

Chapter 1

New Approaches for Hierarchical Image Decomposition, Based on IDP, SVD, PCA and KPCA

Roumen Kountchev and Roumiana Kountcheva

Abstract The contemporary forms of image representation vary depending on the application. There are well-known mathematical methods for image representation, which comprise: matrices, vectors, determined orthogonal transforms, multi-resolution pyramids, Principal Component Analysis (PCA) and Independent Component Analysis (ICA), Singular Value Decomposition (SVD), wavelet sub-band decompositions, hierarchical tensor transformations, nonlinear decompositions through hierarchical neural networks, polynomial and multiscale hierarchical decompositions, multidimensional tree-like structures, multi-layer perceptual and cognitive models, statistical models, etc. In this chapter are analyzed the basic methods for hierarchical decomposition of grayscale and color images, and of sequences of correlated images of the kind: medical, multispectral, multi-view, etc. Here is also added one expansion and generalization of the ideas of the authors from their previous publications, regarding the possibilities for the development of new, efficient algorithms for hierarchical image decompositions with various purposes. In this chapter are presented and analyzed the following four new approaches for hierarchical image decomposition: the Branched Inverse Difference Pyramid (BIDP), based on the Inverse Difference Pyramid (IDP); the Hierarchical Singular Value Decomposition (HSVD) with tree-like computational structure; the Hierarchical Adaptive Principle Component Analysis (HAPCA) for groups of correlated images; and the Hierarchical Adaptive Kernel Principal Component Analysis (HAKPCA) for color images. In the chapter are given the algorithms, used for the implementation of these decompositions, and their computational complexity is evaluated. Some experimental results, related to selected applications are also given, and various possibilities for the creation of new hybrid algorithms for hierarchical decomposition of multidimensional images are specified. On the basis

R. Kountchev (✉)

Department of Radio Communications and Video Technologies,
Technical University of Sofia, 8 Kl. Ohridski Blvd., 1000 Sofia, Bulgaria
e-mail: rkountch@tu-sofia.bg

R. Kountcheva

T&K Engineering Co., Drujba 2, Bl. 404/2, 1582 Sofia, Bulgaria
e-mail: kountcheva_r@yahoo.com

of the results obtained from the executed analysis, the basic application areas for efficient image processing are specified, such as: reduction of the information surplus; noise filtration; color segmentation; image retrieval; image fusion; dimensionality reduction for objects classification; search enhancement in large scale image databases, etc.

Keywords Hierarchical image decomposition · Branched inverse difference pyramid · Hierarchical singular value decomposition · Hierarchical principal component analysis for groups of images · Hierarchical adaptive kernel principal component analysis for color images

1.1 Introduction

The methods for image processing, transmission, registration, restoration, analysis and recognition, are defined at high degree by the corresponding mathematical forms and models for their representation. On the other hand, they all depend on the way the image was created, and on their practical use. The primary forms for image representation depend on the used sources, such as: photo and video cameras, scanners, ultrasound sensors, X-ray, computer tomography, etc. The matrix descriptions are related to the *primary discrete forms*. Each still halftone image is represented by one matrix; the color RGB image—by three matrices; the multi-spectral, hyper spectral and multi-view images, and also some kinds of medical images (for example, computer tomography, IMR, etc.)—by N matrices (for $N > 3$), while the moving images are represented through M temporal sequences, of N matrices each. There are already many *secondary forms* created for image representation, obtained from the primary forms, after reduction of the information surplus, and depending on the application. Various mathematical methods are used to transform the image matrices into reduced (secondary) forms by using: vectors, for each image block, through which are composed vector fields; deterministic and statistical orthogonal transforms; multi-resolution pyramids; wavelet sub-band decompositions; hierarchical tensor transforms; nonlinear decompositions through hierarchical neural networks, polynomial and multiscale hierarchical decompositions, multi-dimensional tree-like structures, multi-layer perceptual and cognitive models, statistical models, fuzzy hybrid methods for image decomposition, etc.

The decomposition methods permit each image matrix to be represented as the sum of the matrix components with different weights, defined by the image contents. Besides, the description of each matrix in the decomposition is much simpler than that of the original (primary) matrix. The number of the matrices in the decomposition could be significantly reduced through analyzing their weights, without significant influence on the approximation accuracy of the primary matrix. To this group could be related the methods for linear orthogonal transforms [1]: the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), the Walsh-Hadamard Transform (WHT), the Hartley Transform (HrT), the Haar

Transform (HT), etc.; the pyramidal decompositions [2]: the Gaussian Pyramid (GP), the Laplacean Pyramid (LP), the Discrete Wavelet Transform (DWT), the Discrete Curvelet Transform (DCuT) [3], the Inverse Difference Pyramid (IDP) [4], etc.; the statistical decompositions [5]: the Principal Component Analysis (PCA), the Independent Component Analysis (ICA) and the Singular Value Decomposition (SVD); the polynomial and multiscale hierarchical decompositions [6, 7]; multi-dimensional tree-like structures [8]; hierarchical tensor transformations [9]; the decompositions based on hierarchical neural networks [10]; etc.

The aim of this chapter is to be analyzed the basic methods and algorithms for *hierarchical image decomposition*. Here are also generalized the following new approaches for hierarchical decomposition of multi-component matrix images: the Branched Inverse Difference Pyramid (BIDP), based on the Inverse Difference Pyramid (IDP), the Hierarchical Singular Value Decomposition (HSVD)—for the representation of single images; the Hierarchical Adaptive Principal Component Analysis (HAPCA)—for the decorrelation of sequences of images, and the Hierarchical Adaptive Kernel Principal Component Analysis (HAKPCA)—for the analysis of color images.

1.2 Related Work

One of the contemporary methods for hierarchical image decomposition is called multiscale decomposition [7]. It is used for noise filtration in the image f , represented by the sum of the clean part u , and the noisy part, v . In accordance to Rudin, Osher and Fatemi (ROF) [11], to define the components u and v it is necessary to calculate the total variation of the functional Q , defined by the relation:

$$Q(f, \lambda) = \inf \left\{ \int_{\Omega} |\nabla u| + \lambda \|v\|_{L^2}^2, f = u + v \right\},$$

where $\lambda > 0$ is a scale parameter; and $f \in L^2(\Omega)$ —the image function, defined in the space $L^2(\Omega)$. The minimization of Q leads to decomposition, in result of which the visual information is divided into a part u that extracts the edges of f , and a part v that captures the texture. Denoising at different scales λ generates a multiscale image representation. In [6], Tadmor, Nezzar and Vese proposed a multiscale image decomposition which offers a hierarchical and adaptive representation for different features in the analyzed images. The image is hierarchically decomposed into the sum of simpler atoms u_k , where u_k extracts more refined information from the previous scale u_{k-1} . To this end, the atoms u_k are obtained as dyadically scaled minimizers of the ROF functionals at increasing λ_k scales. Thus, starting with $v_{-1} := f$ and letting v_k denote the residual at a given dyadic scale, $\lambda_k = 2^k$, the recursive step $[u_k, v_k] = \arg\{\inf[Q_T(v_{k-1}, k)]\}$ leads to the desired hierarchical decomposition, $f = \Sigma T(u_k)$ (here T is a blurring operator).

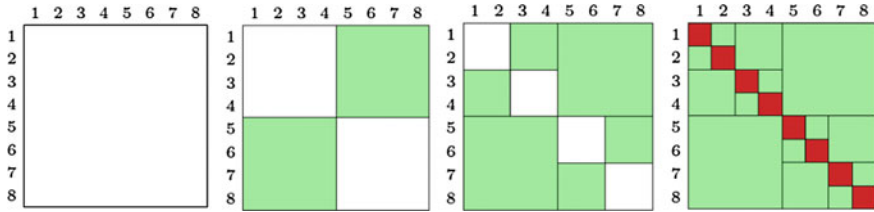


Fig. 1.1 Representation of the matrix of size 8×8 through three hierarchical matrices, or H-matrices

Another well-known approach for hierarchical decomposition is based on the hierarchical matrices [12]. The concept of hierarchical, or H-matrices, is based on the observation that submatrices of a full rank matrix may be of low rank, and respectively—to have low rank approximations. On Fig. 1.1 is given an example for the representation of a matrix of size 8×8 through H-matrices, which contain sub-matrices of three different sizes: 4×4 , 2×2 and 1×1 .

This observation is used for the matrix-skeleton approximation. The inverses of finite element matrices have, under certain assumptions, submatrices with exponentially decaying singular values. This means that these submatrices have also good low rank approximations. The hierarchical matrices permit decomposition by QR or Cholesky algorithms, which are iterative. Unlike them, the new approaches for hierarchical image decomposition, given in this chapter (BIDP and HSVD—for single images, HAPCA—for groups of correlated images, and HAKPCA—for color images), are not based on iterative algorithms.

1.3 Image Representation Based on Branched Inverse Difference Pyramid

1.3.1 Principles for Building the Inverse Difference Pyramid

In this section is given a short description of the inverse difference pyramid, IDP [4, 13], used as a basis for building its modifications. Unlike the famous Gaussian (GP) and Laplacian (LP) pyramids, the IDP represents the image in the spectral domain. After the decomposition, the image energy is concentrated in its first components, which permits to achieve very efficient compression, by cutting off the low-energy components. As a result, the main part of the energy of the original image is retained, despite the limited number of decomposition components used. For the decomposition implementation various kinds of orthogonal transforms could be used. In order to reduce the number of decomposition levels and the computational complexity, the image is initially divided into blocks and for each is then built the corresponding IDP.

In brief, the IDP is executed as follows: At the lowest (initial) level, on the matrix $[B]$ of size $2^n \times 2^n$ is applied the pre-selected “Truncated” Orthogonal Transform (TOT) and are calculated the values of a relatively small number of “retained” coefficients, located in the high-energy area of the so calculated transformed (spectrum) matrix $[S_0]$. These are usually the coefficients with spatial frequencies $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. After Inverse Orthogonal Transform (IOT) of the “truncated” spectrum matrix $[\hat{S}_0]$, which contains the retained coefficients only, is obtained the matrix $[\hat{B}_0]$ for the initial IDP level ($p = 0$), which approximates the matrix $[B]$. The accuracy of the approximation depends on: the positions of the retained coefficients in the matrix $[S_0]$; the values, used to substitute the missing coefficients from the approximating matrix $[\hat{S}_0]$ for the zero level, and on the selected orthogonal transform. In the next decomposition level ($p = 1$), is calculated the difference matrix $[E_0] = [B] - [\hat{B}_0]$. The resulting matrix is then split into 4 sub-matrices of size $2^{n-1} \times 2^{n-1}$ and on each is applied the corresponding TOT. The total number of retained coefficients for level $p = 1$ is 4 times larger than that in the zero level. In case, that Walsh-Hadamard Transform (WHT) is used for this level, the values of coefficients $(0, 0)$ in the IDP decomposition levels 1 and higher are always equal to zero, which permits to reduce the number of retained coefficients with $1/4$. On each of the four spectrum matrices $[\hat{S}_1]$ for the IDP level $p = 1$ is applied IOT and as a result, four sub-matrices are obtained, which build the approximating difference matrix $[\hat{E}_0]$. In the next IDP level ($p = 2$) is calculated the difference matrix $[E_1] = [E_0] - [\hat{E}_0]$. After that, each difference sub-matrix is divided in similar way as in level 1, into four matrices of size $2^{n-2} \times 2^{n-2}$, and for each is performed TOT, etc. In the last (highest) IDP level is obtained the “residual” difference matrix. In case that the image should be losslessly coded, each block of the residual matrix is processed with full orthogonal transform and no coefficients are omitted.

1.3.2 Mathematical Representation of n -Level IDP

The digital image is represented by a matrix of size $(2^n m) \times (2^n m)$. For the processing, the matrix is first divided into blocks of size $2^n \times 2^n$ and on each is applied the IDP decomposition. The matrix $[B(2^n)]$ of each block is represented by the equation:

$$[B(2^n)] = [\hat{B}_0(2^n)] + \sum_{p=1}^r [\hat{E}_{p-1}(2^n)] + [E_r(2^n)] \quad \text{for } r = 1, 2, \dots, n-1. \quad (1.1)$$

Here the number of decomposition components, which are matrices of size $2^n \times 2^n$, is equal to $(r + 2)$. The maximum possible number of decomposition levels for one block is $n + 1$ (for $r = n - 1$). The last component $[E_r(2^n)]$ defines the

approximation error for the block $[B(2^n)]$ for the case, when the decomposition is limited up to level $p = r$. The first component $[\hat{B}_0(2^n)]$ for the level $p = 0$ is the coarse approximation of the block $[B(2^n)]$. It is obtained through 2D IOT on the block $[\hat{S}_0(2^n)]$ in correspondence with the relation:

$$[\hat{B}_0(2^n)] = [T_0(2^n)]^{-1}[\hat{S}_0(2^n)][T_0(2^n)]^{-1} \text{ for } p = 0, \quad (1.2)$$

where $[T_0(2^n)]^{-1}$ is a matrix of size $2^n \times 2^n$, used for the inverse orthogonal transform of $[\hat{S}_0(2^n)]$.

The matrix $[\hat{S}_0(2^n)] = [m_0(u, v)s_0(u, v)]$ is the ‘‘truncated’’ orthogonal transform of the block $[B(2^n)]$. Here $m_0(u, v)$ are the elements of the binary matrix-mask $[M_0(2^n)]$, used to define the retained coefficients of $[\hat{S}_0(2^n)]$ in correspondence to the relation:

$$m_0(u, v) = \begin{cases} 1, & \text{if } s_0(u, v) \text{ is a retained coefficient,} \\ 0 & \text{— otherwise.} \end{cases} \quad (1.3)$$

The values of the elements $m_0(u, v)$ are selected in accordance with the requirement the retained coefficients $\hat{s}_0(u, v) = m_0(u, v)s_0(u, v)$ to be these with maximum energy, calculated for all image blocks. The transform $[S_0(2^n)]$ of the block $[B(2^n)]$ is defined through direct 2D OT:

$$[S_0(2^n)] = [T_0(2^n)][B(2^n)][T_0(2^n)], \quad (1.4)$$

where $[T_0(2^n)]$ is a matrix of size $2^n \times 2^n$ for the decomposition level $p = 0$, used to perform the selected 2D OT, which could be DFT, DCT, WHT, KLT, etc.

The remaining coefficients in the decomposition presented by Eq. 1.1 are the approximating difference matrices $[\hat{E}_{p-1}(2^{n-p})]$ for levels $p = 1, 2, \dots, r$. They comprise the sub-matrices $[\hat{E}_{p-1}^{k_p}(2^{n-p})]$ of size $2^{n-p} \times 2^{n-p}$ for $k_p = 1, 2, \dots, 4^p$, obtained through quadtree division of the matrix $[\hat{E}_{p-1}(2^{n-p})]$. Each sub-matrix $[\hat{E}_{p-1}^{k_p}(2^{n-p})]$ is then defined by the relation:

$$[\hat{E}_{p-1}^{k_p}(2^{n-p})] = [T_p(2^{n-p})]^{-1}[\hat{S}_p^{k_p}(2^{n-p})][T_p(2^{n-p})]^{-1} \text{ for } k_p = 1, 2, \dots, 4^p, \quad (1.5)$$

where 4^p is the number of the quadtree branches in the decomposition level p . Here $[T_p(2^{n-p})]^{-1}$ is a matrix of size $2^{n-p} \times 2^{n-p}$ in the level p , used for the inverse 2D OT.

The elements $\hat{s}_p^{k_p}(u, v) = m_p(u, v) \cdot s_p^{k_p}(u, v)$ of the matrix $[\hat{S}_p^{k_p}(2^{n-p})]$ are defined by the elements $m_p(u, v)$ of the binary matrix-mask $[M_p(2^{n-p})]$:

$$m_p(u, v) = \begin{cases} 1, & \text{if } s_p^{k_p}(u, v) \text{ is a retained coefficient,} \\ 0 & \text{— otherwise.} \end{cases} \quad (1.6)$$

The matrix $[S_p^{k_p}(2^{n-p})]$ is the transform of $[E_{p-1}^{k_p}(2^{n-p})]$ and is defined through direct 2D OT:

$$[S_p^{k_p}(2^{n-p})] = [T_p(2^{n-p})][E_{p-1}^{k_p}(2^{n-p})][T_p(2^{n-p})]. \quad (1.7)$$

Here $[T_p(2^{n-p})]$ is a matrix of size $2^{n-p} \times 2^{n-p}$ in the decomposition level p , used for the 2D OT of each block $[E_p^{k_p}(2^{n-p})]$ (when $k_p = 1, 2, \dots, 4^p$), of the difference matrix for same level, defined by the equation:

$$[E_{p-1}(2^n)] = [B(2^n)] - [\hat{B}_0(2^n)] \text{ for } p = 1; \quad (1.8)$$

$$[E_{p-1}(2^{n-p})] = [E_{p-2}(2^{n-p})] - [\hat{E}_{p-2}(2^{n-p})] \text{ for } p = 2, 3, \dots, r. \quad (1.9)$$

In result of the decomposition represented by Eq. 1.1, for each block $[B(2^n)]$, are calculated the following spectrum coefficients:

- all nonzero coefficients of the transform $[\hat{S}_0(2^n)]$ in the decomposition level $p = 0$;
- all nonzero coefficients of the transforms $[\hat{S}_p^{k_p}(2^{n-p})]$ for $k_p = 1, 2, \dots, 4^p$ in the decomposition levels $p = 1, 2, \dots, r$.

The spectrum coefficients of same spatial frequency (u, v) from all image blocks are arranged in common data sequences, which correspond to their decomposition level p . The transformation of the 2D data massifs into one-dimensional data sequence is executed, using the recursive Hilbert scan, which preserves very well the correlation between neighboring coefficients.

In order to reduce the decomposition complexity, and in accordance with Eq. 1.1, this could be done recursively, as follows:

$$[B'_r(2^n)] = [B'_{r-1}(2^n)] + [\hat{E}_r(2^n)] \text{ for } r = 1, 2, \dots, n-1. \quad (1.10)$$

For the case, when the number of the retained coefficients for each IDP sub-block

k_p of size $2^{n-p} \times 2^{n-p}$ is $\sum_{u=0}^{2^{n-p}} \sum_{v=0}^{2^{n-p}} m_p(u, v) = 4$, then their total number for all levels is:

$$N = \sum_{p=0}^{n-1} 4^{p+1} = (4/3)(4^n - 1) \approx (4/3)4^n. \quad (1.11)$$

In this case the total number of “retained” coefficients is 4/3 times higher than that of the pixels in the block, and hence, the IPD is “overcomplete”.

1.3.3 Reduced Inverse Difference Pyramid

For the building of the Reduced IDP (RIDP) [14], the existing relations between the spectrum coefficients from the neighboring IDP levels are used. Let the retained coefficients $s_p^{k_p}(u, v)$ with spatial frequencies $(0, 0)$, $(1, 0)$, $(0, 1)$ and $(1, 1)$ for the sub-block k_p in the IDP level p , be obtained by using the 2D-WHT. Then, except for level $p = 0$, the coefficients $(0, 0)$ from each of the four neighboring sub-blocks in same IDP level are equal to zero, i.e.:

$$s_p^{k_p}(0, 0) = s_p^{k_p+1}(0, 0) = s_p^{k_p+2}(0, 0) = s_p^{k_p+3}(0, 0) = 0 \text{ for } p = 1, 2, \dots, n-1. \quad (1.12)$$

From this, it follows that the coefficients $s_{p+i}^{k_p}(0, 0)$ for $i = 0, 1, 2, 3$ could be cut-off, and as a result they should not be saved or transferred. Hence, the total number of the retained coefficients N_R for each sub-block k_p in the decomposition levels $p = 1, 2, \dots, n-1$ of the RIDP could be reduced by $1/4$, i.e.

$$N_R = 4 + \sum_{p=1}^{n-1} 4^{p+1} - \sum_{p=1}^{n-1} 4^p = 4 + 3 \sum_{p=1}^{n-1} 4^p = 4 + 3 \frac{4}{3} (4^{n-1} - 1) = 4^n. \quad (1.13)$$

In this case the total number of the “retained” coefficients for all levels is equal to the number of pixels in the block, and hence, the so calculated RIDP is “complete”.

1.3.4 Main Principle for Branched IDP Building

The pyramid BIDP [15, 16] with one or more branches is an extension of the basic IDP. The image representation through the BIDP aims at the enhancement of the image energy concentration in a small number of IDP components. On Fig. 1.2 is shown an example block diagram of the generalized 3-level BIDP. The IDP for each block of size $2^n \times 2^n$ from the original image, called “Main Pyramid”, is of 3 levels ($n = 3$, for $p = 0, 1, 2$). The values of the coefficients, calculated for these 3 levels, compose the inverse pyramid, whose sections are of different color each. The coefficients $s(0, 0)$, $s(0, 1)$, $s(1, 0)$ and $s(1, 1)$ in level $p = 0$ from all blocks compose corresponding matrices of size $m \times m$, colored in yellow. These 4 matrices build the “Branch for level 0” of the Main Pyramids. Each is then divided into blocks of size $2^{n-1} \times 2^{n-1}$, on which in similar way are built the corresponding 3-level IPDs ($p = 00, 01, 02$). The retained coefficients $s(0, 1)$, $s(1, 0)$ and $s(1, 1)$ in level $p = 1$ of the Main Pyramids from all blocks build matrices of size $2m \times 2m$ (colored in pink).

