Stabilization and Optimization of PLUS Factorization and Its Application in Image Coding

Lei Yang[a], Pengwei Hao[b,c], Dapeng Wu[a]

[a]*Department of Electrical and Computer Engineering, University of Florida, FL, 32608, USA*
[b]*Department of Machine Intelligence, Peking University, Beijing, 100871, China*
[c]*Department of Computer Science, Queen Mary University of London, London, E1 4NS, UK*

Abstract

A recently developed PLUS factorization holds great promise in image coding due to its simplicity and integer reversibility. However, existing PLUS factorizations did not consider stability and optimality. To address these problems, we propose methodologies to design stable and optimal PLUS factorization algorithms. Firstly, we propose three stable PLUS factorization algorithms, prove the stability theorem under no perturbation and analyze stability under perturbation. Furthermore, we obtain a closed-form formula for transform error, and use the formula to design an algorithm for optimal PLUS factorization. Then, we apply the PLUS factorization to image coding. The integer DCTs implemented with the optimal PLUS factorizations found by our algorithms outperform the integer DCT with expansion factors in terms of entropy. The optimal PLUS factorizations are superior to the lifting factorization in JPEG-XR. The experimental results agree with analytical results of PLUS factorization, and show superior performance of our algorithms in image coding.

Key words: PLUS factorization, stable algorithm, optimization, transform coding, image compression, integer reversible transform, lapped transform, discrete cosine transform, lifting factorization

1. Introduction

Transforms are widely used in source coding, image processing and computer graphics. Transform coding is a major technique in image and video compression standard, such as JPEG, JPEG 2000, JPEG-XR [1], MPEG [2] and H.264 [3]. Factorization is used to make the transforms, such as Discrete Cosine Transform (DCT) [4] and Discrete Wavelet Transform (DWT) [5], faster, simpler and integer reversible. PLUS factorization is a kind of customizable triangular matrix factorization, proposed by Hao [7] as a new framework of matrix factorization, and encompasses and generalizes quite a few triangular matrix factorizations [8, 20, 21]. The PLUS factorization of a general nonsingular matrix $\boldsymbol{A}$ is formulated as:

$$\boldsymbol{A}=\boldsymbol{PLUS} \tag{1}$$

where matrices $\boldsymbol{P}$, $\boldsymbol{L}$ and $\boldsymbol{U}$ are, almost the same as in LU factorization, permutation, unit lower and upper triangular matrices, respectively, while $\boldsymbol{S}$ is a very special matrix, which is unit, lower and triangular, and only with no more than $N-1$ nonzeros. Different from LU factorization, all the diagonal elements of $\boldsymbol{U}$ in PLUS factorization are customizable, i.e. the diagonal elements can be assigned almost freely, as long as the determinant is equal to that of $\boldsymbol{A}$ up to a possible sign adjustment. With PLUS factorization, a nonsingular matrix $\boldsymbol{A}$ is easily factorized further into a series of special matrices similar to $\boldsymbol{S}$. Furthermore, the permutation matrix can also be substituted with a *pseudo-permutation matrix*, which is a simple unit upper triangular matrix with 0, 1 and -1 as its off-diagonal elements. Besides $\boldsymbol{PLUS}$, a customizable factorization also has other alternatives, $\boldsymbol{LUSP}$, $\boldsymbol{PSUL}$ or $\boldsymbol{SULP}$ with lower $\boldsymbol{S}$, and $\boldsymbol{PULS}$, $\boldsymbol{ULSP}$, $\boldsymbol{PSLU}$ or $\boldsymbol{SLUP}$ with upper $\boldsymbol{S}$, which are all taken as varieties of $\boldsymbol{PLUS}$ factorization.

Currently, the lifting factorization [12] is mostly used to factorize the transforms into lifting steps to simplify the transforms as well as to make the transform integer reversible, such as the factorization in JPEG-XR [30]. But these factorizations are mostly based on experiences and experiments. However, the PLUS factorization provides a general and universal way to factorize transform matrices of any order into products of *Elementary Reversible Matrices* [8]. With the Elementary Reversible Matrices as factor matrices, PLUS factorization is a powerful tool for realizing the integer reversible transform [6], when assisted by the ladder structure and the rounding operations [8]. Therefore, it has promising applications in lossless/lossy coding [8, 9, 10] and reversible image processing [15, 16]. Meanwhile, an elementary reversible matrix is also a triangular shear matrix. Thus, it also has found applications in computer graphics, such as transformation acceleration by shears [13], fast image registration [14], in which matrix-based transformations dominate. However, the existing PLUS factorization suffers from two limitations: instability and sub-optimality. By instability, we mean that the PLUS factorization may stop because of zero or near zero pivoting during the process of Gaussian Elimination. By sub-optimality, we mean that the PLUS factorization of transform matrices found by existing factorization algorithm may lead to large transform error, which may deprive the good properties of original transform matrices, such as orthogonality and high energy-compacting ability, from the products of the factor matrices of PLUS factorization.

