Sliding Window Recursive HAPCA for 3D Image Decomposition

ROUMIANA KOUNTCHEVA
TK Engineering

Mladost 3, Pb. 12, Sofia 1712
BULGARIA
kountcheva_r@yahoo.com

ROUMEN KOUNTCHEV
Radio Communications and Video Technologies
Technical University - Sofia
Bul. Kl. Ohridsky 8, Sofia 1000
BULGARIA
rkountch@tu-sofia.bg http://www.tu-sofia.bg

Abstract: - The famous method Principal Components Analysis (PCA) is the basic approach for decomposition of 3D tensor images (for example, multi- and hyper-spectral, multi-view, computer tomography, video, etc.). As a result of the processing, their information redundancy is significantly reduced. This is of high importance for the efficient compression and for the reduction of the features space needed, when object recognition or search is performed. The basic obstacle for the wide application of PCA is the high computational complexity. One of the approaches to overcome the problem is to use algorithms, based on the recursive PCA. The well-known methods for recursive PCA are aimed at the processing of sequences of images, represented as non-overlapping groups of vectors. In this work is proposed new method, called Sliding Recursive Hierarchical Adaptive PCA, based on image sequence processing in a sliding window. The new method decreases the number of calculations needed, and permits parallel implementation. The results obtained from the algorithm simulation, confirm its efficiency. The lower computational complexity of the new method facilitates its application in the real-time processing of 3D tensor images.

Key-Words: - Hierarchical Adaptive PCA, Sliding Recursive PCA, 3D tensor image decomposition.

1 Introduction

The 3D image processing attracts recently significant attention because of its wide use in various areas: medical diagnostics, analysis of satellite and aero-photo images, compression of video sequences, multi-view images, etc. Each 3D image is mathematically represented as a third-order tensor. In accordance to [1], the tensor image could be transformed into a group of vectors positioned in three different ways: (Mode-1 (columns), Mode-2 (rows), and Mode-3 (fibres). Mode-3 which ensures strongest correlation between the neighbour vectors in the group, is used in this work. The main feature of the 3D images is the very high information redundancy, due to the strong correlation between the neighbor elements in the group. The method Principal Components Analysis (PCA) also known as Karhunen-Loève Transform (KLT), is usually used to achieve optimum orthogonal transform of data, signals and images and to reduce the information redundancy with minimum mean-square error [2-7]. The PCA method is very efficient for processing of sequences of correlated images, represented as a 3D tensor. In accordance with [2], the main problems which impede the wide use of PCA for 3D image processing, are the high computational complexity and the absence of "fast" computational algorithm (such algorithm exists only for the class of images, which could be represented as first-order Markov process). To overcome the mentioned problems, new approaches were developed [8-10], related to the Hierarchical PCA (HPCA). In [3] is introduced hierarchical recursive block processing of matrices. In [9] was used the “divide-and-conquer” strategy, in accordance to which the n-dimensional vectors in the first level of HPCA are divided into groups of vectors of small number of components, and for them are calculated local PCAs of low computational complexity. In the next (second) level, the transformed vectors are arranged in clusters, after which they are processed with PCA, and divided into new groups, where the vectors have lower number of components, etc. Disadvantage of the method is, that for the first component is got full mutual decorrelation, while for the next components the mutual decorrelation is only partial. The main advantage of the Hierarchical Adaptive PCA (HAPCA) [10] is, that it gives full mutual decorrelation for the output components in each level, which is a result of the full transform in all decomposition levels and of the adaptation towards the local statistics of the groups of vectors with 2 or 3 components only. Besides, the above-mentioned HPCA methods are not aimed at recursive calculations.

In the last years, the interest towards the recursive PCA for 3D tensor images executed in a sliding temporal window, was noticeably increased. In significant number of publications [11-13] were introduced methods and algorithms for recursive PCA calculation for the cases, when non-overlapping groups of vectors were used to represent the tensors. In many PCA applications (for example, tensor deconvolution, etc.), special interest attracts the 3D image transform using a sliding window, whose components overlap only in part. In [14] is presented an algorithm for 2D matrix sketching with time-based and sequence-based sliding windows. For the decomposition of time-dynamic 3D data in [15] is offered online tensor decomposition, based on a modification of the famous CANDECOMP/PARAFAC (CP) algorithm. As a result, the CP algorithm is accelerated because the „new“ loading matrices are calculated by using the „old“ matrices (got in earlier time moments), in accordance to the criterion for minimum approximation error.

This work presents new method for recursive transform of 3D tensor images, based on the HAPCA algorithm, developed earlier by the authors. The investigated images represented as 3rd-order tensor, are treated as a group of parallel fibres (vectors). The basic objective of the new method is to achieve lower computational complexity than that of the basic algorithm, HAPCA. As it was proved in [10], HAPCA has lower computational complexity than the "classic" PCA and is better adapted for parallel processing.

2 Method for Sliding Recursive Hierarchical Adaptive PCA

In accordance to Gonzales and Woods [4], the transformation of a group of vectors through "classic" PCA needs the following calculations: 1) vectors' covariance matrix; 2) eigen values and eigen vectors of the covariance matrix; 3) transform matrix, whose rows are composed by the eigen vectors; 4) direct PCA for each vector.

In this work is investigated a PCA-based decomposition for a group of correlated images. The decomposition is executed for the group of images (P), framed at same moment by a sliding window which moves over the processed group of images. In each consecutive position, one image (a matrix of size M×N) drops out of the window, and one new is involved. To simplify the execution of PCA for the group of vectors, which represent the images in the window, it is calculated recursively, using the transform result for the previous position. Besides, to reduce additionally the computational complexity, here is used hierarchical adaptive PCA, related to the fixed window position.

When video sequences are processed, the movement of the sliding window is time-related, while for a group of multispectral images (MSI) the spectral bands are used instead of the time positions. As a 3D-tensor image example, on Fig. 1 is shown a group of multispectral images (MSI), corresponding to four neighbour spectral bands.

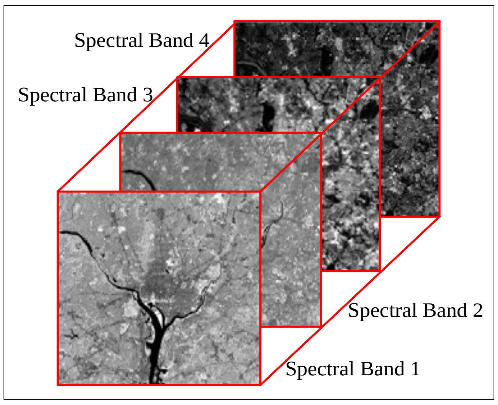


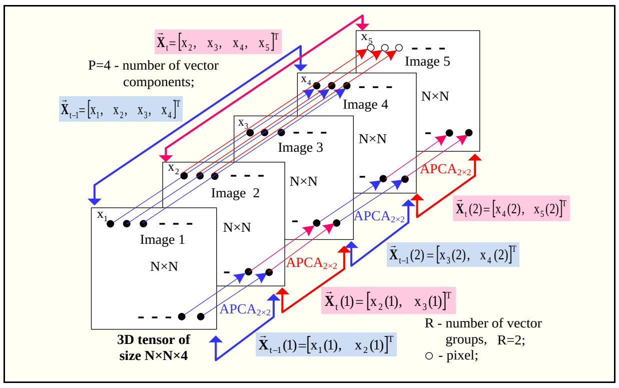
Fig. 1. A group of MSI images of size N×N each, which represents 3D tensor image of size N×N×4.

