1-D AND 2-D TRANSFORMS FROM INTEGERS TO INTEGERS *

Jia Wang, Jun Sun and Songyu Yu

jiawang@cdtv.org.cn
Institute of Image Commun. & Inform. Processing,
Shanghai Jiao Tong Univ., Shanghai 200030, P.R.China

ABSTRACT

Substituting a real valued linear transform with an integer-to-integer mapping has become very important in lots of applications. This paper introduces a new kind of matrix decomposition method called lifting-like factorization, which leads to a theorem: Every 2^n-order real matrix with determinant norm 1 can be expressed as the product of one permutation matrix and at most three unit triangular matrices. Rounding error of this method is analyzed. Realization of 2-D integer transform is also studied and it is shown that a 2-D integer-to-integer transform cannot be realized by performing two 1-D integer transforms separately. Left and right permutation matrices are introduced to reduce rounding error and an application of this method to intDCT is discussed.

1. INTRODUCTION

Linear transform is playing a very important role in signal and image processing. Although the input of these transforms are usually integers, the output are rarely so. Constraining the output of an arbitrary transform to lattices is very important in many applications, such as lossless compression and hardware implementation. It is well known that some integer transform can be realized by factoring matrix into lifting steps. But the factorization is not unique and so far no systematic method has been proposed to find the optimal one. Furthermore, for image processing, the input signal is usually 2-D, while previous works mainly focused on of 1-D integer transforms.

Recently, multiple description coding (MDC) based on correlating transforms has attracted a lot of attentions [1][3]. In general, these linear transforms are real valued and with determinant one. A cascade structure proposed in [3] demand an optimal implementation of 2-D transforms from integers to integers. One may argue that 2-D integer transforms can be realized by performing two 1-D transforms successively. But we will show this is not optimal, which is quite different from float transforms.

*This work was supported in part by the NSF China under Grant 60202006

In literature, factor 2×2 matrix into lifting steps has been widely used. This kind of factorization is not unique, and is usually constraint to size of 2×2. In this work, we propose a new kind of factorization of 2^n-order matrix into at most three unit triangular matrices. The factorization can be done in a recursive manner, and may be considered as an extension of lifting method to higher order condition. So, it can be called as lifting-like factorization. In [8], similar but not the same theorem was obtained and the factorization method is totally different. In this paper, expectation of rounding error is analyzed. Left and right permutation matrices are introduced not only to guarantee the existence of factorization, but also to further reduce the rounding error. Finally, an application of the new factorization method to intDCT is discussed, and example of 4×4 DCT matrix is shown.

2. NEW FACTORIZATION AND MAIN THEOREM

The factorization procedure, which also forms a proof of the theorem shown below, is described in appendix.

Lemma: Every even-order real matrix with determinant norm 1 can have the following factorization:

$$T_{2n} = P_{2n}\begin{bmatrix} I_n & 0 \\ A_n & I_n \end{bmatrix}\begin{bmatrix} B_n & C_n \\ 0 & I_n \end{bmatrix}\begin{bmatrix} I_n & 0 \\ D_n & I_n \end{bmatrix}$$

where $|det(\boldsymbol{B}_n)|$ =1, and $\boldsymbol{P}_{2n}$ is a $2n\times2n$ permutation matrix.

Proof: see appendix.

Theorem 1-1: Every $2^n\times2^n$ real matrix with determinant norm 1 has the following factorization:

$$T_{2^n} = P\cdot L_1\cdot U\cdot L_2$$

Where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are both lower triangular matrices with diagonal entries 1, which is always referred to as UTM (*Unit Triangular Matrix*). $\boldsymbol{U}$ is an upper triangular matrix with diagonal entries 1 except that the first entry may be –1, which is also called UTM in this paper for convenience.

Proof: see appendix. Similarly we have:

Theorem 1-2: Every $2^n\times2^n$ real matrix with determinant norm 1 also has the following factorization:

$$T_{2^n} = P\cdot U_1\cdot L\cdot U_2$$

Where U_1 and U_2 are both upper UTMs. L is a lower UTM.