Each matrix of size $2m \times 2m$ is divided into blocks of size $2^{n-1} \times 2^{n-1}$, on which in similar way are build corresponding 3-level IDPs ($p = 10, 11, 12$). The retained coefficients, calculated after TOT from the blocks of the Residual Difference in the

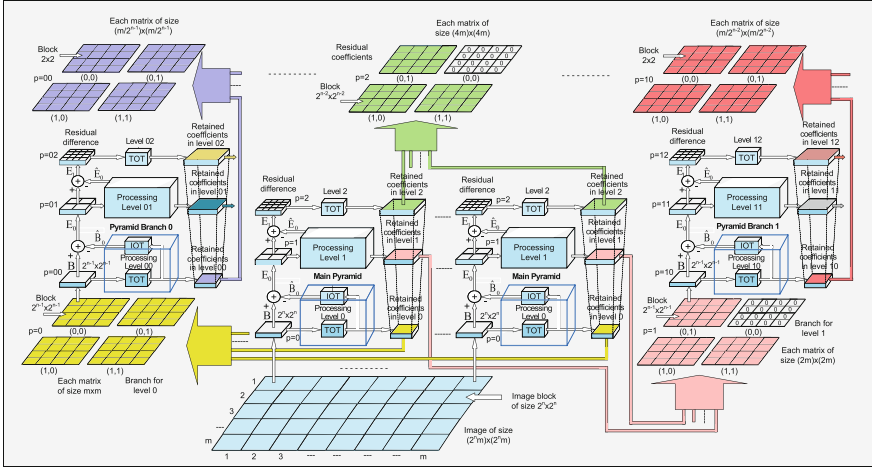


Fig. 1.2 Example of generalized 3-level Branched Inverse Difference Pyramid (BIDP)

last level ($p = 2$) of the Main Pyramids, build matrices of size $4m \times 4m$; from the first level ($p = 00$) of the Pyramid Branch 0—matrices of size $(m/2^{n-1} \times m/2^{n-1})$; and from the first level ($p = 10$) of the “Pyramid Branch 1”—matrices of size $(m/2^{n-2} \times m/2^{n-2})$. In order to reduce the correlation between the elements of the so obtained matrices, on each group of 4 spatially neighboring elements is applied the following transform: the first is substituted by their average value, and each of the remaining 3—by its difference to next elements, scanned counter-clockwise. The coefficients, obtained this way from all levels of the Main and Branch Pyramids are arranged in one-dimensional sequences in accordance with Hilbert scan and after that are quantized and entropy coded using Adaptive RLC and Huffman. The values of the spectrum coefficients are quantized only in case that the image coding is lossy. In order to retain the visual quality of the restored images, the quantization values are related to the sensibility of the human vision to errors in different spatial frequencies. To reduce these errors, retaining the compression efficiency, in the consecutive BIDP levels could be used various fast orthogonal transforms: for example, in the zero level could be used DCT, and in the next levels—WHT.

1.3.5 Mathematical Representation for One BIDP Branch

In the general case, the branch g of the BIDP is built on the matrix $[S_{g(u, v)}]$ of size $2^{n-g-1} \times 2^{n-g-1}$, which comprises all spectrum coefficients $s_p^k(u, v)$ with the same spatial frequency (u, v) from all blocks or sub-blocks k_p in the level $p = g$ of

the Main IDPs. By analogy with Eq. (1.1), the matrix $[S_{g(u,v)}]$ could be decomposed in accordance with the relation, given below:

$$[S_{g(u,v)}] = [\tilde{S}_{0,g(u,v)}] + \sum_{s=1}^r [\tilde{E}_{s-1,g(u,v)}^{k_s}] + [\tilde{E}_{r,g(u,v)}] \text{ for } r = 1, 2, \dots, n-1, \quad (1.14)$$

where

$$[\tilde{S}_{0,g(u,v)}] = [T_{0,g(u,v)}]^{-1} [\hat{S}_{0,g(u,v)}] [T_{0,g(u,v)}]^{-1} \text{ for } s = 0; \quad (1.15)$$

$$[\hat{S}_{0,g(u,v)}] = [\hat{s}_{0,g(u,v)}(k, l)] = [m_{0,g(u,v)}(k, l) s_{0,g(u,v)}(k, l)]; \quad (1.16)$$

$$m_{0,g(u,v)}(k, l) = \begin{cases} 1, & \text{if } s_{0,g(u,v)}(k, l) \text{ are the retained coefficients,} \\ 0 & \text{— otherwise;} \end{cases} \quad (1.17)$$

$$[S_{0,g(u,v)}] = [s_{0,g(u,v)}(k, l)] = [T_{0,g(u,v)}] [S_{g(u,v)}] [T_{0,g(u,v)}]; \quad (1.18)$$

$$[E_{s-1,g(u,v)}] = [S_{g(u,v)}] - [\tilde{S}_{0,g(u,v)}] \text{ for } s = 1; \quad (1.19)$$

$$\begin{aligned} [\tilde{E}_{s-1,g(u,v)}^{k_s}] &= [T_{s,g(u,v)}]^{-1} [S_{s,g(u,v)}^{k_s}] [T_{s,g(u,v)}]^{-1} \text{ for } s = 2, 3, \dots, r \text{ and } k_s \\ &= 1, 2, \dots, 4^s; \end{aligned} \quad (1.20)$$

$$[E_{s-1,g(u,v)}] = [E_{s-2,g(u,v)}] - [\tilde{E}_{s-2,g(u,v)}]. \quad (1.21)$$

All matrices in Eqs. (1.14)–(1.19) are of size $2^{n-g-1} \times 2^{n-g-1}$, and these in Eqs. (1.20) and (1.21)—of size $2^{n-g-s-1} \times 2^{n-g-s-1}$. The decomposition from Eq. (1.14) of the matrix $[S_{g(u,v)}]$ is named Pyramid Branch ($\text{PB}_{g(u,v)}$). It is a pyramid, whose initial and final levels are g and r correspondingly ($g < r$). This pyramid represents the branch g of the Main IDPs and contains all coefficients, whose spatial frequency is (u, v) .

The maximum number of branches for the levels $p = 0, 1, \dots, n-1$ of the Main IDPs, built on a sub-block of size $2^{n-p} \times 2^{n-p}$, is defined by the general number of retained spectrum coefficients $M_p = 4^p \sum_{u=0}^{2^{n-p}} \sum_{v=0}^{2^{n-p}} m_p(u, v)$. For the branch g from the level $p = g$ the corresponding pyramid $\text{PB}_{g(u,v)}$ is of r levels. The number of the coefficients in this branch of the Main IDPs for $p = g, g+1, \dots, r$, without cutting-off the coefficients, calculated for the spatial frequency $(0, 0)$, is:

$$N_{g,r} = M_g \sum_{p=g}^r 4^p = M_g \left[\sum_{p=0}^r 4^p - \sum_{p=0}^{g-1} 4^p \right] = (M_g/3)(4^{r+1} - 4^g). \quad (1.22)$$

In case that the number of the retained spectrum coefficients for each sub-block is set to be $\sum_{u=0}^{2^{n-g}} \sum_{v=0}^{2^{n-g}} m_g(u, v) = 4$, then $M_g = 4^{g+1}$. In this case, from Eq. (1.22) it follows, that the total number of the coefficients in the branch $PB_{g(uv)}$ is $N_{g,r} = (4^{g+1}/3)(4^{r+1} - 4^g)$. Hence, the compression ratio (CR) for $PB_{g(uv)}$ is defined by the relation:

$$CR_{g,r} = \frac{4^{n-g-1}}{N_{g,r}} = \frac{3}{4} \times \frac{4^{n-g-1}}{4^g(4^{r+1} - 4^g)}, \quad (1.23)$$

where 4^{n-g-1} is the number of the elements in one sub-block of size $2^{n-g-1} \times 2^{n-g-1}$ from $PB_{g(uv)}$.

The compression ratio for the Main IDPs, calculated in accordance with Eq. (1.11), is:

$$CR = \frac{4^n}{N} = \frac{3 \times 4^n}{4(4^n - 1)} \approx \frac{3}{4} \text{ for } 4^n \gg 1. \quad (1.24)$$

From the comparison of the Eqs. (1.23) and (1.24) it follows, that:

$$CR_{g,r} > CR, \text{ if } r \leq n - 3. \quad (1.25)$$

In case that the requirement from Eq. (1.25) for the number of levels r of $PB_{g(u,v)}$ for level g of the Main IDPs is satisfied, the compression ratio for the branch g is higher, than that for each of the basic pyramids. From Eq. (1.25) it follows that the condition $r > l$ is satisfied, when $n > 4$, i.e., when the image is divided into blocks of minimum size of 16×16 pixels. For this case, to retain the correlation between their pixels high, is necessary the size of the image $(16m) \times (16m)$ to be relatively large. For example, the image should be of size $2k \times 2k$ (for $m = 128$), or larger. Hence, the BIDP decomposition is efficient mainly for images with high resolution.

The correlation between the elements of the blocks of size $2^{n-1} \times 2^{n-1}$ from the initial level $g = 0$ of the Main IDPs is higher than that, between the elements of the sub-blocks of size $2^{n-g-1} \times 2^{n-g-1}$ from the higher levels $g = 1, 2, \dots, r$. Because of this, the branching of the BIDP should always start from the level $g = 0$.

1.3.6 Transformation of the Retained Coefficients into Sub-blocks of Size 2×2

The aim of the transformation is to reduce the correlation between the retained neighboring spectrum coefficients in the sub-blocks of size 2×2 in each matrix, built by the coefficients of same spatial frequency (u, v) from all blocks (or respectively—from the sub-blocks k_p in the selected level p of the Main IDPs, or

their branches). In order to simplify the presentation, the spectrum coefficients in the sub-blocks k_p for the level p , are set as follows:

$$A_i = s_p^{k_p+i}(0,0); B_i = s_p^{k_p+i}(1,0); C_i = s_p^{k_p+i}(0,1); D_i = s_p^{k_p+i}(1,1) \text{ for } i = 0, 1, 2, 3. \quad (1.26)$$

On Fig. 1.3 are shown matrices of size 2×2 , which contain the retained groups of four spectrum coefficients $s_p^{k_p}(u, v)$, which have same frequencies, (0, 0), (1, 0), (0, 1) and (1, 1) correspondingly, placed in four neighboring sub-blocks ($k_p, k_p + 1, k_p + 2, k_p + 3$) of size $2^{n-p} \times 2^{n-p}$ for the level p of the Main IDPs, or their branches.

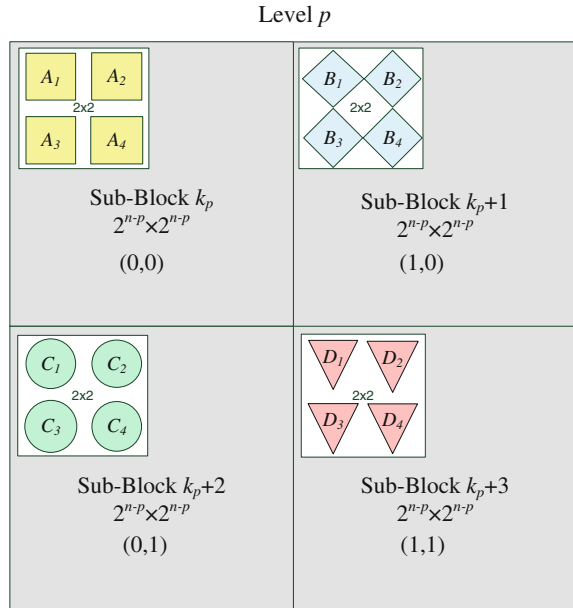
In correspondence with the symbols, used in Fig. 1.3, the transformation of the groups of four coefficients is represented by the relation below [16]:

$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 4 & 0 & -4 \\ -4 & 0 & 4 & 0 \\ 0 & 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}. \quad (1.27)$$

Here P_i , for $i = 1, 2, 3, 4$ represent correspondingly:

- the coefficients A_i , for $i = 1, 2, 3, 4$ with frequencies (0, 0);
- the coefficients B_i , for $i = 1, 2, 3, 4$ with frequencies (1, 0);
- the coefficients C_i , for $i = 1, 2, 3, 4$ with frequencies (0, 1);
- the coefficients D_i , for $i = 1, 2, 3, 4$ with frequencies (1, 1).

Fig. 1.3 Location of the retained groups of four spectrum coefficients from 4 neighboring sub-blocks $k_p + i$ ($i = 0, 1, 2, 3$) of size $2^{n-p} \times 2^{n-p}$ in the decomposition level p



In result of the transform, executed in accordance with Eq. (1.27), each coefficient S_1 has higher value, than the remaining three difference coefficients $S_2, S_3,$ and S_4 .

The inverse transform executed in respect of Eq. (1.27) gives total restoration of the initial coefficients $P_i,$ for $i = 1, 2, 3, 4$:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 4 & -1 & -3 & -2 \\ 4 & 3 & 1 & 2 \\ 4 & -1 & 1 & -2 \\ 4 & -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix}, \quad (1.28)$$

Depending on the frequency $(0, 0), (1, 0), (0, 1),$ or $(1, 1)$ of the restored coefficients $P_1 \sim P_4,$ they correspond to $A_1 \sim A_4, B_1 \sim B_4, C_1 \sim C_4,$ or $D_1 \sim D_4.$ The operation, given in Eq. (1.28) is executed through decoding of the transformed coefficients $S_1 \sim S_4.$ The so described features of the coefficients S_1, S_2, S_3, S_4 permit to achieve significant enhancement of their entropy coding efficiency.

The basic quality of the BIDP is that it offers significant decorrelation of the processed image data. As a result, the BIDP permits the following:









- To achieve highly efficient compression with retained visual quality of the restored image (i.e. visually lossless coding), or efficient lossless coding, depending on the application requirements;
- Layered coding and transfer of the image data, in result of which is obtained low transfer bit-rate with gradually increased quality of the decoded image;
- Lower computational complexity than that of the wavelet decompositions [4];
- Easy adaptation of the coder parameters, so that to ensure the needed concordance of the obtained data stream, to the ability of the communication channel;
- Resistance to noises in the communication channel, or due to compression/decompression. The reason for this is the use of TOT in the decoding of each image block;
- Retaining the quality of the decoded image after multiple coding/decoding;

The BIDP could be further developed and modified in accordance to the requirements of various possible applications. One of these applications for processing of groups of similar images, for example, is a sequence of Computer Tomography (CT) images, Multi-Spectral (MS) images, etc.

1.3.7 Experimental Results

The experimental results, given below, were obtained from the investigation of image database, which contained medical images stored in DICOM (*dcm*) format, of various size and kind, grouped in 24 classes. The database was created at the

Table 1.1 Results for the lossless compression of various classes of medical images

Image name size	CTI013 512 × 512 (Group of 14 images)	CTI069 512 × 512 (Group of 275 images)	CTI022 512 × 512 (Group of 14 images)	CTI002 512 × 512 (Group of 14 images)	MG101 1914 × 2294	NMI001 1024 × 1024	USI 1.2 1020 × 818	USI 1.3 1020 × 818
Image type: CTI/NMI/CR1/MG/USI								
<i>dcm</i>	545 KB	545 KB	39.6 KB	545 KB	4.19 MB	2.00 MB	2.44 MB	4.19 MB
<i>jp2</i>	64.2 KB	52.2 KB	34.3 KB	3.92 KB	820 KB	103 KB	700 KB	189 KB
<i>tk</i>	63.9 KB	50.5 KB	26.3 KB	2.30 KB	801 KB	71 KB	388 KB	134 KB
Comparison of compression efficiency for the examples above								
<i>dcm/jp2</i>	8.48	10.44	1.15	139.03	5.11	19.41	3.48	22.16
<i>dcm/tk</i>	8.52	10.79	1.51	236.95	5.23	28.17	6.28	31.26

Medical University of Sofia, and comprises the following image kinds: CTI—computer tomography images; MGI—mammography images; NMI—nuclear magnetic resonance images; CRI—computer radiography images, and USI—ultrasound images. For the investigation, the DICOM images were first transformed into non-compressed (*bmp* format), and then they were processed by using various lossless compression algorithms. A part of the obtained results is given in Table 1.1.

Here are shown the results for the lossless compression of the *bmp* files of still images, and of image sequences, after their transformation into files of the kind *jp2* and *tk*. The image file format *jp2* is based on the standard JPEG2000LS, and the *tk* format—on the algorithms BIDP for single images, combined with the adaptive run-length lossless coding (ARLE), based on the histogram statistics [17]. For the execution of the 2D-TOT/IOT in the initial levels of all basic pyramids and their branches was used the 2D-DCT, and in their higher levels—the 2D-WHT transform. The number of the pyramid levels for the blocks of the smallest treated images (of size 512×512), is two, and for the larger ones, it is three. The basic IDP pyramids have one branch only, comprising coefficients with spatial frequency (0, 0) for their initial levels.

From the analysis of the obtained results, the following conclusions could be done:

1. The new format *tk* surpasses the *jp2*, especially for images, which contain objects, placed on a homogenous background. From the analyzed 24 classes of images, 17 are of this kind. Some examples are shown in Table 1.1;
2. Together with the enlargement of the analyzed images, the compression ratio for the lossless *tk* compression grows up, compared to that of the *jp2*;
3. The data given in Table 1.1 show that the mean compression ratio for all DICOM images after their transformation into the format *tk* is 41:1, while for the *jp2* this coefficient is 26:1. Hence, the use of the *tk* format for all 24 classes ensures compression ratio which is $\approx 40\%$ higher than that of the *jp2* format.

The experimental results, obtained for the comparison of the coding efficiency for several kinds of medical images through BIDP and JPEG2000 confirmed the basic advantages of the new approach for hierarchical pyramid decomposition, presented here.

1.4 Hierarchical Singular Value Image Decomposition

The SVD is a statistical decomposition for processing, coding and analysis of images, widely used in the computer vision systems. This decomposition was an object of vast research, presented in many monographs [18–22] and papers [23–26]. This is optimal image decomposition, because it concentrates significant part of the image energy in minimum number of components, and the restored image (after reduction of the low-energy components), has minimum mean square error. One of the basic problems, which limit, to some degree, the use of the “classic” SVD, is

related to its high computational complexity, which grows up together with the image size.

To overcome this problem, several new approaches are already offered. The first is based on the SVD calculation through iterative methods, which do not require defining the characteristic polynomials of a pair of matrices. In this case, the SVD is executed in two stages: in the first, each matrix is first transformed into triangular form with the QR decomposition, and then—into bidiagonal, through the Householder transforms [27]. In the second stage on the bidiagonal matrix is applied an iterative method, whose iterations stop when the needed accuracy is achieved. For this could be used the iterative method of Jacobi [21], in accordance with which for the calculation of the SVD with bidiagonal matrix is needed the execution of a sequence of orthogonal transforms with rotation matrix of size 2×2 . The second approach is based on the relation of the SVD with the Principal Component Analysis (PCA). It could be executed through neural networks [28] of the kind generalized Hebbian or multilayer perceptron networks, which use iterative learning algorithms. The third approach is based on the algorithm, known as Sequential KL/SVD [29]. The basic idea here is as follows: the image matrix is divided into blocks of small size, and on each is applied the SVD, based on the QR decomposition [21]. At first, the SVD is calculated for the first block from the original image (the upper left, for example), and then is used iterative SVD calculation for each of the remaining blocks by using the transform matrices, calculated for the first block (by updating the process). In the flow of the iteration process are deleted the SVD components, which correspond to very small eigen values.

For the acceleration of the SVD calculation several methods are already developed [30–32]. The first, is based on the algorithm, called Randomized SVD [30], a number of matrix rows (or columns) is randomly chosen. After scaling, they are used to build a small matrix, for which is calculated the SVD, and it is later used as an approximation of the original matrix. In [31] is offered the algorithm QUIC-SVD, suitable for matrices of very large size. Through this algorithm is achieved fast sample-based SVD approximation with automatic relative error control. Another approach is based on the sampling mechanism, called the cosine tree, through which is achieved best-rank approximation. The experimental investigation of the QUIC-SVD in [32] presents better results than those, from the MATLAB SVD and the Tygert SVD. The so obtained 6–7 times acceleration compared to the SVD depends on the pre-selected value of the parameter δ which defines the upper limit of the approximation error, with probability $(1 - \delta)$.

