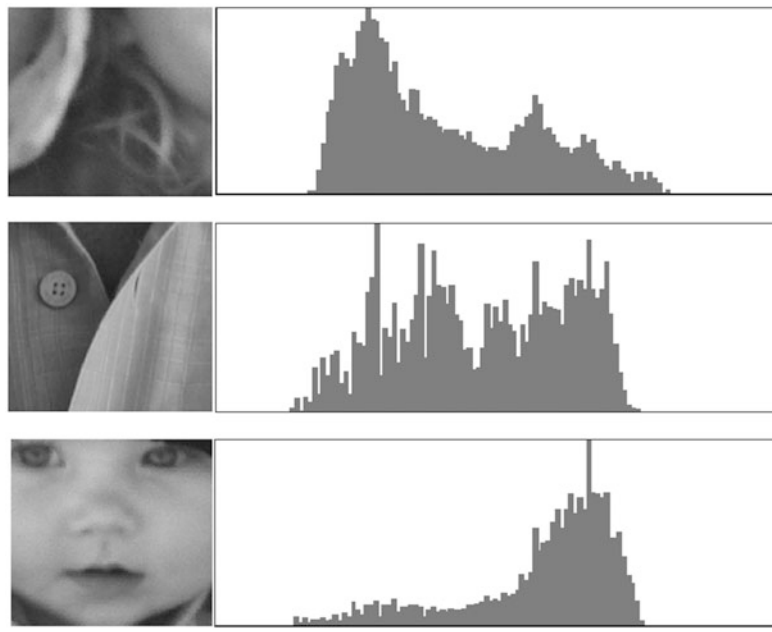


Figure 3.12 2D histogram shapes for different images

contribute to the histogram, with no respect to the distance from any specific point or feature. As discussed in Chap. 2, the histogram itself is the basis of histogram modification methods, allowing the shape of the histogram to be stretched, compressed, or clipped as needed, and then used as an inverse lookup table to rearrange the image pixel intensity levels.

Local Region Histograms

Histograms can also be computed over *local regions* of pixels, such as rectangles or polygons, as well as over sets of feature attributes, such as gradient direction and magnitude or other spectra. To create a polygon region histogram feature descriptor, first a region may be segmented using morphology to create a mask shape around a region of interest, and then only the masked pixels are used for the histogram.

Local histograms of pixel intensity values can be used as attributes of a feature descriptor, and also used as the basis for remapping pixel values from one histogram shape to another, as discussed in Chap. 2, by reshaping the histogram and reprocessing the image accordingly. Chapter 6 discusses a range of feature descriptors such as SIFT, SURF, and LBP which make use of feature histograms to bin attributes such as gradient magnitude and direction.

Scatter Diagrams, 3D Histograms

The *scatter diagram* can be used to visualize the relationship or similarity between two image datasets for image analysis, pattern recognition, and feature description. Pixel intensity from two images or image regions can be compared in the scatter plot to visualize how well the values correspond. Scatter diagrams can be used for feature and pattern matching under limited translation invariance, but they are less useful for affine, scale, or rotation invariance. Fig. 3.13 shows an example using a scatter diagram to look for a pattern in an image, the target pattern is compared at different offsets, the smaller the offset, the better the correspondence. In general, tighter sets of peak features indicate a strong structural or pattern correspondence; more spreading of the data indicates weaker correspondence. The farther away the pattern offset moves, the lower the correspondence.

