Egyptian Computer Science Journal ,ECS ,Vol. 37 No. 3, May 2013 ISSN-1110-2586

Applications of the Hierarchical Adaptive PCA for Processing of Medical CT Images

Roumen Kountchev

Radio communications and Video Technologies Dept., Technical University of Sofia, Sofia, Bulgaria

Abstract

In this work is proposed one new algorithm for processing of sequences (groups) of computer tomography (CT) images, based on the Hierarchical Adaptive PCA (HAPCA). It is an approximation of the well-known Principle Component Analysis (PCA), which ensures maximum power concentration in the firsh images of the processed group (called "eigen") and their full decorrelation. The basic advantages of the HAPCA method as compared to that of the PCA are the high decorrelation achieved for the transformed groups of CT images, very close to that of the "classic" PCA, but with much lower computational complexity, together with the ability for parallel processing for each group. In the paper are also given some results obtained after modeling of the proposed algorithm, applied on sequences of real CT images, which confirm the method ability to achieve high decorrelation. There are also discussed some of the basic applications of the HAPCA algorithm for compression of groups of CT images, object segmentation, and reduction of the features' space, when objects recognition is performed.

Keywords: *Principal Component Analysis, Hierarchical Adaptive PCA, sequences computer tomography images, "eigen" images, decorrelation of groups of images, features space reduction.*

1. Introduction

The studies, based on the analysis of groups of medical images, which ensure the ability for early detection of various diseases, are of significant importance for the contemporary medical diagnostics. This scientific area, known as "diagnostic imaging", utilizes various technical tools to get medical images [1] of the kind: X-Ray (XR), Computer Tomography (CT), Single-Photon Emission Computed Tomography (SPECT), Positron Emission Tomography (PET), Nuclear Magnetic Resonance (NMR), Magnetic Resonance (MR), Magnetic Resonance Tomography (MRT), Ultrasound (US), etc. These images could be single photos of the examined human body parts, or sequences, which visualize the changes of the body in time, or in various spatial positions. Tipical examples are the sequences of CT images, obtained in result of scanning the selected part human body through small spatial steps. For one patient the number of these images is usually significant (could be over 200), each of high resolution. Depending on the sequence visualization on the screen, could be seen moving CT images, or 3D virtual reconstruction of the examined parts of the human body, which permits observation from different view points.

The methods for medical visual information processing depend on the application intended [1,2,3], such as for example:

- Image transfer or archiving, for which are used various compression methods;
- Enhancement of the visual quality (pre-processing);
- Segmentation of the examined objects;
- Extraction of features for objects recognition - methods for features minimization, which to ensure this reduced number of features to be sufficient for the achievement of the needed classification accuracy;
- Objects tracking – metods for image sequences fusion, etc.

The storage of huge volumes of visual medical information needs efficient algorithms for image compression. For still medical images are usually used software solutions, based on the 2D-DCT, 2D-Wavelet Decomposition for prediction or Embedded Zero-Tree/Block Coding of Wavelet Coeffients (EZW) [4,5,6], and also the file format DICOM (Digital Imaging and COmmunications in Medicine) [7]. For the compression of CT image sequences are used: the interframe decorrelation based on Hierarchical INTerpolation (HINT) [8,9], the Spatial Active Appearance Model (SAAM) for echo-cardiographic image sequence [10], the standards JPEG-LS and JPEG2000 with interframe motion compensated prediction [11,12] and the distributed representation of image sets based on Slepian-Wolf coding [13].

One of the most efficient methods for decorrelation and compression of groups of images is based on the Principal Component Analysis (PCA), also known as of Karhunen-Loeve (KL) or Hotelling Transform, [14-23]. For its implementation the pixels with the same spatial position from each in a group of N images compose the corresponding 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. For large values of N, the computational complexity of the algorithm is significantly increased. One of the possible approaches for reduction of the computational complexity of PCA for N-dimensional group of medical images could be based on the "Hierarchical Adaptive PCA" (HAPCA), offered in [24]. Unlike the famous Hierarchical PCA (HPCA) [17-21], HAPCA is not iterative: it is implemented through dividing the image sequence into groups of length, defined by their correlation interval. Each group is divided into sub-groups of images, on which is applied the Adaptive PCA (APCA), of size 3×3 or 2×2. This transform is performed using equations, which are not based on iterative calculations, and as a result, they have lower computational complexity. To obtain high decorrelation for the whole group of medical images is necessary to use APCA of size 3×3 or 2×2, which to be applied in several consecutive stages (hierarchical levels), with rearrangement of the obtained intermediate eigen images after each stage. In result is obtained a decorrelated group of eigen images.

In this paper are presented some results of the HAPCA algorithm application for processing of groups of CT images, aimed at obtaining their high decorrelation and compression. The paper is arranged as follows: detailed description of the HAPCA algorithm for processing of groups of CT images, transformation of CT images sub-groups through

APCA with a matrix of 3×3 or 2×2 elements, evaluation of the computational complexity of the offered transform, results of the HAPCA application for compression of groups of CT images, other possible applications for video coding and object segmentation, and conclusions.

2. Algorithm HAPCA for CT image sequences transform

The algorithm HAPCA for CT image sequences transform comprises the following basic steps: 1) setting the length of each group of CT images through correlation analysis of the whole sequence; 2) dividing the sequence into groups, which comprise sub-groups of 3 or 2 images each; 3) adding new interpolated images, in case that the last sub-group should be filled out; 4) defining the number of hierarchical transform levels depending on the decorrelation achieved; 5) execution of the HAPCA algorithm for each group, in result of which is achieved significant decorrelation of the transformed images.

The next steps of the processing of the so obtained CT images depend on the HAPCA application needed. In case that it will be used for image group compression, follow the next operations: 1) cutting off the low-informative transformed images in each group; 2) block coding of each retained image by applying the selected 2D orthogonal transform, quantization of the obtained coefficients and entropy coding. For the decompression, same operations are executed in reverse order: entropy decoding, dequantization of the transform coefficients, block decoding of the retained images, adding the cut-off images in each group and applying the inverse HAPCA transform. As a result, the sequence of CT images is restored unchanged, or with minimum distortions, unnoticeable for the user. In case that lossless compression is needed, the steps, which execute cutting off for the low-informative transformed images and the coefficients quantization, are omitted. The choise of the compression metod (lossless or visually lossless) depends on the necessity to achieve: 1) full restoration of the sequence of CT images and relatively low compression, or 2) much higher compression at the expense of information loss, which is over the abilities of the human visual perception.

2.1. Setting the number of CT images in each group and sub-group

The basic quality of the HAPCA algorithm is that in result of its execution is achieved significant decorrelation for the processed group of initially highly correlated images. The main criterion used to set the number of images (N) in one group (i.e. - the length of the group), is the correlation interval in the processed CT sequence. In case, that the number N is set to be equal to the length of the correlation interval, the HAPCA efficiency is enhanced.

For a sequence of P images $[C_u]$ ($u=1,2,..,P$) the correlation interval could be defined on the basis of the normalized cross-correlation coefficient $\rho_{1,u}$ for the couple $[C_1]$ and $[C_u]$ [25]:

$$\rho_{1.u} = \frac{\sum_{s=1}^{S}(c_s^1 - \bar{c}_1)\times(c_s^u - \bar{c}_u)}{\sqrt{\sum_{s=1}^{S}(c_s^1 - \bar{c}_1)^2} \times \sqrt{\sum_{s=1}^{S}(c_s^u - \bar{c}_u)^2}} \text{ for } u=1,2,..,P, \quad (1)$$

where S is the number of pixels in each image $[C_u]$; $\bar{c}_1 = E\{c_s^1\}$, $\bar{c}_u = E\{c_s^u\}$ $[\bar{x} = E\{x_s\} = (1/S)\sum_{s=1}^{S} x_s]$; c_s^1, c_s^u - the values of the pixels with same spatial position in the couple of images.

In case that the function of the normalized cross-correlation $\rho(u)$, defined by Eq. (1), is approximated with the exponential model $\rho(u)=\rho(1)exp(-u/N)$, the value of the correlation interval N (called also "interval of statistical influence") is defined by the condition $\rho(u)=\rho(1)exp(-1)=0.36$ for $u=N$. Hence, when the $\rho(u)$ from Eq. (1) is calculated, N is defined by the value of u, for which:

$$\rho(u = N) \le \delta = 0.36 \quad \text{for} \quad u = 1,2,..,P. \tag{2}$$

After the value of N is defined, the sequence of P images is divided into groups of N images and on each is executed the HAPCA algorithm. For this is necessary each Group Of Pictures (GOP) to be divided into sub-groups of N_s=3 or 2 images. However, N is not always divisible to 3 or 2 (i.e. the relations $N = p_1 3^k$ or $p_2 2^n$ are not satisfied). In the general case $N = (p_1 3^k + m_1)$ or $(p_2 2^n + m_2)$ for p_1, k, m_1, p_2, n, m_2 – positive integer numbers. Then the number of images in the sub-group N_s and the corresponding number of sub-groups (p_1 or p_2) are defined by the rules:

$$N_s = \begin{cases} 3, & \text{if } m_1 = 0 \wedge m_2 > 0 \vee m_1 = m_2 = 0; \\ 2, & \text{- in other cases;} \end{cases} \tag{3}$$

In case that $m_1 \neq 0$ and $m_2 \neq 0$ are simultaneously satisfied, the GOP should be expanded by adding new images. In this case the number of images N_e in the expanded GOP_e is defined as follows:

$$N_e = \begin{cases} (p_1+1)3^k \vee (p_1+3)3^{k-1}, & \text{if } N = p_1 3^k + m_1; \\ (p_2+1)2^n, & \text{if } N = p_2 2^n + m_2. \end{cases} \tag{4}$$

The number m_{int} of the images added to the GOP is defined by the relation:

$$m_{int} = N_e - N = min\{(3^k - m_1), [3^{k-1}(3 - 2p_1) - m_1], (2^n - m_2)\} > 0. \tag{5}$$

On Figure (1) is given an example for the GOP expansion. The added new images m_{int}=2 are colored in green. They are defined through zero-order interpolation, by using the last (N^{th}) image in the group of length $(p_1 3^k + m_1)$.