Several SVD-based methods developed, are dedicated to enhancement of the image compression efficiency [33–37]. One of them, called Multi-resolution SVD [33], comprises three steps: image transform, through 9/7 biorthogonal wavelets of two levels, decomposition of the SVD-transformed image, by using blocks of size 2×2 up to level six, and at last—the use of the algorithms SPIHT and gzip. In [34] is offered the hybrid KLT-SVD algorithm for efficient image compression. The method K-SVD [35] for facial image compression, is a generalization of the K-means clusterization method, and is used for iterative learning of overcomplete dictionaries for sparse coding. In correspondence with the combined compression

algorithm, in [36] is proposed a SVD based sub-band decomposition and multi-resolution representation of digital colour images. In the paper [37] is used the decomposition, called Higher-Order SVD (HOSVD), through which the SVD matrix is transformed into a tensor with application in the image compression.

In this chapter, the general presentation of one new approach for hierarchical decomposition of matrix images is given, based on the multiple application of the SVD on blocks of size 2×2 [38]. This decomposition, called Hierarchical SVD (HSVD), has tree-like structure of the kind “binary tree” (full or truncated). The SVD calculation for blocks of size 2×2 is based on the adaptive KLT [5, 39]. The HSVD algorithm aims to achieve a decomposition with high computational efficiency, suitable for parallel and recursive processing of the blocks through simple algebraic operations, and offers the possibility for enhancement of the calculations through cutting-off the tree branches, whose eigen values are small or equal to zero.

1.4.1 SVD Algorithm for Matrix Decomposition

In the general case, the decomposition of each image matrix $[X(N)]$ of size $N \times N$ could be executed by using the direct SVD [5], defined by the equation below:

$$[X(N)] = [U(N)][A(N)]^{1/2}[V(N)]^t = \sum_{s=1}^N \sqrt{\lambda_s} \vec{U}_s \cdot \vec{V}_s^t. \quad (1.29)$$

The inverse SVD is respectively:

$$[A(N)]^{1/2} = [U(N)]^t [X(N)] [V(N)]. \quad (1.30)$$

In the relations above, the terms $[U(N)] = [\vec{U}_1, \vec{U}_2, \dots, \vec{U}_N]$ and $[V(N)] = [\vec{V}_1, \vec{V}_2, \dots, \vec{V}_N]$ are matrices, composed respectively by the vectors \vec{U}_s and \vec{V}_s for $s = 1, 2, \dots, N$; \vec{U}_s are the eigenvectors of the matrix $[Y(N)] = [X(N)][X(N)]^t$ (left-singular vectors of the $[X(N)]$), and \vec{V}_s —the eigenvectors of the matrix $[Z(N)] = [X(N)]^t[X(N)]$ (right-singular vectors of the $[X(N)]$), for which:

$$[Y(N)]\vec{U}_s = \lambda_s \vec{U}_s, [Z(N)]\vec{V}_s = \lambda_s \vec{V}_s; \quad (1.31)$$

$[A(N)] = \text{diag} [\lambda_1, \lambda_2, \dots, \lambda_N]$ is a diagonal matrix, composed by the eigenvalues λ_s which are identical for the matrices $[Y(N)]$ and $[Z(N)]$.

From Eq. (1.29) it follows that for the description of the decomposition for a matrix of size $N \times N$, $N \times (2N + 1)$ parameters are needed in total, i.e. in the general case the SVD is a decomposition of the kind “overcomplete”.

1.4.2 Particular Case of the SVD for Image Block of Size 2×2

In this case, the direct SVD for the block $[X]$ of size 2×2 (for $N = 2$) is represented by the relation:

$$[X] = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [U][A]^{1/2}[V]^t = \sqrt{\lambda_1}\vec{U}_1\vec{V}_1^t + \sqrt{\lambda_2}\vec{U}_2\vec{V}_2^t = \sum_{s=1}^2 \sqrt{\lambda_s}\vec{U}_s\vec{V}_s^t \quad (1.32)$$

or

$$[X] = [C_1] + [C_2], \quad (1.33)$$

where $[C_1] = \sqrt{\lambda_1}\vec{U}_1\vec{V}_1^t$; $[C_2] = \sqrt{\lambda_2}\vec{U}_2\vec{V}_2^t$; a, b, c, d are the elements of the block $[X]$; λ_1, λ_2 are the eigenvalues of the symmetrical matrices $[Y]$ and $[Z]$, defined by the relations below:

$$[Y] = [X][X]^t = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} (a^2 + b^2) & (ac + bd) \\ (ac + bd) & (c^2 + d^2) \end{bmatrix}; \quad (1.34)$$

$$[Z] = [X]^t[X] = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (a^2 + c^2) & (ab + cd) \\ (ab + cd) & (b^2 + d^2) \end{bmatrix}. \quad (1.35)$$

\vec{U}_1 and \vec{U}_2 are the eigenvectors of the matrix $[Y]$, for which: $[Y]\vec{U}_s = \lambda_s\vec{U}_s$ ($s = 1, 2$);

\vec{V}_1 and \vec{V}_2 are the eigenvectors of the matrix $[Z]$, for which: $[Z]\vec{V}_s = \lambda_s\vec{V}_s$ ($s = 1, 2$).

$[U] = [\vec{U}_1, \vec{U}_2]$ and $[V]^t = \begin{bmatrix} \vec{V}_1^t \\ \vec{V}_2^t \end{bmatrix}$ are matrices, composed by the eigen vectors \vec{U}_s and \vec{V}_s .

In accordance with the solution given in [38] for the case when $N = 2$, the couple direct/inverse SVD for the matrix $[X(2)]$ could be represented as follows:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{2A} \left\{ \sigma_1 \begin{bmatrix} \sqrt{rp} & \sqrt{sp} \\ \sqrt{rq} & \sqrt{sq} \end{bmatrix} + \sigma_2 \begin{bmatrix} \sqrt{sq} & -\sqrt{rq} \\ -\sqrt{sp} & \sqrt{rp} \end{bmatrix} \right\} = \sigma_1[T_1] + \sigma_2[T_2] \\ = [C_1] + [C_2], \quad (1.36)$$

$$\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} \sqrt{p} & \sqrt{q} \\ -\sqrt{q} & \sqrt{p} \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \sqrt{r} & -\sqrt{s} \\ \sqrt{s} & \sqrt{r} \end{bmatrix} \text{ for } A \neq 0, \quad (1.37)$$

where

$$\begin{aligned} A &= \sqrt{v^2 + 4\eta^2}, \quad \sigma_1 = \sqrt{\frac{\omega + A}{2}}, \quad \sigma_2 = \sqrt{\frac{\omega - A}{2}}, \quad r = A + v, \quad p = A + \mu, \\ s &= A - v, \quad q = A - \mu, \end{aligned} \quad (1.38)$$

$$\begin{aligned} v &= a^2 + c^2 - b^2 - d^2, \quad \eta = ab + cd, \quad \omega = a^2 + b^2 + c^2 + d^2, \\ \mu &= a^2 + b^2 - c^2 - d^2. \end{aligned} \quad (1.39)$$

Figure 1.4 shows the algorithm for direct SVD for the block $[X]$ of size 2×2 , composed in accordance with the relations (1.36), (1.38) and (1.39). This algorithm is the basic building element—the kernel, used to create the HSVD algorithm.

In accordance with Eq. (1.32) the matrix $[X]$ is transformed into the vector $\vec{X} = [a, b, c, d]^t$, whose components are arranged by using the “Z”-scan. The components of the vector \vec{X} are the input data for the SVD algorithm. After its execution, are obtained the vectors \vec{C}_1 and \vec{C}_2 , from whose components are defined the elements of the matrices $[C_1]$ and $[C_2]$ of size 2×2 , by using the “Z”-scan again. In this case however, this scan is used for the inverse transform of all vectors \vec{C}_1, \vec{C}_2 in the corresponding matrix $[C_1], [C_2]$.

1.4.3 Hierarchical SVD for a Matrix of Size $2^n \times 2^n$

The hierarchical n -level SVD (HSVD) for the image matrix $[X(N)]$ of size $2^n \times 2^n$ pixels ($N = 2^n$) is executed through multiple applying the SVD on image sub-blocks (sub-matrices) of size 2×2 , followed by rearrangement of the so calculated components.

In particular, for the case, when the image matrix $[X(4)]$ is of size $2^2 \times 2^2$ ($N = 2^2 = 4$), then the number of the hierarchical levels of the HSVD is $n = 2$. The flow graph, which represents the calculation of the HSVD, is shown on Fig. 1.5. In the first level ($r = 1$) of the HSVD, the matrix $[X(4)]$ is divided into four sub-matrices of size 2×2 , as shown in the left part of Fig. 1.5. Here the elements of the sub-matrices on which is applied the $SVD_{2 \times 2}$ in the first hierarchical level, are colored in same color (yellow, green, blue, and red). The elements of the sub-matrices are:

$$[X(4)] = \begin{bmatrix} [X_1(2)] & [X_2(2)] \\ [X_3(2)] & [X_4(2)] \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} & \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} \\ \begin{bmatrix} a_3 & b_3 \\ c_3 & d_3 \end{bmatrix} & \begin{bmatrix} a_4 & b_4 \\ c_4 & d_4 \end{bmatrix} \end{bmatrix}. \quad (1.40)$$

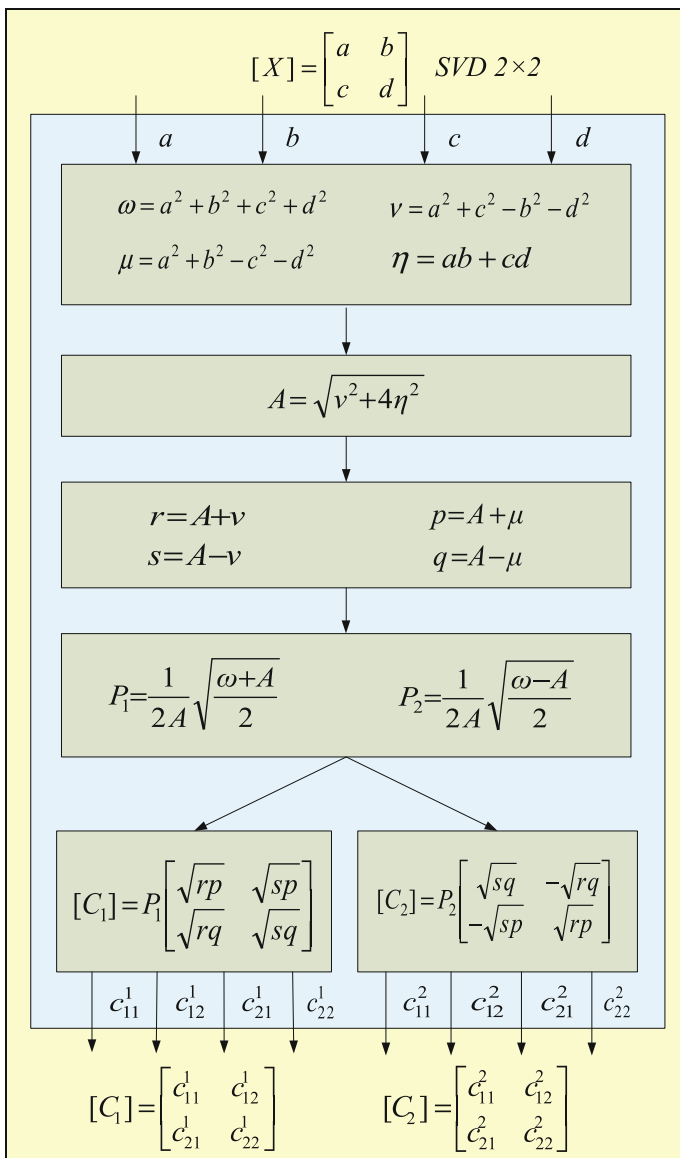


Fig. 1.4 Representation of the SVD algorithm for the matrix $[X]$ of size 2×2

On each sub-matrix $[X_k(2)]$ of size 2×2 ($k = 1, 2, 3, 4$), is applied $SVD_{2 \times 2}$, in accordance with Eqs. (1.36)–(1.39). As a result, it is decomposed into two components:

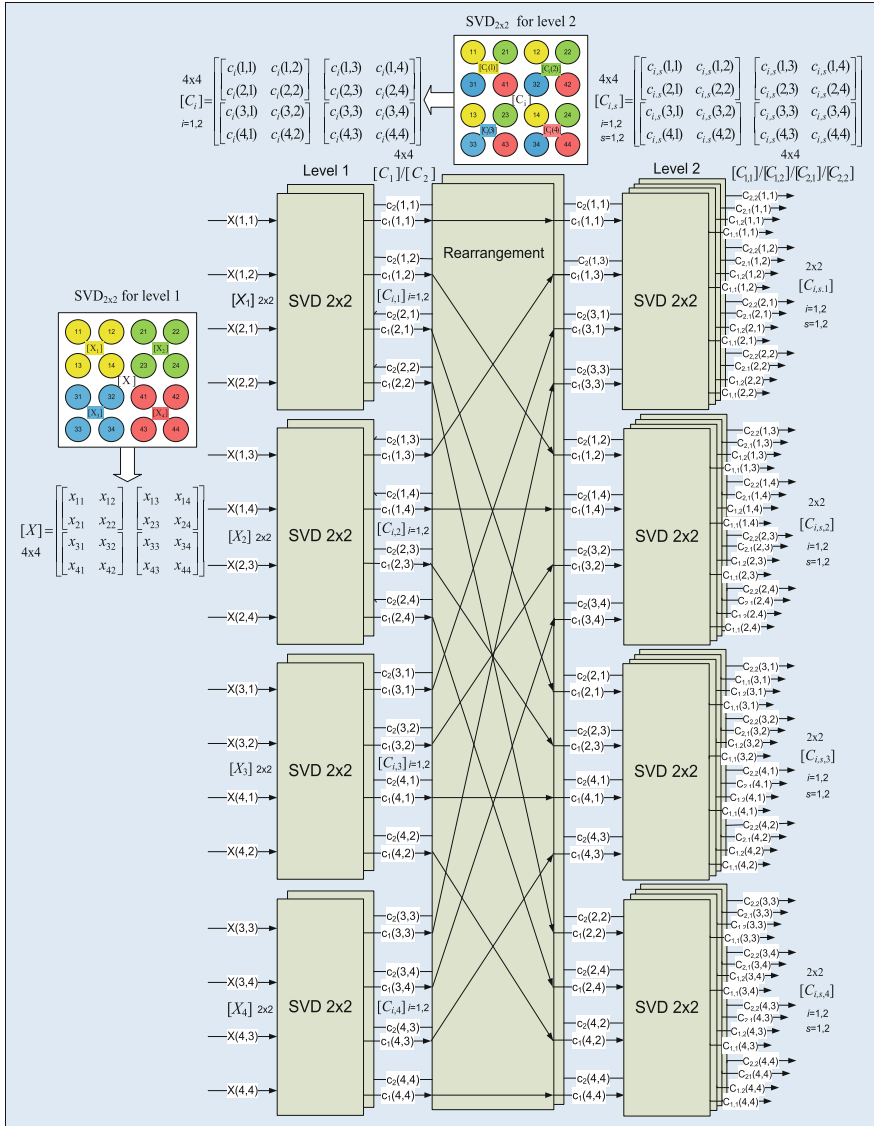


Fig. 1.5 Flowgraph of the HSVD algorithm represented through the vector-radix (2×2) for a matrix of size 4×4

$$[X_k(2)] = \sigma_{1,k}[T_{1,k}(2)] + \sigma_{2,k}[T_{2,k}(2)] = [C_{1,k}(2)] + [C_{2,k}(2)] \quad \text{for } k = 1, 2, 3, 4; \quad (1.41)$$

where $\sigma_{1,k} = \sqrt{\frac{\omega_{1,k} + A_{1,k}}{2}}$, $\sigma_{2,k} = \sqrt{\frac{\omega_{2,k} - A_{2,k}}{2}}$, $[T_{1,k}(2)] = \vec{U}_{1,k} \vec{V}_{1,k}^t$, $[T_{2,k}(2)] = \vec{U}_{2,k} \vec{V}_{2,k}^t$.

Using the matrices $[C_{m,k}(2)]$ of size 2×2 for $k = 1, 2, 3, 4$ and $m = 1, 2$, are composed the matrices $[C_m(4)]$ of size 4×4 :

$$\begin{aligned}
 [C_m(4)] &= \begin{bmatrix} [C_{m,1}(2)] & [C_{m,2}(2)] \\ [C_{m,3}(2)] & [C_{m,4}(2)] \end{bmatrix} \\
 &= \begin{bmatrix} \begin{bmatrix} c_{11}(m,1) & c_{12}(m,1) \\ c_{13}(m,1) & c_{14}(m,1) \end{bmatrix} & \begin{bmatrix} c_{11}(m,2) & c_{12}(m,2) \\ c_{13}(m,2) & c_{14}(m,2) \end{bmatrix} \\ \begin{bmatrix} c_{11}(m,3) & c_{12}(m,3) \\ c_{13}(m,3) & c_{14}(m,3) \end{bmatrix} & \begin{bmatrix} c_{11}(m,4) & c_{12}(m,4) \\ c_{13}(m,4) & c_{14}(m,4) \end{bmatrix} \end{bmatrix} \text{ for } m = 1, 2.
 \end{aligned}
 \tag{1.42}$$

Hence, the SVD decomposition of the matrix $[X]$ in the first level is represented by two components:

$$[X(4)] = [C_1(4)] + [C_2(4)] = \begin{bmatrix} ([C_{1,1}(2)] + [C_{2,1}(2)]) & ([C_{1,2}(2)] + [C_{2,2}(2)]) \\ ([C_{1,3}(2)] + [C_{2,3}(2)]) & ([C_{1,4}(2)] + [C_{2,4}(2)]) \end{bmatrix}.
 \tag{1.43}$$

In the second level ($r = 2$) of the HSVD, on each matrix $[C_m(4)]$ of size 4×4 is applied four times the $SVD_{2 \times 2}$. Unlike the transform in the previous level, in the second level, the $SVD_{2 \times 2}$ is applied on the sub-matrices $[C_{m,k}(2)]$ of size 2×2 , whose elements are mutually interlaced and are defined in accordance with the scheme, given in the upper part of Fig. 1.5. The elements of the sub-matrices, on which is applied the $SVD_{2 \times 2}$ in the second hierarchical level are colored in same color (yellow, green, blue, and red). As it is seen on the figure, the elements of the sub-matrices of size 2×2 in the second level are not neighbors, but placed one element away in horizontal and vertical directions. As a result, each matrix $[C_m(4)]$ is decomposed into two components:

$$[C_m(4)] = [C_{m,1}(4)] + [C_{m,2}(4)] \text{ for } m = 1, 2.
 \tag{1.44}$$

Then, the full decomposition of the matrix $[X]$ is represented by the relation:

$$[X(4)] = [C_{1,1}(4)] + [C_{1,2}(4)] + [C_{2,1}(4)] + [C_{2,2}(4)] = \sum_{m=1}^2 \sum_{s=1}^2 [C_{m,s}(4)],
 \tag{1.45}$$

Hence, the decomposition of an image of size 4×4 comprises four components in total.

The matrix $[X(8)]$ is of size $2^3 \times 2^3$ ($N = 2^3 = 8$ for $n = 3$), and in this case, the HSVD is executed through multiple calculation of the $SVD_{2 \times 2}$ on blocks of size 2×2 , in all levels (the general number of the decomposition components is eight). In the first and second levels, the $SVD_{2 \times 2}$ is executed in accordance with the scheme, shown on Fig. 1.5. In the third level, the $SVD_{2 \times 2}$ is mainly applied on

sub-matrices of size 2×2 . Their elements are defined in similar way, as shown on Fig. 1.5, but the elements of same color (i.e., which belong to same sub-matrix) are moved three elements away in the horizontal and vertical direction.

The described HSVD algorithm could be generalized for the cases when the image $[X(2^n)]$ is of size $2^n \times 2^n$ pixels. Then the relation (1.45) becomes as shown below:

$$[X(2^n)] = \sum_{p_1=1}^2 \sum_{p_2=1}^2 \dots \sum_{p_n=1}^2 [C_{p_1, p_2, \dots, p_n}(2^n)]. \quad (1.46)$$

The maximum number of the HSVD decomposition levels is n , the maximum number of the decomposition components (1.46) is 2^n , and the distance in horizontal and vertical direction between the elements of the blocks of size 2×2 in the level r is correspondingly $(2^{r-1} - 1)$ elements, for $r = 1, 2, \dots, n$.

1.4.4 Computational Complexity of the Hierarchical SVD of Size $2^n \times 2^n$

1.4.4.1 Computational Complexity of the SVD of Size 2×2

The computational complexity could be defined by using the Eq. (1.36), taking into account the number of multiplication and addition operations, needed for the preliminary calculation of the components $\omega, \mu, \delta, v, \eta, A, B, \theta_1, \theta_2, \sigma_1, \sigma_2$, defined by the Eqs. (1.38) and (1.39). Then:

- The number of the multiplications, needed for the calculation of Eq. (1.36) is $\Sigma_m = 39$;
- The number of the additions, needed for the calculation of Eq. (1.36) is $\Sigma_s = 15$.

