

Representing Images by Multiple Kronecker Product Sum

Hongxin ZHANG^{1,2}, Dongxu QI³, Chiew-Lan TAI², Hujun BAO¹

1. State Key Laboratory of CAD&CG, Zhejiang University, Hang Zhou, China

2. Department of Computer Science, Hong Kong University of Science & Technology, Kow loon, Hong Kong

3. Department of Computer Science, Macau University of Science & Technology, Macau

E-mail: zhanghx.taicl@cs.ust.hk, dxqi@must.edu.mo, bao@cad.zju.edu.cn

Abstract

Kronecker product is a powerful operation in matrix computations. This paper presents a technique called multiple Kronecker product sum (or shortly MKPS) approximation. We show that a matrix can be exactly represented or approximated, to within a specific tolerance, by the sum of multiple Kronecker products of lower dimension matrices. Since an image can be viewed as a matrix, MKPS approximation serves as an alternative image representation. We develop an image decomposition-reconstruction framework based on MKPS approximation. Our current results show that MKPS approximation is an attractive and flexible transform which offers an alternative way for understanding images.

Keywords: Kronecker product, matrix approximation, image representation.

1. Introduction

Image transforms are powerful ways for representing information in an image. Their motivation is to exploit properties possessed by the image transform that is not available in the image domain. Most commonly, image transform is done to facilitate image processing, and image compression^[11]. On the other hand, a grey scale image can be viewed as a matrix. Thus most image transforms can be interpreted into a set of matrix computations.

The Kronecker product of matrices^[4], also known as the *tensor product* and *direct matrix product*, plays a central role in mathematics. It also has many applications in engineering and theoretical physics, such as signal processing, statistical physics, quantum groups, and quantum computers. Recently, several researchers began to introduce Kronecker product technique into image processing and related areas. J. Kamm et. al.^{[7][6]} considered Kronecker product and SVD approximations in image restoration. N. P. Pitsianis^[9] applied Kronecker product in fast transform generation. Canta et. al.^[2] proposed Kronecker-product gain-shape vector quantization for multi-spectral and hyper-spectral image coding. However most of them only applied Kronecker product or the sum of Kronecker products as a computation tool for transform matrix computing. Our research work is to create a generic framework for employing Kronecker product and its extension

MKPS defined in Section 2 for image representation and processing.

Much research has been done on the topics of image representations and transforms. In the early works, conveying from the ideas of signal processing, researchers use the discrete Fourier transform (DFT) technique to transform the image space information into the spectrum space and then process the image signal by applying special filters, e.g., noise removal, smoothing and enhancement. Later, with the increased requirement for image compression and transmission, researchers developed 2D discrete cosine transform (DCT) with the aim of finding data redundancy. It is interesting that most existing transforms can be explained through a Kronecker product; both DFT and DCT fall in the category of orthogonal and unitary transformations.

The Karhunen Loeve transform (KLT) is another orthogonal one employed in information reduction. It is derived from principal component analysis (PCA), which tries to decompose the image vector into more important components and less important ones by singular value decomposition (SVD). We will see that SVD can be used in a better and more intuitive way based on Kronecker product for analyzing image signal.

The pyramid structure of image has long been employed in image multi-resolution representation and analysis. This structure provides a much efficient manner to hierarchical access of data. For instance, the mip-mapping technique used in texture mapping employs this idea to achieve a good LOD effect^[13]. The wavelet transforms are well-known multi-resolution approaches. It has been successfully applied to image processing and compression. Recently it is designated as the kernel algorithm for Jpeg2000, which is the newest standard for image compression^[10].

Textures, which always have some implicit structure, is a special class of images. Due to the increasing requirement in rendering and image synthesis applications, texture analysis and synthesis^{[5][1][12]} has been a hot research area for several years. Most works are performed in the image space. A natural question is whether there is other way to view the image structure. This may help researchers understand textures in a new way. Our experiments show that our approach is much suitable for well-structured images, because the Kronecker product can capture potential self similarity and

special structures.

In this paper, we propose an image representation based on our MKPS approximation. It connects several important techniques in image processing and compression including orthogonal transforms, SVD, and multi-resolution analysis. It also gives new cues and perspective for understanding image information.