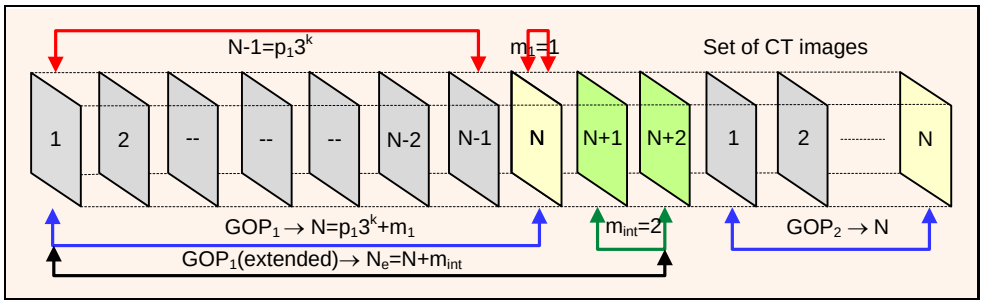


Figure (1): Determination of the GOP parameters for the algorithm HAPCA

In Table 1 are given the values of N_e, p_1, k, m_1, p_2, n, m_2, m_{int}, N_s and L, which define the structure of the HAPCA algorithm for group lengths N=4,5,...,16. For each value of N the number of hierarchical levels L of HAPCA is defined by evaluation of the acceptable decorrelation value, which should be achieved for the transformed GOP. The decorrelation could be evaluated through analysis of the elements $k_{i,j}$ of the covariance matrix $[K_C]$ of the vectors $\vec{C}_s=\left[C_{1,s},C_{2,s},..,C_{N_e,s}\right]^t$ for $s=1,2,..,S$. The components $C_{r,s}$ of these vectors for $r=1,2,..,N_e$ are defined by the pixels s with same spatial position in each image $[C_r]$, which contains S pixels.

Table 1. Parameners, which define the structure of the algorithm HAPCA

N	N_e	$p_1 3^k+m_1$	$p_2 2^n+m_2$	p_1	k	m_1	p_2	n	m_2	m_{int}	N_s	L
4	4	4	4	1	1	1	1	2	0	0	2	2
5	6	5	5	1	1	2	1	2	1	1	3	2
6	6	6	6	2	1	0	3	1	0	0	3	2
7	8	7	7	2	1	1	3	1	1	1	2	2
8	8	8	8	2	1	2	1	3	0	0	2	3
9	9	9	9	1	2	0	1	3	1	0	3	2
10	10	10	10	1	2	1	5	1	0	0	2	3
11	12	11	11	1	2	2	1	3	3	1	3	3
12	12	12	12	4	1	0	3	2	0	0	3	3
13	15	13	13	4	1	1	3	2	1	2	3	3
14	15	14	14	4	1	2	3	2	2	1	3	3
15	15	15	15	5	1	0	3	2	3	0	3	3
16	16	16	16	4	1	4	1	4	0	0	2	4

The decorrelation of the transformed vectors $\vec{L}_s(l)$ in the level l of HAPCA could be evaluated by using the relation $CovR(l)$, calculated on the basis of their covariance matrix $[K_L(l)]$:

$$CovR(l) = \frac{\sum_{i=1}^{N_e} k_{i,i}^2(l)}{\sum_{i=1}^{N_e}\sum_{j=1}^{N_e} k_{i.j}^2(l)_{i\neq j}} \quad \text{for } l=1,2,... \tag{6}$$

Here $k_{i,i}(l)$ are the elements in the main diagonal of the matrix $[K_L]$, and $k_{i,j}(l)$ for $i \neq j$ - its non-diagonal elements. In case of full correlation we obtain $CovR \rightarrow \infty$, but to define the needed number of hierarchical levels L for HAPCA is enough to satisfy the relation:

$$[CovR(l = L)]^{-1} \leq \Delta, \tag{7}$$

where Δ is a pre-defined threshold with a small value, defined experimentally. In Table 1 are given the values of the levels L for each N, defined by Eq. (6) by using the mean covariance matrix $[\overline{K}_L]$ for large number of examined CT images.

2.2. HAPCA Transform Algorithms for a Group of CT images

On Figure (2) is given an example for the structure of the 2-level HAPCA algorithm, applied on the GOP, for $N = 9$.

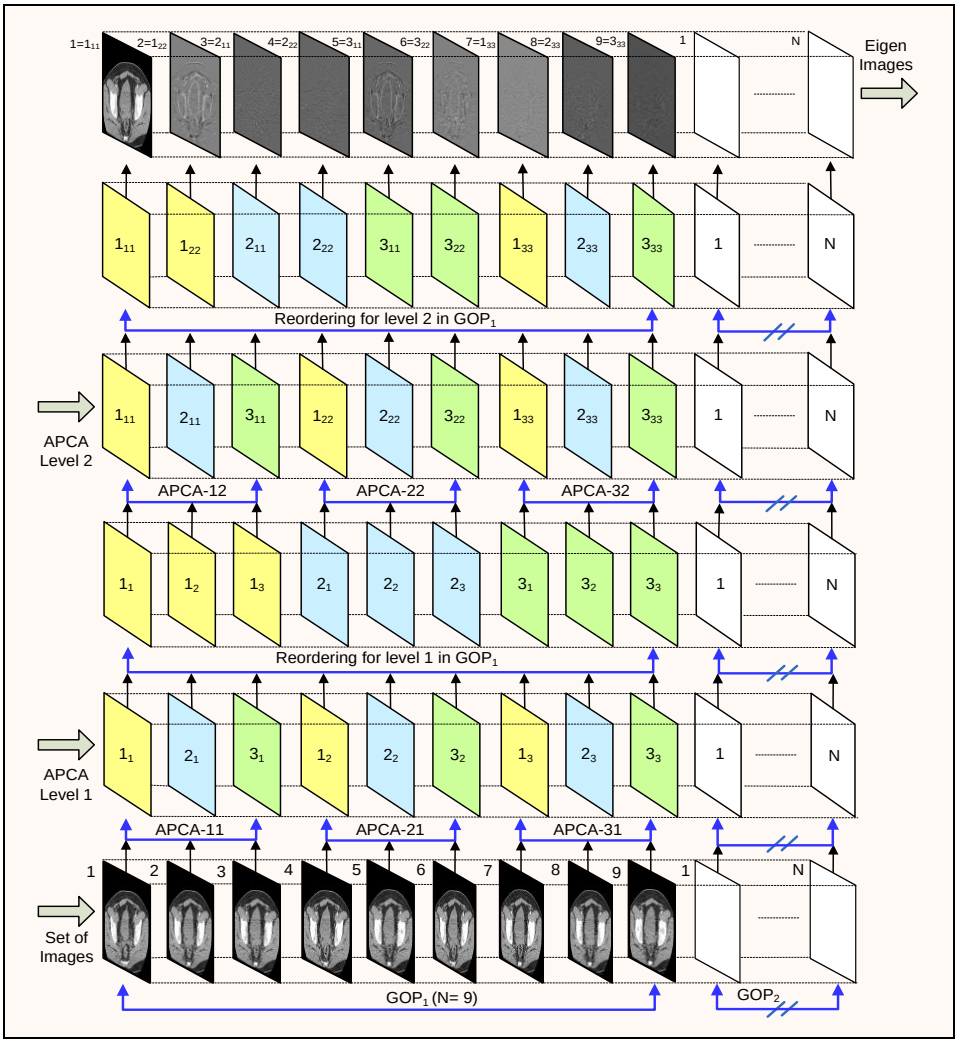


Figure (2): Algorithm of the 2-level HAPCA for a Group of 9 CT images (*N*=9) (each first APCA component is colored in yellow, the second in blue, and the third in green)

As it is seen from the fugure, on each sub-group of 3 CT images from the first hierarchical level of HAPCA is applied APCA with a matrix of size 3×3. In result are obtained 3 eigen images, colored in yellow, blue and green correspondingly. After that, the eigen images are rearranged so that the first sub-group of 3 eigen images to comprise the first images from each group, the second group of 3 eigen images - the second images from each group, etc. For each GOP of 9 intermediate eigen images in the first hierarchical level is executed in similar way the next APCA, (with a 3×3 matrix on each sub-group of 3 eigen values). In result, in the second hierarchical level are obtained 3 new eigen images (i.e. the eigen images of the group of 3 intermediate eigen images), colored in yellow, blue, and green correspondingly. Then the eigen images are rearranged again so, that the first group of 3 eigen images to contain the first images from each group before the rearrangement; the second group of 3 eigen images - the second image before the rearrangement, etc. At the end of the processing is obtained a decorrelated group of eigen images, from which through inverse HAPCA could be restored the original group.

On Figure (3) is shown an example for the structure of a 3-level HAPCA algorithm, applied on a GOP of 8 images (N=8). On each sub-group of 2 CT images from the first hierarchical level of HAPCA is executed APCA with a matrix of size 2×2. In result are obtained 2 eigen images, colored in yellow and blue correspondingly.