Then the total number of the algebraic operations executed with floating point for SVD of size 2×2 is:

$$O_{\text{SVD}}(2 \times 2) = \Sigma_m + \Sigma_s = 54. \quad (1.47)$$

1.4.4.2 Computational Complexity of the Hierarchical SVD of Size $2^n \times 2^n$

The computational complexity is defined on the basis of $\text{SVD}_{2 \times 2}$. In this case, the number M of the sub-matrices of size 2×2 , which comprise the image of size $2^n \times 2^n$, is $2^{n-1} \times 2^{n-1} = 4^{n-1}$, and the number of the decomposition levels is n .

- The number of $\text{SVD}_{2 \times 2}$ in the first level is $M_1 = M = 4^{n-1}$;
- The number of $\text{SVD}_{2 \times 2}$ in the second level is $M_2 = 2 \times M = 2 \times 4^{n-1}$;
-
- The number of $\text{SVD}_{2 \times 2}$ in the level n is $M_n = 2^{n-1} \times M = 2^{n-1} \times 4^{n-1}$;

The total number of $\text{SVD}_{2 \times 2}$ is correspondingly $M_\Sigma = M(1 + 2 + \dots + 2^{n-1}) = 4^{n-1}(2^n - 1) = 2^{2n-2}(2^n - 1)$. Then the total number of the algebraic operations for the HSVD of size $2^n \times 2^n$ is:

$$O_{\text{HSVD}}(2^n \times 2^n) = M_\Sigma \times O_{\text{SVD}}(2 \times 2) = 27 \times 2^{2n-1}(2^n - 1). \quad (1.48)$$

1.4.4.3 Computational Complexity of the SVD of Size $2^n \times 2^n$

For the calculation of the matrices $[Y(N)]$ and $[Z(N)]$ of size $N \times N$ for $N = 2^n$ are needed in total $\Sigma_m = 2^{2n+2}$ multiplications and $\Sigma_s = 2^{n+1}(2^n - 1)$ additions. The total number of the operations is:

$$O_{Y,Z}(N) = 2^{2n+2} + 2^{n+1}(2^n - 1) = 2^{n+1}(3 \times 2^n - 1). \quad (1.49)$$

In accordance with [40], the number of the operations $O(N)$ for the iterative calculation of all N eigenvalues and the eigen N -component vectors of the matrix of size $N \times N$ for $N = 2^n$ with L iterations, is correspondingly:

$$\begin{aligned} O_{\text{val}}(N) &= (1/6)(N - 1)(8N^2 + 17N + 42) \\ &= (1/6)(2^n - 1)(2^{2n+3} + 17 \times 2^n + 42), \end{aligned} \quad (1.50)$$

$$O_{\text{vec}}(N) = N[2N(LN + L + 1) - 1] = 2^n[2^{n+1}(2^n L + L + 1) - 1]. \quad (1.51)$$

From Eq. (1.31) it follows, that two kinds of eigen vectors (\vec{U}_s and \vec{V}_s) should be calculated, so the number of the needed operations in accordance with Eq. (1.51) should be doubled. From the analysis of the Eq. (1.29) it follows that:

- The number of the needed multiplications for all components is: $\Sigma_m = 2^n(2^{2n} + 2^{2n}) = 2^{3n+1}$;
- The number of the needed additions for all components is: $\Sigma_s = 2^n - 1$.

Then the total number of the needed operations for the calculation of Eq. (1.29) is:

$$O_D(N) = 2^{3n+1} + 2^n - 1 = 2^n(2^{2n+1} + 1) - 1 = 2^n(2^{2n+1} + 1) - 1. \quad (1.52)$$

Hence, the total number of the algebraic operations, needed for the execution of the SVD of size $2^n \times 2^n$ is:

Table 1.2 Coefficient $\psi_1(n, L)$ of the relative reduction of the computational complexity of the HSVD versus the SVD as a function of n , for $L = 10$

n	2	3	4	5
$\psi_1(n, 10)$	5.94	4.21	3.67	3.44

$$\begin{aligned}
O_{SVD}(2^n \times 2^n) &= O_{Y,Z}(2^n) + O_{val}(2^n) + 2O_{vec}(2^n) + O_D(2^n) \\
&= 2^{2n+1}[2L(2^n + 1) + 2^{n-1} + 5] + (1/6)(2^{2n+3} + 17 \times 2^n + 42) - 1.
\end{aligned} \tag{1.53}$$

1.4.4.4 Relative Computational Complexity of the HSVD

The relative computational complexity of the HSVD could be calculated on the basis of Eqs. (1.53) and (1.48), using the relation below:

$$\begin{aligned}
\psi_1(n, L) &= \frac{O_{SVD}(2^n \times 2^n)}{O_{HSVD}(2^n \times 2^n)} \\
&= \frac{3 \times 2^{n+1}[2^{n+2}(2^n L + L + 1) + 2^{n+1}(2^n + 3) - 3] + (2^n - 1)(2^{2n+3} + 17 \times 2^n + 42) - 6}{81 \times 2^{2n}(2^n - 1)}.
\end{aligned} \tag{1.54}$$

For $n = 2, 3, 4, 5$ (i.e., for image blocks of size $4 \times 4, 8 \times 8, 16 \times 16$ and 32×32 pixels), the values of $\psi_1(n, L)$ for $L = 10$ are given in Table 1.2.

For big values of n the relation $\psi_1(n, L)$ does not depend on n and trends towards:

$$\psi_1(n, L)_{n \rightarrow \infty} \Rightarrow 0.1 \times (3L + 1). \tag{1.55}$$

Hence, for big values of n , when the number of the iterations $L \geq 4$, the relation $\psi_1(n, L) > 1$, and the computational complexity of the HSVD is lower than that of the SVD. Practically, the value of L is significantly higher than 4. For big values of n the coefficient $\psi_1(n, 10) = 3.1$ and the computational complexity of the HSVD is three times lower than that of the SVD.

1.4.5 Representation of the HSVD Algorithm Through Tree-like Structure

The tree-like structure of the HSVD algorithm of $n = 2$ levels, shown on Fig. 1.6, is built on the basis of the Eq. (1.45), for image block of size 4×4 . As it could be seen, this is a binary tree. For a block of size 8×8 , this binary tree should be of $n = 3$ levels.

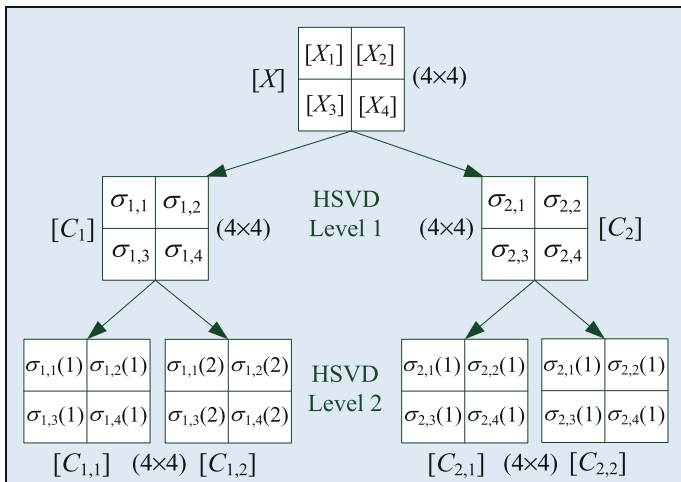


Fig. 1.6 Binary tree, representing the HSVD algorithm for the image matrix $[X]$, of size 4×4

Each tree branch has a corresponding eigen value $\lambda_{s,k}$, or resp. $\sigma_{s,k} = \sqrt{\lambda_{s,k}}$ for the level 1, and $\lambda_{s,k}(m)$ or resp. $\sigma_{s,k}(m) = \sqrt{\lambda_{s,k}(m)}$ —for the level 2 ($m = 1, 2$). The total number of the tree branches shown on Fig. 1.6, is equal to six. It is possible to cut off some branches, if for them the following conditions are satisfied: $\sigma_{s,k} \vee \sigma_{s,k}(m) = 0$ or $\sigma_{s,k} \leq \Delta_{s,k} \vee \sigma_{s,k}(m) \leq \Delta_{s,k}(m)$, i.e., when they are equal to 0, or are smaller than a small threshold $\Delta_{s,k}$, resp. $\Delta_{s,k}(m)$. To cut down one HSVD component $[C_i]$ in one level, it is necessary all values of σ_i , which participate in this component, to be equal to zero, or very close to it. In result, the decomposition in the corresponding branch could be stopped before the last level n . As a consequence, it follows that the HSVD algorithm is adaptive with respect of the contents of each image block. In this sense, the algorithm HSVD is adaptive and could be easily adjusted to the requirements of each particular application.

From the analysis of the presented HSVD algorithm it follows that its basic advantages compared to the “classic” SVD are:

1. The computational complexity of the full-tree HSVD algorithm (without truncation) for a matrix of size $2^n \times 2^n$, compared to SVD for a matrix of same size, is at least three times lower;
2. The HSVD is executed following the tree-like scheme of n levels, which permits parallel and recursive processing of image blocks of size 2×2 in each level. The corresponding SVD is calculated by using simple algebraic relations;
3. The HSVD algorithm retains the quality of the SVD in respect of the high concentration of the main part of the image energy in the first components of the decomposition. After removal of the low-energy components, the restored matrix has minimum mean square error and is the optimal approximation of the original;

4. The tree-like structure of the HSVD algorithm (a binary tree of n levels) makes more feasible the ability to stop the decomposition earlier in some of the tree branches, for which the corresponding eigen value is zero, or approximately zero. As a result, the computational complexity of the HSVD is additionally reduced, compared to the classic SVD;
5. The HSVD algorithm could be easily generalized for matrices of size different from $2^n \times 2^n$. In these cases each matrix should be divided into blocks of size 8×8 , to which is applied the HSVD (i.e., a decomposition of eight components). In case that after the division the blocks at the image borders are incomplete, they should be extended through extrapolation. Such approach is suitable in case, that the number of the decomposition components, which is limited up to 8, is sufficient for the application. If more components are needed, their number could be increased, by dividing the image into blocks of size 16×16 , or larger;
6. The HSVD algorithm opens new possibilities for fast image processing in various application areas, as: compression, filtration, segmentation, merging, watermarking, extraction of minimum number of features for pattern recognition, etc.

1.5 Hierarchical Adaptive Principal Component Analysis for Image Sequences

Image sequences are characterized with the huge volumes of visual information and very high spatial and spectral correlation. The decorrelation of this visual information is the first and basic stage of the processing, related to various publication areas, such as: compression and transfer/storage, analysis, objects recognition, etc. For the decorrelation of correlated image sequences, are developed significant number of methods for interframe prediction with movement compensation for temporal decorrelation of moving images and for transform-coding techniques for intra-frame and inter-frame decorrelation. One of the most efficient methods for decorrelation of groups of images is based on the Principal Component Analysis (PCA), known also as Hotelling transform, and Karhunen-Loeve Transform (KLT). This transform is the object of large number of investigations, presented in many scientific monographs [11, 40–47] and papers [12, 48–53]. The KLT is related to the class of linear statistical orthogonal transforms for groups of vectors, obtained, for example, from the pixels of one image, or from a group of matrix images. The PCA has significant role in image analysis and processing, and also in the systems for computer science and pattern recognition. It has a wide variety of application areas: for the creation of optimal models in the image color space [46], for compression of signals and groups of correlated images [41–44, 47], for the creation of objects descriptors in the reduced features' space [50, 51], for image fusion [52] and segmentation [53], image steganography [54], etc.

The PCA has some significant properties: (1) it is an optimal orthogonal transform for a group of vectors, because as a result of the transform, the maximum part of their energy is concentrated in a minimum number of their components; (2) after reduction of the low energy components of the transformed vectors, the corresponding restored vectors have minimum mean square error (MSE); (3) the components of the transformed vectors are not correlated. In particular, in case that the probability distribution of the vectors is Gaussian, their components become decorrelated and independent after PCA. The Independent Components Analysis (ICA) [55] is very close to the PCA in respect of their calculation and properties.

For PCA implementation the pixels of same spatial position in a group of N images compose an N -dimensional vector. The basic difficulty of the PCA implementation is related to the large size of the covariance matrix. For the calculation of its eigenvectors is necessary to calculate the roots of a polynomial of n th degree (characteristic equation) and to solve a linear system of N equations [21, 56]. For large values of N , the computational complexity of the algorithm for calculation of the transform matrix is significantly increased.

One of the basic problems, which limit the use of the PCA, is due to its high computational complexity, which grows up together with the number of the vectors' components. Various approaches are offered to overcome this problem. One of them is based on the PCA calculation through iterative methods, which do not require the definition of the characteristic polynomial of the vectors' covariance matrix. In this case the PCA is executed in 2 stages: in the first, the original image matrix is transformed into a three-diagonal form through QR decomposition [21], and after that—into a bi-diagonal, by using the Householder's transforms [27]. In the second stage, on the bi-diagonal matrix are applied iterative methods, for which the iterations are stopped, after the needed accuracy is achieved. The iterative PCA calculation through the methods of Jacobi and Givens [21, 56], is based on the execution of a sequence of orthogonal transforms with rotational matrices of size 2×2 .

One well known approach is based on the PCA calculation by using neural networks [28] of the kind Generalized Hebbian, or Multilayer Perceptron Networks. They both use iterative learning algorithms, for which the number of needed operations can reach several hundreds.

The third approach is based on the Sequential KLT/SVD [29], already commented in the preceding section. In [28, 29] is presented one more approach, based on the recursive calculation of the covariance matrix of the vectors, its eigen values and eigen vectors. In the papers [57, 58] is introduced hierarchical recursive block processing of matrices.

The next approach is based on the so-called Distributed KLT [59, 60], where each vector is divided into sub-vectors and on each is applied Partial KLT. Then is executed global iterative approximation of the KLT, through Conditional KLT, based on side information. This approach was further developed in [61], where is offered one algorithm for adaptive two-stage KLT, combined with JPEG2000, and aimed at the compression of hyper-spectral (HS) images. Similar algorithm for enhanced search is the "Integer Sub-optimal KLT" (Int SKLT) [62], which uses the

lifting factorization of matrices. This algorithm is basic for the KLT, executed through a multilevel strategy, also called Divide-and-Conquer (D&C) [63]. In correspondence with this approach, the KLT for a long sequence of images is executed after dividing it into smaller groups, for which the corresponding KLT have lower computational complexity. By applying the KLT on each group, is obtained local decorrelation only. For this reason, the eigen images for the first half of each group in the first decomposition level are used as an input for the next (second) level of the multi-level transform, etc. In the case, when the KLT group contains 2 components only, the corresponding multilevel transform is called Pair-wise Orthogonal Transform (POT) [64]. The experimental results obtained for this transform, when used for HS images, show that it is more efficient than the Wavelet Transform (WT) in respect of Rate-Distortion performance, computational cost, component scalability, and memory requirements.

Another approach is based on the Iterative Thresholding Sparse PCA (ITSPCA) [65] algorithm, aiming at the reduction of the features' space dimension, with minimum dispersion loss.

A fast calculation algorithm (Fast KLT) is known for the particular case, when the images are represented through first order Markov model [66].

In correspondence with the algorithm for PCA randomization [67], on the basis of an accidental choice are selected a certain number of rows (or columns) of the covariance matrix, and on the basis of this approximation, the computational complexity of the KLT is reduced.

In the works [68, 69], are presented hybrid methods for compression of multi-component images through KLT, combined with Wavelets, Adaptive Mixture of Principal Components Model, and JPEG2000.

The analysis of the famous KLT methods shows that: (1) In case of iterative calculations, the number of iterations depends on the covariance matrix of the vectors. In many cases this number is very high, which makes the real-time KLT calculation extremely difficult; (2) In case that the method for multilevel D&C is used, the eigen images from the second half of each group are not transformed in the next levels and as a result, they are not completely decorrelated. Moreover—the selection of the length of each group of images is not optimized.

One of the possible approaches for reducing the computational complexity of PCA for N-dimensional group of images is based on the so-called Hierarchical Adaptive PCA (HAPCA) [70]. Unlike the famous Hierarchical PCA (HPCA) [58], this transform is not related to the image sub-blocks, but to the whole image from one group. For this, the HPCA is implemented through dividing the images into groups of length, defined by their correlation range. Each group is divided into sub-groups of 2 or 3 images each, on which is applied Adaptive PCA (APCA) [71–73], of size 2×2 or 3×3 . This transform is performed using equations, which are not based on iterative calculations, and as a result, they have lower computational complexity. To obtain decorrelation for the whole group of images, it is necessary to use APCA of size 2×2 or 3×3 , which will be applied in several

consecutive stages (hierarchical levels), with rearranging of the obtained intermediate eigen images after each stage. In result, is obtained a decorrelated group of eigen images, on which could be applied other combined approaches to obtain efficient compression through lossless or lossy coding.

1.5.1 Principle for Decorrelation of Image Sequences by Hierarchical Adaptive PCA

The new principle was developed for the transformation of image sequences using the adaptive PCA (APCA) with transform matrix of size 2×2 or 3×3 . The sequence is divided into groups, whose length is harmonized with their correlation range. The corresponding algorithm comprises the following steps: (1) correlation analysis of the image sequence, in result of which is defined the length N of each group; (2) dividing the processed group into sub-groups of two or three images each, depending on the length of the group, (3) adding (when necessary) new interpolated images, which supplements the last sub-group up to two or three images; (4) defining the number of the hierarchical transform levels on the basis of the mutual decorrelation, which should be achieved, (5) executing of the HAPCA algorithm for each group from the image sequence. For this, on each sub-group of two or three images from the first hierarchical level of HAPCA, is applied Adaptive PCA (APCA) with matrix of size 2×2 or 3×3 . In result, are obtained 2 or 3 eigen images. After that, the eigen images are rearranged so that the first sub-group of 2 eigen images to comprise the first images from each group, the second group of 2 or 3 eigen images—the second images from each group, etc. To each group of intermediate eigen images in the first hierarchical level is applied in similar way the next APCA with a 2×2 or 3×3 matrix, on each sub-group of 2 or 3 eigen images. In result are obtained the corresponding new intermediate eigen images in the second hierarchical level. Then the eigen images are rearranged again so, that the first group of 2 or 3 eigen images contains the first images from each group before the rearrangement; the second group of 2 or 3 eigen images—the second image before the rearrangement, etc.

1.5.2 Description of the Hierarchical Adaptive PCA Algorithm

1.5.2.1 Calculation of Eigen Images Through APCA with a 2×2 Matrix

For any 2 digital images of size $S = M \times N$ pixels each, shown on Fig. 1.7, are calculated the vectors $\vec{C}_s = [C_{1s}, C_{2s}]^t$ for $s = 1, 2, \dots, S$.