The paper is organized as follows. We will first give the necessary definitions and several properties of Kronecker product in section 2. In section 3, we explore the theoretical problem of approximating matrix by MKPS. A pair of decomposition-reconstruction algorithms is given in sub-section 3.4. In Section 4, we mainly discuss the experimental results. Finally, we draw the conclusions and outline our future research directions in Section 5.

2. The Kronecker Product

The *Kronecker product* [4] of two matrices B and C of $m_1 \times n_1$ and $m_2 \times n_2$ respectively, is a $m_1 m_2 \times n_1 n_2$ matrix (say A) defined by

$$A = B \otimes C = \begin{pmatrix} b_{11}C & b_{12}C & \cdots & b_{1n_1}C \\ b_{21}C & b_{22}C & \cdots & b_{2n_1}C \\ \vdots & \vdots & \ddots & \vdots \\ b_{m_1 1}C & b_{m_1 2}C & \cdots & b_{m_1 n_1}C \end{pmatrix}$$

Here B and C are called the left and right factor matrix of A respectively. Furthermore, we can recursively define

$$\begin{aligned} \otimes_{i=1}^k A_i &= (\otimes_{i=1}^{k-1} A_i) \otimes A_k = \cdots \\ &= A_1 \otimes A_2 \otimes \cdots \otimes A_k \end{aligned}$$

where $A_i (i=1, 2, \dots, k)$ are matrices of size $m_i \times n_i$. We call this a *multiple Kronecker product*. And our goal is to represent or approximate a given matrix A by the sum of several multi Kronecker products, i.e.,

$$A \cong \sum_i A_{i1} \otimes A_{i2} \otimes \cdots \otimes A_{ik}$$

We call it multiple Kronecker product sum approximation of matrix (or MKPS approximation). The Kronecker product matrix approximation had been analyzed and used by several researchers^{[8][6]}, but till now no work is known on MKPS approximation.

2.1. MKPS representation

The easiest way to represent a matrix as a MKPS is to decompose the matrix into some simplest matrices. Without losing generality, let I_{ij} be a set of 'basis' matrices of size $2^l \times 2^l$, whose element at row i and column j is equal to 1, otherwise it is 0. Each basis matrix can be easily represented by a multi-Kronecker-product, that is

$$I_{ij} = \otimes_{k=1}^l J_{\alpha_k \beta_k},$$

Here $(\alpha_1 \alpha_2 \cdots \alpha_l)$ and $(\beta_1 \beta_2 \cdots \beta_l)$ are the binary representation of i and j respectively, and the $J_{\alpha \beta}$ are

$$\begin{aligned} J_{00} &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, & J_{10} &= \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \\ J_{01} &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, & J_{11} &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

Thus a matrix $A = (a_{ij})$ of size $2^l \times 2^l$ can be represented by

$$A = \sum_{i,j} a_{i,j} (\otimes_{k=1}^l J_{\alpha_k \beta_k}).$$

So we know that it is possible to exactly represent every matrix or image as a MKPS. However, this representation is not efficient. For example, the strong structured and fractal images in Figure 1(a) and (b) can be defined by

$$\left[\otimes_{i=1}^3 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \right] \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \left[\otimes_{i=1}^4 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \right]$$

and

$$\otimes_{i=1}^8 (J_{00} + J_{01} + J_{11})$$

respectively. Then the question is how good a MKPS representation will be. Equivalently, can we use relatively compact representation based on MKPS to describe the information in the image? Moreover, exact representation is not necessary in many applications. This motivates us to consider the approximation problem.

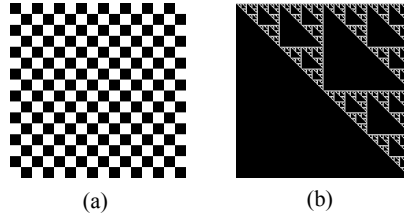


Figure 1: Synthesized image examples by MKPS. (a) The check-board. (b) The fractal pattern known as the Sierpinski triangle.