The main stages of the method *Sliding Recursive Hierarchical Adaptive PCA* (SR-HAPCA), offered here, are given below. These stages comprise recursive calculation of: 1) the Covariance matrix of the 3D images; 2) the Eigen vectors of the covariance matrix; 3) the Adaptive PCA for two-component vectors; 4) the Sliding L-levels HAPCA. After that follows the description of the algorithm.

2.1 Recursive Calculation of the Covariance Matrix of 3D Images

The recursive calculation of the covariance matrix of 3D images, represented as a group of vectors in a sliding temporal window of length P is explained here first for P=4, and after that - generalized for $P=2^L$. The image sequence in the window is treated as a group of P-dimensional vectors, whose components are the pixels with same spatial position in the matrix of each image - the frontal cut of the corresponding third-order tensor.

As an example on Fig. 2 are shown several four-dimensional vectors $\vec{\mathbf{X}}_{t-1,s}$ and $\vec{\mathbf{X}}_{t,s}$, which represent the groups of matrix images 1-4 and 2-5, framed by the sliding window of length P=4 at discrete moments (t-1), and (t). In this case, vectors components are defined by the pixels of same spatial position in all four consecutive images.



$\vec{X}_t=[x_2, x_3, x_4, x_5]^T$
P=4 - number of vector components;
$\vec{X}_{t-1}=[x_1, x_2, x_3, x_4]^T$
x_5
Image 5
x_4
Image 4
N×N
x_3
Image 3
N×N
x_2
Image 2
N×N
x_1
Image 1
N×N
N×N
APCA$_{2\times2}$
APCA$_{2\times2}$
APCA$_{2\times2}$
APCA$_{2\times2}$
$\vec{X}_t(2)=[x_4(2), x_5(2)]^T$
$\vec{X}_{t-1}(2)=[x_3(2), x_4(2)]^T$
$\vec{X}_t(1)=[x_2(1), x_3(1)]^T$
$\vec{X}_{t-1}(1)=[x_1(1), x_2(1)]^T$
3D tensor of size N×N×4
R - number of vector groups, R=2;
○ - pixel;

Fig. 2. Representation of 3D tensor recursive transform, based on APCA$_{2\times2}$.

Each image is a matrix of size N×N and comprises N^2 pixels, shown as black circles. In the example, vectors $\vec{\mathbf{X}}_{t-1,s}$ and $\vec{\mathbf{X}}_{t,s}$ overlap, and the first component of $\vec{\mathbf{X}}_{t,s}$ coincides to the second component of $\vec{\mathbf{X}}_{t-1,s}$, etc.

In correspondence with [10], where the method HAPCA was introduced, each of these vectors is substituted by R=P/2=2 two-component vectors, where $\vec{\mathbf{X}}_{t-1,s}$ is replaced by the couple $\vec{\mathbf{X}}_{t-1,s}(1)$, $\vec{\mathbf{X}}_{t-1,s}(2)$, and $\vec{\mathbf{X}}_{t,s}$ - by the couple $\vec{\mathbf{X}}_{t,s}(1), \vec{\mathbf{X}}_{t,s}(2)$, when s=1,2,..,$N^2$. For each two-component vector from the first and second group, is calculated Adaptive PCA with transform matrix of size 2×2 (APCA$_{2\times2}$). The covariance matrices $\mathbf{K}_{t-1}(1)$, $\mathbf{K}_{t-1}(2)$ of the vectors in groups 1 and 2 at the moment (t-1), and of the covariance matrices $\mathbf{K}_t(1)$, $\mathbf{K}_t(2)$ of the vectors in groups 1 and 2 at the moment (t), have to be calculated in advance.

Here, the following notations are introduced:

$\bar{x}=E\{x_s\}=(1/N^2)\sum_{s=1}^{N^2}x_s$ - operation for „averaging“ the variable x_s for s = 1, 2,..,N^2, where N^2 is the number of vectors;

$\vec{\mathbf{X}}_{t-1,s}(1)=[x_{1,s}(1), x_{2,s}(1)]^T$; $\vec{\mathbf{X}}_{t-1,s}(2)=[x_{3,s}(2), x_{4,s}(2)]^T$ - vectors from group 1 and 2 at the moment (t-1);

$\vec{\mathbf{X}}_{t,s}(1)=[x_{2,s}(1), x_{3,s}(1)]^T$; $\vec{\mathbf{X}}_{t,s}(2)=[x_{4,s}(2), x_{5,s}(2)]^T$ - vectors from group 1 and 2 at the moment (t).

The mean vectors from groups 1 and 2 at time moments (t-1) and (t) are:

$$\vec{\boldsymbol{\mu}}_{t-1}(1)=[E\{x_{1,s}(1)\}, E\{x_{2,s}(1)\}]^T=[\bar{x}_1(1), \bar{x}_2(1)]^T;$$

$$\vec{\boldsymbol{\mu}}_{t-1}(2)=[E\{x_{3,s}(2)\}, E\{x_{4,s}(2)\}]^T=[\bar{x}_3(2), \bar{x}_4(2)]^T;$$

$$\vec{\boldsymbol{\mu}}_t(1)=[E\{x_{2,s}(1)\}, E\{x_{3,s}(1)\}]^T=[\bar{x}_2(1), \bar{x}_3(1)]^T;$$

$$\boldsymbol{\mu}_t(2)=[E\{x_{4,s}(2)\}, E\{x_{5,s}(2)\}]^T=[\bar{x}_4(2), \bar{x}_5(2)]^T.$$

The covariance matrices of the vectors from groups 1 and 2 at time moments (t-1) and (t) are:

$$\mathbf{K}_{t-1}(1)=E\{\vec{\mathbf{X}}_{t-1,s}(1)\vec{\mathbf{X}}^T_{t-1,s}(1)\}-\vec{\boldsymbol{\mu}}_{1,t-1}(1)\vec{\boldsymbol{\mu}}^T_{1,t-1}(1), \quad (1)$$

$$\mathbf{K}_{t-1}(2)=E\{\vec{\mathbf{X}}_{t-1,s}(2).\vec{\mathbf{X}}^T_{t-1,s}(2)\}-\vec{\boldsymbol{\mu}}_{2,t-1}(2).\vec{\boldsymbol{\mu}}^T_{2,t-1}(2), \quad (2)$$

$$\mathbf{K}_t(1)=E\{\vec{\mathbf{X}}_{t,s}(1).\vec{\mathbf{X}}^T_{t,s}(1)\}-\vec{\boldsymbol{\mu}}_{1,t}(1).\vec{\boldsymbol{\mu}}^T_{1,t}(1), \quad (3)$$

$$\mathbf{K}_t(2)=E\{\vec{\mathbf{X}}_{t,s}(2).\vec{\mathbf{X}}^T_{t,s}(2)\}-\vec{\boldsymbol{\mu}}_{1,t}(2).\vec{\boldsymbol{\mu}}^T_{1,t}(2). \quad (4)$$