This work is proposed to address these problems (other problems of PLUS factorization, like blocking factorization and parallelling computing, arising from solving large linear systems, could be found in [17, 18, 19]). Our main contributions include:

1. We propose three stable PLUS algorithms in Matlab pseudo code, and the methodology to stabilize the factorization.
2. We prove the stable theorem and did perturbation analysis of PLUS factorization, to guarantee the stability of our algorithms theoretically.
3. We obtain a closed-form formula of transform error of PLUS factorization, and propose the optimization algorithm based on Tabu Search to quickly find the optimal factorization and to realize integer transform with the least transform error.
4. We apply our algorithms to realize integer DCT and integer Lapped Transform, and test the lossy/lossless image coding performance. Experimental results show the superiority of our algorithms over some existing integer DCT algorithms, and the lapped transform factorization in JPEG-XR.

The paper is organized as below: the stable PLUS factorization algorithms are introduced in Section 2; error analysis and optimization of PLUS factorization with Tabu search algorithm are discussed in Section 3; the numerical examples of factorization and the experimental results on integer reversible transforms for image compression are shown in Section 4. Finally, Section 5 concludes the paper.

2. Stabilization of PLUS Factorization Algorithms

The permutation matrix $\boldsymbol{P}$ in PLUS factorization serves for pivoting. If $\boldsymbol{P}$ is chosen improperly, the algorithm may be unstable, namely division by 0 or extremely small numbers during Gaussian Elimination. In this section, we will introduce three stable PLUS factorization algorithms for any nonsingular matrix based on pivoting and back tracing, in which $\boldsymbol{S}$ is designated as a *single-row elementary matrix* [8] with $n-1$ nonzero off-diagonal elements in the last row. The algorithms could be generalized to factorization with any patterned $\boldsymbol{S}$ [7], as well as polyphase matrix [11, 12] factorization. The analysis of algorithms and stability theorem are also presented, as well as the uniqueness property of PLUS factorization. Given the diagonal entries of $\boldsymbol{U}$ and the pattern of $\boldsymbol{S}$, with a specific permutation matrix $\boldsymbol{P}$, there is one and only one form of PLUS factorization for $\boldsymbol{A}$ under the necessary and sufficient condition of nonzero determinants of the sub-matrices of $\boldsymbol{A}$. Perturbation analysis for PLUS factorization is also provided in Appendix B to show its numerical stability with finite perturbation error bounds.

Definition 1. *Stability of the PLUS factorization algorithm.*
The algorithm is stable when there are no zero pivots during the PLUS factorization, which lead to infinity elements in the factor matrices.

MATLAB's notations and grammar are extensively used to represent our algorithms. Firstly, we summarize the general PLUS factorization algorithm proposed in the constructive proof in [7] in a more direct form with MATLAB notations as below.

Algorithm 1. *General PLUS factorization algorithm.*
For a nonsingular n-by-n matrix $\boldsymbol{A}$, the first $n-1$ diagonal entries of $\boldsymbol{U}$ are given in vector $\boldsymbol{u}$, then the general PLUS factorization algorithm is given as follows:

```
for $i = 1:(n-1)$ do
    $s(i) = (A(i,i) - u(i))/A(i,n)$
    $A(1:n,i) = A(1:n,i) - s(i) \cdot A(1:n,n)$
    $k = (i+1):n$
    $A(k,i) = A(k,i)/A(i,i)$
    $A(k,k) = A(k,k) - kron(A(k,i), A(i,k))$
end for
$L = I + strict\_lower\_tri(A)$
```

```
  $U = upper\_tri(A)$
  $S = I + [zeros(n-1,1); 1] \cdot [s, 0]$   $\square$
```

$kron(\cdot,\cdot)$ stands for the kronecker product of two matrices; $strict\_lower\_tri(A)$ is a strictly-lower triangular matrix obtained from A; $upper\_tri(A)$ is an upper triangular matrix obtained from A.