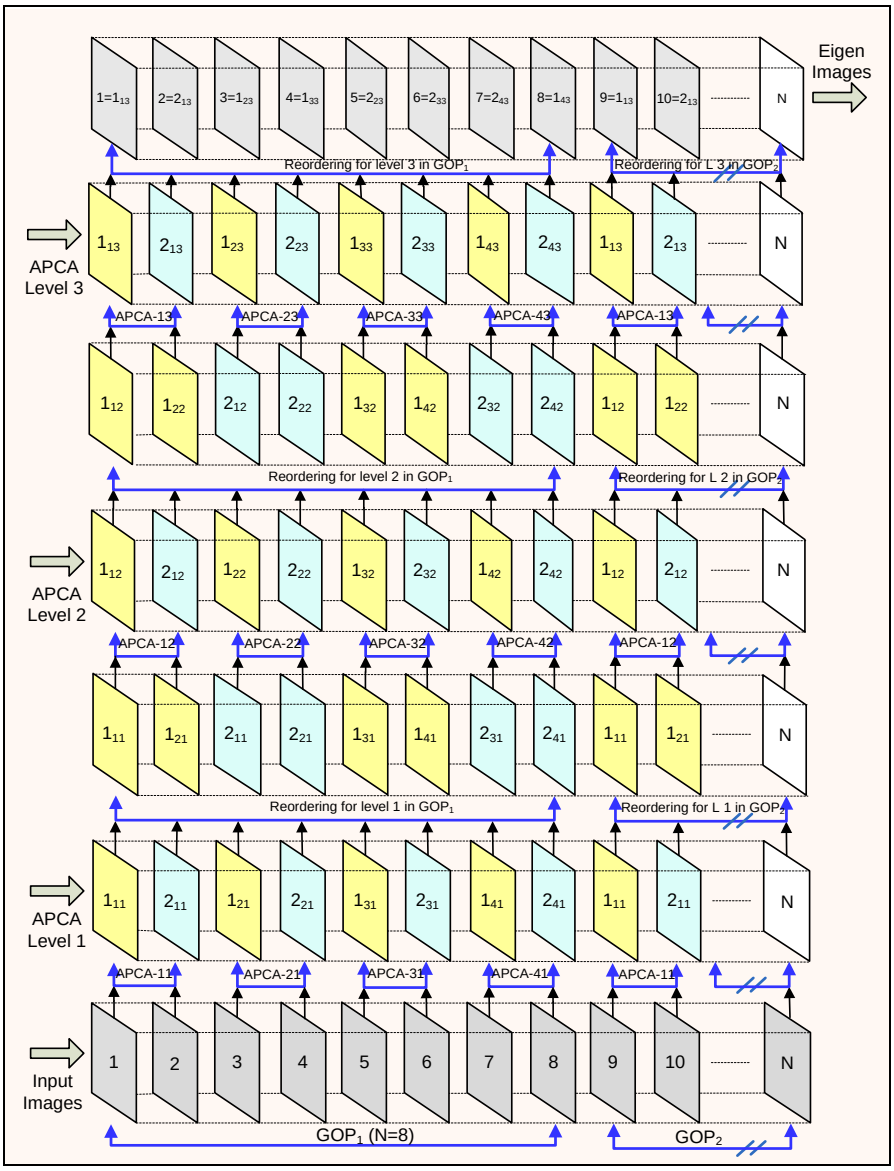


Figure (3): Algorithm for 3-level HAPCA applied on a Group of 8 CT Images (N=8) (each first APCA component is colored in yellow, and each second – in blue)

After execution of all hierarchical HAPCA levels, the obtained intermediate images are rearranged so that these with similar correlation to be in same sub-group. The 3-level HAPCA transform is also reversible. Due to the smaller size of the transform matrix, to achieve the needed decorrelation should be used 3 hierarchical levels (instead of 2 for the previous case).

2.3. Sub-group Transform through APCA with a 3×3 Matrix

For the calculation of eigen images through Adaptive PCA with a 3×3 matrix for one GOP sub-group, is applied the approach given in [26] for 3-component vectors. From each sub-group of 3 CT images of S pixels each, shown on Figure (4), are calculated the vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$ for $s=1,2,..,S$ (on the figure are shown the vectors for the first 4 pixels only: $\vec{C}_1=[C_{11}, C_{21}, C_{31}]^t$, $\vec{C}_2=[C_{12}, C_{22}, C_{32}]^t$, $\vec{C}_3=[C_{13}, C_{23}, C_{33}]^t$, $\vec{C}_4=[C_{14}, C_{24}, C_{34}]^t$). Each vector is transformed into vectors $\vec{L}_s = [L_{1s}, L_{2s}, L_{3s}]^t$ for $s=1,2,..,S$ through APCA with a matrix $[\Phi]$, of size 3×3.

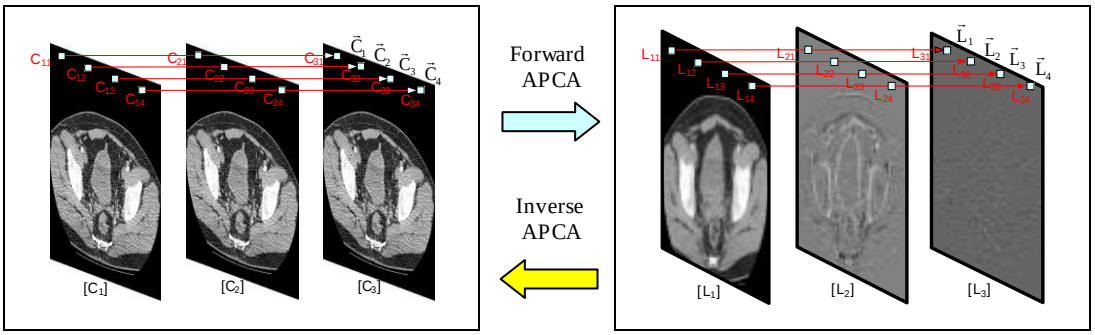


Figure (4): APCA transform for a sub-group of 3 images

The Forward Adaptive PCA for the vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$, from which are obtained vectors the $\vec{L}_s = [L_{1s}, L_{2s}, L_{3s}]^t$, is:

$$\begin{bmatrix} L_{1s} \\ L_{2s} \\ L_{3s} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{21} & \Phi_{31} \\ \Phi_{12} & \Phi_{22} & \Phi_{32} \\ \Phi_{13} & \Phi_{23} & \Phi_{33} \end{bmatrix} \begin{bmatrix} (C_{1s}-\overline{C}_1) \\ (C_{2s}-\overline{C}_2) \\ (C_{3s}-\overline{C}_3) \end{bmatrix} \quad \text{for } s = 1, 2,..., S. \tag{8}$$

Here $\overline{C}_1 = E(C_{1s})$, $\overline{C}_2 = E(C_{2s})$, $\overline{C}_3 = E(C_{3s})$. The elements Φ_{im} of the matrix $[\Phi]$ are:

$$\Phi_{1m} = A_m / P_m;\ \Phi_{2m} = B_m / P_m;\ \Phi_{3m} = D_m / P_m \quad \text{for } m = 1,2,3, \text{ where:} \tag{9}$$

$$A_m = (k_3 - \lambda_m)[k_5(k_2 - \lambda_m) - k_4 k_6],\quad B_m = (k_3 - \lambda_m)[k_6(k_1 - \lambda_m) - k_4 k_5], \tag{10}$$

$$D_m = k_6[2k_4 k_5 - k_6(k_1 - \lambda_m)] - k_5^2(k_2 - \lambda_m),\quad P_m = \sqrt{A_m^2 + B_m^2 + D_m^2}, \tag{11}$$

$$k_1 = E(C_{1s}^2) - (\overline{C}_1)^2, k_2 = E(C_{2s}^2) - (\overline{C}_2)^2, k_3 = E(C_{3s}^2) - (\overline{C}_3)^2, \tag{12}$$

$$k_4 = E(C_{1s}C_{2s}) - (\overline{C}_1)(\overline{C}_2),\ k_5 = E(C_{1s}C_{3s}) - (\overline{C}_1)(\overline{C}_3),\ k_6 = E(C_{2s}C_{3s}) - (\overline{C}_2)(\overline{C}_3), \tag{13}$$

$$\lambda_1 = 2\sqrt{\frac{|p|}{3}}\cos\left(\frac{\varphi}{3}\right) - \frac{a}{3};\ \lambda_2 = -2\sqrt{\frac{|p|}{3}}\cos\left(\frac{\varphi+\pi}{3}\right) - \frac{a}{3};\ \lambda_3 = -2\sqrt{\frac{|p|}{3}}\cos\left(\frac{\varphi-\pi}{3}\right) - \frac{a}{3}, \tag{14}$$

$$\varphi = \arccos\left[-q/2\Big/\sqrt{(|p|/3)^3}\right],\ p = -(a^2/3) + b < 0,\ q = 2(a/3)^3 - (ab)/3 + c, \tag{15}$$

$$a = -(k_1 + k_2 + k_3),\ b = k_1k_2 + k_1k_3 + k_2k_3 - (k_4^2 + k_5^2 + k_6^2), \tag{16}$$

$$c = k_1k_6^2 + k_2k_5^2 + k_3k_4^2 - (k_1k_2k_3 + 2k_4k_5k_6). \tag{17}$$

The equations (9)-(17), which represent the calculation of the matrix [Φ], are given in detail in [26]. The Inverse Adaptive PCA for the vectors $\vec{L}_s$, from which are obtained vectors $\vec{C}_s$, is:

$$\begin{bmatrix} C_{1s} \\ C_{2s} \\ C_{3s} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} \\ \Phi_{21} & \Phi_{22} & \Phi_{23} \\ \Phi_{31} & \Phi_{32} & \Phi_{33} \end{bmatrix} \begin{bmatrix} L_{1s} \\ L_{2s} \\ L_{3s} \end{bmatrix} + \begin{bmatrix} \overline{C}_1 \\ \overline{C}_2 \\ \overline{C}_3 \end{bmatrix} \text{ for } s = 1, 2, \ldots, S. \tag{18}$$