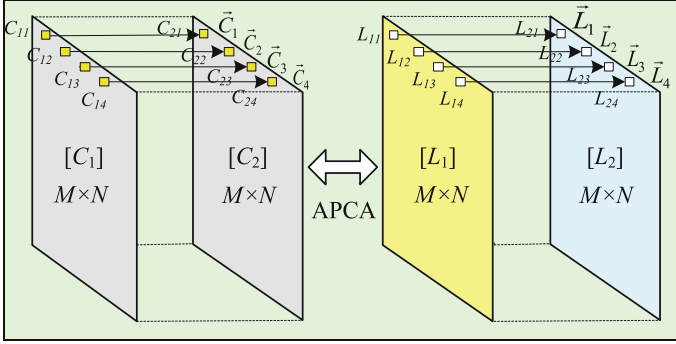


Fig. 1.7 Transformation of the images $[C_1]$, $[C_2]$ into eigen images $[L_1]$, $[L_2]$, through APCA

Each vector is transformed into the corresponding vectors $\vec{L}_s = [L_{1s}, L_{2s}]^t$ through direct APCA using the matrix $[\Phi]$ of size 2×2 in correspondence with the relation:

$$\vec{L}_s = [\Phi](\vec{C}_s - \vec{\mu}) \text{ for } s = 1, 2, \dots, S. \quad (1.56)$$

where

$$\vec{\mu} = E(\vec{C}_s) = (1/S) \sum_{s=1}^S \vec{C}_s = [\bar{C}_1, \bar{C}_2]^t; \quad \bar{C}_1 = E(C_{1s}); \quad \bar{C}_2 = E(C_{2s}); \quad [\Phi] = [\vec{\Phi}_1, \vec{\Phi}_2].$$

On the other hand, the components of the vectors $\vec{\Phi}_1$, $\vec{\Phi}_2$ could be defined using the rotation angle θ of the coordinate system (L_1, L_2) towards the original coordinate system (C_1, C_2) , resulting from the APCA execution. Then:

$$\vec{\Phi}_1 = [\cos \theta, -\sin \theta]^t; \quad \vec{\Phi}_2 = [\sin \theta, \cos \theta]^t, \quad (1.57)$$

where

$$\theta = (1/2) \arctg[2g_3/(g_1 - g_2)]; \quad g_3 = E(C_{1s}C_{2s}) - (\bar{C}_1)(\bar{C}_2);$$

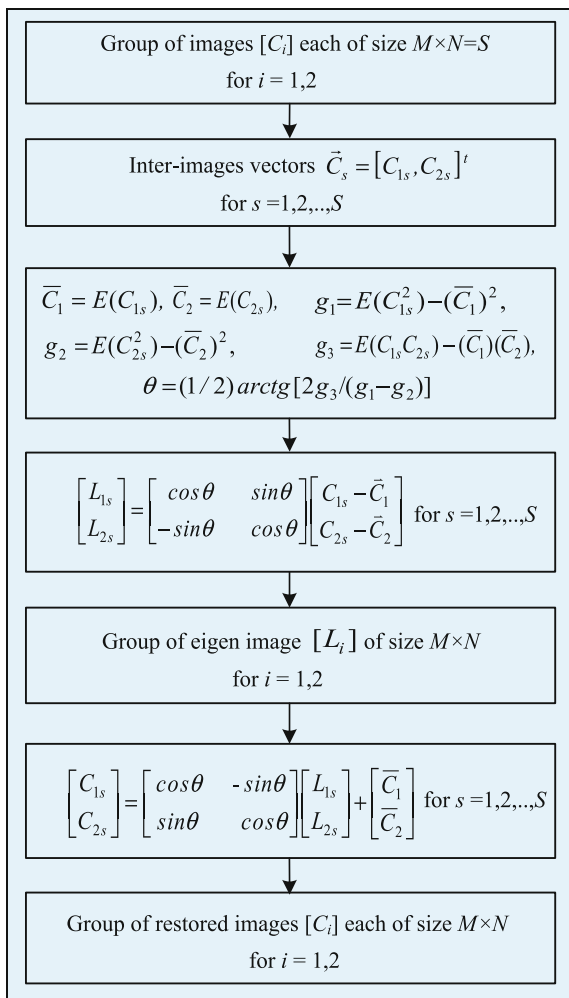
$$g_1 = E(C_{1s}^2) - (\bar{C}_1)^2; \quad g_2 = E(C_{2s}^2) - (\bar{C}_2)^2.$$

As a result of the transform from Eq. (1.56) on all S vectors, are obtained the corresponding two eigen images $[L_1]$, $[L_2]$, shown in the right part of Fig. 1.7. The transformation from Eq. (1.56) is reversible, and the inverse APCA is represented by the relation:

$$\vec{C}_s = [\Phi]^t \vec{L}_s + \vec{\mu} \text{ for } s = 1, 2, \dots, S. \quad (1.58)$$

On Fig. 1.8 is shown the algorithm for direct/inverse APCA for a group of two images.

Fig. 1.8 Algorithm for direct/inverse APCA for a group of two images



1.5.2.2 Hierarchical APCA Algorithm for a Group of 8 Images

On Fig. 1.9 is shown the 3-level HAPCA algorithm for the case, when the number of the correlated images in one group (GOI) is $N = 8$; in one sub-group it is $N_{sg} = 2$; and the number of the sub-groups is $N_g = 4$, i.e. $N = N_{sg} \times N_g$.

As it is shown on Fig. 1.9, on each sub-group of two images from the first hierarchical level of HAPCA is applied APCA with a 2×2 matrix. In result are obtained two “eigen” images, colored in yellow and blue correspondingly. After that, the “eigen” images are rearranged so that the first sub-group of two “eigen” images comprises the first images from each group, the second group of two “eigen” images—the second images from each group, etc. For each GOI of 8

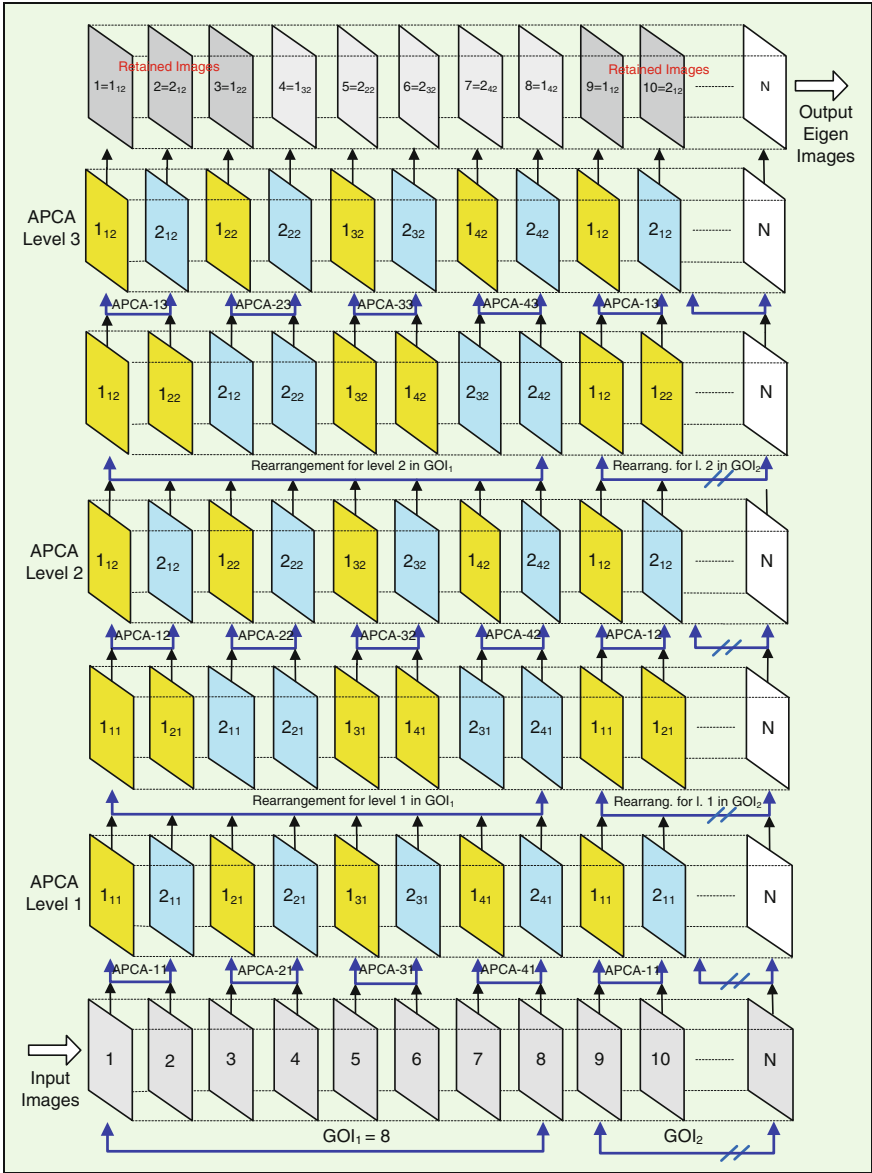


Fig. 1.9 The HAPCA algorithm for direct transform of groups (GOIs) of $N = 8$ images

intermediate eigen images in the first hierarchical level, is applied in similar way the next APCA, with a 2×2 matrix, on each sub-group of two eigen values. In result are obtained two new “eigen” images (i.e. the “eigen” images of the group of two intermediate eigen images), colored in yellow, and blue correspondingly in the

second hierarchical level. Then the eigen images are rearranged again, so that the first group of two “eigen” images to contain the first images from each group before the rearrangement; the second group of two “eigen” images—the second image before the rearrangement, etc. In result, is achieved significant decorrelation for the processed group of images, which is a reliable basis for their efficient compression/restoration. For this is necessary to have information about the transform matrix, used for each couple of images in all hierarchical levels—12 matrices for one GOI altogether (when $N = 8$).

1.5.2.3 Calculation of Eigen Images Through APCA with a 3×3 Matrix

From the three digital images of S pixels each, are obtained the vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$ for $s = 1, 2, \dots, S$. The vectors \vec{C}_s are transformed into the vectors $\vec{L}_s = [L_{1s}, L_{2s}, L_{3s}]^t$ through direct APCA, given in Eq. (1.56), and using the matrix $[\Phi]$ of size 3×3 .

The elements Φ_{ij} of the matrix $[\Phi]$ and the vector $\vec{\mu} = [\bar{C}_1, \bar{C}_2, \bar{C}_3]^t$ for $\bar{C}_1 = E(C_{1s}), \bar{C}_2 = E(C_{2s}), \bar{C}_3 = E(C_{3s})$ are defined below:

$$\Phi_{1m} = A_m/P_m; \Phi_{2m} = B_m/P_m; \Phi_{3m} = D_m/P_m \text{ for } m = 1, 2, 3, \quad (1.59)$$

where

$$\begin{aligned} A_m &= (k_3 - \lambda_m)[k_5(k_2 - \lambda_m) - k_4k_6], \quad B_m = (k_3 - \lambda_m)[k_6(k_1 - \lambda_m) - k_4k_5], \\ D_m &= k_6[2k_4k_5 - k_6(k_1 - \lambda_m)] - k_5^2(k_2 - \lambda_m), \quad P_m = \sqrt{A_m^2 + B_m^2 + D_m^2}; \\ k_1 &= E(C_{1s}^2) - (\bar{C}_1)^2, \quad k_2 = E(C_{2s}^2) - (\bar{C}_2)^2, \quad k_3 = E(C_{3s}^2) - (\bar{C}_3)^2; \\ k_4 &= E(C_{1s}C_{2s}) - (\bar{C}_1)(\bar{C}_2), \quad k_6 = E(C_{2s}C_{3s}) - (\bar{C}_2)(\bar{C}_3), \quad k_5 = E(C_{1s}C_{3s}) - (\bar{C}_1)(\bar{C}_3); \\ \lambda_1 &= 2\sqrt{\frac{|p|}{3}} \cos\left(\frac{\varphi}{3}\right) - \frac{a}{3}; \quad \lambda_2 = -2\sqrt{\frac{|p|}{3}} \cos\left(\frac{\varphi + \pi}{3}\right) - \frac{a}{3}; \quad \lambda_3 = -2\sqrt{\frac{|p|}{3}} \cos\left(\frac{\varphi - \pi}{3}\right) - \frac{a}{3}; \\ q &= 2(a/3)^3 - (ab)/3 + c, \quad p = -(a^2/3) + b, \quad \varphi = \arccos\left[-q/2 / \sqrt{(|p|/3)^3}\right]; \\ a &= -(k_1 + k_2 + k_3), \quad b = k_1k_2 + k_1k_3 + k_2k_3 - (k_4^2 + k_5^2 + k_6^2), \\ c &= k_1k_6^2 + k_2k_5^2 + k_3k_4^2 - (k_1k_2k_3 + 2k_4k_5k_6). \end{aligned}$$

The inverse APCA, using the matrix $[\Phi]$ of size 3×3 , is defined by Eq. (1.58).

1.5.2.4 Hierarchical APCA Algorithm for a Group of 9 Images

In this case, the HAPCA algorithm for a group of nine images $N = 9$ is executed in similar way, as that, shown on Fig. 1.9 for a group of eight images ($N = 8$). Each GOI is divided into $N_g = 3$ sub-groups, each containing $N_{sg} = 3$ images, and

the number of the HAPCA decomposition levels is $n = 2$. In the first HAPCA level, in accordance with Eq. (1.56) on the vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$ for each of the three sub-groups the APCA is executed. In this case, the elements of the matrix $[\Phi]$ of size 3×3 are defined by Eq. (1.59). In result, for each sub-group the vectors $\vec{L}_s = [L_{1s}, L_{2s}, L_{3s}]^t$ are calculated. After the rearrangement of the vectors components from all sub-groups and their second division into sub-groups of same size ($N_g = 3$), are obtained the corresponding input vectors $\vec{L}_s^1(r) = [L_{1s}^1(r), L_{2s}^1(r), L_{3s}^1(r)]^t$ for the next HAPCA level, etc.

1.5.3 Setting the Number of the Levels and the Structure of the HAPCA Algorithm

1.5.3.1 Number of the HAPCA Levels

The minimum number of levels n_{min} needed for the execution of the HAPCA algorithm for a group of N images could be defined through the analysis of the mutual correlation of the group of transformed N -dimensional vectors, obtained after each hierarchical level. For this, after the execution of the first HAPCA level for the transformed vectors \vec{L}_s for each sub-group (with two or three components), are obtained the N -dimensional vectors $\vec{L}_s^1 = [L_{1s}^1, L_{2s}^1, \dots, L_{N_s}^1]^t$. After the rearrangement of the components of each vector \vec{L}_s^1 , it is transformed into the vector $\vec{L}_s^1(r) = [L_{1s}^1(r), L_{2s}^1(r), \dots, L_{N_s}^1(r)]^t$. The decision to continue with the next (second) HAPCA is based on the analysis of the covariance matrix $[K_L^1(r)]$ of the rearranged vectors $\vec{L}_s^1(r)$ for $s = 1, 2, \dots, S$, from which could be calculated the achieved decorrelation in the first level. In case that full decorrelation is achieved, the matrix $[K_L^1(r)]$ is diagonal. The HAPCA algorithm could be stopped before the second level even if the decorrelation is not full, provided that the relation below is satisfied:

$$\left\{ \frac{\sum_{i=1}^N \sum_{j=1}^N [k_{i,j}(r)]_{|(i \neq j)}^2}{\sum_{i=1}^N \sum_{j=1}^N [k_{i,j}(r)]_{|(i=j)}^2} \right\} \leq \delta. \quad (1.60)$$

Here $k_{i,j}(r)$ is the element (i, j) of the matrix $[K_L^1(r)]$, and δ is a threshold with preliminary set small value. In case that the condition from Eq. (1.60) is not satisfied, the processing continues with the second HAPCA level. After all calculations are finished, the condition in Eq. (1.60) is checked again, but here $k_{i,j}(r)$ are the elements of the matrix $[K_L^2(r)]$ of the rearranged vectors $\vec{L}_s^2(r)$ in the second level, etc.

1.5.3.2 Structure of the HAPCA Algorithm

The structure of the HAPCA algorithm for one group (GOI) depends on the number of images (N) in it. This number is defined through correlation analysis of the whole image sequence, and most frequently it is in the range from 4, up to 16. In some cases, the number of images in the group is not divisible by the number of the images in a sub-group (N_g), which should be two or three, and then the number of the images N has to be extended by adding m_{int} interpolated images to the GOI. In result, the new value $N_e = N + m_{int}$ becomes divisible by two or three. In Table 1.3, are given the basic parameters of HAPCA for one GOI: N —number of images in the group, n —the number of transform levels, N_{sg} —the number of the sub-groups, N_g —the number of the images in one sub-group, N_e —the number of images in the extended GOI, and m_{int} —the number of the interpolated images in the extended GOI.

The number of the levels n in Table 1.3 is defined through correlation analysis of the whole GOI, and the values of N_{sg} and N_g —on the basis of the requirement for minimum value of the number of interpolated images, m_{int} .

1.5.3.3 Computational Complexity of HAPCA

The computational complexity of the n -levels HAPCA algorithm can be calculated and compared with the classic PCA for a covariance matrix of size $N \times N$ for group of N images with N_{sg} sub-groups for the APCA of size 2×2 or 3×3 . In case of classic PCA, this number is $n = N_{sg} = 1$, because there are no hierarchical levels or sub-groups. For this, both algorithms are compared regarding the number of operations O (additions and multiplications) [74] needed for the calculation of the following components:

Table 1.3 Basic parameters of the HAPCA algorithm

N	n	N_{sg}	N_g	$N_e = N_{sg} \times N_g$	$m_{int} = N_e - N$
4	2	2	2	4	0
5	3	3	2	6	1
6	3	3	2	6	0
7	3	4	2	8	1
8	3	4	2	8	0
9	2	3	3	9	0
10	3	4	3	12	2
11	3	4	3	12	1
12	3	4	3	12	0
13	3	5	3	15	2
14	3	5	3	15	1
15	3	5	3	15	0
16	5	8	2	16	0

- Covariance matrices $[K_C]$ of size $N \times N$ for the classic PCA algorithm and for the APCA with size of the transform matrix 2×2 or 3×3 [73]:

$$O_{cov}(N) = (1/2)N(N+1)[N(N-1) + 2(N+2)] \text{ for the classic PCA; } (1.61)$$

$$O_{cov}(2) = 30 \text{ for APCA of size } 2 \times 2 \text{ (} N = 2\text{); } (1.62)$$

$$O_{cov}(3) = 96 \text{ for APCA of size } 3 \times 3 \text{ (} N = 3\text{). } (1.63)$$

- Calculation of the eigen values of the corresponding $[K_C]$ covariance matrix when the QR decomposition and the Householder transform of $(N-1)$ steps are used for the classic PCA [73]:

$$O_{val}(N) = (N-1)\left(\frac{4}{3}N^2 + \frac{17}{6}N + 7\right) \text{ for classic PCA; } (1.64)$$

$$O_{val}(2) \approx 12 \text{ for APCA of size } 2 \times 2 \text{ (} N = 2\text{); } (1.65)$$

$$O_{val}(3) = 55 \text{ for APCA of size } 3 \times 3 \text{ (} N = 3\text{). } (1.66)$$

- Calculation of the eigen vectors of the corresponding $[K_C]$ covariance matrix in case that iterative algorithm with 4 iterations is used for the classic PCA [73]:

$$O_{vec}(N) = N[2N(4N+5) - 1] \text{ for classic PCA; } (1.67)$$

$$O_{vec}(2) = 102 \text{ for APCA of size } 2 \times 2 \text{ (} N = 2\text{); } (1.68)$$

$$O_{vec}(3) = 303 \text{ for APCA of size } 3 \times 3 \text{ (} N = 3\text{). } (1.69)$$

- The number of operations needed for the calculation of a group of N eigen images (each of S pixels), obtained in result of direct PCA transform for zero mean vectors, is:

$$O(N, S) = SN(2N - 1) \text{ for classic PCA; } (1.70)$$

$$O(2, S) = 6S \text{ for APCA of size } 2 \times 2 \text{ (} N = 2\text{); } (1.71)$$

$$O(3, S) = 15S \text{ for APCA of size } 3 \times 3 \text{ (} N = 3\text{). } (1.72)$$

- Using Eqs. (1.61)–(1.72) the total number of operations (TO) needed for both algorithms (the classic PCA and the HAPCA-based algorithms with APCA of size 2×2 or 3×3) is:

$$\begin{aligned}
TO_{PCA}(N, S) &= \frac{1}{2}N(N+1)[N-1] + 2(N+2) \\
&+ (N-1)\left(\frac{4}{3}N^2 + \frac{17}{6}N + 7\right) + N[2N(4N+5) - 1] \quad (1.73) \\
&+ SN(2N-1);
\end{aligned}$$

$$TO_{HAPCA-2}(N, S) = nN_{sg}(30 + 12 + 102 + 6S) = nN_g(144 + 6S); \quad (1.74)$$

$$TO_{HAPCA-3}(N, S) = nN_{sg}(96 + 55 + 303 + 15S) = nN_g(454 + 15S). \quad (1.75)$$

Having obtained the total number of operations required by the algorithms (1.73)–(1.75), we can compare the computational complexity of both the classic PCA and the proposed algorithms. The reduction of the number of operations needed for these algorithms can be described by the coefficient:

$$\eta_2(N, S) = \frac{TO_{PCA}(N, S)}{TO_{HAPCA-2}(N, S)} = \frac{O_{cov}(N) + O_{val}(N) + O_{vec}(N) + O(N, S)}{nN_{sg}[O_{cov}(2) + O_{val}(2) + O_{vec}(2) + O(2, S)]}, \quad (1.76)$$

is the ratio of the number of operations for the classic PCA and the proposed HAPCA-2 algorithm (with APCA of size 2×2), and:

$$\eta_3(N, S) = \frac{TO_{PCA}(N, S)}{TO_{HAPCA-3}(N, S)} = \frac{O_{cov}(N) + O_{val}(N) + O_{vec}(N) + O(N, S)}{nN_{sg}[O_{cov}(3) + O_{val}(3) + O_{vec}(3) + O(3, S)]}, \quad (1.77)$$

is the ratio of the number of operations for the classic PCA and the proposed HAPCA-3 algorithm (with APCA of size 3×3).

For example, for $N = 8$, $n = 3$ and $N_g = 4$, from Eq. (1.76) is obtained:

$$\eta_2(8, S) = \frac{TO_{PCA}(8, S)}{TO_{HAPCA-2}(8, S)} = \frac{8269 + 120S}{1730 + 72S}. \quad (1.78)$$

For $N = 9$, $n = 2$ and $N_g = 3$, from Eq. (1.77) it follows:

$$\eta_3(9, S) = \frac{TO_{PCA}(9, S)}{TO_{HAPCA-3}(9, S)} = \frac{11987 + 153S}{2724 + 90S}. \quad (1.79)$$

If $S = 2^{18}$, then $\eta_2(8, 2^{18}) = 1.66$ and $\eta_3(9, 2^{18}) = 1.7$, i.e., the coefficient $\eta(S)$ is at least 1.66 times larger than 1 for images of size 512×512 , or higher (in average, about 2 times). For higher values of N (for example, between 9 and 16), and for big values of S , the coefficient $\eta(S) > 2$.