From Algorithm 1, it is obvious that except for the additional $n-1$ memory units for $\boldsymbol{S}$, $\boldsymbol{L}$ and $\boldsymbol{U}$ can be calculated in the place of $\boldsymbol{A}$, and there are only $n-1$ iterations in the main body of the algorithm. There are two pivots located in line 2 and 5 of Algorithm 1, where lies the instability. Simply substituting the zero pivots is practical for stabilization, but not enough to obtain smaller rounding error accumulated during elimination. Thus we propose PLUS factorization with partial pivoting in Algorithm 2, with the two pivots of the largest magnitudes in $\boldsymbol{A}(i:n,n)$ and $\boldsymbol{A}(i:n,i)$. Then we show the theorem of its stability .

Algorithm 2. *PLUS factorization with partial pivoting.*
For a nonsingular n-by-n matrix $\boldsymbol{A}$, the first $n-1$ diagonal entries of $\boldsymbol{U}$ are given in vector $\boldsymbol{u}$, then the PLUS factorization algorithm with partial pivoting is given as follows:

```
  $P = 1:n$
  for $i = 1:(n-1)$ do
    Determine $\mu_1$ with $i \le \mu_1 \le n$, so that $|A(\mu_1, n)| = \| A(i:n, n) \|_\infty$
    $P(i) \leftrightarrow P(\mu_1)$
    $A(i, 1:n) \leftrightarrow A(\mu_1, 1:n)$
    $s(i) = (A(i,i) - u(i))/A(i,n)$
    $A(1:n, i) = A(1:n, i) - s(i) \cdot A(1:n, n)$
    Determine $\mu_2$ with $i \le \mu_2 \le n$, so that $|A(\mu_2, i)| = \| A(i:n, i) \|_\infty$
    $P(i) \leftrightarrow P(\mu_2)$
    $A(i, 1:n) \leftrightarrow A(\mu_2, 1:n)$
    $k = (i+1):n$
    $A(k,i) = A(k,i)/A(i,i)$
    $A(k,k) = A(k,k) - kron(A(k,i), A(i,k))$
  end for
  $L = I + strict\_lower\_tri(A)$
  $U = upper\_tri(A)$
  $S = I + [zeros(n-1,1); 1] \cdot [s, 0]$   $\square$
```

Theorem 1. *Stability of Algorithm 2.*
For a nonsingular n-by-n matrix $\boldsymbol{A}$, the first $n-1$ diagonal entries of $\boldsymbol{U}$ are given in vector $\boldsymbol{u}$, then PLUS factorization algorithm with partial pivoting can avoid division by zeros and give a stable factorization.
Proof : (by contradiction)
If Algorithm 2 is disabled in the i-th iteration ($1 \le i \le n-1$), i.e. $\boldsymbol{A}(j,n) = 0, j = i, \cdots, n$, then matrix $\boldsymbol{A}$ becomes:

$$\left(\begin{array}{cccc|cccc}
u_1 & a_{1,2}^{(i)} & \cdots & a_{1,i-1}^{(i)} & a_{1,i}^{(i)} & \cdots & & a_{1,n}^{(i)} \\
0 & u_2 & a_{2,3}^{(i)} & \cdots & \vdots & & & \vdots \\
\vdots & \ddots & \ddots & \cdots & & & & \\
0 & \cdots & 0 & u_{i-1} & a_{i-1,i}^{(i)} & \cdots & & a_{i-1,n}^{(i)} \\
\hline
0 & \cdots & & 0 & a_{i,i}^{(i)} & \cdots & a_{i,n-1}^{(i)} & 0 \\
\vdots & & & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & & 0 & a_{n,i}^{(i)} & \cdots & a_{n,n-1}^{(i)} & 0
\end{array}\right) \tag{2}$$

where the superscripts (i) indicate i-th iteration.
Because of

$$\det(\begin{pmatrix} \boldsymbol{B}_{r\times r} & \boldsymbol{C}_{r\times s} \\ \boldsymbol{O} & \boldsymbol{D}_{s\times s} \end{pmatrix}) = \det(\boldsymbol{B}_{r\times r}) \cdot \det(\boldsymbol{D}_{s\times s}),$$

we deduce $\det(\boldsymbol{A}) = 0$, *which contradicts with the prerequisite that $\boldsymbol{A}$ is nonsingular.* ■