3. ROUNDING ERROR ANALYSIS

It is well known that a Unit Triangular Matrix is very suitable for integer transform. For example, assuming the original transform matrix is L (UTM) with float entries. The input is an integer matrix A. Then the integer transform pair is as follows:

$$B_{i,j} = \left[\sum_{k<j} L_{i,k} A_{k,j} \right] + A_{i,j}$$

$$A_{i,j} = B_{i,j} - \left[\sum_{k<j} L_{i,k} A_{k,j} \right]$$

where [·] denotes rounding. This kind of transform is invertible.

In integer-to-integer transform, the properties of the original transform need to be preserved, so the smaller the rounding error, the better. Because multiplication with permutation matrix does not introduce rounding error, we assume the original floating point matrix be factored into:

$$T_{2^n} = L_1 \cdot U \cdot L_2$$

Where L_1, L_2 and U are lower and upper UTMs with the first entry of U may be –1. The realization of the integer transform of T is then divided into three steps containing one left-multiplication with a UTM and rounding each. Denote $\vec{\Delta}_i = (\Delta_{i,1}, \Delta_{i,2}, ..., \Delta_{i,m})^T$, $m = 2^n, i = 1,2,3$ as the rounding error vector of each step if the input is an integer vector. Obviously the total rounding error can be calculated by:

$$\vec{\delta} = \vec{\Delta}_3 + L_1 \cdot \vec{\Delta}_2 + L_1 \cdot U \cdot \vec{\Delta}_1 \qquad (1)$$

And $\Delta_{1,1} = \Delta_{2,m} = \Delta_{3,1} = 0$ .For simplicity, we make the assumption that all non-zero $\Delta_{i,j}$ s are independently, identically distributed (*i.i.d.*) with zero expectation and variance σ^2. The final expectation of rounding error can be obtained by taking expectation of (1). This procedure can be done by using the following algorithm:

1. Calculate the expectation of $\vec{\Delta}_i$ s. For example:

$$E(\Delta_{1,i}) = \begin{cases} \sigma^2 & \text{If there is any float entry in the } i\text{-th row of } L_2. \\ 0 & \text{Else} \end{cases}$$

   where σ^2 is always set to 1/12.

2. If $\vec{\Delta}_i$ is further multiplied by X, then the rounding error is propagated to:

$$\sum_j (X^T X)^2_{jj} E\Delta_{i,j}$$

3. Calculate the total error expectation:

$$\| \vec{\delta} \|^2 = \sum_i E\Delta_{3,i} + \sum_j (L_1^T L_1)^2_{jj} E\Delta_{2,j} + \sum_j ((L_1 U)^T L_1 U)^2_{jj} E\Delta_{1,j}$$

More specifically, if the original transform matrix T is orthogonal, then because $L_1 U = P^{-1} T L_2^{-1}$, we have:

$$(L_1 U)^T L_1 U = (P^{-1} T L_2^{-1})^T P^{-1} T L_2^{-1} = (L_2^{-1})^T L_2^{-1}$$

so:

$$\| \vec{\delta} \|^2 = \sum_i E\Delta_{3,i} + \sum_j (L_1^T L_1)^2_{jj} E\Delta_{2,j} + \sum_j ((L_2^{-1})^T L_2^{-1})^2_{jj} E\Delta_{1,j}$$

4. 2-D INTEGER-TO-INTEGER TRANSFORM

A 2-D float transform can be realized by performing two 1-D transforms separately. But in 2-D integer-to-integer transform condition, this is not the case. Divide 2-D integer transform into two separate 1-D integer transform will generate more rounding steps and error.