For the restoration of vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$ through inverse APCA are needed not only the vectors $\vec{L}_s = [L_{1s}, L_{2s}, L_{3s}]^t$, but also the elements Φ_{ij} of the matrix $[\Phi]$, and the values of $\overline{C}_1, \overline{C}_2, \overline{C}_3$ as well. The total number of needed elements could be reduced representing the matrix $[\Phi]$ as the product of matrices $[\Phi_1(\alpha)]$, $[\Phi_2(\beta)]$, $[\Phi_3(\gamma)]$, and the rotation around the coordinate axes for each transformed vector in Euler angles α, β and γ correspondingly:

$$[\Phi] = \begin{bmatrix} \Phi_{11} & \Phi_{21} & \Phi_{31} \\ \Phi_{12} & \Phi_{22} & \Phi_{32} \\ \Phi_{13} & \Phi_{23} & \Phi_{33} \end{bmatrix} = [\Phi_1(\alpha)][\Phi_2(\beta)][\Phi_3(\gamma)] = [\Phi(\alpha, \beta, \gamma)], \tag{19}$$

where

$$[\Phi_1(\alpha)] = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}; [\Phi_2(\beta)] = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}; [\Phi_3(\gamma)] = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{20}$$

In this case the elements of the matrix [Φ] are represented by the relations:

$$\Phi_{11} = \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma;\ \Phi_{21} = -(\cos\alpha\cos\beta\sin\gamma + \sin\alpha\cos\gamma);\ \Phi_{31} = -\cos\alpha\sin\beta;$$

$$\Phi_{12} = \sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma;\ \Phi_{22} = -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma;\ \Phi_{32} = -\sin\alpha\sin\beta;$$

$$\Phi_{13} = \sin\beta\cos\gamma; \qquad \Phi_{23} = -\sin\beta\sin\gamma; \qquad \Phi_{33} = \cos\beta. \tag{21}$$

The matrix of the inverse AKLT is defined by the relation:

$$[\Phi]^{-1} = [\Phi_3(-\gamma)][\Phi_2(-\beta)][\Phi_1(-\alpha)]. \tag{22}$$

Then, for the calculation of the elements of the inverse matrix is enough to know the values of the rotation angles α, β and γ, defined by the relations:

$$\alpha = -\arcsin\left(\Phi_{32} / \sqrt{1 - \Phi_{33}^2}\right);\ \beta = \arccos(\Phi_{33});\ \gamma = \arccos\left(\Phi_{13} / \sqrt{1 - \Phi_{33}^2}\right). \tag{23}$$

In result, the number of needed values for the calculation of the matrix $[\Phi]^{-1}$ is reduced from 9 down to 3, i.e. 3 times reduction. The elements L_{1s}, L_{2s}, L_{3s} for s=1, 2,...,S comprise the pixels of the first, second and third eigen image in the sub-group of CT images.

2.4. Sub-group Transform through APCA with a 2×2 Matrix

For the calculation of eigen images through APCA with a 2×2 matrix for one GOP sub-group is used the approach for the transformation of 2-component vectors through PCA, given in detail in [27].

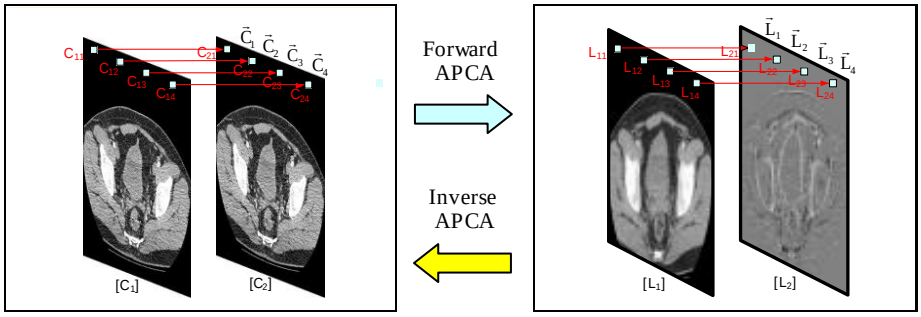


Figure (5): APCA transform for sub-group of 2 images from the GOP

From each sub-group of 2 CT images of S pixels each, shown on Figure (5), are calculated the vectors $\vec{C}_s = [C_{1s}, C_{2s}]^t$ for s=1,2,..,S (on the figure are shown the vectors for the first 4 pixels only: $\vec{C}_1 = [C_{11}, C_{21}]^t$, $\vec{C}_2 = [C_{12}, C_{22}]^t$, $\vec{C}_3 = [C_{13}, C_{23}]^t$, $\vec{C}_4 = [C_{14}, C_{24}]^t$). Each vector $\vec{C}_s = [C_{1s}, C_{2s}]^t$ is then transformed into vectors $\vec{L}_s = [L_{1s}, L_{2s}]^t$ through forward APCA, using the matrix $[\Phi]$ of size 2×2:

$$\begin{bmatrix} L_{1s} \\ L_{2s} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{21} \\ \Phi_{12} & \Phi_{22} \end{bmatrix} \begin{bmatrix} (C_{1s} - \overline{C}_1) \\ (C_{2s} - \overline{C}_2) \end{bmatrix} \text{ for s = 1, 2,..., S.} \tag{24}$$

Here $\overline{C}_1 = E(C_{1s})$, $\overline{C}_2 = E(C_{2s})$. The elements Φ_{ij} of the matrix $[\Phi]$ are [27]:

$$\Phi_{11} = \frac{\alpha + \gamma}{\sqrt{2(\gamma^2 + \alpha\gamma)}}, \quad \Phi_{21} = \frac{\beta}{\sqrt{2(\gamma^2 + \alpha\gamma)}}, \tag{25}$$

$$\Phi_{12} = \frac{\alpha - \gamma}{\sqrt{2(\gamma^2 - \alpha\gamma)}}, \quad \Phi_{22} = \frac{\beta}{\sqrt{2(\gamma^2 - \alpha\gamma)}}, \tag{26}$$

where $\alpha = k_1 - k_2$, $\beta = 2k_3$ and $\gamma^2 = \alpha^2 + \beta^2$.

$$k_1 = E(C_{1s}^2) - (\overline{C}_1)^2, \; k_2 = E(C_{2s}^2) - (\overline{C}_2)^2, \; k_3 = E(C_{1s}.C_{2s}) - (\overline{C}_1)(\overline{C}_2). \tag{27}$$

To satisfy the requirement $\sqrt{2(\gamma^2 - \alpha\gamma)} \neq 0$ in Eqs. (26), it's necessary to have $k_3 \neq 0$ (i.e. the mutual covariation between the vectors $\vec{C}_s$ has to be positive or negative). If the

opposite is true (for $k_3 = 0$) the PCA 2×2 is not applied because the vectors $\vec{C}_s$ are decorrelated.

The inverse APCA of the vectors $\vec{L}_s$ transforms them into vectors $\vec{C}_s = [C_{1s}, C_{2s}]^t$:

$$\begin{bmatrix} C_{1s} \\ C_{2s} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} L_{1s} \\ L_{2s} \end{bmatrix} + \begin{bmatrix} \overline{C}_1 \\ \overline{C}_2 \end{bmatrix} \text{ for } s = 1, 2, \ldots, S. \tag{28}$$

The elements Φ_{ij} of $[\Phi]$ are functions of the angle θ to which the rotation of the coordinate system (L_1, L_2) is done in relation to the initial system (C_1, C_2) as a result of the APCA. If both coordinate systems (C_1, C_2) and (L_1, L_2) are selected to be right-oriented, the matrix $[\Phi(\theta)]$ used for rotation to angle θ in the counter-clockwise direction, is given by the expression:

$$[\Phi(\theta)] = \begin{bmatrix} \Phi_{11}(\theta) & \Phi_{21}(\theta) \\ \Phi_{12}(\theta) & \Phi_{22}(\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{29}$$

where $\theta = arctg\left(\dfrac{\Phi_{21}(\theta)}{\Phi_{11}(\theta)}\right) = arctg\left(\dfrac{\beta}{\alpha+\gamma}\right)$.

The elements of the rotation matrix $[\Phi(\theta)]$ are:

$$\cos\theta = \cos[arctg\left(\frac{\beta}{\alpha+\gamma}\right)] = \frac{\alpha+\gamma}{\sqrt{2\gamma(\alpha+\gamma)}},\ \sin\theta = \sin[arctg\left(\frac{\beta}{\alpha+\gamma}\right)] = \frac{\beta}{\sqrt{2\gamma(\alpha+\gamma)}}. \tag{30}$$

Then from Eqs. (29) - (30) follows that the rotation matrix for 2×2 is:

$$[\Phi(\theta)] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \frac{\alpha+\gamma}{\sqrt{2\gamma(\alpha+\gamma)}} \begin{bmatrix} 1 & \dfrac{\beta}{\alpha+\gamma} \\ -\dfrac{\beta}{\alpha+\gamma} & 1 \end{bmatrix}. \tag{31}$$

From Eqs. (24)-(27) and (28)-(31) follows that for the executation of one transform of the forward/inverse APCA, 3 parameters only should be known: θ, $\overline{C}_1$, and $\overline{C}_2$.

3. Evaluaton of the Computational Complexity of HAPCA

The computational complesity (CC) of the HAPCA algorithm is evaluated for N=9 and L=2 (in this case the transform matrix $[\Phi]$ used for the processing of each sub-group, is of size 3×3). In this work it is compared with the CC for the PCA algorithm when N=9 and the transform matrix $[\Phi]$ is of size 9×9. As a basis for the evaluation is used the total number of operations O (additions and multiplications), needed for the calculation of the following elements of both algorithms:

- The covariance matrices: in total, 6 matrices for the first algorithm, each of size 3×3, and one matrix of size 9×9 – for the second algorithm;
- The eigen values and the eigen vectors of these matrices;
- The eigen images in each GOP, obtained as a result of the algorithm execution.