The difference covariance matrices for both groups of vectors at the time moment (t) are:

$$\Delta\mathbf{K}_t(1)=\mathbf{K}_t(1)-\mathbf{K}_{t-1}(1), \;\; \Delta\mathbf{K}_t(2)=\mathbf{K}_t(2)-\mathbf{K}_{t-1}(2). \quad (5)$$

The covariance matrix for each group of vectors could be recursively calculated as given below:

$$\mathbf{K}_t(1)=\mathbf{K}_{t-1}(1)]+\Delta\mathbf{K}_t(1), \; \mathbf{K}_t(2)=\mathbf{K}_{t-1}(2)]+\Delta\mathbf{K}_t(2) \quad (6)$$

for t=1,2,.., where:

$$\Delta\mathbf{K}_t(1)=\Delta\mathbf{R}_t(1)-\Delta\boldsymbol{\mu}_t(1), \; \Delta\mathbf{K}_t(2)=\Delta\mathbf{R}_t(2)-\Delta\boldsymbol{\mu}_t(2). \quad (7)$$

Here $\Delta\mathbf{R}_t(1)$ and $\Delta\mathbf{R}_t(2)$ are the difference covariance matrices of the vectors from groups 1 and 2. For the moment (t) these matrices could be represented in the following way:

$$\Delta\mathbf{K}_t(1)=\begin{bmatrix}\Delta k_{1,t}(1) & \Delta k_{3,t}(1)\\ \Delta k_{3,t}(1) & \Delta k_{2,t}(1)\end{bmatrix} \text{ for} \quad (8)$$

$$\Delta k_{1,t}(1)=[\overline{x_2^2(1)}-\overline{x_1^2(1)}]-[\bar{x}_2^2(1)-\bar{x}_1^2(1)];$$

$$\Delta k_{2,t}(1)=[\overline{x_3^2(1)}-\overline{x_2^2(1)}]-[\bar{x}_3^2(1)-\bar{x}_2^2(1)];$$

$$\Delta k_{3,t}(1)=\overline{x_2(1)[x_3(1)-x_1(1)]}-\bar{x}_2(1)[\bar{x}_3(1)-\bar{x}_1(1)].$$

$$\Delta\mathbf{K}_t(2)=\begin{bmatrix}\Delta k_{1,t}(2) & \Delta k_{3,t}(2)\\ \Delta k_{3,t}(2) & \Delta k_{2,t}(2)\end{bmatrix} \text{ for} \quad (9)$$

$$\Delta k_{1,t}(2)=[\overline{x_4^2(2)}-\overline{x_3^2(2)}]-[\bar{x}_4^2(2)-\bar{x}_3^2(2)];$$

$$\Delta k_{2,t}(2)=[\overline{x_5^2(2)}-\overline{x_4(2)}]-[\bar{x}_5^2(2)-\bar{x}_4^2(2)];$$

$$\Delta k_{3,t}(2)=\overline{x_4(2)[x_5(2)-x_3(2)]}-\bar{x}_4(2)[\bar{x}_5(2)-\bar{x}_3(2)].$$

In accordance with [16], these matrices are symmetric in respect to their main diagonal and from this it follows that their eigen values are positive numbers. In particular, for $\vec{\boldsymbol{\mu}}_{t-1}(1)=\vec{\boldsymbol{\mu}}_{t-1}(2)=\vec{\boldsymbol{\mu}}_t(1)=\vec{\boldsymbol{\mu}}_t(2)=0$. Then:

$$\mathbf{K}_{t-1}(1)=\mathbf{R}_{t-1}(1)=\begin{bmatrix}\overline{x_1^2(1)} & \overline{x_1(1)x_2(1)}\\ \overline{x_1(1)x_2(1)} & \overline{x_2^2(1)}\end{bmatrix}, \quad (10)$$

$$\mathbf{K}_{t-1}(2)=\mathbf{R}_{t-1}(2)=\begin{bmatrix}\overline{x_3^2(2)} & \overline{x_3(2)x_4(2)}\\ \overline{x_3(2)x_4(2)} & \overline{x_4^2(2)}\end{bmatrix}, \quad (11)$$

$$\mathbf{K}_t(1)=\mathbf{R}_t(1)=\begin{bmatrix}\overline{x_2^2(1)} & \overline{x_2(1)x_3(1)}\\ \overline{x_2(1)x_3(1)} & \overline{x_3^2(1)}\end{bmatrix}, \quad (12)$$

$$\mathbf{K}_t(2)=\mathbf{R}_t(2)=\begin{bmatrix}\overline{x_4^2(2)} & \overline{x_4(2)x_5(2)}\\ \overline{x_4(2)x_5(2)} & \overline{x_5^2(2)}\end{bmatrix}, \quad (13)$$

$$\Delta\mathbf{K}_t(1)=\begin{bmatrix}\overline{x_2^2(1)}-\overline{x_1^2(1)} & \overline{x_2(1)[x_3(1)-x_1(1)]}\\ \overline{x_2(1)[x_3(1)-x_1(1)]} & \overline{x_3^2(1)}-\overline{x_2^2(1)}\end{bmatrix}, \quad (14)$$

$$\Delta\mathbf{K}_t(2)=\begin{bmatrix}\overline{x_4^2(2)}-\overline{x_3^2(2)} & \overline{x_4(2)[x_5(2)-x_3(2)]}\\ \overline{x_4(2)[x_5(2)-x_3(2)]} & \overline{x_5^2(2)}-\overline{x_4(2)}\end{bmatrix}. \quad (15)$$

In accordance with [12], in case, that vectors do not overlap at the time moments (t) and (t-1), the correlation matrix $\mathbf{R}_t$ for (t) is calculated recursively:

$$\mathbf{R}_t=\frac{1}{t}\sum_{s=1}^{t}\vec{\mathbf{X}}_s.\vec{\mathbf{X}}_s^T=\frac{t-1}{t}\mathbf{R}_{t-1}+\frac{1}{t}\vec{\mathbf{X}}_t.\vec{\mathbf{X}}_t^T, \quad (16)$$

where t is the index, which defines the number of the current vector $\vec{\mathbf{X}}_t$ from the sequence of vectors for t=1,2,..,N^2. Obviously, this approach for recursive calculation of covariance matrices $\mathbf{R}_t(1)$, and $\mathbf{R}_t(2)$, is not applicable for the overlapping vectors, defined by the sliding window at the time moments (t) and (t-1).

2.2 Recursive Calculation of the Eigen Vectors of the Covariance Matrix

The eigen vectors $\vec{\mathbf{\Phi}}_{k,t}$ of the matrix $\mathbf{K}_t$ for the moment (t), are defined by the system of equations:

$$\mathbf{K}_t\vec{\mathbf{\Phi}}_{k,t}=\lambda_{k,t}\vec{\mathbf{\Phi}}_{k,t},\ \vec{\mathbf{\Phi}}_{k,t}^T\vec{\mathbf{\Phi}}_{k,t}=1 \quad \text{for } k=1,2, \quad (17)$$

where $\lambda_{k,t}$ are the eigen values of the matrix $\mathbf{K}_t$.