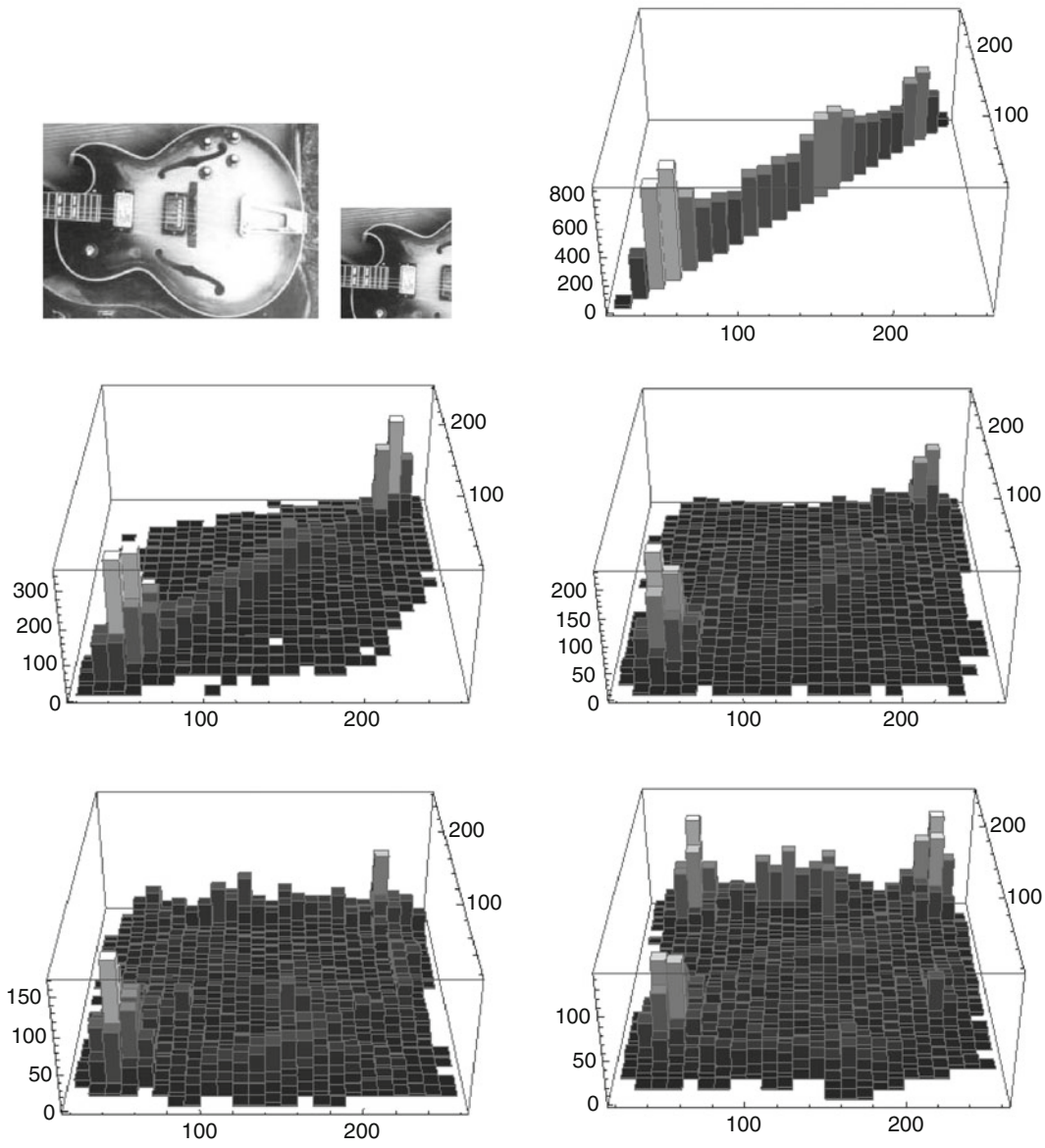


Figure 3.13 Scatter diagrams, rendered as 3D histograms, of an image and a target pattern at various displacements. *Top row*: (*left*) image, (*center*) target pattern from image, (*right*) SDM of pattern with itself. *Center row*: (*left*) target and image offset 1,1 (*right*) target and image offset 8,8. *Bottom row*: (*left*) target and image offset 16,16 (*right*) target and image offset 32,32

Note that by analyzing the peak features compared to the low-frequency features, correspondence can be visualized. Fig. 3.14 shows scatter diagrams from two separate images. The lack of peaks along the axis and the presence of spreading in the data show low structural or pattern correspondence.

The scatter plot can be made, pixel by pixel, from two images, where pixel pairs form the Cartesian coordinate for scatter plotting using the pixel intensity of image 1 is used as the x coordinate, and the pixel intensities of image 2 as the y coordinate, then the count of pixel pair correspondence is binned in the scatter plot. The bin count for each coordinate can be false colored for visualization. Fig. 3.15 provides some code for illustration purposes.

For feature detection, as shown in Fig. 3.12, the scatter plot may reveal enough correspondence at coarse translation steps to reduce the need for image pyramids in some feature detection and pattern matching applications. For example, the step size of the pattern search and compare could be optimized by striding or skipping pixels, searching the image at 8 or 16 pixel intervals, rather than at every pixel, reducing feature detection time. In addition, the scatter plot data could first be thresholded to a binary image, masked to show just the peak values, converted into a bit vector, and measured for correspondence using HAMMING distance for increased performance.

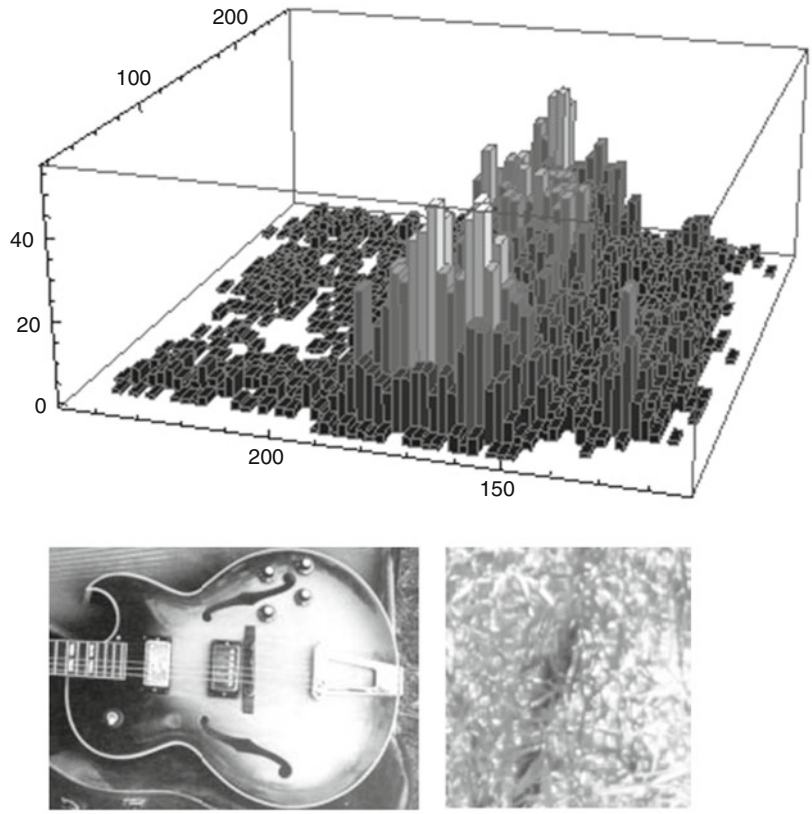


Figure 3.14 Scatter diagram from two different images showing low correspondence along diagonal

```
r1.x = sarea.x;
r1.y = sarea.y;
r1.z = sarea.z;
r1.dx = dx;
r1.dy = 1;
r1.dz = 1;

r2.x = darea.x;
r2.y = darea.y;
r2.z = darea.z;
r2.dx = dx;
r2.dy = 1;
r2.dz = 1;

/* INITIALIZE DATA */
for (x=0; x < 0x10000; mbin[x] = (int)0, x++);

gf = c->grain;
if (gf <= 0) gf = 1;
if (gf > dx) gf = dx;

z=0;
while (z < dz) {
  r1.y = sarea.y;
  r2.y = darea.y;
  y=0;
  while (y < dy) {

    pix_read(c->soid, &r1, data1);
    pix_read(c->doid, &r2, data2);
    for (x=0; x < dx; mbin[ ((data2[x] << 8)&0xff00) + (data1[x] & 0xff) ]++, x += gf);

    y += gf;
    r1.y += gf;
    r2.y += gf;
  }
  z += gf;
  r1.z += gf;
  r2.z += gf;
}
```

Figure 3.15 Code to illustrate binning 8-bit data for a scatter diagram comparing two images pixel by pixel and binning the results for plotting

Multi-resolution, Multi-scale Histograms

Multi-resolution histograms have been used for texture analysis [46], and also for feature recognition [47]. The PHOG descriptor described in Chap. 6 makes use of multi-scale histograms of feature spectra—in this case, gradient information. Note that the multi-resolution histogram provides scale invariance for feature description. For texture analysis [46], multi-resolution histograms are constructed using an image pyramid, and then a histogram is created for each pyramid level and concatenated together [10], which is referred to as a *multi-resolution histogram*. This histogram has the desirable properties of algorithm simplicity, fast computation, low memory requirements, noise tolerance, and high reliability across spatial and rotational variations. See Fig. 3.16. A variation on the pyramid is used in the method of Zhao and Pietikainen [15], employing a multidimensional pyramid image set from a volume.