3. MKPS Approximation

In this section, we will try to answer the central question of this paper: whether it is possible to approximate a matrix (indeed an image) by decomposing it into small simpler matrices (or image elements) in an efficient way. Like most mathematical problems, we will solve this in several steps.

3.1. Single Kronecker product approximation

Firstly, we consider the problem of approximating with single Kronecker product. Given a matrix A of dimension $m \times n$ with $m = m_1 \times m_2$ and $n = n_1 \times n_2$, we want to find the matrices B and C of size $m_1 \times n_1$ and $m_2 \times n_2$ respectively that minimize the following error

$$f_A(B, C) = \|A - B \otimes C\|_F.$$

In other words, we want to find the best single decomposition of image with respect to the given measure.

Note that $f_A(B, C) = f_A(\alpha B, \alpha^{-1}C)$ for all $\alpha \neq 0$, so the solution of the problem is not unique. Then we can define the equivalent class for the matrix pair, i.e., $(B, C) \approx (\alpha B, \alpha^{-1}C)$. For conciseness, we define the *vec* operation of a given matrix $X \in \mathfrak{R}^{m \times n}$

$$\text{vec}(X) = \begin{pmatrix} X(1:m, 1) \\ X(1:m, 2) \\ \vdots \\ X(1:m, n) \end{pmatrix} \in \mathfrak{R}^{m \times n}$$

and re-shape operation

$$\text{res}(A) = (A_1, A_2, \dots, A_{n_1})^T,$$

$$A_j = (\text{vec}(A_{1,j}^T), \text{vec}(A_{2,j}^T), \dots, \text{vec}(A_{m_1,j}^T))^T,$$

where

$$A_{ij} = A((i-1)m_2 + 1 : im_2, (j-1)n_2 + 1 : in_2)$$

are blocks of A with dimension $m_2 \times n_2$. It follows that

$$f_A(B, C) = \|\text{res}(A) - \text{vec}(B)\text{vec}(C)^T\|_F$$

Then we obtain the following useful theorem^{[8] [3]}.

Theorem 3.1 Assume that $A \in R^{m \times n}$ with $m = m_1 \times m_2$ and $n = n_1 \times n_2$. If $\tilde{A} = \text{res}(A)$ has the SVD

$$U^T \tilde{A} V = S = \text{diag}(\sigma_1, \dots, \sigma_n),$$

where $(\sigma_1, \dots, \sigma_n)$ is the singular value sequence in descending order, and $\text{diag}(\cdot)$ means to generate the diagonal matrix with this sequence. Let σ_1 be the largest singular value, and U_1, V_1 are the corresponding singular vectors, then the matrix $B \in R^{m_1 \times n_1}$ and $C \in R^{m_2 \times n_2}$ defined by

$$\text{vec}(B) = \sqrt{\sigma_1} U_1, \quad \text{vec}(C) = \sqrt{\sigma_1} V_1$$

minimize $f_A(B, C)$.

This theorem builds up the relationship between Kronecker product decomposition and the SVD technique, which has been used in image processing and information reduction for many years. The advantage here is that the matrix is first re-constructed by the reshape operation which takes a