The eigen vectors $\Delta\vec{\mathbf{\Phi}}_{k,t}$ of the difference matrix $\Delta\mathbf{K}_t$ are defined in similar way:

$$\Delta\mathbf{K}_t\Delta\vec{\mathbf{\Phi}}_{k,t}=\Delta\lambda_{k,t}\Delta\vec{\mathbf{\Phi}}_{k,t}, \quad (18)$$

$$\Delta\vec{\mathbf{\Phi}}_{k,t}=\vec{\mathbf{\Phi}}_{k,t}-\vec{\mathbf{\Phi}}_{k,t-1}. \quad (19)$$

Hence, the recursive calculation of the eigen vectors is defined by the relation:

$$\vec{\mathbf{\Phi}}_{k,t}=\vec{\mathbf{\Phi}}_{k,t-1}+\Delta\vec{\mathbf{\Phi}}_{k,t} \quad \text{for } k=1,2. \quad (20)$$

2.3 Recursive APCA Calculation for Two-Component Vectors

The direct APCA$_{2\times2}$ for the two-component vectors $\vec{\mathbf{X}}_{s,t}$ with zero mean vector $(\vec{\boldsymbol{\mu}}_t=0)$ is defined by the relation:

$$\vec{\mathbf{Y}}_{s,t}=\mathbf{\Phi}_t\vec{\mathbf{X}}_{s,t} \text{ for } s=1,2,..,N^2 \text{ and } t=1,2,..., \quad (21)$$

where $\mathbf{\Phi}_t^T=[\vec{\mathbf{\Phi}}_{1,t},\vec{\mathbf{\Phi}}_{2,t}]$ is the APCA$_{2\times2}$ transform matrix at the time moment (t), and $\vec{\mathbf{Y}}_{s,t}$ - the transformed two-component vector. The above-given transform could be represented also as follows:

$$\vec{\mathbf{Y}}_{s,t}=\begin{bmatrix}\vec{\mathbf{\Phi}}_{1,t-1}^T+\Delta\vec{\mathbf{\Phi}}_{1,t}^T\\ \vec{\mathbf{\Phi}}_{2,t-1}^T+\Delta\vec{\mathbf{\Phi}}_{2,t}^T\end{bmatrix}\vec{\mathbf{X}}_{s,t}=\vec{\mathbf{Y}}_{s,t-1}+\Delta\vec{\mathbf{Y}}_{s,t-1}, \quad (22)$$

$$\vec{\mathbf{Y}}_{s,t-1}=\begin{bmatrix}\vec{\mathbf{\Phi}}_{1,t-1}^T\\ \vec{\mathbf{\Phi}}_{2,t-1}^T\end{bmatrix}\vec{\mathbf{X}}_{s,t}=\mathbf{\Phi}_{t-1}\vec{\mathbf{X}}_{s,t}. \quad (23)$$

2.4 Recursive Calculation of the Sliding L-levels HAPCA

To apply HAPCA of L levels (for L=log$_2$P) on N^2 vectors of P components, each vector is replaced by R=P/2 two-component vectors. In each level, on the two-component vector $\vec{\mathbf{X}}_{s,t}(r)$ from the group r, which has zero mean vector, is executed the transform:

$$\vec{\mathbf{Y}}_{s,t}(r)=\begin{bmatrix}\vec{\mathbf{\Phi}}_{1,t-1}^T(r)+\Delta\vec{\mathbf{\Phi}}_{1,t}^T(r)\\ \vec{\mathbf{\Phi}}_{2,t-1}^T(r)+\Delta\vec{\mathbf{\Phi}}_{2,t}^T(r)\end{bmatrix}\vec{\mathbf{X}}_{s,t}(r)=\vec{\mathbf{Y}}_{s,t-1}(r)+\Delta\vec{\mathbf{Y}}_{s,t-1}(r) \quad (24)$$

for $r=1,2,..R$, where:

$$\vec{\mathbf{Y}}_{s,t-1}(r)=\begin{bmatrix}\vec{\mathbf{\Phi}}_{1,t-1}^T(r)\\ \vec{\mathbf{\Phi}}_{2,t-1}^T(r)\end{bmatrix}\vec{\mathbf{X}}_{s,t}(r)=\mathbf{\Phi}_{t-1}(r)\vec{\mathbf{X}}_{s,t}(r), \quad (25)$$

$$\Delta\vec{\mathbf{Y}}_{s,t-1}(r)=\begin{bmatrix}\Delta\vec{\mathbf{\Phi}}_{1,t}^T(r)\\ \Delta\vec{\mathbf{\Phi}}_{2,t}^T(r)\end{bmatrix}\vec{\mathbf{X}}_{s,t}(r)=\Delta\mathbf{\Phi}_t(r)\vec{\mathbf{X}}_{s,t}(r). \quad (26)$$

The solutions of the system of equations (17) and (18) used to define matrices $\mathbf{\Phi}_{t-1}(r)$ and $\Delta\mathbf{\Phi}_t(r)$, are similar to these for the non-recursive HAPCA [10]. They are defined by the relations below:

$$\mathbf{\Phi}_{t-1}(r)=\begin{bmatrix}\Phi_{11,t-1}(r) & \Phi_{21,t-1}(r)\\ \Phi_{12,t-1}(r) & \Phi_{22,t-1}(r)\end{bmatrix}=\begin{bmatrix}\cos\theta_{t-1}(r) & \sin\theta_{t-1}(r)\\ -\sin\theta_{t-1}(r) & \cos\theta_{t-1}(r)\end{bmatrix} \quad \text{for r=1,2,..R,} \quad (27)$$

$$\theta_{t-1}(r)=\operatorname{arctg}\left(\frac{\Phi_{21,t-1}(r)}{\Phi_{11,t-1}(r)}\right)= \quad (28)$$

$$=\operatorname{arctg}\left(\frac{2k_{3,t-1}(r)}{k_{1,t-1}(r)-k_{2,t-1}(r)+\sqrt{[k_{1,t-1}(r)-k_{2,t-1}(r)]^2+4k_{3,t-1}^2(r)}}\right),$$

$$k_{1,t-1}(r)=\overline{x_1^2(r)},\ k_{2,t-1}(r)=\overline{x_2^2(r)},\ k_{3,t-1}(r)=\overline{x_1(r)x_2(r)}.$$

and correspondingly:

$$\Delta\mathbf{\Phi}_t(r)=\begin{bmatrix}\Delta\Phi_{11,t}(r) & \Delta\Phi_{21,t}(r)\\ \Delta\Phi_{12,t}(r) & \Delta\Phi_{22,t}(r)\end{bmatrix}=\begin{bmatrix}\cos[\Delta\theta_t(r)] & \sin[\Delta\theta_t(r)]\\ -\sin[\Delta\theta_t(r)] & \cos[\Delta\theta_t(r)]\end{bmatrix} \quad \text{for r=1,2,..,R,} \quad (29)$$

$$\Delta\theta_t(r) = \mathrm{arctg}\left(\frac{\Delta\Phi_{21,t}(r)}{\Delta\Phi_{11,t}(r)}\right) = \tag{30}$$