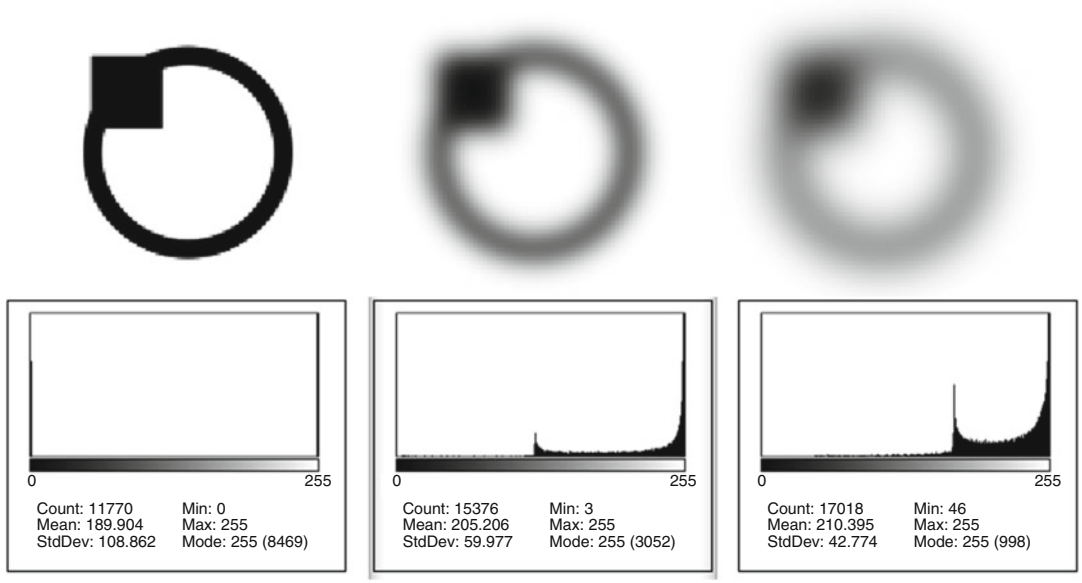


Figure 3.16 Multi-resolution histogram image sequence. Note that the multiple histograms are taken at various Gaussian blur levels in an attempt to create more invariant feature descriptors

Steps involved in creating and using multi-resolution histograms are as follows:

1. Apply Gaussian filter to image.
2. Create an image pyramid.
3. Create histograms at each level.
4. Normalize the histograms using L1 norm.
5. Create cumulative histograms.
6. Create difference histograms or DOG images (differences between pyramid levels).
7. Renormalize histograms using the difference histograms.
8. Create a feature vector from the set of difference histograms.
9. Use L1 norm as distance function for comparisons between histograms.

Radial Histograms

For some applications, computing the histogram using radial samples originating at the shape centroid can be valuable [128, 129]. To do this, a line is cast from the centroid to the perimeter of the shape, and pixel values are recorded along each line and then binned into histograms. See Fig. 3.17.

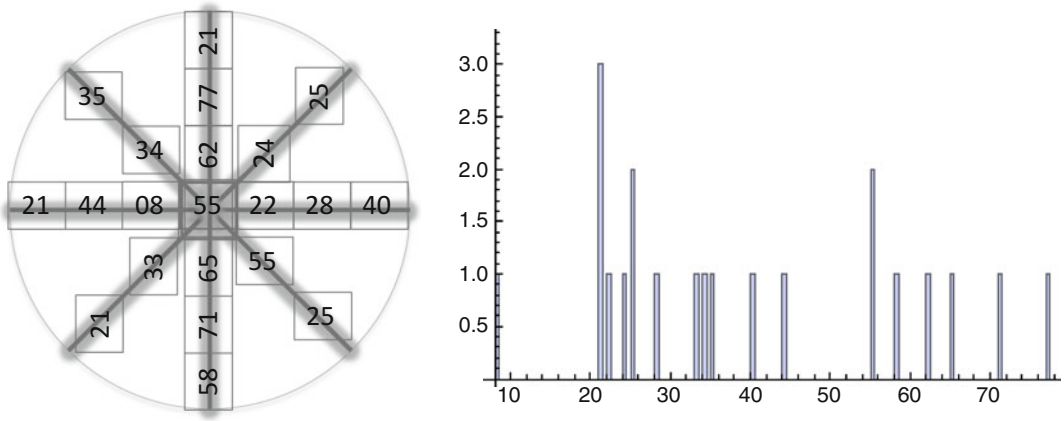


Figure 3.17 Radial histogram illustrations [128, 129]

Contour or Edge Histograms

The perimeter or shape of an object can be the basis of a shape histogram, which includes the pixel values of each point on the perimeter of the object binned into the histogram. Besides recording the actual pixel values along the perimeter, the chain code histogram (CCH) that is discussed in Chap. 6 shows the direction of the perimeter at connected edge point coordinates. Taken together, the CCH and contour histograms provide useful shape information.

Basis Space Metrics

Features can be described in a *basis space*, which involves transforming pixels into an alternative basis and describing features in the chosen basis, such as the frequency domain. What is a basis space and what is a transform? Consider the decimal system, which is base 10, and the binary system which is base 2. We can change numbers between the two number systems by using a transform. A Fourier transform uses sine and cosine as basis functions in frequency space, so that the Fourier transform can move pixels between the time-domain pixel space and the frequency space. Basis space moments describe the projection of a function onto a basis space [500]—for example, the Fourier transform projects a function onto a basis of harmonic functions.