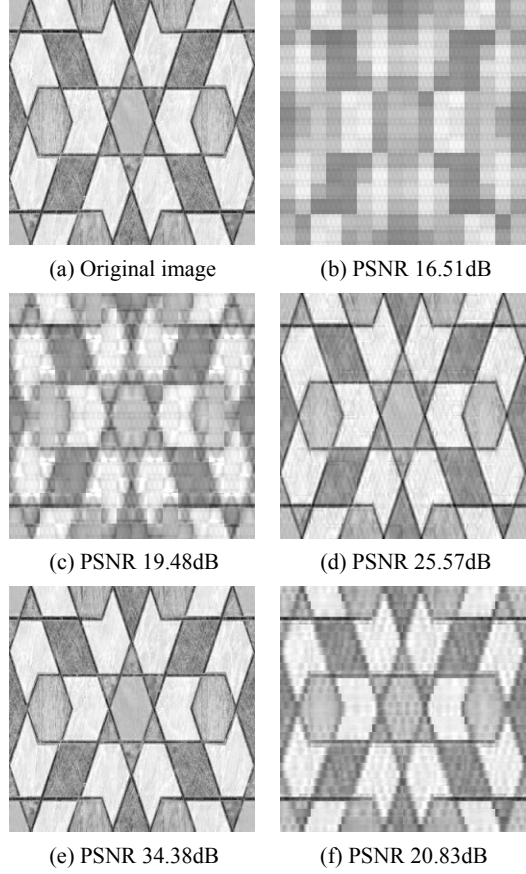


Figure 2: Approximating image by Kronecker product sum. (a) is the original image in size 256×256 . (b), (c), (d) and (e) are the 1st, 4th, 16th and 64th level reconstruction results respectively with the left-factor matrix of size 16×16 . (f) is the 8th level reconstruction with the left-factor matrix of size 8×32 .

small block in the original matrix into a row of the new one. Since each block is a cluster of sampling, the approximation problem becomes that of finding the most important pattern in these blocks. Then matrix B turns out to be a good down-sampling version of A , and C measures the similarity among these blocks. Figure 2(b) is a Kronecker product approximation example.

3.2. Kronecker product sum approximation

In a very similar way, we can find the subsequence components. This gives the following theorem^[8].

Theorem 3.2 Same as the assumption as Theorem 3.1. And let U_i, V_i be the corresponding singular vectors of σ_i , then the matrix $B_i \in R^{m_1 \times n_1}$ and $C_i \in R^{m_2 \times n_2}$ defined by $\text{vec}(B_i) = \sqrt{\sigma_i} U_i$ and

$$\text{rec}(C_i) = \sqrt{\sigma_i} V_i \quad \text{minimize} \left\| A - \sum_{i=1}^k B_i \otimes C_i \right\|_F.$$

Then it is obvious that A can be exactly represented by Kronecker product sum when the number k is equal to the rank of \tilde{A} , since the approximation will theoretically have no error.

Corollary 3.3 Let A , B_i and C_i be defined as

Theorem 3.2. If $\text{rank}(\tilde{A}) = r$, then

$$A \equiv \sum_{i=1}^k B_i \otimes C_i.$$

By Theorem 3.2 and Corollary 3.3, we can easily build a progressive transmission procedure based on the Kronecker product sum approximation. Since we can choose the size of B_i and C_i freely, the procedure framework exhibits much flexibility. For example, if the original image is in size 256×256 , it is possible to obtain at most 256 levels by selecting B_i and C_i to be the same size 16×16 , or to obtain only 8 levels by selecting B_i to be 128×64 and C_i to be 2×4 . Figure 2 gives an example of Kronecker product sum approximation.

The procedure error can be controlled by some features in the image, specifically, by the singular value sequence of the re-shaped image matrix. From our experiments, typical real-scene images always have singular value sequences that shape like power function curves if the values are in increasing order. In the sequence, the largest singular value is almost 10^4 times of the first several smallest values which are act as the redundant information.

3.3. Adaptive MKPS approximation

Our MKPS approximation procedure is as follows. Given a matrix A with size $2^l \times 2^l$, we denote A as $A^{(0)}$ which means it is in decomposition layer 0. By Corollary 3.3, there is a full decomposition

$$A^{(0)} = \sum_{i_1=1}^4 B_{i_1}^{(1)} \otimes A_{i_1}^{(1)} \quad (1)$$

with $B_{i_1}^{(1)}$ of size 2×2 . By Equation 7, we can set

$$\|B_{i_1}^{(1)}\|_F = 1. \quad \text{It follows that } \|A^{(0)}\|_F^2 = \left\| \sum_{i_1=1}^4 A_{i_1}^{(1)} \right\|_F^2.$$

Applying the same procedure for each $A_{i_1 \dots i_k}^{(k)}$ in the decomposition layer k , we have the following iteration equation

$$A_{i_1 i_2 \dots i_k}^{(k)} = \sum_{i_{k+1}=1}^4 B_{i_1 i_2 \dots i_{k+1}}^{(k+1)} \otimes A_{i_1 i_2 \dots i_{k+1}}^{(k+1)},$$

and the norm estimation

$$\|A_{i_1 i_2 \dots i_k}^{(k)}\|_F^2 = \sum_{i_{k+1}=1}^4 \|A_{i_1 i_2 \dots i_{k+1}}^{(k+1)}\|_F^2. \quad (2)$$