$$= \mathrm{arctg}\left(\frac{2\Delta k_{3,t}(r)}{\Delta k_{1,t}(r) - \Delta k_{2,t}(r) + \sqrt{[\Delta k_{1,t}(r) - \Delta k_{2,t}(r)]^2 + 4\Delta k_{3,t}^2(r)}}\right),$$

$$\Delta k_{1,t}(r) = \overline{x_2^2(r)} - \overline{x_1^2(r)},\ \Delta k_{2,t}(r) = \overline{x_3^2(r)} - \overline{x_2^2(r)},$$

$$\Delta k_{3,t}(r) = \overline{x_2(r)[x_3(r) - x_1(r)]}. \tag{31}$$

Hence, the transform matrix for the vectors in the group r (for r=1,2,...,R) at the time moment (t) is:

$$\mathbf{\Phi}_t(r) = \begin{bmatrix} \Phi_{11,t}(r) & \Phi_{21,t}(r) \\ \Phi_{12,t}(r) & \Phi_{22,t}(r) \end{bmatrix} = \begin{bmatrix} \cos[\theta_t(r)] & \sin[\theta_t(r)] \\ -\sin[\theta_t(r)] & \cos[\theta_t(r)] \end{bmatrix} \tag{32}$$

$$\theta_t(r) = \theta_{t-1}(r) + \Delta\theta_t(r). \tag{33}$$

If $\Delta\theta_t(r) \approx 0$, then $\theta_t(r) \approx \theta_{t-1}(r)$ and we get $\vec{\mathbf{Y}}_{s,t} = \vec{\mathbf{Y}}_{s,t-1}$. In this case the mutual correlation between two sequential images from the sequence is close to 100 %, which permits the calculations needed for the corresponding APCA$_{2\times2}$ for the vectors from the group r to be reduced N^2 times. For this, it is enough the condition for approximately zero value of the difference $\Delta\theta_t(r)$, defined by Eq. (30), to be satisfied.

The direct SR-APCA$_{2\times2}$ for the vectors from the group r, which have zero mean vector, is defined by the relation:

$$\vec{\mathbf{Y}}_{s,t}(r) = \mathbf{\Phi}_t(r)\vec{\mathbf{X}}_{s,t}(r), \tag{34}$$

where

$$\mathbf{\Phi}_t(r) = \mathbf{\Phi}_{t-1}(r) + \Delta\mathbf{\Phi}_t(r), \tag{35}$$

Taking into account Eqs. (31) and (32), is obtained:

$$\mathbf{\Phi}_t(r) = \begin{bmatrix} \cos[\theta_{t-1}(r) + \Delta\theta_t(r)] & \sin[\theta_{t-1}(r) + \Delta\theta_t(r)] \\ -\{\sin[\theta_{t-1}(r) + \Delta\theta_t(r)]\} & \cos[\theta_{t-1}(r) + \Delta\theta_t(r)] \end{bmatrix} \tag{36}$$

Let us assume:

$$\alpha_{t-1,r} = \frac{2k_{3,t}(r)}{k_{1,t}(r) - k_{2,t}(r) + \sqrt{[k_{1,t}(r) - k_{2,t}(r)]^2 + 4k_{3,t}^2(r)}}, \tag{37}$$

$$k_{1,t}(r) = \overline{x_2^2(r)},\ \ k_{2,t}(r) = \overline{x_3^2(r)},$$
$$k_{3,t}(r) = \overline{x_2(r)x_3(r)}. \tag{38}$$

and

$$\beta_{t,r} = \frac{2\Delta k_{3,t}(r)}{\Delta k_{1,t}(r) - \Delta k_{2,t}(r) + \sqrt{[\Delta k_{1,t}(r) - \Delta k_{2,t}(r)]^2 + 4\Delta k_{3,t}^2(r)}}, \tag{39}$$

$$\Delta k_{1,t}(r) = \overline{x_2^2(r)} - \overline{x_1^2(r)},\ \ \Delta k_{2,t}(r) = \overline{x_3^2(1)} - \overline{x_2^2(1)},$$
$$\Delta k_{3,t}(r) = \overline{x_2(1)[x_3(1) - x_1(1)]}. \tag{40}$$

Then

$$\theta_{t-1}(r) = \mathrm{arctg}\left(\alpha_{t-1,r}\right),\ \ \Delta\theta_t(r) = \mathrm{arctg}\left(\beta_{t,r}\right), \tag{41}$$

$$\theta_t(r) = \theta_{t-1}(r) + \Delta\theta_t(r) = \mathrm{arctg}\left(\alpha_{t-1,r}\right) +$$
$$+ \mathrm{arctg}\left(\beta_{t,r}\right) = \mathrm{arctg}\frac{\alpha_{t-1,r} + \beta_{t,r}}{1 - \alpha_{t-1,r}\beta_{t,r}} \text{ for } \alpha_{t-1,r}\beta_{t,r} < 1. \tag{42}$$

Hence,

$$\cos[\theta_{t-1}(r) + \Delta\theta_t(r)] = \cos(\mathrm{arctg}\frac{\alpha_{t-1,r} + \beta_{t,r}}{1 - \alpha_{t-1,r}\beta_{t,r}}) =$$
$$= \frac{1 - \alpha_{t-1,r}\beta_{t,r}}{\sqrt{1 + \alpha_{t-1,r}^2 + \beta_{t,r}^2 + \alpha_{t-1,r}^2\beta_{t,r}^2}}, \tag{43}$$

$$\sin[\theta_{t-1}(r) + \Delta\theta_t(r)] = \sin(\mathrm{arctg}\frac{\alpha_{t-1,r} + \beta_{t,r}}{1 - \alpha_{t-1,r}\beta_{t,r}}) =$$
$$= \frac{\alpha_{t-1,r} + \beta_{t,r}}{\sqrt{1 + \alpha_{t-1,r}^2 + \beta_{t,r}^2 + \alpha_{t-1,r}^2\beta_{t,r}^2}}. \tag{44}$$

As a result, for the matrix for the SR-APCA$_{2\times2}$ at the time moment (t) we get:

$$\mathbf{\Phi}_t(r) = \frac{1 - \alpha_{t-1,r}\beta_{t,r}}{\sqrt{1 + \alpha_{t-1,r}^2 + \beta_{t,r}^2 + \alpha_{t-1,r}^2\beta_{t,r}^2}} \begin{bmatrix} 1 & \dfrac{\alpha_{t-1,r} + \beta_{t,r}}{1 - \alpha_{t-1,r}\beta_{t,r}} \\ -\dfrac{\alpha_{t-1,r} + \beta_{t,r}}{1 - \alpha_{t-1,r}\beta_{t,r}} & 1 \end{bmatrix}. \tag{45}$$

If $\Delta\theta_t(r) \approx 0$, then $\beta_{t,r} \approx 0$. In this case, the matrix for SR-APCA$_{2\times2}$ is simpler:

$$\mathbf{\Phi}_t(r) = \mathbf{\Phi}_{t-1}(r) = \frac{1}{\sqrt{1 + \alpha_{t-1,r}^2}} \begin{bmatrix} 1 & \alpha_{t-1,r} \\ -\alpha_{t-1,r} & 1 \end{bmatrix}. \tag{46}$$

where:

$$\alpha_{t-1,r} = \frac{2\overline{x_2(r)x_3(r)}}{[\overline{x_2^2(r)} - \overline{x_3^2(r)}] + \sqrt{[\overline{x_2^2(r)} - \overline{x_3^2(r)}]^2 + 4\overline{x_2(r)x_3(r)}^2}}. \tag{47}$$