Due to finite precision of computers, some ill-conditioned matrices, with near singular sub-matrices, may halt in the i-th iteration of PLUS factorization with a midway product as:

$$\left(\begin{array}{cccc|cccc}
u_1 & a_{1,2}^{(i)} & \cdots & a_{1,i-1}^{(i)} & a_{1,i}^{(i)} & \cdots & & a_{1,n}^{(i)} \\
0 & u_2 & a_{2,3}^{(i)} & \cdots & \vdots & & & \vdots \\
\vdots & \ddots & \ddots & \cdots & & & & \\
0 & \cdots & 0 & u_{i-1} & a_{i-1,i}^{(i)} & \cdots & & a_{i-1,n}^{(i)} \\
\hline
0 & \cdots & & 0 & a_{i,i}^{(i)} & \cdots & a_{i,n-1}^{(i)} & \varepsilon_{i,n} \\
\vdots & & & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & & 0 & a_{n,i}^{(i)} & \cdots & a_{n,n-1}^{(i)} & \varepsilon_{n,n}
\end{array}\right) \tag{3}$$

where $|\varepsilon_{k,n}| < \epsilon, k = i, \cdots, n$, and ϵ is the precision of a computer. For such ill-conditioned matrices, some improvements could be done based on Algorithm 2.

An upper triangular pseudo-permutation matrix instead of a permutation matrix can be used in Algorithm 2 to deal with some ill-conditioned cases. Because $\boldsymbol{L}^{-1} = \boldsymbol{L}_{n-1}(\boldsymbol{P}_{n-1}\boldsymbol{L}_{n-2}\boldsymbol{P}_{n-1}^{-1})\cdots(\boldsymbol{P}_{n-1}\boldsymbol{P}_{n-2}\cdots\boldsymbol{P}_2\boldsymbol{L}_1\ \boldsymbol{P}_2^{-1}\cdots\boldsymbol{P}_{n-2}^{-1}\boldsymbol{P}_{n-1}^{-1})$ where $\boldsymbol{P}_i = \boldsymbol{I} + \boldsymbol{E}_{i,i_m}, i < i_m, i = 2, \cdots, n-1$, are pseudo-permutation matrices in the i-th iteration, $\boldsymbol{L}_i$ is unit lower triangular matrices composed of the i-th column of $\boldsymbol{L}$. The operations of pseudo-permutation matrices $\boldsymbol{P}_i$ applied to $\boldsymbol{A}$ are equivalent to that the elements of the i-th row of $\boldsymbol{A}$ are added to or subtracted from the i_m-th row. This stratege can make Algorithm 2 in effect, when $|\varepsilon_{i,n}| + |\varepsilon_{i_m,n}| > \epsilon$ in the i-th iteration.

Because of Gaussian Elimination, the pivot chosen in the current iteration helps to cope with ill-conditioned cases only in the current iteration, but can not determine the fate of the later iterations. Thus, a trace-back algorithm is a good candidate for global control of ill-condition as in Algorithm 3 with a little increase of algorithm complexity. The Algorithm 3 traces back to the last iteration to re-choose a pivot, namely to multiply a new permutation matrix, when the algorithm encounters a failure in the current iteration.

Algorithm 3. *Trace-back PLUS factorization with partial pivoting.*

```
function PLUS ( pass, P )
if pass == n then
  return
else
  for i=pass : n do
    if $|A(i,n)| > \epsilon$ then
      $P(pass) = i$
      $A(pass, 1:n) \leftrightarrow A(i, 1:n)$
      s(pass)=(A(pass,pass)-u(pass))/A(pass,n)
      A(1:n,pass)=A(1:n,pass)-s(pass)·A(1:n,n)
      for j=pass : n do
        if $|A(j,pass)| > \epsilon$ then
          $P(pass) = j$
          $A(pass, 1:n) \leftrightarrow A(j, 1:n)$
          $k = (pass + 1) : n$
          $A(k, pass) = A(k, pass)/A(pass, pass)$
          A(k,k)=A(k,k)-kron(A(k,pass),A(pass,k))
```

```
            PLUS ( pass+1, P )
            $A(pass, 1:n) \leftrightarrow A(j, 1:n)$
          end if
        end for
        $A(pass, 1:n) \leftrightarrow A(i, 1:n)$
      end if
    end for
  end if  $\square$
```