Finally in the layer $l-1$, all the decomposed matrices $A_{i_1 i_2 \dots i_{l-1}}^{(l-1)}$ have the same 2×2 size. Combining all the layers, we conclude that A can be decomposed into

$$\sum_{i_1=1}^4 \dots \sum_{i_l=1}^4 B_{i_1}^{(1)} \otimes \dots \otimes B_{i_{l-1}}^{(l-1)} \otimes A_{i_1 i_2 \dots i_l}^{(l)}$$

and the norm relationship equation is

$$\|A\|_F^2 = \sum_{i_1=1}^4 \dots \sum_{i_l=1}^4 \|A_{i_1 i_2 \dots i_l}^{(l)}\|_F^2.$$

It is obvious that the whole procedure is equivalent to building a complete quad-tree. Every leaf in this tree is a multiple Kronecker product. The image can be represented by this tree. At first look, the size of the tree is huge. However, most-right factor matrices in layer $l-1$, i.e. $A_{i_1 i_2 \dots i_{l-1}}^{(l-1)}$ have extremely tiny norm, which means they will contribute almost nothing to the whole representation. Thus we obtain our MKPS approximation by ignoring these items. Furthermore, by Equation (2), the approximation procedure can be implemented with adaptive strategy, i.e., the decomposed branch can be terminated while less than a tolerance. Thus the total size of MKPS approximations dramatically decreased.

Another observation is that the algorithm performs in a manner similar to the multi-resolution analysis framework. On each layer, the procedure subdivides the matrices into 4 parts, which are sorted by their importance according to the singular values. If we fix the left matrices to be

$$\begin{cases} B_1 &= (J_{00} + J_{01} + J_{10} + J_{11}) / 2; \\ B_2 &= (J_{00} + J_{01} - J_{10} - J_{11}) / 2; \\ B_3 &= (J_{00} - J_{01} + J_{10} - J_{11}) / 2; \\ B_4 &= (J_{00} - J_{01} - J_{10} + J_{11}) / 2; \end{cases}$$

in the 1-4 decomposition (refer to Equation (1)), then the whole procedure is equivalent to the Harr wavelet transform. But the advantage of our algorithm is that the singular values or the norm of right matrices explicitly indicate the error during the approximation procedure.

3.4. Fast MKPS approximation

The decomposition procedure in the last sub-section works well for small images. However, it performs less efficiently for large-scale images in practice. According to Golub and Van Loan's book^[3], the best algorithms for SVD computation of an $m \times n$ matrix take time that is proportional to is $O(km^2n + k'n^3)$ (k and k' are constants which are 4 and 22 for an algorithm called R-SVD). Since the

re-shaped matrix is thin and long in the MKPS decomposition, the runtime would be long. So we propose a more efficient solution to decompose an image into MKPS representations in this section. The main idea is to make the re-shaped matrix squarer. The brief steps of our algorithms are as follows.

Algorithm 3.4 Fast MKPS decomposition

1. Input a matrix A in size of $2^{2^l} \times 2^{2^l}$ and the maximal decomposition levels k in layer 1.
2. By Theorem 3.2, decompose A into two matrix sequences B_i and C_i ($i=1,2,\dots,k$) with size $2^{2^{(l-1)}} \times 2^{2^{(l-1)}}$
3. Merge the two sequences to obtain as layer 1 decomposition sequence.
4. For each matrix in the sequence perform Kronecker product sum decomposition.
5. Merge all the sequence together as layer l decomposition sequence.
6. Do the step 4 and 5, until layer 1 and all the matrices are of size 2×2 .

Algorithm 3.5 Fast MKPS reconstruction

1. Input layer 1 decomposition sequence, i.e., the 2×2 matrix sequence.
2. Use Kronecker product sum to create layer $l-1$ decomposition sequence, i.e., the 4×4 matrix sequence.
3. Use Kronecker product sum to create layer $l-i$ decomposition sequence, i.e., the $2^{2^i} \times 2^{2^i}$ matrix sequence.
4. Repeat step 3 until the sequence in layer 1, and output the reconstruction result matrix.