In case, that for the components of vectors $\vec{\mathbf{X}}_{t,s}(r) = [x_{2,s}(r),\ x_{3,s}(r)]^T$ for s =1, 2,.., N^2 is satisfied the condition:

$$\overline{x_2^2(r)} - \overline{x_3^2(r)} = 0, \tag{48}$$

and if the difference between the averaged squares of the pixels in two consecutive images in the group P is equal to zero, then from Eq. (47) is got $\alpha_{t-1,r} = 1$. In this case, the matrix for SR-APCA$_{2\times2}$ at the time moments (t-1) and (t) becomes:

$$\mathbf{\Phi}_{t-1}(r) = \mathbf{\Phi}_t(r) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \tag{49}$$

In this particular case, it turns into a Hadamard matrix of size 2×2 [4]. As a result of the execution of the SR-APCA$_{2\times2}$, at the time moments (t-1) and (t) the coordinate system of vectors $\vec{\mathbf{X}}_{t,s}(r)$ rotates on same angle, $\theta_{t-1}(r) = \theta_t(r) = \pi/4$, for r=1,2,..,R.

2.5 Algorithm for Sliding Recursive HAPCA

On the basis of the relations given above, here is generated the global algorithm for recursive calculation of the Sliding Recursive HAPCA (SR-HAPCA) of L levels, when $L=\log_2 P$. As an input data is used a sequence of correlated halftone images of size N×N. In particular, this could be a video sequence, or a group of multispectral images. When multispectral images are concerned, the time sequence is replaced by the band sequence, because their mutual correlation is highly related to the frequency band. The selection of the initial band depends on the correlation between participating images - the image with highest correlation to all others, is treated as the leading (initial) one.

The basic steps of the SR-HAPCA algorithm follow below:

Step 1. Initialization: at the initial moment/band t=0, the HAPCA algorithm is executed for the first P images (sections of the 3D tensor), represented by N^2 vectors of P components each, when $P=2^L$ (L - the number of hierarchical HAPCA levels). In the frame of the sliding window, which envelopes these images, they are divided into R=P/2 couples. In the consecutive HAPCA levels each couple is transformed by using $APCA_{2\times2}$ with transform matrix $\mathbf{\Phi}_0(r)$, as defined in Eq. (46). The images, obtained after the transform, are rearranged into new groups with similar energy, and then - divided into couples again. For each couple is once more calculated the $APCA_{2\times2}$, etc., until the last decomposition level of HAPCA is calculated;

Step 2. At the moment t=t+1, the algorithm HAPCA is executed for the next P images from the sections of the 3D tensor, framed by the sliding window, which moved meanwhile one position ahead. Besides, the first image from the group, processed at the moment t=0, is already out of the current window. Instead, new image with sequential number (P+1) is included in the frame;

Step 3. In the first hierarchical level of SR-HAPCA, for each group of vectors r=1,2,..,R (respectively, a couple of images) is checked the condition:

$$\Delta\theta_t(r) = \mathrm{arctg}\,\beta_{t,r} \approx 0, \qquad (50)$$

which is satisfied for $\beta_{t,r} \approx 0$. The calculation of Eq. (50) is easier, if the relation below is used:

$$\Delta k_{3,t}(r) = \overline{x_2(r)[x_3(r) - x_1(r)]} \approx 0. \qquad (51)$$

If the threshold δ<<1 is chosen, Eq. (51) could be represented as follows:

$$|\overline{x_2(r)[x_3(r) - x_1(r)]}| \leq \delta \quad \text{for} \quad r=1,2,..,R. \qquad (52)$$

In case, that the condition from Eq. (52) is not satisfied, for the corresponding group (r) is used SR-$APCA_{2\times2}$ with transform matrix $\mathbf{\Phi}_t(r)$, as defined by Eq. (45). For the remaining couples the transformed images are defined by the values, calculated in the preceding position of the sliding window at the time moment (t-1). These values are stored in a special buffer memory;

Step 4. The transformed groups of vectors are calculated, and their corresponding images are rearranged into new groups in accordance to their energy. After that they are sequentially arranged in couples again;

Step 5. In the next (second) hierarchical level of SR-HAPCA, the operations performed for the groups of two-component vectors are executed in a way, similar to that in the first level;

Step 6. At the highest level (L) of SR-HAPCA and after the last rearrangement, are calculated the images which are the main components of the input images (tensor sections), enveloped by the window at the moment (t);

Step 7. The processing continues with the next images. For this, the processing goes back to Step 2 and then steps 3-7 are executed, until all input images from the sequence are processed. As a result of the L-level HAPCA are obtained P mutually uncorrelated eigen images. The first one is the principle component, whose energy is maximum, compared to that of the next output components

As an illustration, on Fig. 3 is shown the three-level algorithm SR-HAPCA (L=3), used to transform a sequence of images (tensor sections) with fixed rearrangement in a window of length P=8. In each hierarchical level the transformed images, which correspond to the first (highest-energy) component of SR-$APCA_{2\times2}$, are coloured in yellow, and the images, which correspond to the second (lower-energy) component - in blue. The output images are the principle components, arranged in accordance to the lessening of their energies (the first main component of maximum energy E_1 is coloured in orange, and the last E_8 - in dark blue). In each hierarchical level the numbers of the components got after the transform, are retained after the rearrangement. In case, that in some hierarchical level for a given couple of images r and at the time moment t is got $|\Delta k_{3,t}(r)| \leq \delta$, then SR-$APCA_{2\times2}$ is not calculated. In the next level, the transformed images for these couples are calculated recursively using the already calculated data in the preceding discrete moment, (t-1). As a result, significant reduction of the needed computations is achieved, compared to the full recursive calculation of the input image sequence, as shown on Fig. 3. Besides, the used memory should be enlarged, because all results calculated for the SR-APCA$_{2\times2}$ at the moment (t-1) for all decomposition levels should be stored there.