1.5.4 Experimental Results

The presented experimental results are for sequences of multispectral (MS) images. As an example, was used the test MS sequence “balloons” shown on Fig. 1.10; on Fig. 1.11 is shown the corresponding color image with RGB values, obtained after lighting with neutral daylight. This sequence is from the free-access image database of the Columbia University, USA (<http://www1.cs.columbia.edu/CAVE/databases/multispectral/>). It contains $N = 15$ MS images of size 512×512 pixels, 16 *bpp*. On

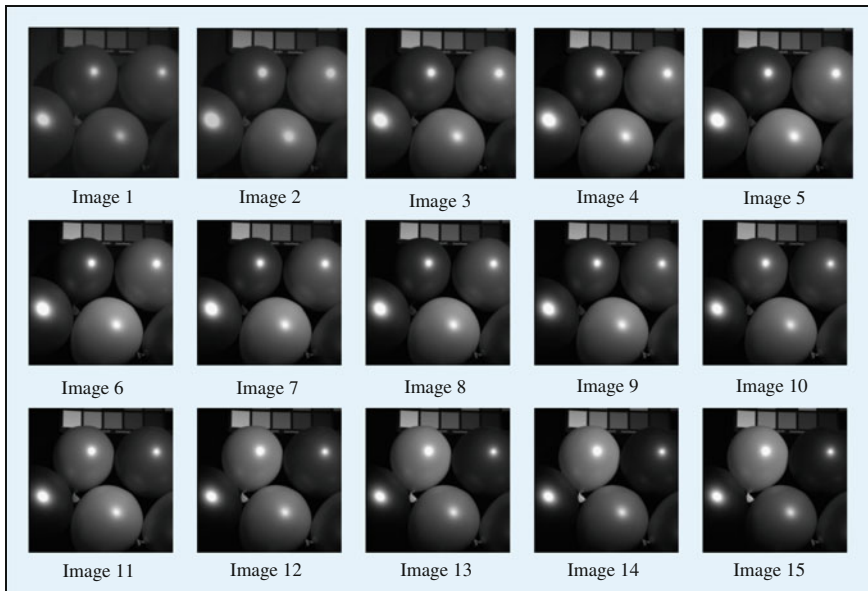
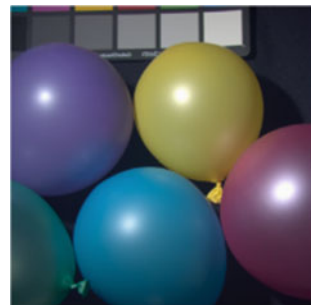


Fig. 1.10 Group of 15 consecutive MS images “balloons”

Fig. 1.11 The color image of “balloons”, obtained after lighting with neutral daylight



it was applied the 3-level HAPCA algorithm. The sequence was divided into $N_{sg} = 5$ sub-groups, each of $N_g = 3$ MS images, and the number of the vectors in each sub-group is $S = 2^{18}$.

On Fig. 1.12 are shown the corresponding eigen MS images, obtained after applying the 3-level HAPCA algorithm on the group of images. As it could be seen from the results shown on Fig. 1.13, the main part of the energy of these 15 images is concentrated on the first eigen MS image, and the energy of the next eigen images decreases rapidly.

The graphics on Fig. 1.13 represent the power distribution of all 15 eigen images in levels 1, 2, 3 before and after the rearrangement. In the first three eigen MS images are concentrated 99, 88 % of the total power of all 15 images in the GOI.

The basic qualities of the HAPCA algorithm for processing of groups of MS images are:

1. Lower computational complexity than PCA for the whole GOI, due to the lower complexity of APCA with matrices of size 2×2 and 3×3 compared to the case, when for the calculation of the PCA matrix are used iterative methods;
2. HAPCA could be used not only for efficient compression of sets of MS images, but also for sequences of medical CT images, video sequences, obtained from stationary TV camera, compression of multi-view images, image fusion, face recognition, etc.;

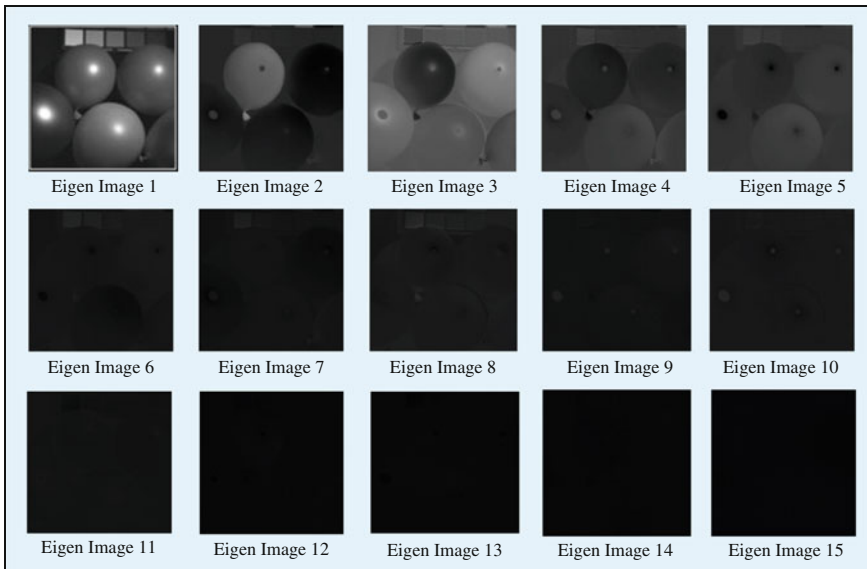


Fig. 1.12 Eigen images, obtained after executing the 3-levels HAPCA

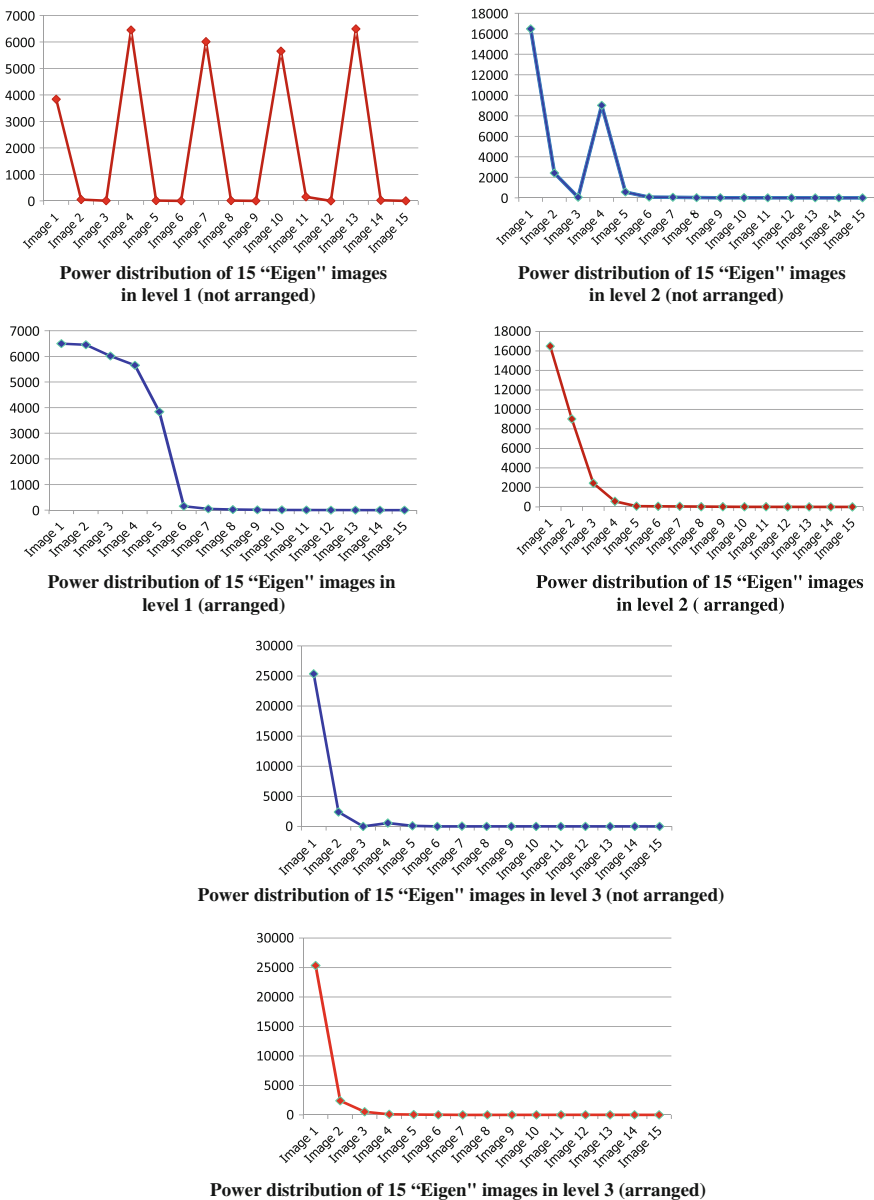


Fig. 1.13 Power distribution of all 15 eigen images in levels 1, 2, 3 before and after the rearrangement

3. There is also a possibility for further development of the HAPCA algorithms, through: the use of Integer PCA for lossless coding of MS images; HAPCA with a matrix of size $N \times N$ (N —a digit, divisible by 2 or 3), but without using numerical methods, etc.

1.6 Hierarchical Adaptive Kernel Principal Component Analysis for Color Image Segmentation

The color image segmentation is of high significance in computer vision as the first stage of the processing, concerning the detection and extraction of objects with predefined color, the shape of the visible part of the surface, and the texture. The existing color image segmentation techniques can be classified into seven main approaches based on: edge detection, region growing, neural network based, fuzzy logic, histogram analysis, Support Vector Machine and principal color [75–79]. One of the contemporary methods for color image segmentation is based on the adaptive models in the perceptual color space, using neural networks as multilayer perceptrons with multi-sigmoid activation function [80]. Recently special attention attracted the methods for human skin segmentation in color images [81–85]. These methods are mainly based on different color spaces, adaptive color space switching, skin color models and detection techniques.

The color space representation based on the PCA [86–88] offers significant advantages in the efficient image processing, as image compression and filtration, color segmentation, etc. In this section, a new approach for adaptive object color segmentation is presented through combining the linear and nonlinear PCA. The basic problem of PCA, which makes its application for efficient representation of the image color space relatively difficult, is related to the hypothesis for Gaussian distribution of the primary RGB vectors. One of the possible approaches for solving the problem is the use of PCA variations, such as: the nonlinear Kernel PCA (KPCA) [88], Fast Iterative KPCA [89], etc. In this section, for the color space representation an adaptive method for transform selection is used: linear PCA or nonlinear KPCA. The first transform (the linear PCA) could be considered as a particular case of the KPCA. The linear PCA is carried out on the basis of the already described Color Adaptive PCA (CAPCA) [85, 87]. The choice of CAPCA or KPCA is made through evaluation of the kind of distribution of the vectors, which describe the object color: Gaussian or not.

1.6.1 Mathematical Representation of the Color Adaptive Kernel PCA

In the general case, through KPCA is executed nonlinear transform of the original centered vectors \vec{X}_s over S pixels ($\vec{X}_s = \sum_{s=1}^S \bar{X}_s$) into the high-dimensional space, and then, for the obtained transformed vectors $\Phi(\vec{X}_s)$, the PCA is applied. The aim is, in the new multidimensional space the vertices of the vectors $\Phi(\vec{X}_s)$ to be concentrated in an area, which is accurately enough enveloped by a hyperellipsoid, whose axes

are the eigenvectors of the covariance matrix of the vectors $\Phi(\vec{X}_s)$. Figure 1.14 illustrates the idea of the new 3D color space of eigenvectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ [85].

In particular, it is possible that the vectors $\Phi(\vec{X}_s)$ in the transformed space are represented by their projections on the first eigenvector \vec{v}_1 of their covariance matrix, as shown in Fig. 1.15. For the example, shown in this figure, on the eigenvector \vec{v}_1 is projected the basic part of the multitude of all transformed vectors $\Phi(\vec{X}_s)$. The original 3D color vectors \vec{C}_s are first centered:

$$\vec{X}_s = \vec{C}_s - \vec{m}_C \text{ for } s = 1, 2, \dots, S, \tag{1.80}$$

where \vec{m}_C is the mean value of the color vector and then follows some kind of nonlinear transform, which uses the selected nonlinear function $\Phi(\cdot)$. In result, the corresponding N -dimensional vectors, $\Phi(\vec{X}_s)$ ($N \geq 3$) are obtained. The value of N depends on the selected function $\Phi(\cdot)$, used for the nonlinear transform [88].

The covariance matrix $[\tilde{K}_x]$ of the transformed color vectors $\Phi(\vec{X}_s)$ is of size $N \times N$ and can be calculated in accordance with the relation:

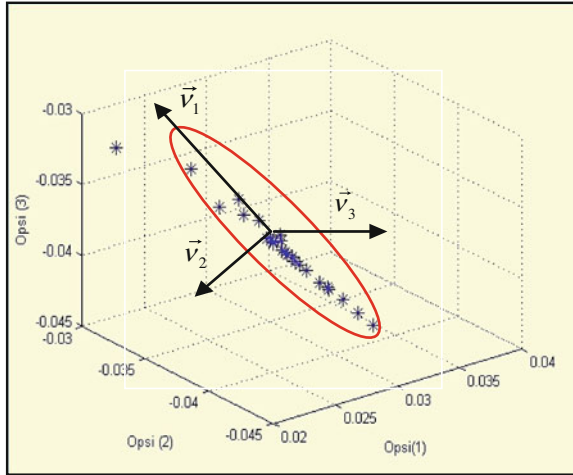
$$[\tilde{K}_x] = \frac{1}{S} \sum_{s=1}^S \Phi(\vec{X}_s) \cdot \Phi(\vec{X}_s)^t = E\{\Phi(\vec{C}_s - \vec{m}_c) \cdot \Phi(\vec{C}_s - \vec{m}_c)^t\}, \tag{1.81}$$

where $\Phi(\vec{X}_s) = [\Phi(x_{s1}), \Phi(x_{s2}), \dots, \Phi(x_{sN})]^t$ for $s = 1, 2, \dots, S$.

For each eigenvalue $\tilde{\lambda}_i$ and eigenvector $\vec{v}_i = [v_{i1}, v_{i2}, \dots, v_{iN}]^t$ of the matrix $[\tilde{K}_x]$ the following relation is performed:

$$[\tilde{K}_x] \vec{v}_i = \tilde{\lambda}_i \vec{v}_i \text{ for } i = 1, 2, \dots, N. \tag{1.82}$$

Fig. 1.14 Plot of skin color samples in the $\vec{v}_1, \vec{v}_2, \vec{v}_3$ eigenvectors space of CAPCA



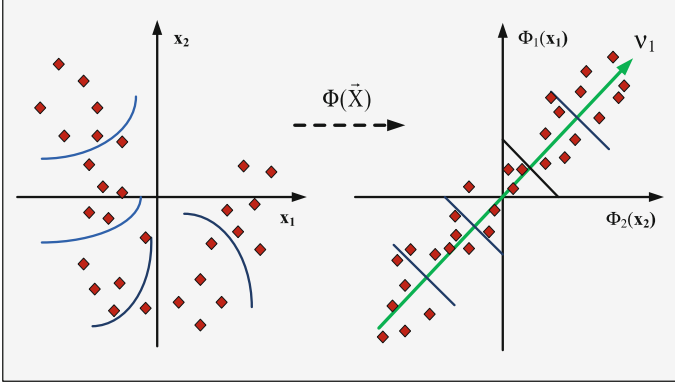


Fig. 1.15 Color space transform with KPCA

After substitution in Eq. (1.82) using Eq. (1.81), is got:

$$[\tilde{K}_x] \vec{v}_i = \frac{1}{S} \sum_{s=1}^S \Phi(\vec{X}_s) \Phi(\vec{X}_s)^t \vec{v}_i = \tilde{\lambda}_i \vec{v}_i \quad (1.83)$$

In result of the transformation of Eq. (1.83), known as the “kernel trick” [90], for the i th eigenvector is obtained:

$$\vec{v}_i = \frac{1}{S \tilde{\lambda}_i} \sum_{s=1}^S [\Phi(\vec{X}_s)^t \cdot \vec{v}_i] \Phi(\vec{X}_s) = \sum_{s=1}^S \alpha_{si} \Phi(\vec{X}_s), \quad (1.84)$$

where for $\tilde{\lambda}_i \neq 0$ the coefficient $\alpha_{si} = \frac{\Phi(\vec{X}_s)^t \cdot \vec{v}_i}{S \tilde{\lambda}_i}$.

From this, it follows, that:

$$[\tilde{K}_x] \vec{v}_i = \tilde{\lambda}_i \vec{v}_i = \tilde{\lambda}_i \sum_{s=1}^S \alpha_{si} \Phi(\vec{X}_s). \quad (1.85)$$

Substituting Eq. (1.84) in Eq. (1.83), is obtained:

$$\left[\frac{1}{S} \sum_{s=1}^S \Phi(\vec{X}_s) \Phi(\vec{X}_s)^t \right] \times \left[\sum_{l=1}^S \alpha_{il} \Phi(\vec{X}_l) \right] = \tilde{\lambda}_i \sum_{l=1}^S \alpha_{li} \Phi(\vec{X}_l)$$

or

$$\frac{1}{S} \sum_{s=1}^S \sum_{l=1}^S \Phi(\vec{X}_s) \Phi(\vec{X}_s)^t \Phi(\vec{X}_l) \alpha_{il} = \tilde{\lambda}_i \sum_{l=1}^S \alpha_{li} \Phi(\vec{X}_l),$$

from which follows:

$$\sum_{s=1}^S \sum_{l=1}^S \Phi(\vec{X}_s) \Phi(\vec{X}_s)^t \Phi(\vec{X}_l) \alpha_{li} = S \tilde{\lambda}_i \sum_{l=1}^S \alpha_{li} \Phi(\vec{X}_l). \quad (1.86)$$

After multiplying the left side of the above equation with the vector $\Phi(\vec{X}_s)^t$, is obtained:

$$\sum_{s=1}^S \sum_{l=1}^S \Phi(\vec{X}_s)^t \Phi(\vec{X}_s) \Phi(\vec{X}_s)^t \Phi(\vec{X}_l) \alpha_{li} = S \tilde{\lambda}_i \sum_{l=1}^S \alpha_{li} \Phi(\vec{X}_l)^t \Phi(\vec{X}_s). \quad (1.87)$$

The dot product of the vectors $\Phi(\vec{X}_s)$ and $\Phi(\vec{X}_l)$ could be represented through the kernel function $k(\vec{X}_s, \vec{X}_l)$, defined by the relation:

$$k(\vec{X}_s, \vec{X}_l) = \Phi(\vec{X}_s)^t \cdot \Phi(\vec{X}_l) \text{ for } s, l = 1, 2, \dots, S. \quad (1.88)$$

Here, the term $k(\vec{X}_s, \vec{X}_l)$ represents the elements (s, l) of the Gram matrix $[\mathbf{K}]$ of size $S \times S$, called “kernel matrix”. After substituting Eq. (1.88) in Eq. (1.87), is obtained:

$$[\mathbf{K}]^2 \cdot \vec{\alpha}_i = S \tilde{\lambda}_i [\mathbf{K}] \vec{\alpha}_i. \quad (1.89)$$

Under the condition, that the matrix $[\mathbf{K}]$ is positively defined (i.e. when it eigenvalues are positive) is got a shorter representation than in Eq. (1.89). Then:

$$[\mathbf{K}] \vec{\alpha}_i = S \tilde{\lambda}_i \vec{\alpha}_i. \quad (1.90)$$

From this relation it follows, that $S \tilde{\lambda}_i$ are the eigenvalues of the matrix $[\mathbf{K}]$, and $\vec{\alpha}_i = [\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{iS}]^t$ are the corresponding eigenvectors of same matrix. Taking into account the requirement $\vec{v}_i^t \vec{v}_i = 1$, from Eq. (1.84) is obtained the relation:

$$\sum_{s=1}^S \sum_{l=1}^S \alpha_{li} \alpha_{si} \Phi(\vec{X}_l)^t \cdot \Phi(\vec{X}_s) = 1 \text{ or } \vec{\alpha}_i^t [\mathbf{K}] \vec{\alpha}_i = 1. \quad (1.91)$$

After substituting Eq. (1.90) in Eq. (1.91) is obtained $S \tilde{\lambda}_i \vec{\alpha}_i^t \vec{\alpha}_i = 1$, from which is defined the square of the module of the vector $\vec{\alpha}_i = [\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{iS}]^t$:

$$\|\vec{\alpha}_i\|^2 = \vec{\alpha}_i^t \cdot \vec{\alpha}_i = \sum_{s=1}^S \alpha_{si}^2 = 1/S \tilde{\lambda}_i. \quad (1.92)$$

In the general case, the vectors $\Phi(\bar{X}_s)$ in Eq. (1.88) are not centered. In order to apply the PCA on them, they should be centered in advance, and in result are obtained the vectors:

$$\check{\Phi}(\bar{X}_s) = \Phi(\bar{X}_s) - E\{\Phi(\bar{X}_s)\}, \quad (1.93)$$

where $\check{m}_{\phi}^- = E\{\Phi(\bar{X}_s)\} = \frac{1}{S} \sum_{s=1}^S \Phi(\bar{X}_s)$.