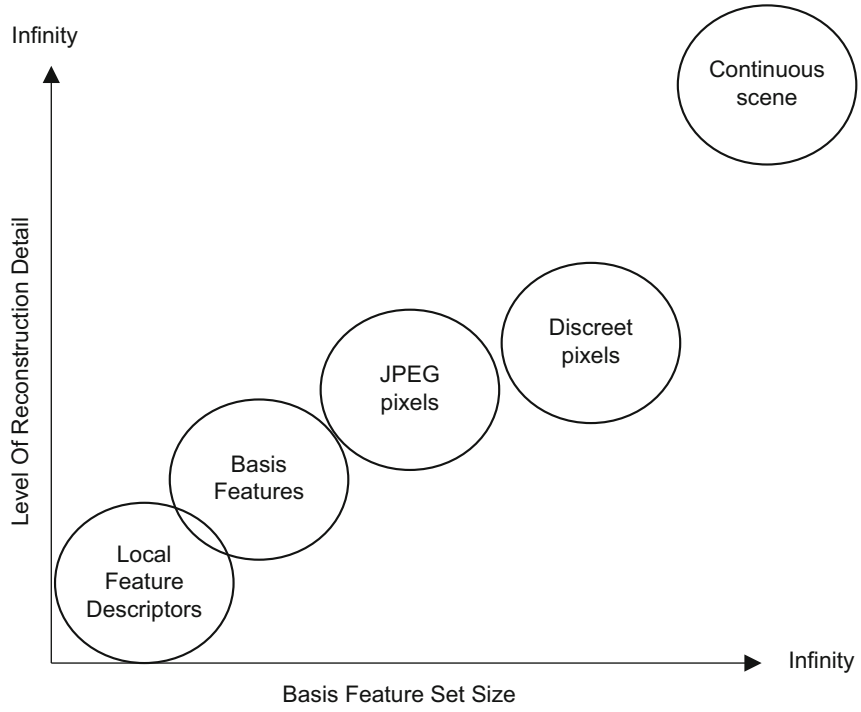


Figure 3.18 An oversimplified spectrum of basis space options, showing feature set size and complexity of description and reconstruction

Basis spaces and transforms are useful for a wide range of applications, including image coding and reconstruction, image processing, feature description, and feature matching. As shown in Fig. 3.18, image representation and image coding are closely related to feature description. Images can be described using *coding methods* or *feature descriptors*, and images also can be reconstructed from the encodings or from the feature descriptors. Many methods exist to reconstruct images from alternative basis space encodings, ranging from lossless RLE methods to lossy JPEG methods; in Chap. 4, we provide illustrations of images that have been reconstructed from only local feature descriptors (see Figs. 4.12, 4.13 and 4.14).

As illustrated in Fig. 3.18, a spectrum of basis spaces can be imagined, ranging from a continuous real function or live scene with infinite complexity, to a complete raster image, a JPEG compressed image, a frequency domain, or other basis representations, down to local feature descriptor sets. Note that the more detail that is provided and used from the basis space representation, the better the real scene can be recognized or reconstructed. So the trade-off is to find the best representation or description, in the optimal basis space, to reach the invariance and accuracy goals using the least amount of compute and memory.

Transforms and basis spaces are a vast field within mathematics and signal processing, covered quite well in other works, so here we only introduce common transforms useful for image coding and feature description. We describe their key advantages and applications, and refer the reader to the literature as we go. See Fig. 3.19.

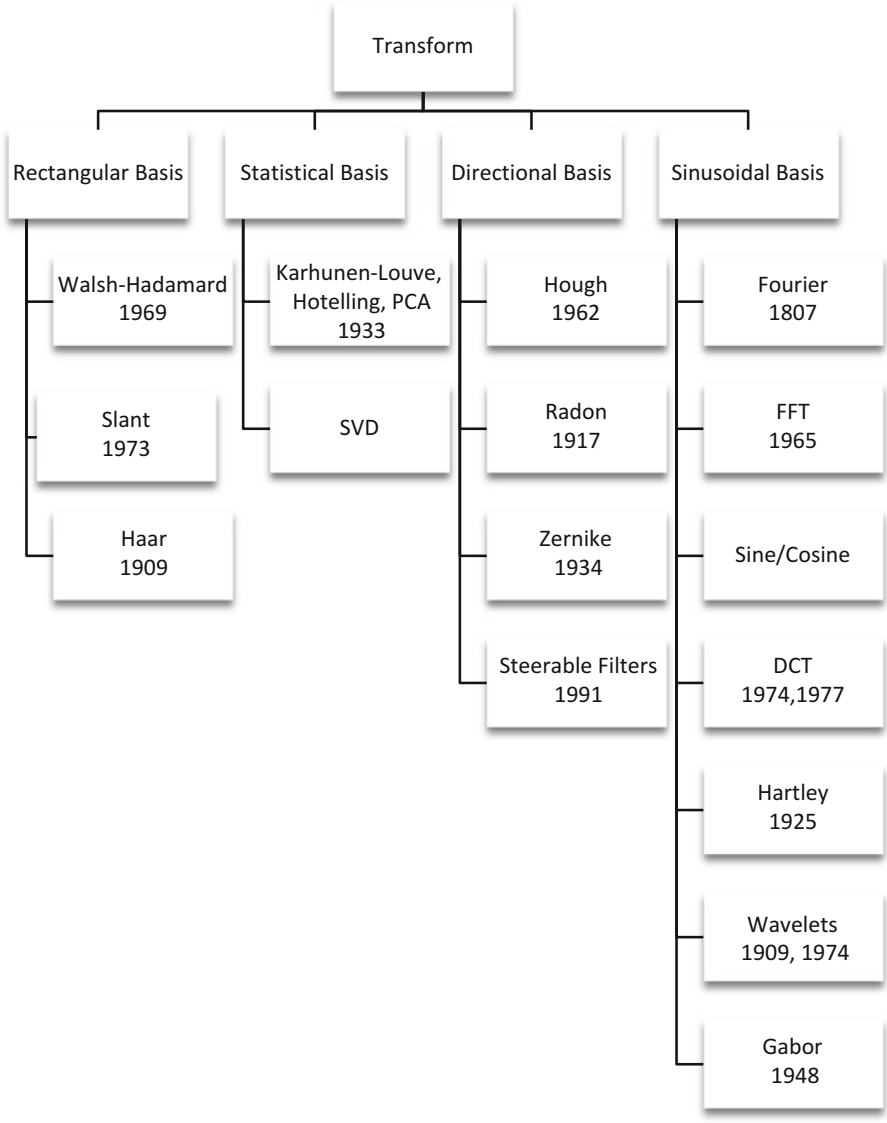


Figure 3.19 Various basis transforms used in image processing and computer vision