PLUS factorization with complete pivoting is equivalent to applying both left and right permutation matrices to $\boldsymbol{A}$ in each iteration. Not only is the following algorithm stable without 0 divisors, but also avoids subtraction between two very close numbers when calculating $s(pass)$ in line 9. The column pivots must avoid the elements in the last column to guarantee that $\boldsymbol{S}_i\boldsymbol{P}_{R_{i+1}} = \boldsymbol{P}_{R_{i+1}}\boldsymbol{S}_i$. Therefore, $\boldsymbol{P}_L\boldsymbol{A}\boldsymbol{P}_{R_1}\boldsymbol{S}_1\boldsymbol{P}_{R_2}\boldsymbol{S}_2\cdots\boldsymbol{P}_{R_{n-1}}\boldsymbol{S}_{n-1} = \boldsymbol{A}\boldsymbol{P}_L\boldsymbol{A}\boldsymbol{P}_R\boldsymbol{S}$, where $\boldsymbol{P}_R = \boldsymbol{P}_{R_1}\boldsymbol{P}_{R_2}\cdots\boldsymbol{P}_{R_{n-1}}$ and $\boldsymbol{S} = \boldsymbol{S}_1\boldsymbol{S}_2\cdots\boldsymbol{S}_{n-1}$, where $\boldsymbol{P}_L$ is left permutation matrix, $\boldsymbol{P}_R$ is right permutation matrix and $\boldsymbol{P}_{R_i}$ is the right permutation matrix in the i-th iteration.

Algorithm 4. *PLUS factorization with complete pivoting.*

```
  $P_L = 1:n$
  $P_R = 1:n$
  for $i = 1:(n-1)$ do
    Determine $\mu$ with $i \le \mu \le n$ , so that $|A(\mu, n)| = \| A(i:n, n) \|_\infty$
    Determine $\lambda$ with $i \le \lambda \le n-1$, so that $|A(\mu, \lambda)| = max\{|A(\mu, m) - d(i)|, m = i:(n-1)\}$
    $P_L(i) \leftrightarrow P_L(\mu)$
    $P_R(i) \leftrightarrow P_R(\lambda)$
    $A(i, 1:n) \leftrightarrow A(\mu, 1:n)$
    $A(1:n, i) \leftrightarrow A(1:n, \lambda)$
    if $|A(i, n)| > \epsilon$ then
      $s(i) = (A(i, i) - u(i))/A(i, n)$
      $A(1:n, i) = A(1:n, i) - s(i) \cdot A(1:n, n)$
      Determine $\nu$ with $i \le \nu \le n$ , so that $|A(\nu, i)| = \| A(i:n, i) \|_\infty$
      $P_L(i) \leftrightarrow P_L(\nu)$
      $A(i, 1:n) \leftrightarrow A(\nu, 1:n)$
      $k = (i+1):n$
      $A(k, i) = A(k, i)/A(i, i)$
```

```
        $A(k,k) = A(k,k) - kron(A(k,i), A(i,k))$
      **end if**
    **end for**
    $L = I + strict\_lower\_tri(A)$
    $U = upper\_tri(A)$
    $S = I + [zeros(n-1,1); 1] \cdot [s, 0]$   $\square$
```

Similar to Algorithm 2, Algorithm 4 can also have a trace-back version, which is not difficult to obtain. Besides, it is natural for users to customize pivoting strategies of these algorithms to obtain other expected properties.

Collorary 1. *The algorithm of trace-back PLUS factorization with partial pivoting and PLUS factorization with complete pivoting are both stable. Similar to the proof of Theorem 1, this proof is easy to obtain.*

Besides the n-by-n nonsingular matrices, by using our algorithm, an n-by-m full-rank rectangular matrix can also have a PLUS factorization in the following forms:
For $m < n$,

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \boldsymbol{P} \cdot \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{m1} & l_{m2} & \cdots & 1 \end{pmatrix}$$

$$\cdot \begin{pmatrix} u_1 & u_{12} & \cdots & & u_{1n} \\ & u_2 & \ddots & & u_{2n} \\ & & \ddots & & \vdots \\ & & & u_m & u_{mn} \end{pmatrix} \cdot \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ \vdots & & \ddots & \\ s_1 & \cdots & s_{n-1} & 1 \end{pmatrix};$$

for $m > n$,

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \boldsymbol{P} \cdot \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ l_{m1} & l_{m2} & \cdots & l_{mn} \end{pmatrix}$$

$$\cdot\begin{pmatrix} u_1 & u_{12} & \cdots & u_{1n} \\ & u_2 & \ddots & \vdots \\ & & \ddots & u_{n-1,n} \\ & & & u_n \end{pmatrix}\cdot\begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ \vdots & & \ddots & \\ s_1 & \cdots & s_{n-1} & 1 \end{pmatrix}$$

Next, the uniqueness of PLUS factorization is discussed.