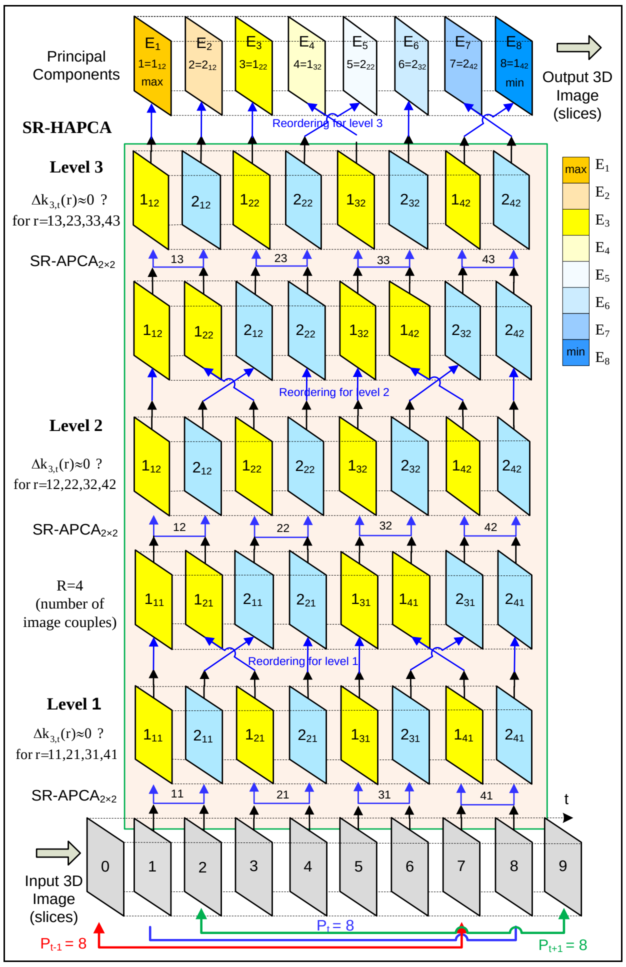


Fig. 3. The SR-HAPCA algorithm for full recursive transform of the image sequence of 8 images

Here should be also noted, that if the calculation SR-APCA$_{2\times2}$ for only one couple of images is missed, this results in N^2 times reduction of the number of vectors transforms in the corresponding group. In case, that the missed transform is in the first level, the transforms of the images in the next levels, which are related to the missed couple, will disappear also. In this way, the offered algorithm SR-HAPCA for a sequence of input images (tensor sections) is adapted in respect of their contents (mutual correlation), in result of which the needed computational operations are reduced even more, compared to these of the basic HAPCA algorithm.

3 Results

3.1 Evaluation of Computational Complexity

In this work, the computational complexity is evaluated on the basis of the needed mathematical operations. The execution time is not compared, because for this the evaluated algorithms should be represented in similar way, run on same computer, etc. Such comparison is not quite accurate, if all conditions are not same.

The basic algorithm used for the comparison, is HAPCA. Its computational complexity [17] can be calculated and compared with this of the regular PCA transform for P×P covariance matrix i.e. for $P=2^L$ input images, $L=\log_2 P$ hierarchical levels and R=P/2 groups of APCA$_{2\times2}$. For the classic PCA, L=R=1 because there are no hierarchical levels or sub-groups. In accordance with [18], both algorithms are compared regarding the number of operations O (additions and multiplications), needed for the calculation of the following components:

- Covariance matrices $\mathbf{K}_C$ of size P×P for the classic PCA algorithm and size 2×2 for the APCA:

$$O_{cov}(P)=(1/2)P(P+1)[P(P-1)+2(P+2)], \quad (53)$$

for the classic PCA

$$O_{cov2}(P)=30, \text{ for APCA}_{2\times2} \text{ and P=2.} \quad (54)$$

- Calculation of the eigen values of the corresponding $\mathbf{K}_C$ covariance matrix when the QR decomposition and the Householder transform of (P-1) steps are used for the classic PCA [16]:

$$O_{val}(P)=(P-1)(\frac{4}{3}P^2+\frac{17}{6}P+7) \text{ for the classic PCA} \quad (55)$$

$$O_{val2}(P)=16 \text{ for APCA}_{2\times2} \text{ and P=2.} \quad (56)$$

- Calculation of the eigen vectors of the covariance matrix $\mathbf{K}_C$ in case that iterative algorithm of four iterations is used for the classic PCA [16]:

$$O_{vec}(P)=P[2P(4P+5)-1] \text{ for the classic PCA} \quad (57)$$

$$O_{vec2}(P)=72 \text{ for APCA}_{2\times2} \text{ and P=2.} \quad (58)$$

- Transformations of eigen images, each of $S=N^2$ pixels:

$$O_{img}(P,S)=SP(2P-1) \text{ for the classic PCA} \quad (59)$$

$$O_{img}(2,S)=6S \text{ for APCA}_{2\times2} \text{ and P=2.} \quad (60)$$

- The total number of operations needed for the execution of both algorithms (the classic PCA and the Hierarchical PCA, based on APCA$_{2\times2}$) is correspondingly:

$$O(P,S)=(1/2)P(P+1)[P(P-1)+2(P+2)]+(P-1)(\frac{4}{3}P^2+\frac{17}{6}P+7)+$$
$$+P[2P(4P+5)-1]+SP(2P-1) \text{ - for the classic PCA.} \quad (61)$$

$$O_2(S)=LR(118+6S) \text{ - for APCA}_{2\times2} \text{ and P=2.} \quad (62)$$

Having got the total number of operations, we can compare the computational complexity of both: the classic PCA and the proposed algorithm. The reduction of the number of needed operations can be represented by the parameter η:

$$\eta(P,S)=O(P,S)/O_2(P,S)=LP(118+6S)^{-1}\times \left[P(P+1)(P^2+P+4)+(1/3)(P-1)(8P^2+17P+42)+4P(4P^2-4P-1)\right] \quad (63)$$

Since we work with images whose number of pixels vary, then we can evaluate the computational complexity on the basis of the parameter $\eta_{S\to\infty}(P,S)$ which covers all cases and gives minimum computational reduction:

$$\lim \eta_{S\to\infty}(P,S) \to (2P-1)/3L \approx 2P/3\log_2 P. \quad (64)$$

Hence, the reduction of the computational complexity of HAPCA compared to PCA, is appropximately $2P/3\log_2 P$ (for example, for P=8 the reduction is 1.77 times). As it was mentioned above, in case, that at given moment and in given hierarchical level for a selected couple of images is satisfied the condition from Eq. (52), for it is calculated the SR-APCA$_{2\times2}$. In the next hierarchical level the transformed images for this couple are defined recursively, using the images from the preceding moment (t-1). This results in additional reduction of the needed computational operations of SR-HAPCA, compared to HAPCA.

3.2 Simulation of Algorithm SR-HAPCA

For illustration of the algorithm SR-HAPCA, on Fig. 4*a* is shown a sequence of computer tomography (CT) images, framed by a sliding window of size P=8 at the time moment (t). Each CT image in the group is of size of 512×512 pixels and 8 bpp (bits per pixel).

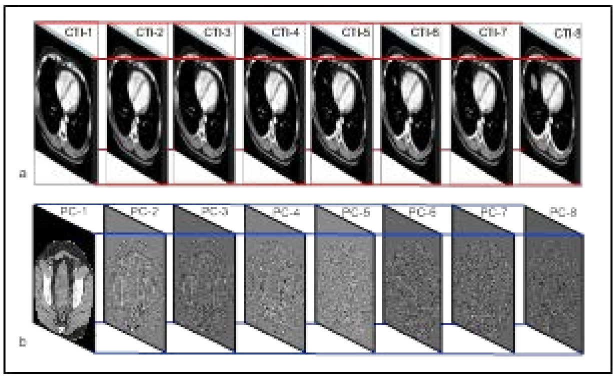


Fig. 4. A sequence of eight (P=8) computer tomography images (CTI) and the corresponding Principal Components (PC) got after the execution of SR-HAPCA.