From the analysis, given in [26] for the CC of APCA with a 3×3 matrix and of PCA with a $N \times N$ matrix it follows that for HAPCA with L=2 and 3×3 matrices and PCA with a 9×9 matrix, we have:

- The total number of operations needed to calculate all 6 covariance matrices of size 3×3 - for HAPCA, and one matrix of size 9×9 – for PCA, are corespondingly:

$$O_{kov}(N)\big|_{N=3} = 3N(N+1)[N(N-1)+2(N+2)] = 576. \quad (32)$$

$$O_{kov}(N)\big|_{N=9} = (1/2)N(N+1)[N(N-1)+2(N+2)] = 4230. \quad (33)$$

- The total number of operations needed to calculate the eigen values of same matrices for HAPCA in correspondence with Eqs. (14)-(17), and for PCA - when the QR decomposition and the Householder transfomation of (*N-1*) steps are used [15], we have:

$$O_{val}(N)\big|_{N=3} = 282. \quad (34)$$

$$O_{val}(N)\big|_{N=9} = (N-1)(\frac{4}{3}N^2 + \frac{17}{6}N + 7) = 1124. \quad (35)$$

- The total number of operations needed to calculate the eigen vectors of same matrices for 2-level HAPCA and for PCA - by using an iterative Gauss-Seidel method [25] with 4 iterations, we have:

$$O_{vec}(N)\big|_{N=3} = 275. \quad (36)$$

$$O_{vec}(N)\big|_{N=9} = N[2N(4N+5)-1] = 6633. \quad (37)$$

- The total number of operations needed to calculate a group of 9 eigen images, comprising S pixels each, through the forward 2-level HAPCA and the PCA with zero mean vectors, is correspondingly:

$$O_{HAPCA}(N)\big|_{N=3} = 6SN(2N-1) = 90S. \quad (38)$$

$$O_{PCA}(N)\big|_{N=9} = SN(2N-1) = 153S. \quad (39)$$

The total number of operations O for 2-level HAPCA and PCA is:

$$O_1(3) = [O_{kov}(3) + O_{val}(3) + O_{vec}(3) + O_{HAPCA}(3)] = 576 + 282 + 275 + 90S = 1133 + 90S, \quad (40)$$

$$O_2(9) = [O_{kov}(9) + O_{val}(9) + O_{vec}(9) + O_{PCA}(9)] = 4230 + 1124 + 6633 + 153S = 11987 + 153S \quad (41)$$

The reduction of the total number of operations needed for the 2-levels HAPCA, compared to that of the PCA, could be evaluated by using the coefficient η:

$$\eta(S)=\frac{O_2(9)}{O_1(3)}=\frac{11987+153S}{1133+90S}. \quad (42)$$

From Eq. (42) follows that for S=100, $\eta(100)=2.69$; for S=1000, $\eta(1000)=1.81$; and for $S = \infty$, $\eta(\infty)\to 1.7$. Hence, for each value of S the total number of operations $O_1(S)$ is at least 1.7 times smaller than $O_2(S)$ (or in average, about 2 times). In respect to the value of η we get similar result in the case, when the HAPCA is for N=8 and L=3.

4. Experimental Results of the HAPCA Algorithm Application in Compression of CT Images

For the experiments were used image sets of the image database of the Technical Universuty of Sofia. The algorithm implementation was in Visual C. The time needed for the HAPCA execution is up to several milliseconds. A part of the experimental results is given below. The algorithm of the 2-level HAKLT, was applied on a sequence of P=64 CT images of size 512×512 pixels (S=2^{18}), 8 bpp. In result of the correlation analysis this sequence was divided into 7 groups (Set 1,..,Set 7), each comprising N=9 CT images. On Figure (6) is shown one of the groups (Set 3), which comprises images: Image 1,..,Image 9.

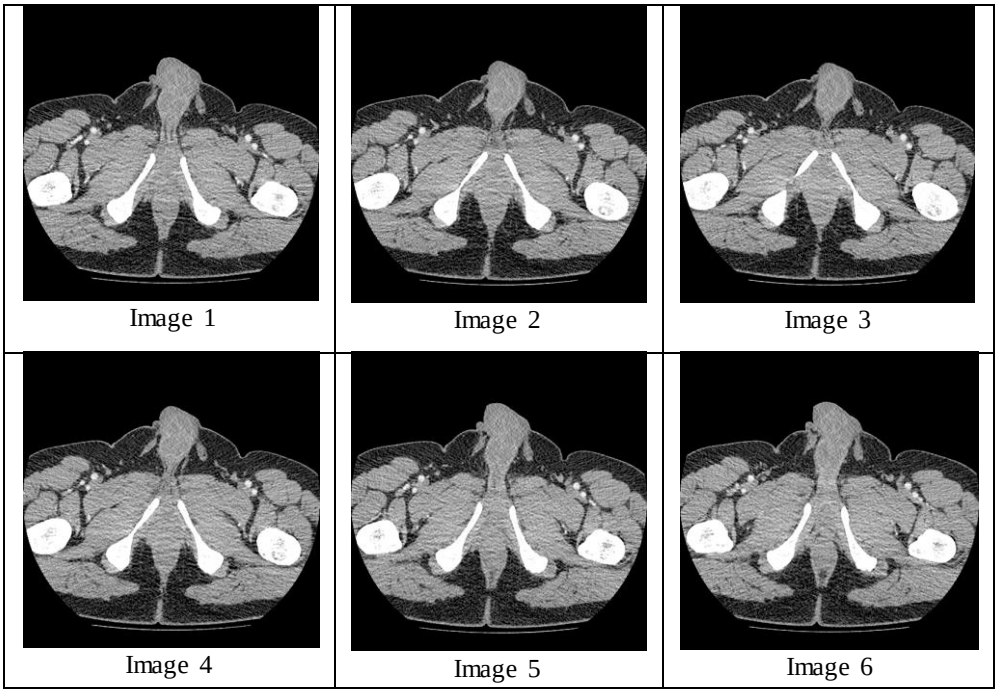


Image 1 | Image 2 | Image 3

Image 4 | Image 5 | Image 6

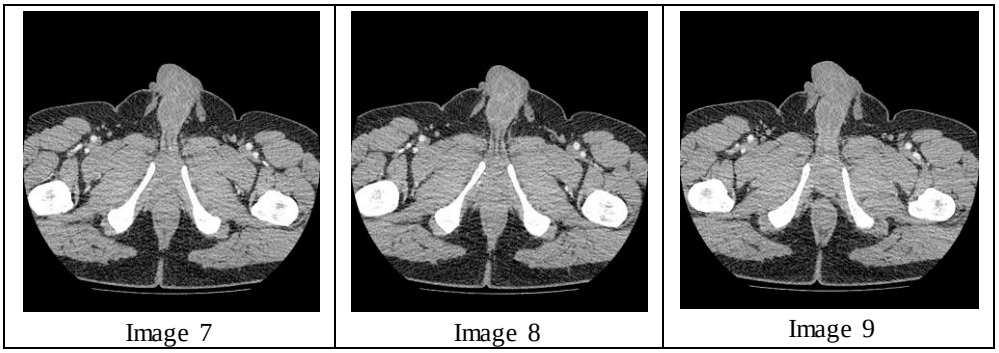


Figure (6): A group of N=9 consecutive CT images from Set 3

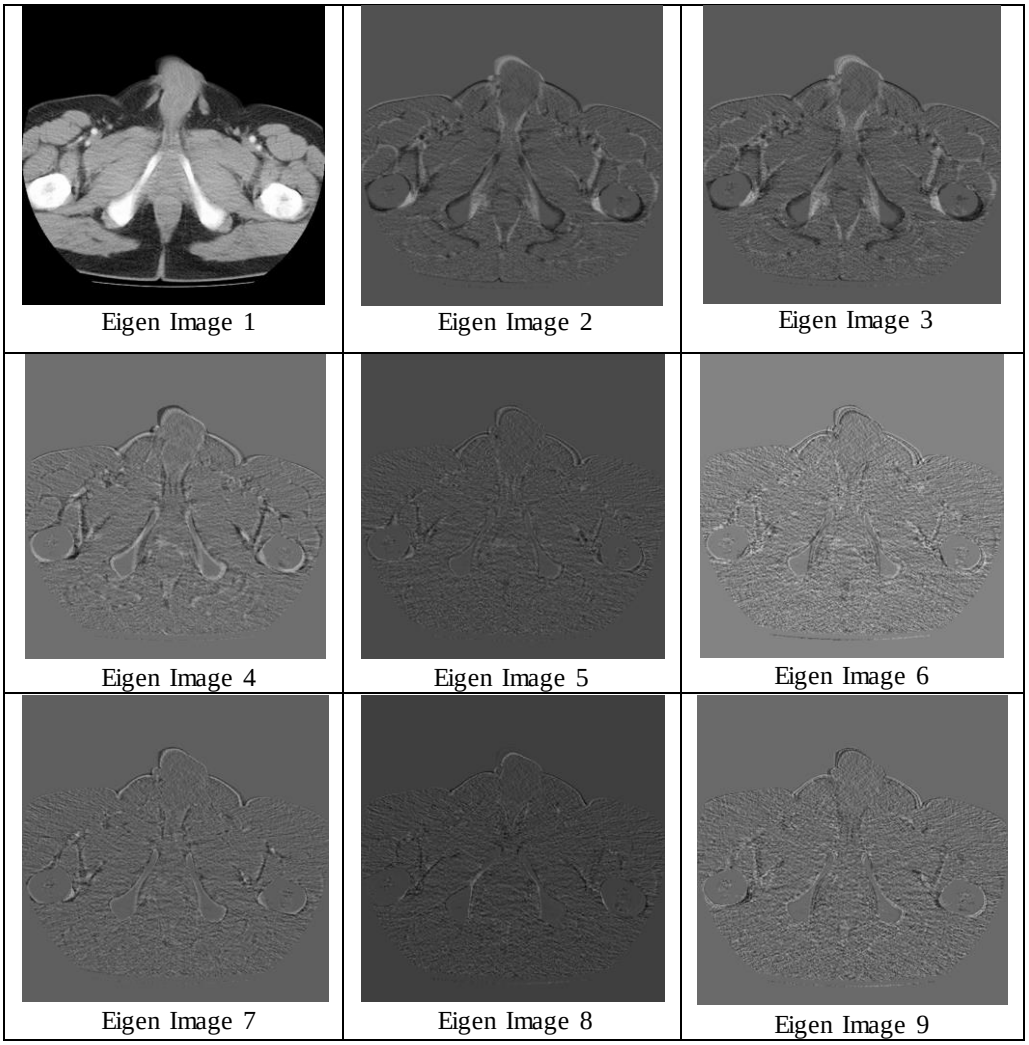


Figure (7): Eigen images, obtained from images in Set 3 as a result of the 2-level HAPCA