The procedure of 2-D integer transform is described below. A 2-D matrix transform can be expressed by:

$$Y = T_1 \cdot X \cdot T_2^T$$

Factoring T_1 and T_2 into three UTMs respectively using the above method, the 2-D integer transform can also be divided into three steps. Each step consists one of the following matrix multiplications and rounding.

$$Y = L \cdot X \cdot U$$

$$Y = U \cdot X \cdot L$$

Where L is a lower UTM and U is an upper UTM.

For the first matrix multiplication:

$$Y_{ij} = \sum_k \sum_l L_{il} \cdot X_{lk} \cdot U_{kj} \qquad (2)$$

Because $L_{il} = 0, U_{kj} = 0$ when $l>i$, $k>j$, (2) can be rewrite into:

$$Y_{ij} = \sum_{k \le j} \sum_{l \le i} L_{il} \cdot X_{lk} \cdot U_{kj} = X_{ij} + \sum_{k<j} \sum_{l<i} L_{il} \cdot X_{lk} \cdot U_{kj}$$

So the 2-D integer transform pair is:

$$\bar{Y}_{ij} = X_{ij} + \left[\sum_{k<j} \sum_{l<i} L_{il} \cdot X_{lk} \cdot U_{kj} \right] \qquad X_{ij} = \bar{Y}_{ij} - \left[\sum_{k<j} \sum_{l<i} L_{il} \cdot X_{lk} \cdot U_{kj} \right]$$

Where $\bar{Y}_{ij}$ is the integer output. In order to recover X_{ij}, all entries $X_{lk}, l < i, k < j$ must have already been recovered. So, for inverse transform, if the scan order is normal or zig-zag, the above transform is invertible.

Similarly, if the matrix multiplication is of the form $Y = U \cdot X \cdot L$, the transform pair is as follows:

$$\bar{Y}_{ij} = X_{ij} + \left[\sum_{k>j} \sum_{l>i} U_{il} \cdot X_{lk} \cdot L_{kj} \right] \qquad X_{ij} = \bar{Y}_{ij} - \left[\sum_{k>j} \sum_{l>i} U_{il} \cdot X_{lk} \cdot L_{kj} \right]$$

The scan order of inverse transform must be inverted. The rounding error analysis of 2-D case can be performed similar to that in part 3, but is omitted here for lack of space.

5. LEFT AND RIGHT PERMUTATION MATRICES AND INT-DCT

The permutation matrix in the theorem 1 is used to guarantee the existence of factorization into UTMs. But in this part, we show that the permutation matrix affects the total rounding error. Rewrite the factorization as follows:

$$T = P_L \cdot L_1 \cdot U \cdot L_2 \cdot P_R$$

where P_L and P_R are left and right permutation matrices (LPM and RPM). Consider a 1-D matrix transform:

$$T\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \equiv \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

In 2×2 case, there are at most 4 permutation methods (considering that the corresponding denominator must not be zero). The four integer transforms are listed below:

$$\begin{cases} B_1 = A_1 + \left[b \cdot (A_2 + \left[(a-1)A_1 / b\right])\right] \\ B_2 = A_2 + \left[(a-1)A_1 / b\right] + \left[(d-1)B_1 / b\right] \end{cases} \quad \text{if } b \neq 0$$

$$\begin{cases} B_2 = A_2 + \left[c \cdot (A_1 + \left[(d-1)A_2 / c\right])\right] \\ B_1 = A_1 + \left[(d-1)A_2 / c\right] + \left[(a-1)B_2 / c\right] \end{cases} \quad \text{if } c \neq 0$$

$$\begin{cases} B_2 = \left[d \cdot (A_2 + \left[(c+1)A_1 / d\right])\right] - A_1 \\ B_1 = A_2 + \left[(c+1)A_1 / d\right] + \left[(b-1)B_2 / d\right] \end{cases} \quad \text{if } d \neq 0$$

$$\begin{cases} B_1 = \left[a \cdot (A_1 + \left[(b+1)A_2 / a\right])\right] - A_2 \\ B_2 = A_1 + \left[(b+1)A_2 / a\right] + \left[(c-1)B_1 / a\right] \end{cases} \quad \text{if } a \neq 0$$