The covariance matrix $[\check{\mathbf{K}}]$ of the centered vectors $\check{\Phi}(\bar{X}_s)$ is of size $S \times S$ and is defined by the relation:

$$[\check{\mathbf{K}}] = \frac{1}{S} \sum_{s=1}^S \check{\Phi}(\bar{X}_s)^t \cdot \check{\Phi}(\bar{X}_s) = E\{\check{\Phi}(\bar{X}_s)^t \cdot \check{\Phi}(\bar{X}_s)\}. \quad (1.94)$$

The matrix kernel is:

$$\check{k}(\bar{X}_s, \bar{X}_l) = \check{\Phi}(\bar{X}_s)^t \cdot \check{\Phi}(\bar{X}_l) = [\Phi(\bar{X}_s) - \check{m}_{\phi}^-]^t \cdot [\Phi(\bar{X}_l) - \check{m}_{\phi}^-]. \quad (1.95)$$

The relation between the covariance matrices $[\check{\mathbf{K}}]$ and $[\mathbf{K}]$ is:

$$[\check{\mathbf{K}}] = [\mathbf{K}] - 2[\mathbf{I}_{1/S}][\mathbf{K}] + [\mathbf{I}_{1/S}][\mathbf{K}][\mathbf{I}_{1/S}], \quad (1.96)$$

where $[\mathbf{I}_{1/S}]$ is a matrix of size $S \times S$, whose elements are equal to $1/S$.

The projection of the vector $\Phi(\bar{X}_s)$ on the eigenvector \bar{v}_i in the S -dimensional space is:

$$Pr_{si} = \Phi(\bar{X}_s)^t \cdot \bar{v}_i = \sum_{s=1}^S \alpha_{is} \Phi(\bar{X}_i)^t \cdot \Phi(\bar{X}_s) = \sum_{s=1}^S \alpha_{is} k(\bar{X}_i, \bar{X}_s) \text{ for } i = 1, 2, 3, \dots, N. \quad (1.97)$$

Using the projections Pr_{si} of the vector $\Phi(\bar{X}_s)$ on each of the first $k \leq N$ eigenvectors \bar{v}_i (for $i = 1, 2, \dots, k$), could be taken the decision for the classification of the s th pixel to the dominant color of the selected object, using some of the well-known classifiers, as: SVM, LDA, k -nearest neighbors, neural networks, etc. [89]

To carry out the KPCA, one could use different kinds of kernel functions, such as the polynomial, the Gaussian, the sigmoid, etc. By substituting $\Phi(\bar{X}_s) = \bar{x}$ and $\Phi(\bar{X}_l) = \bar{y}$ the polynomial kernel function of degree d is defined by the relation:

$$k(\vec{x}, \vec{y}) = (\vec{x}^t \cdot \vec{y})^d. \quad (1.98)$$

For $d = 2$ and if assumed that for the transformation of the 3-component vectors $\vec{X}_s = [x_{s1}, x_{s2}, x_{s3}]^t$ and $\vec{X}_l = [x_{l1}, x_{l2}, x_{l3}]^t$ into N -component is used the nonlinear function $\Phi(\cdot)$, then:

$$\vec{x} = \Phi(\vec{X}_s) = [\Phi_{s1}, \Phi_{s2}, \dots, \Phi_{sN}]^t, \quad \vec{y} = \Phi(\vec{X}_l) = [\Phi_{l1}, \Phi_{l2}, \dots, \Phi_{lN}]^t \quad (1.99)$$

where the vectors components are defined by the relations:

$$\Phi_{si} = x_{s_i p_1}^{r_1} x_{s_i p_2}^{r_2}, \quad \Phi_{li} = x_{l_i p_1}^{r_1} x_{l_i p_2}^{r_2} \quad (1.100)$$

for $r_1, r_2 = 0, 1, 2, 3, i = 1, 2, \dots, N$ and $s, l = 1, 2, \dots, S$.

In this case the maximum value of N is $N = 9$. In order to reduce the needed calculations, it is suitable to use smaller number of the possible 9 components of the quadratic function $\Phi(\cdot)$.

For example, if assumed $N = 3$ and if only mixed products of the vectors components \vec{X}_s and \vec{X}_l are chosen, then from Eq. (1.100) it follows:

$$\vec{x} = \Phi(\vec{X}_s) = [x_{s1}x_{s2}, x_{s1}x_{s3}, x_{s2}x_{s3}]^t, \quad \vec{y} = \Phi(\vec{X}_l) = [x_{l1}x_{l2}, x_{l1}x_{l3}, x_{l2}x_{l3}]^t. \quad (1.101)$$

Then the corresponding kernel function of vectors $\Phi(\vec{X}_s)$ and $\Phi(\vec{X}_l)$ is represented by the polynomial below:

$$k(\vec{x}, \vec{y}) = [\Phi_{s1}, \Phi_{s2}, \Phi_{s3}]^t \cdot [\Phi_{l1}, \Phi_{l2}, \Phi_{l3}] = x_{s1}x_{s2}x_{l1}x_{l2} + x_{s1}x_{s3}x_{l1}x_{l3} + x_{s2}x_{s3}x_{l2}x_{l3}. \quad (1.102)$$

In particular, for $d = 1$, $\Phi(\vec{X}_s) = \vec{X}_s$ and $\Phi(\vec{X}_l) = \vec{X}_l$ the corresponding kernel function is linear:

$$k(\vec{x}, \vec{y}) = [x_{s1}, x_{s2}, x_{s3}]^t \times [x_{l1}, x_{l2}, x_{l3}] = x_{s1}x_{l1} + x_{s2}x_{l2} + x_{s3}x_{l3}. \quad (1.103)$$

From the above, it follows that KPCA is transformed into linear PCA (i.e. PCA is a particular case of KPCA).

1.6.2 Algorithm for Color Image Segmentation by Using HAKPCA

The general algorithm for objects segmentation in the extended color space, based on the Hierarchical Adaptive Kernel PCA (HAKPCA) and the classifier of the reduced vectors, is shown on Fig. 1.16.

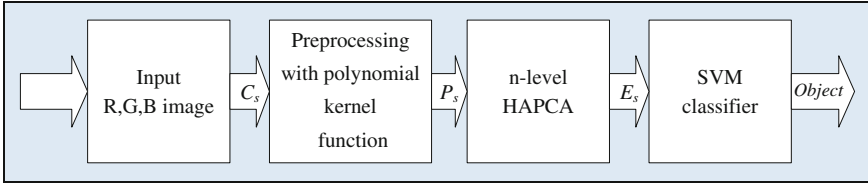


Fig. 1.16 Block diagram of the algorithm for image segmentation in the expanded color space

In the preprocessing block, each color vector $\vec{C}_s = [R_s, G_s, B_s]^t$ is transformed into the corresponding expanded vector \vec{P}_s . If the chosen kernel-function is polynomial, and the 3D color space is transformed into a 9-dimensional, then the components p_{is} of vectors \vec{P}_s could be defined as follows:

$$\begin{aligned} \vec{P}_s &= [R_s, G_s, B_s, R_s^2, G_s^2, B_s^2, R_s G_s, B_s G_s, R_s B_s]^t \\ &= [P_{1s}, P_{2s}, P_{3s}, P_{4s}, P_{5s}, P_{6s}, P_{7s}, P_{8s}, P_{9s}]^t \text{ for } s = 1, 2, \dots, S. \end{aligned}$$

In order to put all components p_{is} in the range $[0, 255]$, for $i = 4, 5, \dots, 9$: $R_s^2, G_s^2, B_s^2, R_s G_s, B_s G_s, R_s B_s$, are normalized in the range $0-255$. The vectors \vec{P}_s are then transformed by the 2-level HAKPCA, whose algorithm is shown in Fig. 1.17. As a result of the transform are obtained the 2-component vectors $\vec{E}_s = [E_{1s}, E_{2s}]^t$, which are used to substitute the input 9-components vectors $\vec{P}_s = [P_{1s}, P_{2s}, P_{3s}, P_{4s}, P_{5s}, P_{6s}, P_{7s}, P_{8s}, P_{9s}]^t$. In this way the performance of the classifier is also simplified, because it has to process the vectors \vec{E}_s in the two-component, instead of the nine-dimensional space.

At its output are separated all pixels in the image, whose corresponding vectors \vec{E}_s are in the area of the cluster, belonging to the object. With this, the color segmentation is finished. In accordance with the algorithm shown in Fig. 1.17, for the 2-level HAKPCA [91], the nine components of each input vector \vec{P}_s are divided into three groups, which contain the three-components vectors

$$\vec{P}_{1s} = [P_{11s}, P_{12s}, P_{13s}]^t, \vec{P}_{2s} = [P_{21s}, P_{22s}, P_{23s}]^t, \vec{P}_{3s} = [P_{31s}, P_{32s}, P_{33s}]^t,$$

At the first level of HAKPCA, on each group of the three-component vectors $\vec{P}_{ks} = [P_{k1s}, P_{k2s}, P_{k3s}]^t$ for $k = 1, 2, 3$, is performed color APCA with a transform matrix of size 3×3 . The so obtained vectors from each group comprise three “eigen” images, shown in Fig. 1.18. These images are rearranged in accordance to the rule:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_9. \quad (1.104)$$

where $\lambda_l \geq 0$ for $l = 1, 2, \dots, 9$ are eigen values of the covariance matrices of the three-component vectors \vec{P}_{ks} for each group ($k = 1, 2, 3$) in the first level of

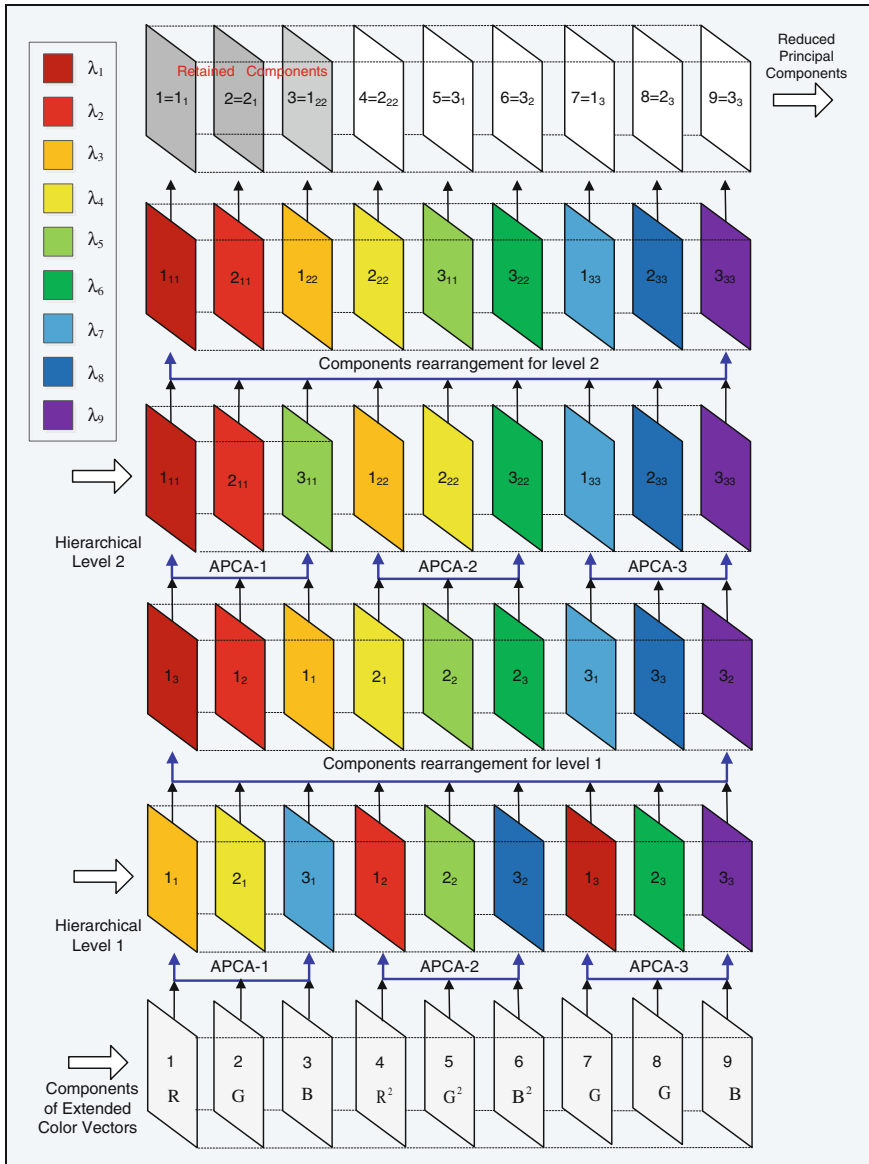


Fig. 1.17 HAKPCA algorithm for direct transform of the extended color image with components $R, G, B, R^2, G^2, B^2, RG, BG, RB$

HAKPCA, arranged as monotonously decreasing sequence of eigen values. After that these components are divided again, this time into 3 groups, of 3 images each.

The vectors, obtained from the pixels with same coordinates in the images from each group, are of 3 components. For the second level of HAKPCA for each group

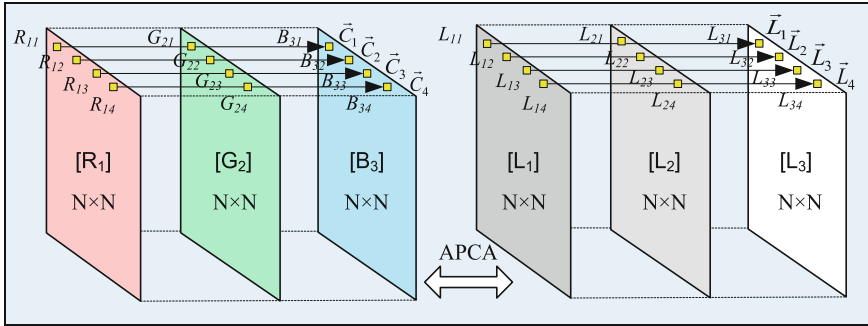


Fig. 1.18 Color APCA transform of the $[R]$, $[G]$, $[B]$ components of the color image

of 3-component vectors is executed direct APCA with a transform matrix of size 3×3 .

The algorithm for direct/inverse transform of the components $[R]$, $[G]$, $[B]$ of the color image, calculated using APCA in three “eigen” images $[L_1]$, $[L_2]$, $[L_3]$, is shown on Fig. 1.19. The vectors from each group build the three eigen images, which are rearranged again in accordance with Eq. (1.25). As a result, the nine eigen images $E_1 \sim E_9$ are obtained, from which are retained the first two (E_1 and E_2) only, which carry the main information, needed for the color objects segmentation.

As a result, the computational complexity of HAKPCA is lower than that of the KPCA, for the case, when it is used to transform directly the 9-component vectors \vec{P}_s . In this way, the general computational complexity of HAKPCA with a classifier, needed for the processing of the vectors \vec{P}_s is lower than that, needed for the processing of same vectors with KPCA with a classifier. From the pixels with same coordinates in the images E_1 and E_2 are obtained the vectors $\vec{E}_s = [E_{1s}, E_{2s}]^t$, which are then used by the classifier.

1.6.3 Experimental Results

To verify the feasibility of the proposed algorithm, skin pigmentation images were tested and evaluated. Figures 1.20 and 1.21 show the original tested images and their color vectors distribution in the RGB space, respectively. It can be seen that their color distributions are considered as non-linear Gaussian ones.

These images are passed through the HAKPCA algorithm (shown on Fig. 1.17). The obtained transformed vectors \vec{E}_s in the new color space E_{1s}, E_{2s}, E_{3s} are plotted in the 3D domain shown in Fig. 1.22. It is easy to notice that the proposed techniques concentrate the energy of the different skin color into very small and close components of transformed vectors.

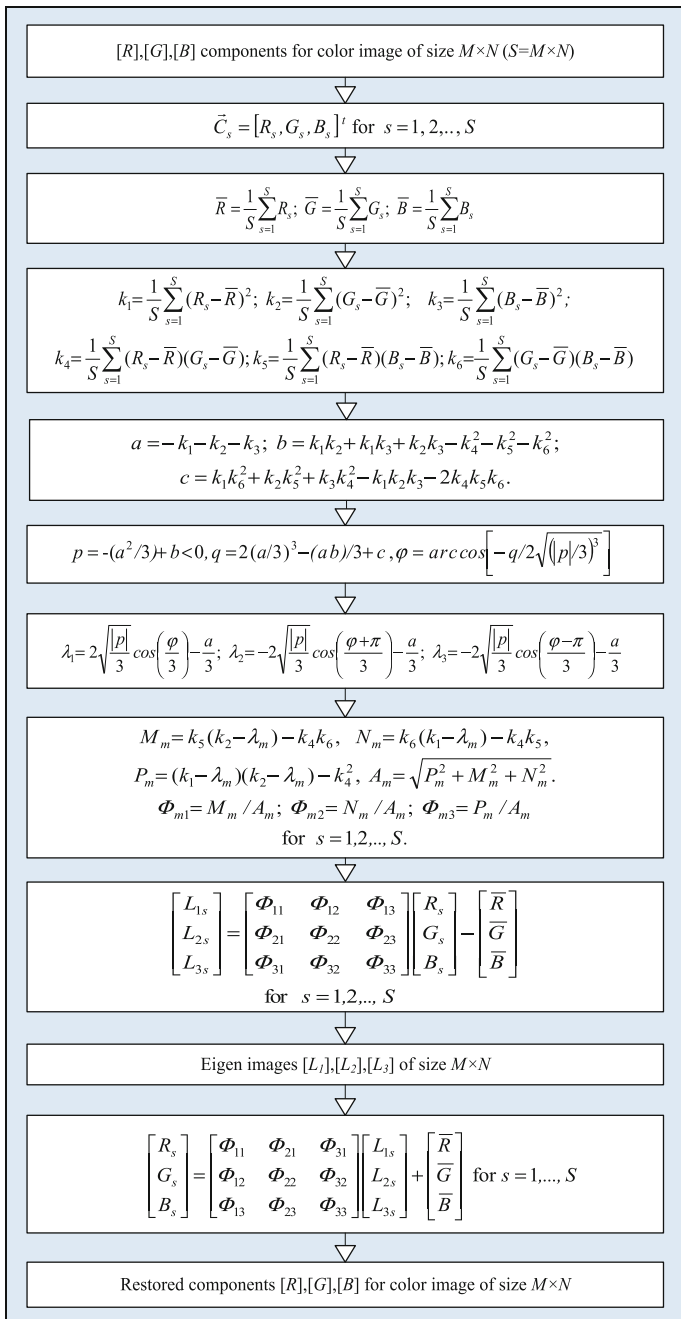


Fig. 1.19 Algorithm for direct/inverse APCA of the components [R], [G], [B] of the color image

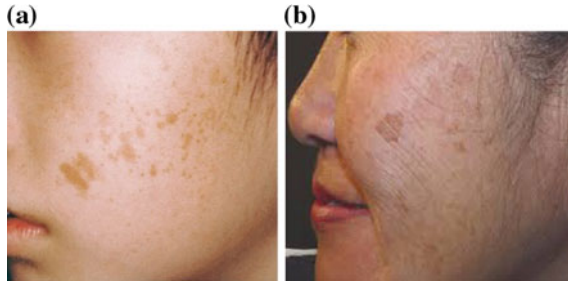


Fig. 1.20 a, b Original skin pigmentation images

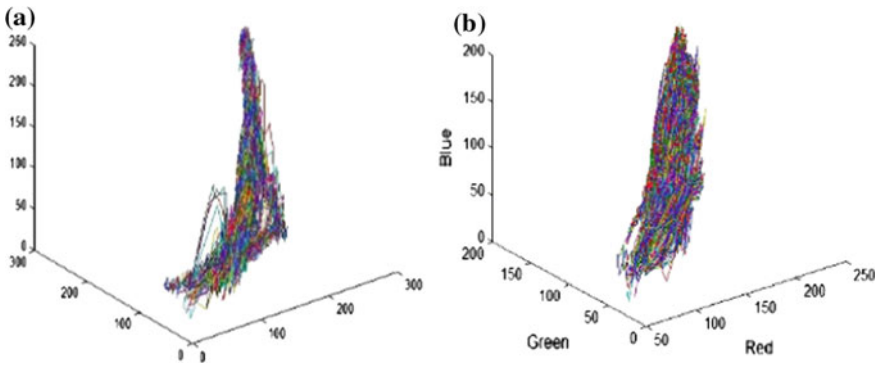


Fig. 1.21 a, b Color vectors distribution in RGB space for original images in Fig. 1.20 a, b

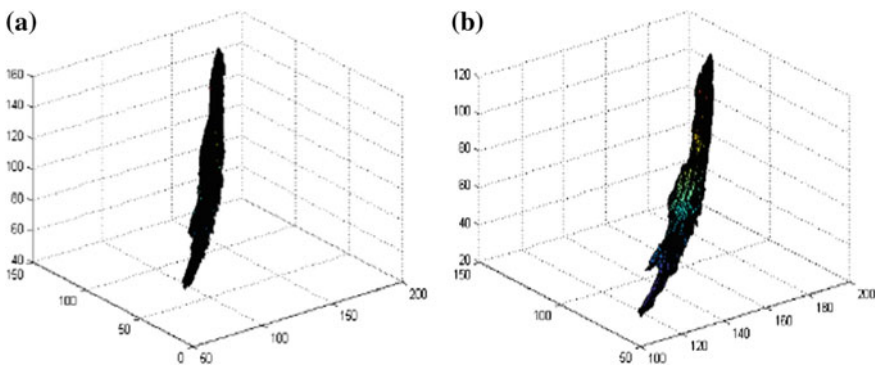
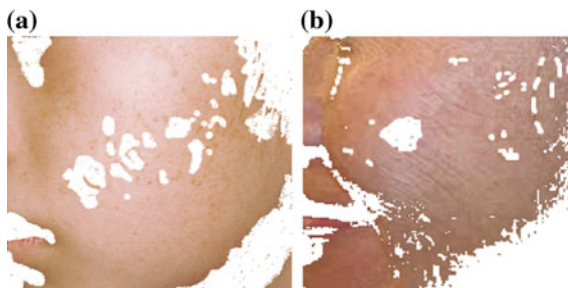


Fig. 1.22 a, b. Distribution of the transformed vectors \vec{E}_s in the new color space E_{1s}, E_{2s}, E_{3s}

Fig. 1.23 Skin color segmentation based on HAKPCA



The HAKPCA transformed coefficients are then used to train a classifier. For briefing, fuzzy K-means clustering is used. The segmentation results are shown in Fig. 1.23a, b respectively.