Since we are dealing with discrete pixels in computer vision, we are primarily interested in discrete transforms, especially those which can be accelerated with optimized software or fixed-function hardware. However, we also cover a few integral transform methods that may be slower to compute and less used. Here is an overview:

- **Global or local feature description**. It is possible to use transforms and basis space representations of images as a global feature descriptor, allowing scenes and larger objects to be recognized and compared. The 2D FFT spectrum is only one example, and it is simple to compare FFT spectrum features using SAD or SSD distance measures.
- **Image coding and compression**. Many of the transforms have proved valuable for image coding and image compression. The basic method involves transforming the image, or block regions of

the image, into another basis space. For example, transforming blocks of an image into the Fourier domain allows the image regions to be represented as sine and cosine waves. Then, based on the amount of energy in the region, a reduced amount of frequency space components can be stored or coded to represent the image. The energy is mostly contained in the lower-frequency components, which can be observed in the Fourier power spectrum such as shown in Fig. 2.16; the high-frequency components can be discarded and the significant lower-frequency components can be encoded, thus some image compression is achieved with a small loss of detail. Many novel image coding methods exist, such as that using a basis of scaled Laplacian features over an image pyramid [310].

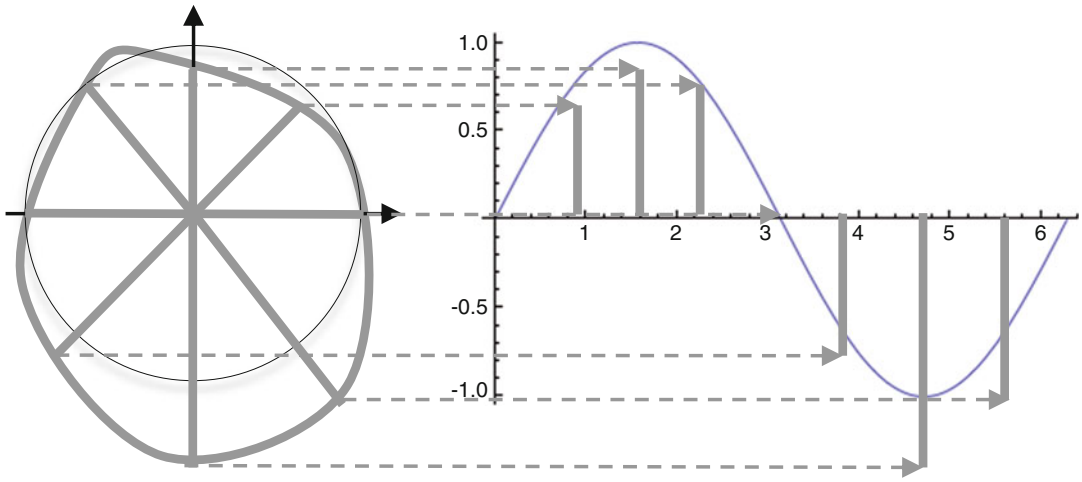


Figure 3.20 Fourier descriptor of the odd shaped polygon surrounding the circle on the *left*

Fourier Description

The Fourier family of transforms was covered in detail in Chap. 2, in the context of image preprocessing and filtering. However, the Fourier frequency components can also be used for feature description. Using the forward Fourier transform, an image is transformed into frequency components, which can be selectively used to describe the transformed pixel region, commonly done for image coding and compression, and for feature description.

The Fourier descriptor provides several invariance attributes, such as rotation and scale. Any array of values can be fed to an FFT to generate a descriptor—for example, a histogram. A common application is illustrated in Fig. 3.20, describing the circularity of a shape and finding the major and minor axis as the extrema frequency deviation from the sine wave. A related application is finding the endpoints of a flat line segment on the perimeter by fitting FFT magnitude's of the harmonic series as polar coordinates against a straight line in Cartesian space.

In Fig. 3.20, a complex wave is plotted as a dark gray circle unrolled around a sine wave function or a perfect circle. Note that the Fourier transform of the lengths of each point around the complex function yields an approximation of a periodic wave, and the Fourier descriptor of the shape of the complex wave is visible. Another example illustrating Fourier descriptors is shown in Fig. 6.29.

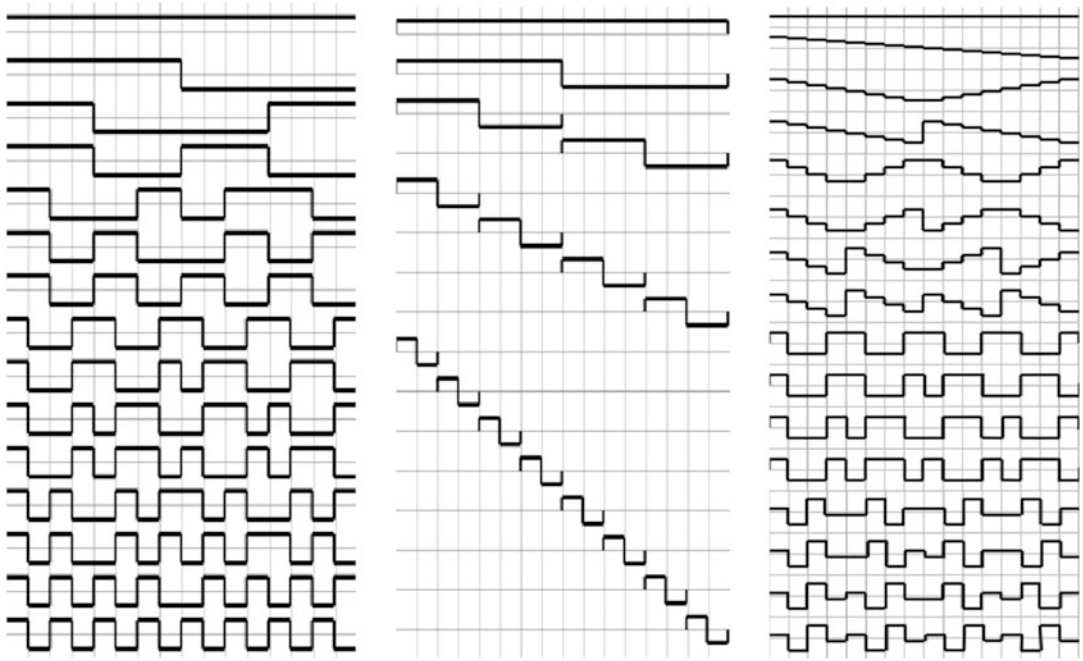


Figure 3.21 (*Left*) Walsh Haramaard basis set. (*Center*) HAAR basis set. (*Right*) Slant basis set

Walsh–Hadamard Transform

The Hadamard transform [4, 9] uses a series of square waves with the value of +1 or −1, which is ideal for digital signal processing. It is amenable to optimizations, since only signed addition is needed to sum the basis vectors, making this transform much faster than sinusoidal basis transforms. The basis vectors for the harmonic Hadamard series and corresponding transform can be generated by sampling Walsh functions, which make up an orthonormal basis set; thus, the combined method is commonly referred to as the Walsh–Hadamaard transform; see Fig. 3.21.