And the corresponding expectation of rounding error:

$$\Delta_1 = \left(2 + b^2 + d^2 + (d-1)^2 / b^2\right) \quad \Delta_2 = \left(2 + a^2 + c^2 + (a-1)^2 / c^2\right)$$

$$\Delta_3 = \left(2 + b^2 + d^2 + (b-1)^2 / d^2\right) \quad \Delta_4 = \left(2 + a^2 + c^2 + (c-1)^2 / a^2\right)$$

Thus, the left and right permutation matrices can be chosen to minimize the expectation of rounding error.

A generalized formula of how to choose the best LPM and RPM has not been derived. The algorithm we could use now is exhaustive search. This does not cause any inconvenience if the transform matrix is fixed such as DCT etc. We only need to store the three UTMs.

In many situations, say, wireless communication, there is a large demand for fast and integer DCT transform, which has been referred to as intDCT in literature.

For 4×4 DCT, the best LPM and RPM and three UTMs chosen by the above method are listed below:

$$DCT_4 = \begin{pmatrix} 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5000 & -0.5000 & -0.5000 & 0.5000 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{pmatrix} \quad P_L = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$P_R = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad L_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -0.5142 & 1 & 0 & 0 \\ 0.5142 & -1 & 1 & 0 \\ -0.213 & 0.8011 & 0 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} -1 & 1.523 & 0.5 & -0.6533 \\ 0 & 1 & 0.7571 & 0.6065 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad L_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.3935 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -0.1989 & 0.8011 & 0 & 1 \end{pmatrix}$$

6. CONCLUSION

In this paper, a new kind of matrix factorization method which is called lifting-like factorization is introduced. The theorem that every 2^n-order real matrix with determinant norm 1 can be factored into products of one permutation matrix and at most three unitary triangular matrices is proved in a recursive manner. Rounding error is analyzed, and left and right permutation matrices are introduced to reduce expectation of rounding error. 2-D integer transform is studied as a foundation of 2-D correlated transform used in MDC. A useful application of this method to intDCT is discussed.

7. APPENDIX

Proof of **Lemma**: First, for any non-singular real matrix T_{2n}, there must exist a permutation matrix $\boldsymbol{P}_{2n}$, such that:

$$T_{2n} = P_{2n} \cdot S = P_{2n} \cdot \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$$

And $\boldsymbol{S}_{12}$ is a non-singular $n \times n$ matrix. The following equation is easily shown to be true by directly expanding the right side:

$$T_{2n} = P_{2n} \cdot S = P_{2n} \cdot \begin{bmatrix} I_n & 0 \\ -S_{12}^{-1} + S_{22} \cdot S_{12}^{-1} & I_n \end{bmatrix} \cdot \begin{bmatrix} S_{12} \cdot U^{-1} & S_{12} \\ 0 & I_n \end{bmatrix} \begin{bmatrix} I_n & 0 \\ -U^{-1} + S_{12}^{-1} S_{11} & I_n \end{bmatrix}$$

where $U = (S_{22} S_{12}^{-1} S_{11} - S_{21})^{-1}$. The remaining task is to show the existence of $\boldsymbol{U}$ and $|det(S_{12} \cdot U^{-1})| = 1$. Observe the following identity:

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} 0 & I_n \\ I_n & -S_{12}^{-1} \cdot S_{11} \end{bmatrix} = \begin{bmatrix} S_{12} & 0 \\ S_{22} & -U^{-1} \end{bmatrix}$$

By taking determinant on both side, we can get $|det(S_{12} \cdot U^{-1})| = 1$, and $\boldsymbol{U}$ is obviously non-singular.

Proof of **Theorem1-1**: Here we directly show the decomposition steps, which can be considered as the algorithm as well as a proof of the theorem.