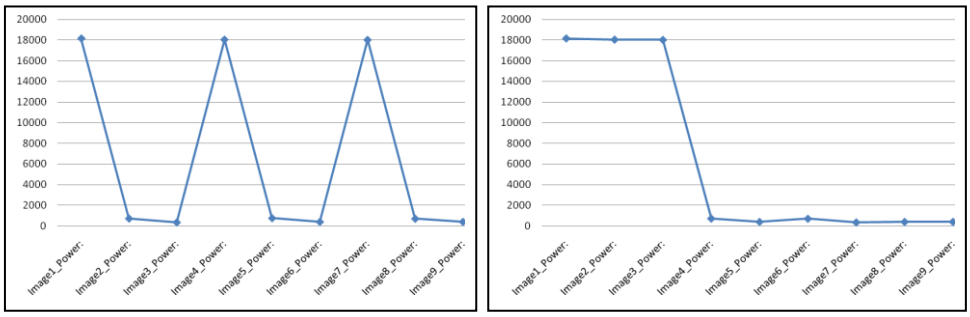
On Figure (7) are shown the eigen images, obtained after the execution of 2-levels HAPCA on the images from Set 3. As it could be noticed, the main part of the power of all 9 images is concentrated in the first eigen image, and the power of each consecutive eigen

image is quickly getting lower. This conclusion is confirmed also by the data shown in Table 2, which presents the power distribution for the pixels of the eigen images from Set 3 after the first and the second HAPCA levels, before and after their rearrangement in correspondence with the algorithm from Figure (2).

The power distribution of all eigen images in Set 3 before and after rearrangement and the relative mean power distribution are given in Table 2. On the basis of the data given in this table, on Figures (8-10) are shown the corresponding graphics for the power distribution of the obtained 9 eihgen images.

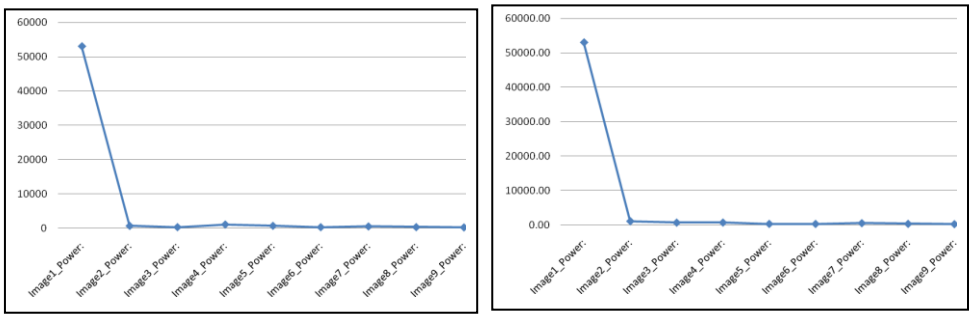
Table 2. Power distribution of all eigen images in Set 3 before and after each level and the relative mean power distribution.

Name	Level 1 (not arranged)	Level 1 (arranged)	Level 2 (not arranged)	Level 2 (arranged)	Relative mean
Eigen Im. 1	18170	18170	53041	53041	220
Eigen Im. 2	715	18056	686	1100	5
Eigen Im. 3	341	18029	316	686	3
Eigen Im. 4	18056	715	1100	710	3
Eigen Im. 5	748	748	710	316	1
Eigen Im. 6	389	694	305	305	1
Eigen Im. 7	18029	341	523	523	2
Eigen Im. 8	694	389	326	326	1
Eigen Im. 9	394	394	242	242	1



a **b**

Figure (8): Power distibution for Set 3, level 1: a) not arranged; b) arranged.



a **b**

Fig. 9. Power distibution for Set 3, level 2: a - not arranged, b - arranged.

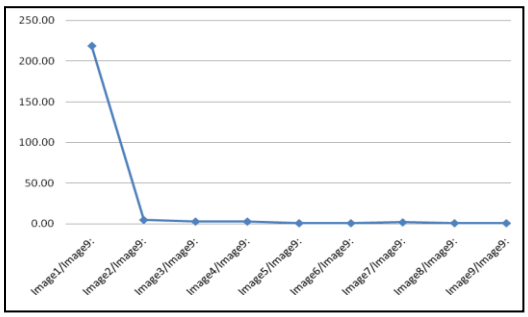


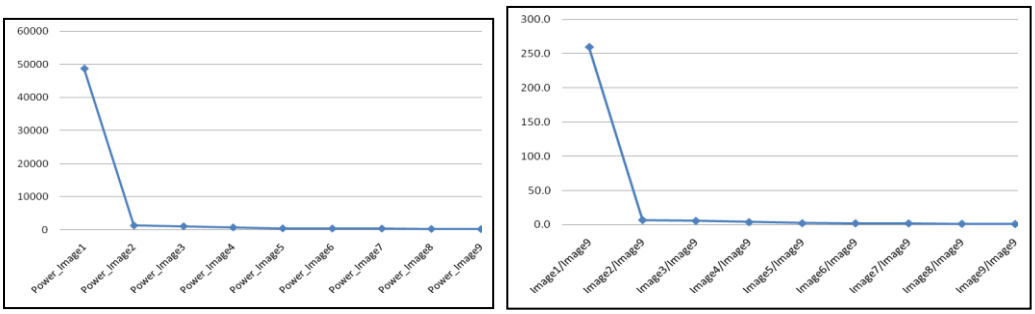
Figure (10) Relative mean power distribution for Set 3, level 2 (arranged)

In Table 3 are given the mean and the relative mean power distribution for the pixels in all 9 eigen images from Set 1,...,Set 7. On Figure (11) are shown the corresponding graphics of these distributions. From the graphic representation on Figure (11,b) it follows, that the mean power of the first eigen image for all examined sets is 220 times larger than that of the half of the next 8 eigen images. The data in the last column of Table 3 show that in the first 3 eigen images is concentrated 95.7 % of the global mean power for all 9 images in the GOP.

The pixels of the eigen images are obtained through the forward 2-level HAPCA followed by Arithmetic Coding (AC). After inverse HAPCA and AC decoding, the quality of the restored images of the processed GOP, evaluated as Peak Signal-to-Noise Ratio (PSNR), is ≥ 52 dB. This is also confirmed by the results shown on Figure (12), obtained for the eigen images in Set 1,.., Set 7. Hence, after the forward HAPCA-AC on the CT images in one GOP, followed by Inverse HAPCA-AC the original sequence could be restored with retained image quality.

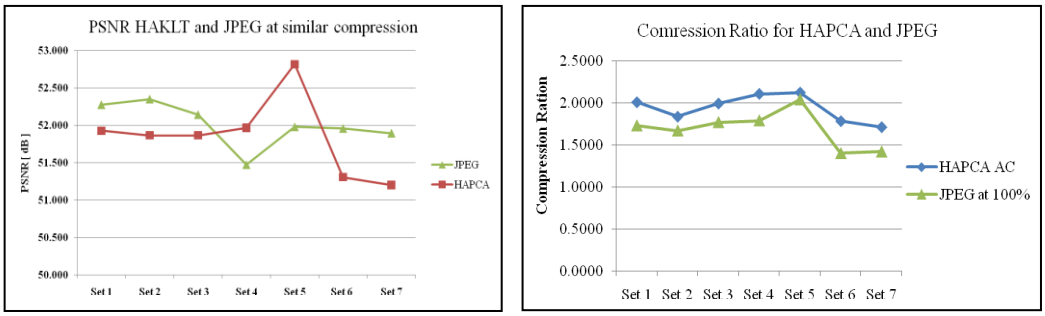
Table 3. Power Distribution, Mean Power Distribution, Relative Mean Power Distribution and Relative Mean % of the Power Distribution for all eigen images in Set 1,..,Set 7.

Image	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Mean	Relative mean	Relative mean %
Eig. 1	49992	49749	53041	53547	53774	43272	37701	48725	259.6	91.4
Eig. 2	949	811	1100	875	2331	1770	1094	1276	6.8	93.8
Eig. 3	683	2325	686	1062	625	834	1144	1051	5.6	95.7
Eig. 4	808	710	710	512	460	811	950	709	3.8	97.1
Eig. 5	522	566	316	425	300	442	364	419	2.2	97.8
Eig. 6	350	529	305	306	317	402	435	378	2.0	98.6
Eig. 7	206	222	523	317	554	306	430	365	1.9	99.2
Eig. 8	172	198	326	261	312	251	218	248	1.3	99.6
Eig. 9	130	171	242	173	254	167	177	188	1.0	100.0



a **b**

Figure (11) a Mean Power Distribution; b) Relative Mean Power Distribution for all eigen images



a **b**

Figure (12): Comparison of the quality (resp. PSNR) and of the Compression Ratio for sequences of CT images in Set 1,.., Set 7 through 2-level HAPCA-AC and JPEG (for 100% quality)

This result shows that the method could achieve higher compression if the application permits to use restored images of lower quality (for example, images with PSNR ≥ 32 are visually lossless also, and in this case the compression ratio will be much higher).

5. HAPCA Applications for Video Coding and Objects Segmentation

Here are discussed some application possibilities for HAPCA, aimed at the compression of video sequences and at the segmentation of objects or regions of interest in static color images. The offered algorithms could be also used for the processing of various medical images (static or dynamic).