HAAR Transform

The HAAR transform [4, 9] is similar to the Fourier transform, except that the basis vectors are HAAR features resembling square waves, and similar to wavelets. HAAR features, owing to their orthogonal rectangular shapes, are suitable for detecting vertical and horizontal images features that have near- constant gray level. Any structural discontinuities in the data, such as edges and local texture, cannot be resolved very well by the HAAR features; see Figs. 3.21 and 6.21.

Slant Transform

The Slant transform [276], as illustrated in Fig. 3.21, was originally developed for television signal encoding, and was later applied to general image coding [4, 275]. The Slant transform is analogous to

the Fourier transform, except that the basis functions are a series of slant, sawtooth, or triangle waves. The slant basis vector is suitable for applications where image brightness changes linearly over the length of the function. The slant transform is amenable to discrete optimizations in digital systems. Although the primary applications have been image coding and image compression, the slant transform is amenable to feature description. It is closely related to the Karhunen–Loeve transform and the Slant–Hadamard transform [494].

Zernike Polynomials

Fritz Zernike, 1953 Nobel Prize winner, devised Zernike polynomials during his quest to develop the phase contrast microscope, while studying the optical properties and spectra of diffraction gratings. The Zernike polynomials [264–266] have been widely used for optical analysis and modeling of the human visual system, and for assistance in medical procedures such as laser surgery. They provide an accurate model of optical wave aberrations expressed as a set of basis polynomials, illustrated in Fig. 3.22.

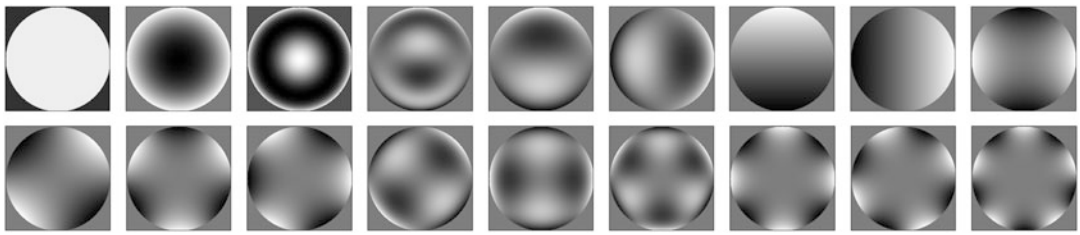


Figure 3.22 The first 18 Zernike modes. Note various aberrations from a perfect filter; *top left image* is the perfect filter. (Images © Dr. Thomas Salmon at Northeastern State University and used by permission)

Zernike polynomials are analogous to steerable filters [370], which also contain oriented basis sets of filter shapes used to identify oriented features and take moments to create descriptors. The Zernike model uses radial coordinates and circular regions, rather than rectangular patches as used in many other feature description methods.

Zernike methods are widely used in optometry to model human eye aberrations. Zernike moments are also used for image watermarking [270] and image coding and reconstruction [271, 273]. The Zernike features provide scale and rotational invariance, in part due to the radial coordinate symmetry and increasing level of detail possible within the higher-order polynomials. Zernike moments are used in computer vision applications by comparing the Zernike basis features against circular patches in target images [268, 269].

Fast methods to compute the Zernike polynomials and moments exist [267, 272, 274], which exploit the symmetry of the basis functions around the x and y axes to reduce computations, and also to exploit recursion.

Steerable Filters

Steerable filters are loosely considered as basis functions here, and can be used for both filtering or feature description. Conceptually similar to Zernike polynomials, steerable filters [370, 382] are composed by synthesizing steered or oriented linearly combinations of chosen basis functions, such as quadrature pairs of Gaussian filters and oriented versions of each function, in a simple transform.

Many types of filter functions can be used as the basis for steerable filters [371, 373]. The filter transform is created by combining together the basis functions in a filter bank, as shown in Fig. 3.23. Gain is selected for each function, and all filters in the bank are summed, then adaptively applied to the image. Pyramid sets of basis functions can be created to operate over scale. Applications include convolving oriented steerable filters with target image regions to determine filter response strength, orientation and phase. Other applications include filtering images based on orientation of features, contour detection, and feature description.

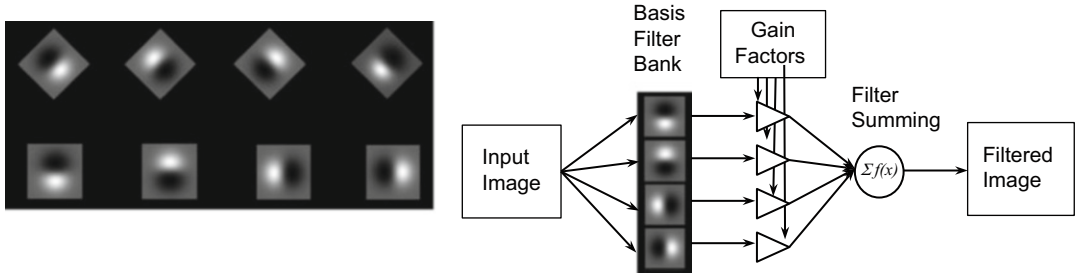


Figure 3.23 (*Left*) Steerable filters basis set showing eight orientations of the first-order Gaussian filter. (*Right*) How steerable filters can be combined for directional filtering. Filter images generated using ImageJ Fiji SteerableJ plugin from Design of Steerable Filters for Feature Detection Using Canny-Like Criteria, M. Jacob, M. Unser, PAMI 2004

For feature description, there are several methods that could work—for example, convolving each steerable basis function with an image patch. The highest one or two filter responses or moments from all the steerable filters can then be chosen as the set-ordinal feature descriptor, or all the filter responses can be used as a feature descriptor. As an optimization, an interest point can first be determined in the patch, and the orientation of the interest point can be used to select the one or two steerable filters closest to the orientation of the interest point; then the closest steerable filers are used as the basis to compute the descriptor.