The corresponding principal components (PC$_p$), calculated by using the algorithm SR-HAPCA, are shown on Fig. 4*b*. As it could be easily seen, the main part of the energy in the group is concentrated on the first and second decomposition components. In this case, components 3-8 could be neglected without influencing the visual quality of the restored group of CT images, got after the execution of the inverse SR-HAPCA. The simulation results for the proposed algorithm for different kinds of 3D images (CTI, MSI, MVI, etc.) confirmed its efficiency in various application areas. The energy of the matrix **PC**$_p$ of size K×J and elements $pc_{k,j,p}$, which represents the principal component p, is calculated in accordance to the relation below:

$$E_p=\sum_{k=1}^{K}\sum_{j=1}^{J}(pc_{k,j,p})^2 \text{ for } p=1,2,.., P. \quad (65)$$

On Fig. 5 is shown the energy distribution (E_p) of the principal components (PC$_p$), which represent the decomposition of the group of CTI, got after the execution of the 3-level SR-HAPCA. The energy concentration on the first and second components confirms the theoretical analysis, given above. This is a good basement for significant reduction of the information redundancy in the transformed group of CT images, which was one of the main objectives of the presented method.

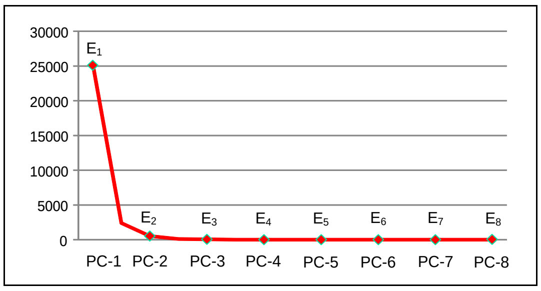


Fig. 5. Energy distribution for a group of eight CT images, after the execution of 3-level SR-HAPCA

The algorithm SR-HAPCA could be used for efficient processing of correlated image sequences in the following application areas:

- Optimum filtration of 3D tensor images in the frame of the sliding temporal window;
- Compression of 3D tensor images with minimum mean square error, got in result of the threshold truncation of the low-energy transformed images (principal components) in the window for each discrete moment t;
- Reduction of the features' space used for the classification of objects contained in the 3D tensor images, framed by the sliding window, at each discrete moment (t), etc.

The new method could be used not only for processing of 3D images, but also for decorrelation of multidimensional data in systems of various kind.

4 Conclusions

The presented new method SR-HAPCA retains all advantages of HAPCA towards PCA (the reduced number of calculations and the ability for parallel implementation). Besides, because of the relative simplicity of SR-HAPCA, it is possible to achieve real-time calculation of the tensor images in the frame of the sliding window. The output sequence contains the principle components of the input sequence (tensor slices) framed by the selected window, at each discrete time moment. The efficiency of SR-HAPCA could be enhanced by:

- Replacement of the fixed rearrangement of the couples of components obtained after APCA (in each hierarchical level), by adaptive rearrangement in accordance to the lessening of their energy;
- Selection of the best orientation (vertical, horizontal, or lateral) for the group of vectors, used for the 3D tensor representation depending on its creation (source);
- Depending on the application, it is possible to calculate only the output principle components which have maximum energy, and to miss the calculation of the low-energy components. As a result, the number of needed calculations for the Truncated SR-HAPCA will be additionally reduced.

The method SR-HAPCA opens new promising opportunities for various real-time applications of the basic HAPCA method in the area of the 3D digital processing (filtration, compression, segmentation, etc.) of tensor images in the contemporary systems for objects analysis and recognition, video communications, process monitoring, etc.

References:

[1] T. Kolda, B. Bader, Tensor decompositions and applications, *SIAM Review*, Vol. 51, No 3, 2009, pp. 455-500.

[2] M. Torun, A. Akansu, An Efficient Method to Derive Explicit KLT Kernel for First-Order Discrete Autoregressive Process, *IEEE Transactions on Signal Processing*, Vol. 61, No 15, 2013, pp. 3944-3953.

[3] H. Abdi, L. Williams, Principal Component Analysis, *Wiley Interdisciplinary Reviews: Computational Statistics*, Vol. 2, Iss. 4, 2010, pp. 433-459.

[4] R. Gonzales, R. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, 2008.

[5] S. Orfanidis, SVD, PCA, KLT, CCA, and all that, Rutgers University Electrical & Computer Engineering Department, *Optimum Signal Processing*, 2007, pp. 1-77.

[6] I. Jolliffe, *Principal Component Analysis*, 2nd ed., Springer, NY, 2002.

[7] C. Maccone, A simple introduction to the KLT (Karhunen-Loève Transform), *Deep Space Flight and Communications*, C. Maccone (Ed.), Springer, 2009, pp. 151-179.

[8] M. Hanafi, A. Kohler, E. Qannari, Shedding new light on Hierarchical Principal Component Analysis, *Journal of Chemometrics*, Vol. 24, Iss. 11-12, 2010, pp. 703-709.

[9] I. Blanes, J. Sagristà, M. Marcellin, J. Rapesta, Divide-and-conquer strategies for hyperspectral image processing, *IEEE Signal Processing Magazine*, Vol. 29, No 3, 2012, pp. 71-81.

[10] R. Kountchev, R. Kountcheva, Decorrelation of Multispectral Images, Based on Hierarchical Adaptive PCA, *Intern. Journal WSEAS Trans. on Signal Processing*, Iss. 3, No 9, July 2013, pp. 120-137.

[11] W. Li, H. Yue, S. Cervantes, S. Qin, Recursive PCA for adaptive process monitoring, *Journal of Process Control*, Vol.10, No 5, October 2000, pp. 471-486.

[12] D. Erdogmus, Y. Rao, H. Peddaneni, A. Hegde, J. Principe, Recursive Principal Components Analysis using Eigenvector Matrix Perturbation, *EURASIP Journal on Advances in Signal Processing*, Vol. 13, Dec. 2004:13, pp. 2034-2041.

[13] A. Amar, A. Leshem, M. Gastpar, Recursive Implementation of the Distributed Karhunen-Loève Transform, *IEEE Trans. on Signal Processing*, Vol. 58, No 10, 2010, pp. 5320-5330.

[14] Z. Wei, X. Liu, F. Li, S. Shang, X. Du, J. Wen, Matrix Sketching Over Sliding Windows, *Proc. of the Intern. Conf. on Management of Data (SIGMOD'16)*, June 26-July 01, San Francisco, USA, 2016, pp. 1465-1480.

[15] S. Zhou, N. Vinh, J. Bailey, Y. Jia, I. Davidson, Accelerating Online CP Decompositions for Higher Order Tensors, *Proc. of the 22nd ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining (KDD'16)*, August 13-17, San Francisco, USA, 2016, pp. 1375-1384.

[16] J. Faires, R. Burden, *Numerical Methods*, 4th ed, Brooks/Cole, Boston MA, USA, 2013.

[17] P. Ivanov, R. Kountchev, Hierarchical Principal Component Analysis based Transformation of Multispectral Images, *Intern. Journal of Reasoning-Based Intelligent Systems (IJRIS)*, Vol. 5, No 4, 2013, pp. 260-273.

[18] S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, NY, USA, 2009.