Note that the image size of $2^l \times 2^l$ is not a necessary constraint. One can modify our fast decomposition-reconstruction algorithms slightly to fit special requirements, for instance, by enlarging the image size. Another possible extension is to combine Algorithm 3.4 with the method described in the last subsection.

4. Results

We have implemented our algorithms in a hybrid environment using Microsoft VC++ and MATLAB 6.5 to take advantage of its powerful matrix computation ability. We implicitly represent MKPS approximating result into a tree structure, and retrieve the true MKPS for well-structured images from the tree structure.

The two algorithms described in Section 3.4 perform very efficiently for well-structured images. For instance, we obtain only one non-zero component to represent the check-board or the Sierpinski triangle (Figure 1) by Algorithm 3.4 in the first layer decomposition. Finally we extract only one

multi-Kronecker-product for both examples as desired.

Our Kronecker product-based decomposition can be viewed as an extension of basic matrix SVD. For a given image in size of $m \times n$, our decomposition method is equivalent to SVD when we choose $m \times 1$ as the size of the left-factor matrix in the first layer decomposition, since the re-shaped matrix will be the same as the original one. The advantage of the Kronecker product is that it produces flexible choices for sampling pattern and size. In our framework, we choose the squared factor matrices. Thus we can run the decomposition recursively.

We have applied our algorithm on different kinds of images, and use the well-known PSNR measure to estimate the quality of a reconstructed image compared with an original image. Figure 4 shows the reconstruction PSNR curves of several typical images. For comparing the results, all images are in 256×256 size. The horizontal axis represents the first 160 levels in layer 1. It indicates that the better structured the image is, the better the reconstruction result will be.

The singular values give an alternative viewpoint beside DFT. The image is transformed into a component sequence by MKPS approximation. If we only save the components related to the larger singular values, then we obtain smoothed and blur images compared with the original one (Figure 4(b)), and a potential information reduction procedure at the same time. On the other side, if we enhance the components related to smaller singular values then we can get sharper images, which appear much similar to pen drawing effects (Figure 4(c, d)).

As an experiment to test the robustness of our algorithm, we quantize the elements of the final decomposed matrices before reconstruction. It is amazing that the reconstruction result still seems acceptable. We used 32 degrees in Figure 5(c) and 16 in Figure 5(d). So the approach is promising for compression.

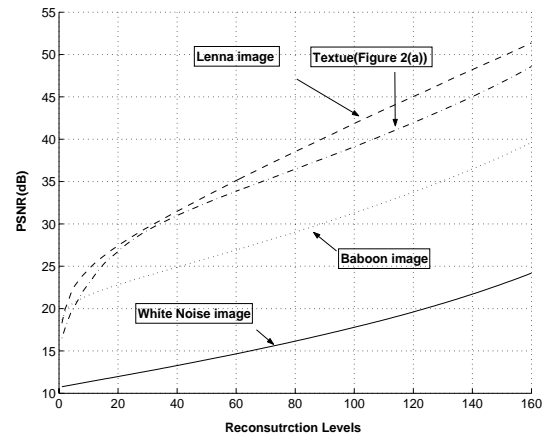


Figure 3: Comparison the reconstruction results.

5. Conclusion

We proposed an alternative image representation based on our novel MKPS approximation technique. A flexible progressive transmission procedure can be generated by this technique. With the technique, we have explored several image processing functions. We will further investigate the properties and develop practical applications based on the MKPS approximation.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 60133020) and the National Natural Science Foundation for Innovative Research Groups (No. 60021201).