Condition 1. *Uniqueness Condition.*
$\det([\boldsymbol{a}_n, \boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_{k-1}]^{(k)}) \neq 0$ *when* $k > 1$, *and* $a_{1n} \neq 0$ *when* $k = 1$, *where* $\boldsymbol{A}^{(k)}$ *denotes the* k*-th leading sub-matrix of* $\boldsymbol{A}$.

Theorem 2. *Uniqueness of PLUS factorization.*
Given a permutation matrix $\boldsymbol{P}$, *diagonal entries of* $\boldsymbol{U}$ *and the pattern of* $\boldsymbol{S}$, *if a nonsingular* n*-by-*n *matrix* $\boldsymbol{A}$ *has PLUS factorization,* $\boldsymbol{A} = \boldsymbol{PLUS}$ *and Condition 1 is satisfied for* $\boldsymbol{P}^T\boldsymbol{A}$, *then PLUS factorization is unique.*
By induction, Theorem 2 is proved in Appendix A.

Condition 1 is necessary and sufficient, and satisfied in our algorithms. For a counterexample, if $\boldsymbol{P^T A} = \boldsymbol{I}$, which does not satisfy Condition 1, then $\boldsymbol{I} = \boldsymbol{I}\boldsymbol{S}_0^{-1}\boldsymbol{I}\boldsymbol{S}_0$ is the PLUS factorization, where $\boldsymbol{S}_0$ could be any single-row elementary reversible matrix with nonzeros in the last row, i.e. PLUS factorization for $\boldsymbol{P^T A}$ is not unique.

Based on Theorem 2, the property of uniqueness of PLUS factorization could be extended to reciprocal determination between two sets. Let $V = \{u_1, u_2, \cdots, u_{n-1}, s_1, s_2, \cdots, s_{n-1}\}$, and Q, T compose a partition of V, with $n-1$ elements respectively. Let $Q = \{q_1, q_2, \cdots, q_{n-1}\}$ and $T = \{t_1, t_2, \cdots, t_{n-1}\}$, q and t could be either u or s. If Q is pre-designated, then T could be determined uniquely by Q in our algorithms, and vice visa.

3. Optimization of PLUS Factorization

For the better performance in applications, optimization of PLUS factorization aims to minimize three types of transform error, which has three main origins. The first one is due to the precision limitation of computers. The second one results from the rounding operations for integer reversible transformations. The third one comes from the possible quantization of compression. The error will be further propagated and amplified after multiplications by factor matrices. The transform error causes the differences between the coefficients after transform with factor matrices of PLUS factorization and those after the original transform. It is well known that the traditional linear transforms such as DCT and DWT hold many good properties like orthogonality, high de-correlation and energy-concentration ability for effective image coding. Therefore, to keep the same merits as the original transform matrices is an important concern, when using PLUS factorization as the effective tool for realizing integer reversible reversion of the traditional transforms. The least transform error is desired. This optimization problem of PLUS factorization is denoted as *PLUS FOP* in the following discussion.

PLUS factorization is really diversified, even with a given pattern of $\boldsymbol{S}$ and ± 1 as diagonal entries of $\boldsymbol{U}$. For an n-by-n matrix $\boldsymbol{A}$, $\boldsymbol{A} = \boldsymbol{P}_L\boldsymbol{L}\boldsymbol{U}\boldsymbol{S}\boldsymbol{P}_R$, there are up to $n!$ possible left permutation matrices $\boldsymbol{P_L}$, $n!$ possible right permutation matrices $\boldsymbol{P_R}$ and 2^{n-1} possible combinations of first $n-1$ diagonal entries of $\boldsymbol{U}$. It means that there are $n! \times n! \times 2^{n-1}$ possible PLUS factorizations. It is an N-P hard problem to find the optimal PLUS factorization. Thus, enumerating all the possible solutions to find the best is out of the question for the high-order matrices, but its results for low-order matrices can be used as a ground truth for comparison between algorithms. To solve PLUS FOP, we present our error analysis of PLUS factorization, propose E_2 error metric, and design the optimization algorithm with Tabu Search to find the factorizations with the least transform error. The optimal PLUS factorization with the least transform error can help improve the performance of various systems, e.g., lossless/lossy image coding.