Let $k = 1$, according to lemma, we get:

$$T_{2^n} = P_{2^n} \cdot \begin{bmatrix} I_{2^{n-1}} & 0 \\ -S_{12}^{-1} + S_{22} \cdot S_{12}^{-1} & I_{2^{n-1}} \end{bmatrix} \cdot \begin{bmatrix} S_{12} \cdot U^{-1} & S_{12} \\ 0 & I_{2^{n-1}} \end{bmatrix} \begin{bmatrix} I_{2^{n-1}} & 0 \\ -U^{-1} + S_{12}^{-1} S_{11} & I_{2^{n-1}} \end{bmatrix}$$

$$= P_{2^n} \cdot \begin{bmatrix} I_{2^{n-1}} & 0 \\ \varphi_1 & I_{2^{n-1}} \end{bmatrix} \begin{bmatrix} X & \varphi_2 \\ 0 & I_{2^{n-1}} \end{bmatrix} \begin{bmatrix} I_{2^{n-1}} & 0 \\ \varphi_3 & I_{2^{n-1}} \end{bmatrix}$$

Assume after k step, original matrix can be factored into:

$$T_{2^n} = P_{2^n}{}^{(k)} \cdot \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_1 & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} X_{2^{n-k}} & *_2 \\ 0 & D_{2^n - 2^{n-k}} \end{bmatrix} \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_3 & I_{2^n - 2^{n-k}} \end{bmatrix} \quad (3)$$

Where $|det(X_{2^{n-k}})| = 1$, $D_{2^n - 2^{n-k}}$ is an upper UTM. According to Lemma, $X_{2^{n-k}}$ can be decomposed into:

$$X_{2^{n-k}} = P_{2^{n-k}} \cdot \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} = P_{2^{n-k}} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ \boldsymbol{\Phi}_1 & I_{2^{n-k-1}} \end{bmatrix} \cdot \begin{bmatrix} Z_{2^{n-k-1}} & Y_{12} \\ 0 & I_{2^{n-k-1}} \end{bmatrix} \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ \boldsymbol{\Phi}_2 & I_{2^{n-k-1}} \end{bmatrix}$$

Where $det(Z_{2^{n-k-1}}) = \pm 1$, $\boldsymbol{\Phi}_1 = -Y_{12}^{-1} + Y_{22} Y_{12}^{-1}$, $\boldsymbol{\Phi}_2 = -Y_{22} Y_{12}^{-1} Y_{11} + Y_{21} + Y_{12}^{-1} Y_{11}$. So, the last but one matrix in (3) can be rewrite into:

$$\begin{bmatrix} X_{2^{n-k}} & *_2 \\ 0 & D_{2^n - 2^{n-k}} \end{bmatrix} = \begin{bmatrix} P_{2^{n-k}} & 0 \\ 0 & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 & \\ \boldsymbol{\Phi}_1 & I_{2^{n-k-1}} & 0 \\ 0 & & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} Z_{2^{n-k-1}} & Y_{12} & \\ 0 & I_{2^{n-k-1}} & * \\ & 0 & D_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 & \\ \boldsymbol{\Phi}_2 & I_{2^{n-k-1}} & 0 \\ 0 & & I_{2^n - 2^{n-k}} \end{bmatrix}$$

Where:

$$* = \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ \boldsymbol{\Phi}_1 & I_{2^{n-k-1}} \end{bmatrix}^{-1} P_{2^{n-k}}{}^{-1} \cdot *_2 = \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ -\boldsymbol{\Phi}_1 & I_{2^{n-k-1}} \end{bmatrix} P_{2^{n-k}}{}^{T} \cdot *_2$$