References

- [1] J. S. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images", *Computer Graphics*, 31(Annual Conference Series), 1997, pp.361-368.
- [2] G. R. Canta and G. Poggi, "Kronecker-product gain-shape vector quantization for multi-spectral and hyper-spectral image coding", *IEEE Trans. on Image Processing*, 1998, 7(5), pp.668-678.
- [3] G. H. Golub and C. F. Van Loan, *Matrix computation*. The Johns Hopkins University Press, 1st edition, 1989.
- [4] A. Graham, *Kronecker Products and Matrix Calculus With Applications*, Halsted Press, John Wiley and Sons, NY, 1981.
- [5] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis", In *SIGGRAPH 1995 proceeding*, 1995, pp. 229-238.
- [6] J. Kamm, *Singular value decomposition-based methods for signal and image restoration*. Phd thesis, Southern Methodist University, Southern Methodist University, Dallas, TX, 1998.
- [7] J. Kamm and J. G. Nagy, "Optimal Kronecker product approximation of block Toeplitz matrices", *SIAM Journal on Matrix Analysis and Applications*, 2001, 22(1), pp.155-172.
- [8] C. F. Van Loan and N. Pitsianis, "Approximations with Kronecker products", (CTC92TR109), 1992.
- [9] N. P. Pitsianis, "*The Kronecker Product in Approximation and Fast Transform Generation*". Phd thesis, Cornell University, Ithaca, NY, 1997.
- [10] D. Santa-Cruz and T. Ebrahimi, "An analytical study of jpeg 2000 functionalities". In *Proc. of the International Conference on Image Processing (ICIP)*, 2000, vol 2, pp.49-52.
- [11] A. Watt and F.Policarpo, *The computer image*. Addison-wesley press, 1999.
- [12] L. Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization". In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pp.479-488. ACM Press, ACM SIGGRAPH, Addison Wesley Longman, 2000.

- [13] L. Williams, "Pyramidal parametrics", *Computer Graphics*, 1983, 17(3), pp.1-11.

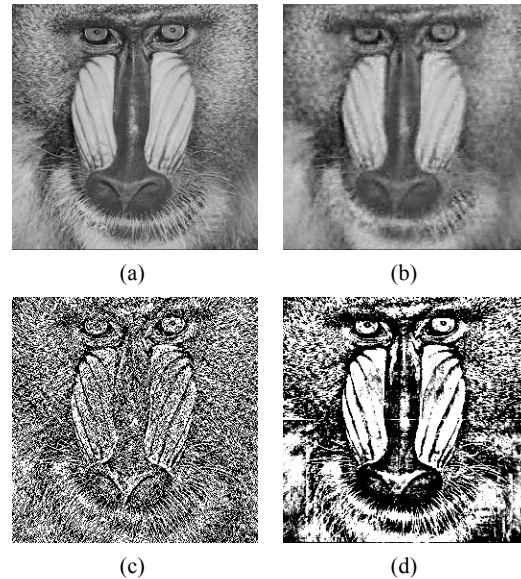


Figure 4: MKPS approximation examples: Baboon image (grey scale). (a) is the original image in size of 256×256 . (b) is the reconstruction result by algorithm 4.2 using only first 10 components in first layer. (c) and (d) are two examples of enhancing the 'less important' components.

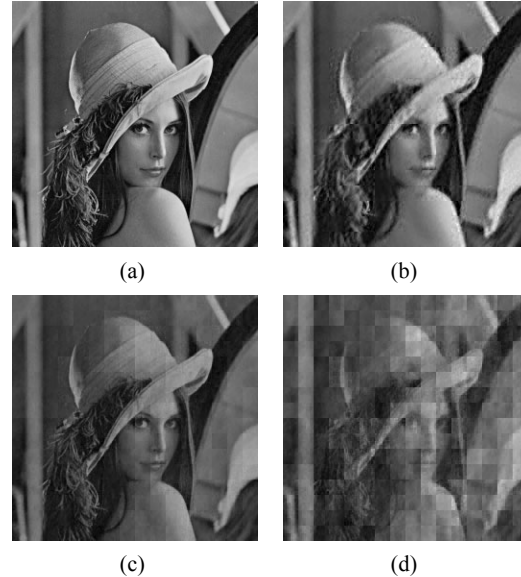


Figure 5: Decomposition and reconstruction example. (a) the original Lenna grey scale image in size of 256×256 . In (b) the reconstruction result only use first 20 components in the first layer. (c) and (d) are two reconstruction results that use 64 components in the first layer after quantized of 32 and 16 degrees respectively.