5.1. HAPCA Application for Video Coding

One of the applications of the HAPCA could besuccessfully aimed at the enhancement of the compression efficiency of videosequences, represented in one of the well-known standards MPEG-1/2/4 [9,28]. For the example, shown on Figure (13), the algorithm HAPCA could be used to transform the 3 *P*-frames in the GOP, comprising *N*=12 TV frames in total.

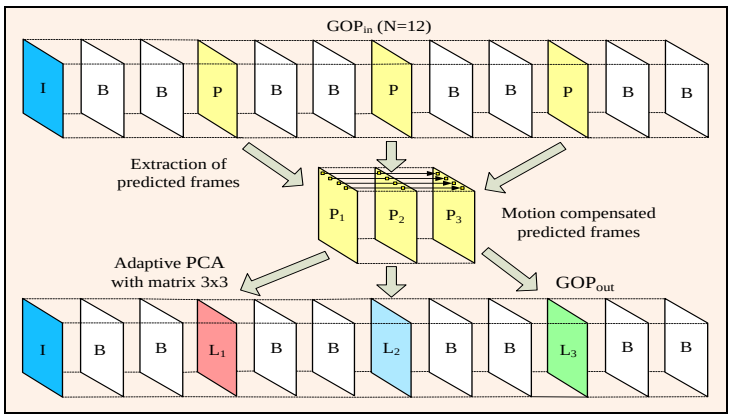


Figure (13): HAPCA compression for a video sequence, coded in one of the standard formats MPEG-1/2/4, for a GOP of *N*=12 TV frames

To achieve higher decorrrelation for the processed 3 *P*-frames, after their APCA transform is necessary to change the way for the calculation of vectors $\vec{C}_s = [C_{1s}, C_{2s}, C_{3s}]^t$ for s=1,2,..,S. Unlike the case, shown on Figure (4) for static images, the components of vectors $\vec{C}_s$ here contain the pixels with same spatial position in the *P* frames, only for the case, when they are placed in static regions (i.e., without movement). For the example on Figure (14) in such region are placed only pixels, which comprise the vector $\vec{C}_1 = [C_{11}, C_{21}, C_{31}]^t$ (for s=1).

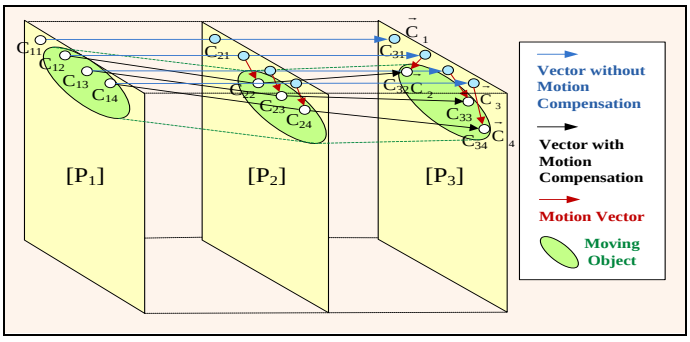


Figure (14): APCA transform for 3 motion-compensated *P*-images from one GOP

In regions with movement, colored in green on Figure (14), the pixels, which comprise the components of the corresponding vectors $\vec{C}_s=[C_{1s},C_{2s},C_{3s}]^t$ for s=2,3,4 should be moved away in relation to their positions from the regions without movement. The displacement $(\Delta i,\Delta j)$ in the region with movement for every pixel (i,j) from the frame $(k+1)$ in relation to the foregoing frame k in the group, defines the corresponding movement vector, colored in red on Figure (14). The movement vectors permit to define the positions of pixels in the regions with movement in the second and third *P*-frames, as shown on Figure (14). The so obtained vectors $\vec{C}_s$ (s=2,3,4) are with movement compensation. The APCA transform for these vectors enhances the concentration of their power mainly in the first component of the calculated vectors $\vec{L}_s$, which corresponds to the first eigen image. In this way it is possible to retain and to transfer to the decoding side the first eigen image only, together with the motion vectors for the pixels from the regions with movement of the remaining 2 eigen images (for fixed TV camera the regions with movement in every frame correspond to relatively small part of the global number of pixels). On the basis of this approach, the P-frames are restored with satisfactory quality through inverse APCA, and together with this the compression ratio for the whole GOP is increased.

5.2. The application of HAPCA algorithm for object segmentation

One of the approaches for objects segmentation in the image is based on the use of the color information. As it is known the color of every pixel in the image is represented by the vector $\vec{C}_s=[R_s,G_s,B_s]^t$ for s=1,2,..,S, where R_s,G_s,B_s are the values of the red, green and blue components for the pixel s, and S - number of pixels. In case that the colors of the pixels on the object surface are close, their corresponding vectors $\vec{C}_s$ constitute a cluster in the color space *R,G,B*. Then, the object color segmentation in the image could be performed, using: linear discriminant functions, defined through the famous methods Linear Discriminant Analysis (LDA), Fisher's Linear Discriminant (FLD), Support Vector Machine (SVM); Principal Colors with fuzzy clustering - Fuzzy Principal Component Analysis (FPCA), etc. [29-31]. In many cases however, the colors of the pixels, belonging to same object, are not close, and their corresponding color vectors do not constitute convex multitudes in the color space. Then, in order to achieve linear separability of the classes is applied preliminary expansion of the vector space, through non-linear transform using a "kernel" function of the kind: polynomial, exponential, etc. In the general case, the color space is significantly expanded in result of the non-linear transform, and in order to reduce the needed calculations, the color vectors components, which carry relatively small part of the information, should be reduced, retaining their principal components only. Such reduction could be achieved using the Kernel Principal Component Analysis (KPCA) [32]. In this case, however, is necessary to apply various iterative methods for the calculation of the eigen values and vectors of the covariance matrix of large size, defined using the vectors in the color space, expanded through kernel-function [33-36]. One more alternative is the introduction of the corresponding weight coefficients for the eigen vectors components in the expanded color space, defined using Weighted KPCA (WKPCA) [37]. The computational complexity of KPCA and WKPCA is high in these both cases.

Here is offered one new approach for objects segmentation in the color space, expanded through KPCA, which permits significant reduction of the computational complexity. For this, the number of vectors components in the expanded space is reduced by using HAPCA instead of the PCA. To illustrate this new approach, here is given one algorithm for objects color segmentation in the expanded 6-dimensional space, using a polynomial kernel-function, a space reduction through 2-level HAPCA, and SVM classifier for pixels indexation in every segmented object.

The general algorithm for objects segmentation in the extended color space, based on the Kernel HAPCA (KHAPCA) and SVM classifier of the reduced vectors, is given in Figure (15). In the block for preprocessing, 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 3-dimensional color space is transformed into a 6-dimensional, then the components p_{is} of the vectors $\vec{P}_s$ could be defined as follows:

$$\vec{P}_s = \left[R_s, G_s, B_s, \sqrt{R_s G_s}, \sqrt{B_s G_s}, \sqrt{R_s B_s}\right]^t = [P_{1s}, P_{2s}, P_{3s}, P_{4s}, P_{5s}, P_{6s}]^t \text{ for } s=1,2,..,S.$$

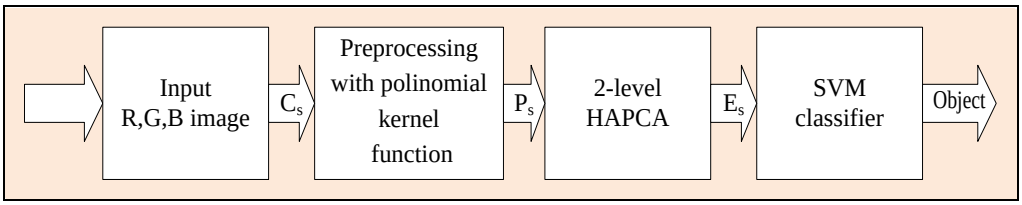


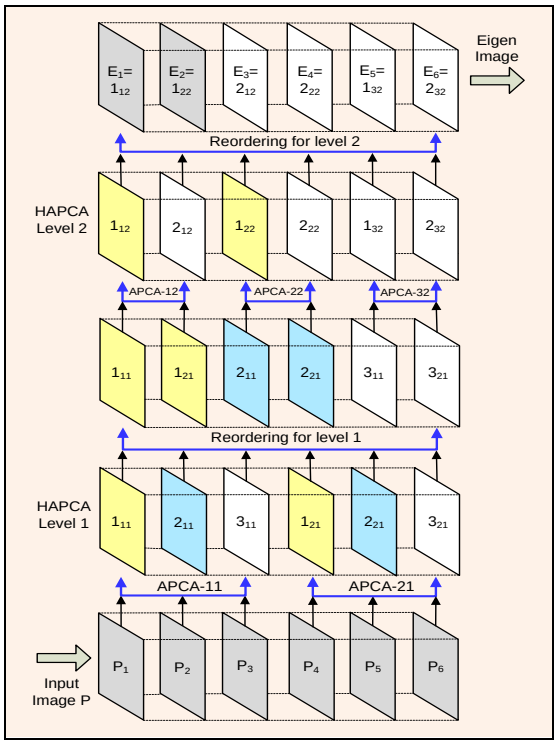
Figure (15): Block diagram of the algorithm for objects segmentation in the expanded color space

In order to put all components p_{is} in the range 0 ÷ 255, these with a consecutive number *i=4,5,6* are normalized by calculating the root of the products $R_s G_s$, $B_s G_s$, $R_s B_s$, followed by a quantization of their values in the range 0 ÷ 255. The vectors $\vec{P}_s$ are then transformed using the 2-level HAPCA, whose algorithm is shown in Fig. 16. 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 6-components vectors $\vec{P}_s = [P_{1s}, P_{2s}, P_{3s}, P_{4s}, P_{5s}, P_{6s}]^t$. In this way the performance of the SVM classifier is also simplified, because it have to process the vectors $\vec{E}_s$ in the 2-dimensional, instead of the 6-dimensional space. At its output are separated (indexed) 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 Figure (16), for the 2-level HAPCA, the 6 components of each input vector $\vec{P}_s = [P_{1s}, P_{2s}, P_{3s}, P_{4s}, P_{5s}, P_{6s}]^t$ are divided into 2 sub-groups, which contain the 3-components vectors $\vec{P}_{1s} = [P_{11s}, P_{12s}, P_{13s}]^t$ and $\vec{P}_{2s} = [P_{21s}, P_{22s}, P_{23s}]^t$ for *s=1,2,..,S.*

In the first level of HAPCA on each group of vectors $\vec{P}_{ks} = [P_{k1s}, P_{k2s}, P_{k3s}]^t$ for *k=1,2* is performed APCA with a transform matrix of size 3×3. The so obtained vectors from each

group comprise 3 "eigen" images, shown in Figure (16) with different colors (yellow, blue and white). These images are rearranged as shown in the figure, and after that are separated again, this time into 3 groups, of 2 images each. The vectors, obtained from pixels with same coordinates in the images from each group, are of 2 components.