3.1. Transform Error Analysis

The transform error after the transformation steps with the factor matrices of PLUS is actually a mixture of direct round-off error and indirect error propagated and accumulated from the round-off error in the previous steps. For $\boldsymbol{A} = \boldsymbol{P}_L\boldsymbol{L}\boldsymbol{U}\boldsymbol{S}\boldsymbol{P}_R$, the transform error can be formulated as:

$$\boldsymbol{e} = \boldsymbol{P}_L(\boldsymbol{e}_L + \boldsymbol{L}(\boldsymbol{e}_U + \boldsymbol{U}(\boldsymbol{e}_S + \boldsymbol{S}\boldsymbol{P}_R\boldsymbol{e}_0))) \tag{4}$$

where $\boldsymbol{e}_L$, $\boldsymbol{e}_U$ and $\boldsymbol{e}_S$ are round-off error vectors directly brought in after the transformation with $\boldsymbol{L}$, $\boldsymbol{U}$ and $\boldsymbol{S}$, respectively, and $\boldsymbol{e}_0$ is the system error vector before transformation. Compared with the transform error using the original matrix $\boldsymbol{A}$, $\boldsymbol{e} = \boldsymbol{e}_A + \boldsymbol{A}\boldsymbol{e}_0$, $\boldsymbol{e}_0$ can be disregarded, and the following error model is considered for PLUS factorization:

$$\boldsymbol{e} = \boldsymbol{P}_L(\boldsymbol{e}_L + \boldsymbol{L}(\boldsymbol{e}_U + \boldsymbol{U}\boldsymbol{e}_S)) \tag{5}$$

where $\boldsymbol{e}_L = [0, 1, 1, \cdots, 1]^T$, $\boldsymbol{e}_U = [1, 1, \cdots, 1, 0]^T$ and $\boldsymbol{e}_S = [0, 0, \cdots, 0, 1]^T$ as the upper bound of the magnitudes of error vectors, when the floor() operator is used. $\boldsymbol{e}$ can be evaluated by its norm as follows:

$$\begin{aligned} \parallel \boldsymbol{e} \parallel &= \parallel \boldsymbol{P}_L(\boldsymbol{e}_L + \boldsymbol{L}(\boldsymbol{e}_U + \boldsymbol{U}\boldsymbol{e}_S)) \parallel \\ &= \parallel \boldsymbol{e}_L + \boldsymbol{L}\boldsymbol{e}_U + \boldsymbol{L}\boldsymbol{U}\boldsymbol{e}_S \parallel \\ &\leq \parallel \boldsymbol{e}_L \parallel + \parallel \boldsymbol{L}\boldsymbol{e}_U \parallel + \parallel \boldsymbol{L}\boldsymbol{U}\boldsymbol{e}_S \parallel \end{aligned} \tag{6}$$

In our random numerical experiments, such upper error bound can sometimes be reached. Therefore, in the integer transform domain, an error metric of PLUS factorization can be defined by the error bound in Equation (6):

$$E_p(\boldsymbol{LUS}) = \parallel \boldsymbol{e}_L \parallel_p + \parallel \boldsymbol{L}\boldsymbol{e}_U \parallel_p + \parallel \boldsymbol{L}\boldsymbol{U}\boldsymbol{e}_S \parallel_p \tag{7}$$

where $\parallel \bullet \parallel_p$ is the p-norm operator.

3.2. Statement of Optimization Problem

Take the error metric defined in Equation (7) as the objective function, the optimization of PLUS factorization for any nonsingular matrix $\boldsymbol{A}$ is formulated as:

$$\begin{aligned} &\min_{\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u}} E_p(\boldsymbol{LUS}) \\ &s.t. \;\; \boldsymbol{A} = \boldsymbol{P}_L \boldsymbol{LUS} \boldsymbol{P}_R \end{aligned} \tag{8}$$

where $\boldsymbol{u}$ is the vector composed by the first $n-1$ diagonal elements of $\boldsymbol{U}$.

For the vector norm in Equation (7), we test L_1, L_2 and L_∞ in our experiments, and use E_1, E_2 and E_∞ to represent the corresponding error metric, respectively. Our experimental results in Section 4 show that L_2, which is a continuous function of vectors, consists with the definition of Mean Square Error to achieve Least Mean Square Error. Thus, the transform error metric is defined as E_2.