Karhunen–Loeve Transform and Hotelling Transform

The Karhunen–Loeve transform (KLT)[4, 9] was devised to describe a continuous random process as a series expansion, as opposed to the Fourier method of describing periodic signals. Hotelling later devised a discrete equivalent of the KLT using principal components. "KLT" is the most common name referring to both methods.

The basis functions are dependent on the eigenvectors of the underlying image, and computing eigenvectors is a compute-intensive process with no established fast transform known. The KLT is not separable to optimize over image blocks, so the KLT is typically used for PCA on small datasets such as feature vectors used in pattern classification, clustering, and matching.

Wavelet Transform and Gabor Filters

Wavelets, as the name suggests, are short waves or wave-lets [326]. Think of a wavelet as a short-duration pulse such as a seismic tremor, starting and ending at zero, rather than a continuous or resonating wave. Wavelets are convolved with a given signal, such as an image, to find similarity and statistical moments. Wavelets can therefore be implemented like convolution kernels in the spatial domain. See Fig. 3.24.

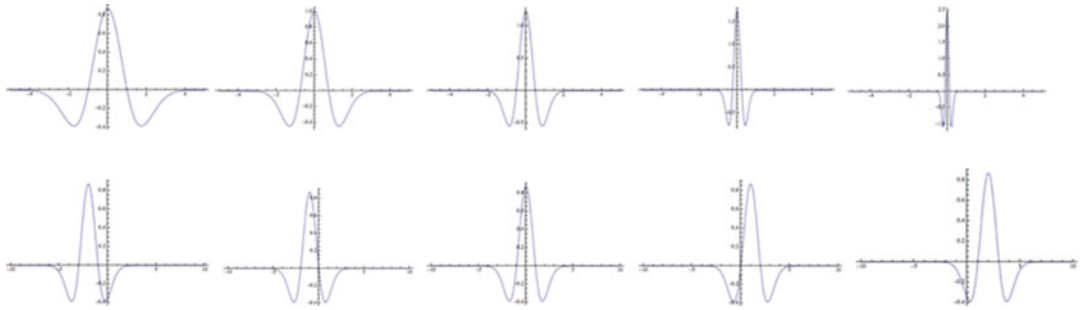


Figure 3.24 Wavelet concepts using a "Mexican top hat" wavelet basis. (*Top*) A few scaled Mexican top hats derived from the mother wavelet. (*Bottom*) A few translated wavelets

Wavelet analysis is a vast field [283, 284] with many applications and useful resources available, including libraries of wavelet families and analysis software packages [281]. Fast wavelet transforms (FWTs) exist in common signal and image processing libraries. Several variants of the wavelet transform include:

- Discrete wavelet transform (DWT)
- Stationary wavelet transform (SWT)
- Continuous wavelet transform (CWT)
- Lifting wavelet transform (LWT)
- Stationary wavelet packet transform (SWPT)
- Discrete wavelet packet transform (DWPT)
- Fractional Fourier transform (FRFT)
- Fractional wavelet transform (FRWT)

Wavelets are designed to meet various goals, and are crafted for specific applications; there is no single wavelet function or basis. For example, a set of wavelets can be designed to represent the musical scale, where each note (such as middle C) is defined as having a duration of an eighth note wavelet pulse, and then each wavelet in the set is convolved across a signal to locate the corresponding notes in the musical scale.

When designing wavelets, the mother wavelet is the basis of the wavelet family, and then daughter wavelets are derived using translation, scaling, or compression of the mother wavelet. Ideally, a set of wavelets are overlapping and complementary so as to decompose data with no gaps and be mathematically reversible.

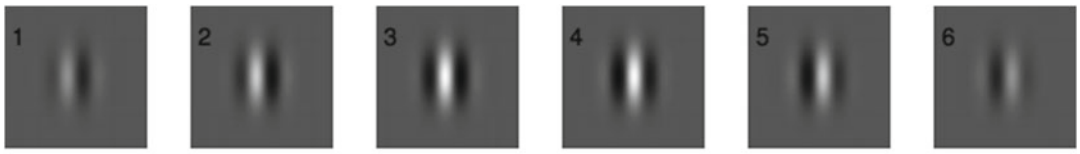
Wavelets are used in transforms as a set of nonlinear basis functions, where each basis function can be designed as needed to optimally match a desired feature in the input function. So, unlike transforms which use a uniform set of basis functions—as the Fourier transform uses sine and cosine functions—wavelets use a dynamic set of basis functions that are complex and nonuniform in nature. See Fig. 3.25.

Wavelets have been used as the basis for scale and rotation invariant feature description [280], image segmentation [277, 278], shape description [279], and obviously image and signal filtering of all the expected varieties, denoising, image compression, and image coding. A set of application-specific wavelets could be devised for feature description.

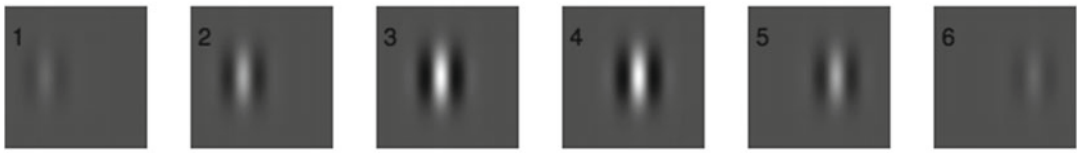
Figure 3.25 Various 2D wavelet shapes: (*left* to *right*) Top hat, Shannon, Dabechies, Smylet, Coiflett

Gabor Functions

Wavelets can be considered an extension of the earlier concept of Gabor functions [285, 325], which can be derived for imaging applications as a set of 2D oriented bandpass filters. Gabor's work was centered on the physical transmission of sound and problems with Fourier methods involving time-varying signals like sirens that could not be perfectly represented as periodic frequency information. Gabor proposed a more compact representation than Fourier analysis could provide, using a concept called *atoms* that recorded coefficients of the sound that could be transmitted more compactly. See Fig. 3.26.