The proposed approach depends mainly on the evaluation of the color vectors distribution. For a non-Gaussian distribution of the vectors, is used HAKPCA. The selected nonlinear transform results in negligible expansion of the original color space, which increases slightly the number of needed calculations. The main advantage of the new approach is that in result of its adaptation in respect to the color vectors distribution, it could be used as universal tool for efficient image processing. One more advantage of HAKPCA towards the KPCA is the lower computational complexity.

On the basis of the presented approach, new algorithm for objects color segmentation was developed, which was distinguished by its high accuracy. This algorithm could be used in the CBIR systems for extraction of objects with preset color, in the computer vision systems for detection and tracking of objects in correspondence to their color under changing surveillance conditions, for automatic control of various manufacturing processes, etc.

1.7 Conclusions

The new approaches for image decomposition, presented in this chapter, are distinguished from the well-known methods by their low computational complexity and high efficiency, which makes them very attractive for future applications in computer vision, video communications, image content protection through digital Watermarking [92], image search in large databases, etc. Besides, the methods for hierarchical decomposition could be also used for the creation of new hybrid algorithms [93, 94] for processing of some specific kinds of images (static, or sequences), of the kind: medical, multi- and hyper spectral, multiview, panoramic satellite photos, etc. Depending on the requirements of the applications, in respect of their efficiency and the computation time needed for the processing, for each image kind could be selected the most suitable from the four decompositions

(BIDP, HSVD, HAPCA and HAKPCA) i.e., they can complement each other and have their own place and significance.

The future development of the presented new algorithms will be focused at the investigation of the possibilities for their integration in the contemporary systems for parallel processing, analysis and image recognition.

References

1. Rao, R., Kim, D., Hwang, J.: Fast Fourier Transform: Algorithms and Applications. Springer, Heidelberg (2010)
2. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Prentice Hall, NJ (2008)
3. Do, M., Vetterli, M.: Contourlets: a directional multiresolution image representation. In: Proceedings of International Conference on Image Processing, 2002, vol. 1, pp. 357–360 (2002)
4. Kountchev, R., Vl, Todorov, Kountcheva, R.: Linear and non-linear inverse pyramidal image representation: algorithms and applications, Chapter. In: Kountchev, R., Nakamatsu, K. (eds.) Advances in Reasoning-based Image Processing, Analysis and Intelligent Systems, pp. 35–89. Springer, Berlin (2012)
5. Orfanidis, S.: SVD, PCA, KLT, CCA, and all that. In: Optimum Signal Processing, pp. 332–525. Rutgers University (2007)
6. Ahuja, N.: On approaches to polygonal decomposition for hierarchical image representation. *Comput. Vis. Graph. Image Proc.* **24**, 200–214 (1983)
7. Tadmor, E., Nezzar, S., Vese, L.: Multiscale hierarchical decomposition of images with applications to deblurring, denoising and segmentation. *Commun. Math. Sci.* **6**, 281–307 (2008)
8. Hidane, M., Lezoray, O., Ta, V., Elmoataz, A.: Nonlocal multiscale hierarchical decomposition on graphs. In: Daniilidis K, Maragos P, Paragios N (eds.) Computer Vision—ECCV 2010, Part IV, pp. 638–650, LNCS 6314. Springer, Berlin (2010)
9. Wu, Q., Xia, T., Yu, Y.: Hierarchical tensor approximation of multidimensional images. In: IEEE International Conference on Image Processing (ICIP'07), San Antonio, TX, 6, IV-49, IV-52 (2007)
10. Cherkashyn, V., He, D., Kountchev, R.: Image decomposition on the basis of an inverse pyramid with 3-layer neural networks. *J. Commun. Comput.* **6**(11), 1–9 (2009)
11. Mach, T.: Eigenvalue algorithms for symmetric hierarchical matrices. Dissertation, Chemnitz University of Technology, March 13, 175p (2012)
12. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. *Physica D* **60**, 259–268 (1992)
13. Kountchev, R.: New Approaches in Intelligent Image Processing, 322p. WSEAS Press, Athene (2013)
14. Kountchev, R., Kountcheva, R.: Image representation with reduced spectrum pyramid. In: Tshirintzis, G., Virvou, M., Howlett, R., Jain, L. (eds.) New Directions in Intelligent Interactive Multimedia, pp. 275–284. Springer, Berlin, Heidelberg (2008)
15. Kountchev, R., Kountcheva, R.: Compression of CT images with modified inverse pyramidal decomposition. In: 12th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision (ISCGAV'12), pp. 74–79, Istanbul, Turkey (2012)
16. Draganov, I., Kountchev, R., Georgieva, V.: Compression of CT images with branched inverse pyramidal decomposition. In: Iantovics, B., Kountchev, R. (eds.) Advanced Intelligent Computational Technologies and Decision Support Systems, vol. 486, VII, pp. 71–81. Springer (2014)

17. Kountchev, R., Todorov, V., Kountcheva, R.: New method for adaptive lossless compression of still images based on the histogram statistics. In: Tshirintzis, G., Virvou, M., Jain, L., Howlett, R. (eds.) *Intelligent Interactive Multimedia Systems and Services*, Proceedings of the 4th International Conference on Intelligent Interactive Multimedia Systems and Services (KES-IIMSS 2011), pp. 61–70. Springer (2011)
18. Deprettere, E.: *SVD and Signal Processing*. Elsevier, New York (1988)
19. Vaccaro, R.: *SVD and Signal Processing II*. Elsevier, New York (1991)
20. Moonen, M., Moor, B.: *SVD and Signal Processing III*. Elsevier, New York (1995)
21. Carlen, E.: *Calculus++*. In: *The Symmetric Eigenvalue Problem*. Georgia Tech (2003)
22. Andrews, H., Patterson, C.: Singular value decompositions and digital image processing. *IEEE Trans. Acoust. Speech Signal Process. ASSP-24*, 26–53 (1976)
23. Gerbrands, J.: On the relationships between SVD KLT, and PCA, *Pattern Recogn.* **14**(6), 375–381 (1981)
24. Kalman, D.: A singularly valuable decomposition: the SVD of a matrix. *Coll. Math. J* **27**(1), 2–23 (1996)
25. Rehna, V., Jeyakumar, M.: Singular value decomposition based image coding for achieving additional compression to JPEG images. *Int. J. Image Process. Vis. Sci.* **1**(2), 56–61 (2012)
26. Sadek, R.: SVD based image processing applications: state of the art, contributions and research challenges. *Int. J. Adv. Comput. Sci. Appl.* **3**(7), 26–34 (2012)
27. Householder, A.: *The Theory of Matrices in Numerical Analysis*. Dover, New York (1975)
28. Diamantaras, K., Kung, S.: *Principal Component Neural Networks*. Wiley, New York (1996)
29. Levy, A., Lindenbaum, M.: Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Trans. Image Process.* **9**(8), 1371–1374 (2000)
30. Drinea, E., Drineas, P., Huggins, P.: A randomized singular value decomposition algorithm for image processing applications. In: Manolopoulos, Y., Evripidou, S. (eds.) *Proceedings of the 8th Panhellenic Conference on Informatics*, pp. 278–288, Nicosia, Cyprus (2001)
31. Holmes, M., Gray, A., Isbell, C.: QUIC-SVD: fast SVD using cosine trees. In: *Proceedings of NIPS*, pp. 673–680 (2008)
32. Foster, B., Mahadevan, S., Wang, R.: A GPU-based approximate SVD algorithm. In: *9th International Conference on Parallel Processing and Applied Mathematics*, 2011, LNCS 7203, pp. 569–578 (2012)
33. Yoshikawa, M., Gong, Y., Ashino, R., Vaillancourt, R.: Case study on SVD multiresolution analysis, CRM-3179, pp. 1–18 (2005)
34. Waldemar, P., Ramstad, T.: Hybrid KLT-SVD image compression. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2713–2716. IEEE Computer Society Press, Los Alamitos (1997)
35. Aharon, M., Elad, M., Bruckstein, A.: The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process.* **54**, 4311–4322 (2006)
36. Singh, S., Kumar, S.: Singular value decomposition based sub-band decomposition and multi-resolution (SVD-SBD-MRR) representation of digital colour images. *Pertanika J Sci. Technol. (JST)* **19**(2), 229–235 (2011)
37. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.* **21**, 1253–1278 (2000)
38. Kountchev, R., Kountcheva, R.: Hierarchical SVD-based image decomposition with tree structure. *Int. J. Reasoning-Based Intell. Syst. (IJRIS)* **7**(1/2), 114–129 (2015)
39. Kountchev, R., Nakamatsu, K.: One approach for grayscale image decorrelation with adaptive multi-level 2D KLT. In: Graña, M., Toro, C., Posada, J., Howlett, R., Jain, L. (eds.) *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 1303–1312. IOS Press (2012)
40. Fieguth, P.: *Statistical image processing and multidimensional modeling*. Springer, Science +Business Media (2011)
41. Reed, T.: *Digital Image Sequence Processing, Compression, and Analysis*. CRC Press (2004)

42. Shi, Y., Sun, H.: *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, 2nd edn. CRC Press, Taylor & Francis Group, LLC (2008)
43. Mukhopadhyay, J.: *Image and Video Processing in the Compressed Domain*. CRC Press (2011)
44. Hyvarinen, A., Hurri, J., Hoyer, P.: *Natural Image Statistics, A Probabilistic Approach to Early Computational Vision*. Springer (2009)
45. Jolliffe, I.: *Principal Component Analysis*, 2nd edn. Springer-Verlag, NY (2002)
46. Dony, R.: Karhunen-Loeve Transform. In: Rao, K., Yip, P. (eds.) *The Transform and Data Compression Handbook*. CRC Press, Boca Raton (2001)
47. Fleury, M., Dowton, A., Clark, A.: *Karhunen-Loeve Transform—Image Processing*. University of Essex, Wivenhoe (1997)
48. Miranda, A., Borgne, Y., Bontempi, G.: New routes from minimal approximation error to principal components, pp. 1–14. Kluwer Academic Publishers (2007)
49. Landqvist, R., Mohammed, A.: Comparative performance analysis of three algorithms for principal component analysis. *Radioengineering* **15**(4), 84–90 (2006)
50. Tipping, M., Bishop, C.: Probabilistic principal component analysis. *J. Roy. Stat. Soc. B*, **61**(3), 611–622 (1999)
51. Liwicki, S., Tzimiropoulos, G., Zafeiriou, S., Pantic, M.: Euler principal component analysis. *Int. J. Comput. Vis.* **101**(3), 498–518 (2013)
52. Ujwala, P., Uma, M.: Image fusion using hierarchical PCA. In: *International Conference on Image Information Processing (ICIIP'11)*. pp. 1–6 (2011)
53. Abadpour, A., Kasaei, S.: Color PCA eigen images and their application to compression and watermarking. *Image Video Comput.* **26**, 878–890 (2008)
54. Chadha, A., Satam, N., Sood, R., Bade, D.: Image steganography using Karhunen-Loève transform and least bit substitution. *Int. J. Comput. Appl.* **79**(9), 31–37 (2013)
55. Naik, G., Kumar, D.: An overview of independent component analysis and its applications. *Informatica* **35**, 63–81 (2011)
56. Press, W., Teukolsky, S., Vetterling, W.: *Numerical Recipes in C, The Art of Scientific Computing*, 2nd edn. Cambridge University Press (2001)
57. Erdogmus, D., Rao, Y., Peddaneni, H., Hegde, A., Principe, J.: Recursive PCA using eigenvector matrix perturbation. *EURASIP J. Adv. Signal Process.* **13**, 2034–2041 (2004)
58. Hanafi, M., Kohler, A., Qannari, E.: Shedding new light on hierarchical principal component analysis. *J. Chemom.* **24**(11–12), 703–709 (2010)
59. Amar, A., Leshem, A., Gastpar, M.: Recursive implementation of the distributed Karhunen-Loève transform. *IEEE Trans. Signal Process.* **58**(10), 5320–5330 (2010)
60. Thirumalai, V.: *Distributed compressed representation of correlated image sets*. Thesis no 5264, Lausanne, EPFL (2012)
61. Saghi, J., Schroeder, S., Tescher, A.: An adaptive two-stage KLT scheme for spectral decorrelation in hyperspectral bandwidth compression. *SPIE Proc.* **7443**, 72–84 (2009)
62. Wongsawat, Y., Oraintara, S., Rao, K.: Integer sub-optimal Karhunen Loeve transform for multi-channel lossless EEG Compression. In: *14th European Signal Processing Conference (EUSIPCO'06)*, Florence, Italy (2006)
63. Blanes, I., Sagristà, J., Marcellin, M., Rapesta, J.: Divide-and-conquer strategies for hyperspectral image processing. *IEEE Signal Process. Mag.* **29**(3), 71–81 (2012)
64. Blanes, I., Sagristà, J.: Pairwise orthogonal transform for spectral image coding. *IEEE Trans. Geosci. Remote Sens.* **49**(3), 961–972 (2011)
65. Ma, Z.: Sparse principal component analysis and iterative thresholding. *Ann. Stat.* **41**(2), 772–801 (2013)
66. Jain, A.: A fast Karhunen-Loeve transform for a class of random processes. *IEEE Trans. Commun.* **COM-24**, 1023–1029 (1976)
67. Rokhlin, V., Szlam, A., Tygert, M.: A randomized algorithm for principal component analysis. *SIAM J. Matrix Anal. Appl.* **31**, 1100–1124 (2009)
68. Kambhatla, N., Haykin, S., Dony, R.: Image compression using KLT, wavelets and an adaptive mixture of principal components model. *J. VLSI Signal Process.* **18**, 287–296 (1998)

69. Bita, I., Barret, M., Pham, D.: On optimal transforms in lossy compression of multi-component images with JPEG2000. *Sig. Process.* **90**(3), 759–773 (2010)
70. Kountchev, R., Kountcheva, R.: PCA-based adaptive hierarchical transform for correlated image groups. In *Proceedings of the International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS'13)*, pp. 323–332. IEEE, Serbia (2013)
71. Kountchev, R.: Applications of the hierarchical adaptive PCA for processing of medical CT images. *Egypt. Comput. Sci. J.* **37**(3), 1–25 (2013)
72. Kountchev, R., Kountcheva, R.: Decorrelation of multispectral images, based on hierarchical adaptive PCA. *Int. J. WSEAS Trans. Signal Process.* **3**(9), 120–137 (2013)
73. Kountchev, R., Kountcheva, R.: Hierarchical adaptive KL-based transform: algorithms and applications. In: Favorskaya, M., Jain, L. (eds.) *Computer Vision in Advanced Control Systems: Mathematical Theory*, pp. 91–136. Springer, Heidelberg (2015)
74. Arora, S., Barak, B.: *Computational complexity: A modern approach*. Cambridge University Press (2009)
75. Jie, X., Fei, S.: Natural color image segmentation. In: *Proceedings of IEEE IC on Image Processing (ICIP'03)*, pp. 973–976, Barcelona, Spain (2003)
76. Navon, E., Miller, O., Averabuch, A.: Color image segmentation based on adaptive local thresholds. *Image Vis. Comput.* **23**, 69–85 (2005)
77. Deshmukh, K., Shinde, G.: An adaptive color image segmentation. *Electron. Lett. Comput. Vis. Image Anal.* **5**(4), 12–23 (2005)
78. Yu, Z., Wong, H., Wen, G.: A modified support vector machine and its application to image segmentation. *Image Vis. Comput.* **29**, 29–40 (2011)
79. Wang, X., Wang, T., Bu, J.: Color image segmentation using pixel wise support vector machine classification. *Pattern Recogn.* **44**, 777–787 (2011)
80. Bhoyar, K., Kakde, O.: Skin color detection model using neural networks and its performance evaluation. *J. Comput. Sci.* **6**(9), 963–968 (2010)
81. Stern, H., Efros, B.: Adaptive color space switching for face tracking in multi-colored lighting environments. In: *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 249–255 (2002)
82. Kakumanu, P., Makrogiannis, S., Bourbakis, N.: A survey of skin-color modeling and detection methods. *Pattern Recogn.* **40**, 1106–1122 (2007)
83. Ionita, M., Corcoran, P.: Benefits of using decorrelated color information for face segmentation/tracking. *Adv. Optical Technol.* **2008**(583687) (2008)
84. Hassanpour, R., Shahbahrami, A., Wong, S.: Adaptive Gaussian mixture model for skin color segmentation. *World Acad. Sci. Eng. Technol.* **41**, 1–6 (2008)
85. Hikal, N., Kountchev, R.: Skin color segmentation using adaptive PCA and modified elliptical boundary model. In: *International Proceedings of the IEEE International Conference on Advanced Computer Science and Information Systems (IEEE ICACISIS'11)*, pp. 407–412, Jakarta, Universitas Indonesia (2011)
86. Abadpour, A., Kasaei, S.: Principal color and its application to color image segmentation. *Scientia Iranica* **15**(2), 238–245 (2008)
87. Kountchev, R., Kountcheva, R.: Image color space transform with enhanced KLT. In: Nakamatsu, K., Wren, G., Jain, L., Howlett, R. (eds.) *New Advances in Intelligent Decision Technologies*, pp. 171–182. Springer-Verlag (2009)
88. Gunter, S., Schraudolph, N., Vishwanathan, S.: Fast iterative kernel principal component analysis. *J. Mach. Learn. Res.* **8**, 1893–1918 (2007)
89. Scholkopf, B., Smola, A., Muller, K.: Kernel principal component analysis. In: Scholkopf, B., Burges, C., Smola, A. (eds.) *Advances in Kernel Methods—Support Vector Learning*, pp. 327–352. MIT Press, Cambridge, MA (1999)
90. Lee, J., Verleysen, M.: *Nonlinear Dimensionality Reduction*. Springer (2007)
91. Kountchev, R., Hikal, N., Kountcheva, R.: A new approach for color image segmentation with hierarchical adaptive kernel PCA. In: *18th International Conference on Circuits, Systems,*

- Communications and Computers (CSCC'14), Proceedings of Advances in Information Science and Applications, pp. 38–43, Santorini Island, I (2014)
92. Kountchev, R., Milanova, M., Todorov, V., Kountcheva, R.: Image watermarking based on pyramid decomposition with CH transform, In: Chan, Y., Talburt, J., Talley, T. (eds.) *Data Engineering: Mining, Information, and Intelligence*, pp. 353–387. Springer (2010)
 93. Draganov, I.: Inverse pyramid decomposition of wavelet spectrum for image compression. *Int. J. Reasoning-based Intell. Syst. (IJRIS)* **6**(1/2), 19–23 (2014)
 94. Martino, F., Loia, V., Sessa, S.: A fuzzy hybrid method for image decomposition. *Int. J. Reasoning-based Intell. Syst. (IJRIS)* **1**(1/2), 77–84 (2009)