In the second level of HAPCA for each group of 2-dimensional vectors is performed APCA with a transform matrix of size 2×2. The so obtained vectors from each group build the 2 eigen images, shown in Fig. 16 in yellow and white color correspondingly. These images are rearranged again, as shown in the figure. As a result, are obtained the 6 eigen images E_1-E_6, from which are retained the first two (E_1 and E_2) only, which carry the main information, needed for the color objects segmentation. From the algorithm, shown in Figure (16) follows, that to define E_1 and E_2 is not necessary to calculate the eigen images in the HAPCA first and second levels, colored in white. As a result, the computational complexity of HAPCA is smaller than that of PCA, for the case, when it is used to transform directly the 6-dimensional vectors $\vec{P}_s$. In this way, the general computational complexity of HAPCA and SVM, needed for the processing of the vectors $\vec{P}_s$ is lower than that, needed for the processing of same vectors with PCA and SVM. 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 SVM classifier.



Figute (16): 2-level HAPCA algorithm, used for the transformation of vectors $\vec{P}_s$ (each first APCA component is colored in yellow, each second - in blue, and each third in the first level only - in white; the reduced components are $E_3 \div E_6$)

8. Conclusions

The basic qualities of the offered HAPCA for processing a group of CT images are:

1. The lower computational complexity than that of the PCA for the whole GOP, due to the lower complexity of APCA compared to the case, for which for the calculation of the PCA matrix are used numerical methods [14,15];
2. The ability to implement efficient lossy compression for a GOP with retained visual quality of the restored images and for lossless compression also;
3. The ability for reduction of the features space in the regions of interest for a group of medical images, which comprise objects of different classes;
4. The ability for parallel processing of each sub-group of CT images in one GOP;
5. There is also a possibility for further development of the HAPCA algorithms, through: use of Integer PCA for lossless coding of medical images by analogy approach with [22,23]; compression of video sequences from stationary TV camera; segmentation of regions of interest in sequences of medical, multispectral and multi-view images; object detection in extended color space; image fusion; face recognition, etc.

Acknowledgements

This paper was supported by the Joint Research Project Bulgaria-Romania (2010-2012): “Electronic Health Records for the Next Generation Medical Decision Support in Romanian and Bulgarian National Healthcare Systems”, DNTS 02/19.

References

[1] Dougherty G., "Digital Image Processing for Medical Applications", Cambridge University Press, 2009.

[2] Suri J., Wilson D., Laxminarayan S. (Eds.), "Handbook of Biomedical Image Analysis", Vol. I and Vol. II: Segmentation Models, Kluwer Academic/Plenum Publishers, 2005.

[3] Rangayyan R., "Biomedical Image Analysis", CRC Press, Boca Raton, FL. 2005.

[4] Wu Y., "Medical image compression by sampling DCT coefficients", IEEE Trans. on Information Technology in Biomedicine, 6 (1), 86-94, 2002.

[5] S. Ramesh, D. Shanmugam, "Medical image compression using wavelet decomposition for prediction method", Intern. J. of Computer Science and Information Security (IJCSIS), 7(1), 262-265, 2010.

[6] Lalitha Y., Latte M., "Image compression of MRI image using planar coding". Intern. J. of Advanced Computer Science and Applications (IJACSA), 2 (7), 23-33, 2011.

[7] Graham R., Perriss R., Scarsbrook A., "DICOM demystified: A review of digital file formats and their use in radiological practice", Clinical Radiology, 60,1133-1140, 2005.

[8] Roos P., Viergever M., "Reversible interframe compression of medical images: a comparison of decorrelation methods", IEEE Trans. on Medical Imaging, 10(4), 538-547, 1991.

[9] Reed T. (Ed.), "Digital Image Sequence Processing, Compression, and Analysis", CRC Press, July 2004.

[10] Szilágyi S., Szilágyi L., Benyó Z., "Echocardiographic Image Sequence Compression Based on Spatial Active Appearance Model", Progress in Pattern Recognition, Image Analysis and Applications, LN in Computer Science, 2007, 4756, 841-850, 2007.

[11] Miaou S., Ke F., Chen S., "A Lossless Compression Method for Medical Image Sequences Using JPEG-LS and Interframe Coding", IEEE Trans. on Inform. Techn. in Biomedicine, 13(5), 818-821, 2009.

[12] Bitaa I., Barretb M., Phamc D.. "On Optimal Transforms in Lossy Compression of Multicomponent Images with JPEG2000", Signal Processing, 90(3), 759-773, 2010.

[13] Thirumalai V., "Distributed Compressed Representation of Correlated Image Sets", Thesis No 5264, Lausanne, EPFL, 2012.

[14] Dony R., "Karhunen-Loeve Transform", Book chapter in "The transform and data compression handbook", K. Rao, P. Yip (Eds.), Boca Raton, CRC Press, 2001.

[15] Jolliffe I., "Principal Component Analysis", 2nd ed., Springer-Verlag, NY, 2002.

[16] Ujwala P., Uma M., "Image Fusion using Hierarchical PCA", International Conference on Image Information Processing (ICIIP), 1-6, Nov. 2011.

[17] Hanafi M., Kohler A., Qannari E., "Shedding new light on Hierarchical Principal Component Analysis", J. of Chemometrics, 24(11-12), 703-709, 2010.

[18] Langs G., Bischof H., Kropatsch W., "Hierarchical Top Down Enhancement of Robust PCA", Caelli T. et al. (Eds.), SSPR&SPR'02, LNCS 2396, Springer, 234-243, 2002.

[19] Grasedyck L., "Hierarchical Singular Value Decomposition of Tensors", Preprint 20, AG Numerik/Optimierung, Philipps-Universitat Marburg, 1-29, 2009.

[20] Diamantaras K., Kung S., "Principal Component Neural Networks: Theory and Applications", John Wiley & Sons, 1996, New York.

[21] Solo V., Kong X., "Performance analysis of adaptive eigen analysis algorithms", IEEE Trans. Signal Processing, 46 (3), 1998, 636–645.

[22] Hao P., Shi Q., "Reversible Integer KLT for Progressive-to-Lossless Compression of Multiple Component Images", IEEE ICIP, Barcelona, Spain, 2003.

[23] Liu Y., Bouganis C., Cheung P., Leong P. and Motley S., "Hardware Efficient Architectures for Eigenvalue Computation", EDAA, 953-958, 2006.

[24] Kountchev R., Ivanov P., "Decorrelation of Sequences of Medical CT Images based on the Hierarchical Adaptive KLT", Intern. Workshop on Next Generation Intelligent Medical Decision Support Systems (MedDecSup'12), Sofia, Bulgaria, 2012, Ch. in "Advances in Intelligent Analysis of Medical Data and Decision Support Systems", Kountchev R., Iantovics B. (Eds.). Springer, 41-55, 2013.

[25] Korn G., Korn T., "Mathematical Handbook for Scientists and Engineers", Mc Graw-Hill Book Company, NY, 2000.

[26] Kountchev R., Kountcheva R., "Image Color Space Transform with Enhanced KLT", Book chapter in "New Advances in Intelligent Decision Technologies", Nakamatsu K., Wren G., Jain L., Howlett R. (Eds.), Springer-Verlag, 171-182, 2009.

[27] Kountchev R., Nakamatsu K., "One Approach for Grayscale Image Decorrelation with Adaptive Multi-level 2D KLT", Chapter in: "Advances in Knowledge-Based and Intelligent Information and Engineering Systems", Graña M., Toro C., Posada J., Howlett R., Jain L. (Eds.), IOS Press, 1303-1312, 2012.

[28] Mukhopadhyay J., "Image and Video Processing in the Compressed Domain", CRC Press, 2011.

[29] Duda R., Hart P., Stork D., "Pattern Classification", Wiley Interscience, 2005.

[30] Abe S., "Support Vector Machines for Pattern Classification", Springer, 2005.

[31] Abadpour A. and Kasaei S., "Principal Color and Its Application to Color Image Segmentation", Scientia Iranica, 15(2), 238-245, 2008.

[32] Scholkopf B., Smola A., and Muller K. "Nonlinear component analysis as a kernel eigenvalue problem", Neural Computation, 10(5), 1299-1319, 1998.

[33] Oja E. "Principal components, minor components, and linear neural networks", Neural Networks, 5, 927-935, 1992.

[34] Kim K., Franz M., and Scholkopf B., "Iterative kernel principal component analysis for image modeling", IEEE Trans. on Pattern Analysis and Machine Learning, 27(9), 1351-1366, 2005.

[35] Schraudolph N., Günter S., Vishwanathan S.. "Fast Iterative Kernel PCA". In "Advances in Neural Information Processing Systems" (NIPS), MIT Press, Cambridge, MA, 1225-1232, 2007.

[36] Lee J., Verleysen M., "Nonlinear Dimensionality Reduction", Springer, 2007.

[37] Alzate C., Suykens J., "Image Segmentation using a Weighted Kernel PCA Approach to Spectral Clustering", Proc. of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing (CIISP'2007), Honolulu, USA, 208-213, 2007.