(a) frequency-tuned (static envelope)



(b) velocity-tuned (moving envelope)

Figure 3.26 This figure showing Gabor filters (*top*) frequency tuned, and (*bottom*) velocity tuned. Images © - Springer-Verlag, taken from CVPR 2010, "Facial expression recognition using Gabor motion energy filters, Tingfan Wu, Bartlett, M.S. Movellan, Javier R."

Hough Transform and Radon Transform

The Hough transform [220–222] and the Radon transform [291] are related, and the results are equivalent, in the opinion of many [287, 292]; see Fig. 3.27. The Radon transform is an integral transform, while the Hough transform is a discrete method, therefore much faster. The Hough method is widely used in image processing, and can be accelerated using a GPU [290] with data parallel methods. The Radon algorithm is slightly more accurate and perhaps more mathematically sound, and is often associated with X-ray tomography applied to reconstruction from X-ray projections. We focus primarily on the Hough transform, since it is widely available in image processing libraries.

Key applications for the Hough and Radon transforms are shape detection and shape description of lines, circles, and parametric curves. The main advantages include:

- Robust to noise and partial occlusion
- Fill gaps in apparent lines, edges, and curves
- Can be parameterized to handle various edge and curve shapes

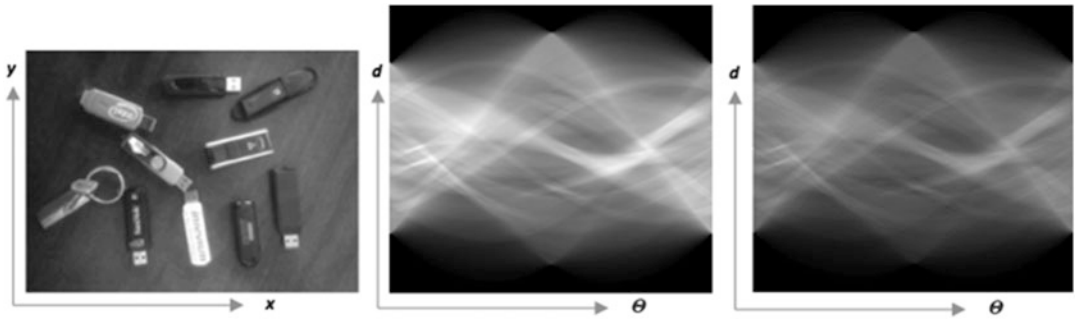


Figure 3.27 Line detection: (*Left*) Original image. (*Center*) Radon Transform. (*Right*) Hough Transform. The brightness of the transform images reveals the relative strength of the accumulators, and the sinusoidal line intersections indicate the angular orientation of features

The disadvantages include:

- Look for one type or parameterization of a feature at a time, such as a line
- Colinear segments are not distinguished and lumped together
- May incorrectly fill in gaps and link edges that are not connected
- Length and position of lines are not determined, but this can be done in image space

The Hough transform is primarily a global or regional descriptor and operates over larger areas. It was originally devised to detect lines, and has been subsequently generalized to detect parametric shapes [293], such as curves and circles. However, adding more parameterization to the feature requires more memory and compute. Hough features can be used to mark region boundaries described by regular parametric curves and lines. The Hough transform is attractive for some applications, since it can tolerate gaps in the lines or curves and is not strongly affected by noise or some occlusion, but morphology and edge detection via other methods is often sufficient, so the Hough transform has limited applications.

The input to the Hough transform is a gradient magnitude image, which has been thresholded, leaving the dominant gradient information. The gradient magnitude is used to build a map revealing all the parameterized features in the image—for example, lines at a given orientation or circles with a given diameter. For example, to detect lines, we map each gradient point in the pixel space into the Hough parameter space, parameterized as a single point $(\boldsymbol{d},\boldsymbol{\theta})$ corresponding to all lines with orientation angle $\boldsymbol{\theta}$ at distance $\boldsymbol{d}$ from the origin. Curve and circle parameterization uses different variables [293]. The parameter space is quantized into cells or accumulator bins, and each accumulator is updated by summing the number of gradient lines passing through the same Hough points. The accumulator method is modified for detecting parametric curves and circles. Thresholding the accumulator space and reprojecting only the highest accumulator values as overlays back onto the image is useful to highlight features.

Summary

This chapter provides a selected history of global and regional metrics, with the treatment of local feature metrics deferred until Chaps. 4 and 6. Some historical context is provided on the development of structural and statistical texture metrics, as well as basis spaces useful for feature description, and several common regional and global metrics. A wide range of topics in texture analysis and statistical analysis are surveyed with applications to computer vision.

Since it is difficult to cleanly partition all the related topics in image processing and computer vision, there is some overlap of topics in here and in Chaps. 2, 4, 5, and 6.

Chapter 3: Learning Assignments

1. Discuss when to use a global image processing operation vs. a local or regional image processing operation.
2. Discuss in general how global image statistics can guide image preprocessing for computer vision applications, and specifically name one global image metric and discuss how it can be applied.
3. Compare global image feature metrics and local feature descriptors in general, and discuss a specific example global feature metric and compare it to a specific local feature descriptor.
4. Describe global image texture in general terms.
5. Discuss how a 2d histogram of an image can be used to understand image texture.
6. Discuss how the 2d Fourier Series of an image is used to understand image texture.
7. Discuss how the Haralick texture metrics based on the co-occurrence matrix are used to understand image texture.
8. Discuss how Spatial Dependency Matrix (SDM) plots are used to understand image texture.
9. Discuss statistical moments of an image histogram, including at least the mean value and variance, and how these features are useful as global image descriptors.
10. Describe a multi-resolution histogram built from an image pyramid, and how to interpret the results of the histogram.
11. Describe how a Fourier description of the shape of a circle is created from the Fourier Series, and how it is useful as a shape descriptor.
12. Describe basis features for the HAAR transform, Slant Transform, and Walsh–Hadamard Transform.
13. Compare Wavelet features to Fourier Series features.
14. Describe the Hough Transform and the Radon Transform algorithms, and how they are used as a global image metric for shape detection.