So, (3) can be further factored into:

$$T_{2^n} = P_{2^n}{}^{(k)} \cdot \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_1 & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} P_{2^{n-k}} & 0 \\ 0 & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 & \\ \boldsymbol{\Phi}_1 & I_{2^{n-k-1}} & 0 \\ 0 & & I_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} Z_{2^{n-k-1}} & Y_{12} & \\ 0 & I_{2^{n-k-1}} & * \\ & 0 & D_{2^n - 2^{n-k}} \end{bmatrix} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 & \\ \boldsymbol{\Phi}_2 & I_{2^{n-k-1}} & 0 \\ 0 & & I_{2^n - 2^{n-k}} \end{bmatrix} \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_3 & I_{2^n - 2^{n-k}} \end{bmatrix} \quad (4)$$

Because

$$P_{2^n}{}^{(k)} \cdot \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_1 & I_{2^n - 2^{n-k}} \end{bmatrix} \begin{bmatrix} P_{2^{n-k}} & 0 \\ 0 & I_{2^n - 2^{n-k}} \end{bmatrix}$$

$$= P_{2^n}{}^{(k)} \cdot \begin{bmatrix} P_{2^{n-k}} & 0 \\ *_1 \cdot P_{2^{n-k}} & I_{2^n - 2^{n-k}} \end{bmatrix}$$

$$= P_{2^n}{}^{(k)} \cdot \begin{bmatrix} P_{2^{n-k}} & 0 \\ 0 & I_{2^n - 2^{n-k}} \end{bmatrix} \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_1 \cdot P_{2^{n-k}} & I_{2^n - 2^{n-k}} \end{bmatrix}$$

$$= P_{2^n}{}^{(k+1)} \cdot \begin{bmatrix} I_{2^{n-k}} & 0 \\ *_1 \cdot P_{2^{n-k}} & I_{2^n - 2^{n-k}} \end{bmatrix}$$

Multiplying matrices in (4) leads to:

$$T_{2^n} = P_{2^n}{}^{(k+1)} \cdot \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ *'_1 & I_{2^n - 2^{n-k-1}} \end{bmatrix} \cdot \begin{bmatrix} Z_{2^{n-k-1}} & *'_2 \\ 0 & D_{2^n - 2^{n-k-1}} \end{bmatrix} \begin{bmatrix} I_{2^{n-k-1}} & 0 \\ *'_3 & I_{2^n - 2^{n-k-1}} \end{bmatrix} \quad (5)$$

Where $det(Z_{2^{n-k-1}}) = \pm 1$, $D_{2^n - 2^{n-k-1}}$ is an upper UTM. Because (5) has the same form with (3), the factorization can be finished after $k = n-1$, and the theorem holds.

8. REFERENCES

[1] Y.Wang, M.T.Orchard and A.R.Teibman, "Multiple Description Coding Using Pairwise Correlating Transforms," *IEEE Trans. Image Processing*, vol. 10, No.3, pp.351-365, Mar. 2001.

[2] X. Li, B.Tao and M.T.Orchard, "On implementing transforms from integers to integers," in *Proc. Int. Conf. Image Processing (ICIP'98)*, Chicago, IL, pp. 881-885, Oct. 1998.

[3] V.K.Goyal and J.Kovacevic, "Generalized Multiple Description Coding With Correlating Transforms," *IEEE Trans. Inform. Theory,* vol. 47, No.6, pp. 2199-2223, Sept. 2001.

[4] R.A.Horn and C.R.Johnson, *Matrix analysis*, Camgridge University Press 1985.

[5] Ying-Jui Chen, S. Oraintara and Truong Nguyen, "Video compression using integer DCT," in *Proc. Int. Conf. Image Processing (ICIP'00)*, pp. 844 -847 vol.2 2000.

[6] W.Philips, "The lossless DCT for combined lossy/lossless image coding," in *Proc. Int. Conf. Image Processing (ICIP'98)*, Chicago, IL, pp. 871-875, Oct. 1998.

[7] I.Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *J. Fourier Anal. Appl.,* Vol. 4, No. 3, pp. 247-269, 1998.

[8] P. Hao and Q. Shi, "Matrix Factorizations for Reversible Integer Mapping," *IEEE Trans. Signal Processing,* Vol. 49, No. 10, pp. 2314-2324, Oct. 2001.