The global optimal factorization results for the DCT matrices found by exhaustive search are given in Table 2, when matrix order $n = 2, 4, 8$. It is easy to obtain the optima when $n = 2$ or 4. But when $n = 8$, a PC with 0.7G CPU and 128M Memory takes nearly 3 weeks to try all the possibilities, not to mension the matrices of higher orders. The facts founded by exhaustive search are: (i) Due to symmetry of solution space, the optimal solutions are not unique: 4 optimal results for 2×2 DCT, 4 for 4×4 DCT, and 32 for

8×8 DCT are found in our experiments. (ii) There are a lot of factorization results, which approximate the optimal, scattering in the feasible solution space. In our experiments, there are 8 suboptimal factorizations near the optimal one for 4×4 DCT, with 0.08 more transform error than the least transform error of the optimal factorization.

Therefore, attempts to solve this NP-hard nonlinear combinatorial optimization problem can be achieved by heuristic methods, which yield the approximately optimal solutions in a polynomial time. Such methods include Neural Network (NN), Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search (TS) and so on [25]. In this paper, we use Tabu Search to solve the PLUS FOP, since it is a combinatorial optimization problem, and can be well modelled in the TS framework and well solved as shown in the experiments.

3.3. Optimization Algorithm with Tabu Search

Tabu search (TS) is a meta-heuristic technique proposed by Glover [23] to solve combinatorial optimization problems, such as vehicle routing and shop scheduling problems [25]. TS avoids being trapped at local minima by allowing the temporal acceptance of the worse solutions, and avoids cyclically revisiting solutions by keeping track of the recent migrations of the solutions in tabu list. It consistently outperforms the algorithms in Section 2 and provides amazing optimization performance. Based on the essential procedures of the TS algorithm and the characteristics of possible solutions mentioned in Section 3.2, we see that PLUS FOP is a typical problem in the TS framework and TS works well for our PLUS FOP, which is also verified by the experiments.

Some key terms in the TS algorithm are:

1. *Objective function*
   The objective function E is defined in Equation (8) with $p = 2$.
2. *Possible solution set*
   A possible solution can be represented as a triplet $(\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u})$. The possible solution set is $\Omega = \{(\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u})\}$, and $|\Omega| = n! \times n! \times 2^{n-1}$.
3. *Neighbors*
   $\forall X \in \Omega, X = (\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u})$, the neighbors of X is defined as: $B(X) = \{(\boldsymbol{P}^{'}_L, \boldsymbol{P}^{'}_R, \boldsymbol{u}^{'}) | d((\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u}), (\boldsymbol{P}^{'}_L, \boldsymbol{P}^{'}_R, \boldsymbol{u}^{'})) = 1\}$, where
   $d((\boldsymbol{P}_L, \boldsymbol{P}_R, \boldsymbol{u}), (\boldsymbol{P}^{'}_L, \boldsymbol{P}^{'}_R, \boldsymbol{u}^{'})) = d^{'}(\boldsymbol{P}_L, \boldsymbol{P}^{'}_L) + d^{'}(\boldsymbol{P}_R, \boldsymbol{P}^{'}_R) + d^{''}(\boldsymbol{u}, \boldsymbol{u}^{'})$, $d^{'}(\boldsymbol{P}, \boldsymbol{P}^{'}) = (\sum_{i,j} \bar{\delta}_{\boldsymbol{P}(i,j), \boldsymbol{P}^{'}(i,j)})/2$, $d^{''}(\boldsymbol{u}, \boldsymbol{u}^{'}) = \sum_i \bar{\delta}_{u(i), u^{'}(i)}$, and $\bar{\delta}_{i,j} = 0$ when $i = j$, $\bar{\delta}_{i,j} = 1$ when $i \neq j$.

4. *Candidate list*
   $\forall X \in \Omega$, $C(X) \subseteq B(X)$, is composed of the top k better candidates among the neighbors of the current point with less E. The size of the candidate list $|C(X)| = k$ is a tunable parameter.
5. *Tabu tenets*, *tabu list* and *tabu tenure*
   Tabu tenets are implemented using tabu list and tabu tenure. Tabu list records the recent migrations of current point and forbids revisiting these points in tabu tenure.
6. *Aspiration criteria*
   Criterion1. If $E(X_{next}) < E_{min}$, then the next candidate X_{next} will be set as the current candidate $X_{current}$, even if X_{next} is in the tabu list.
   Criterion2. If the candidates in the candidate list are all tabu-active, then the best one among the candidates will be set as $X_{current}$.
7. *Termination criteria*
   Criterion1. If the number of iterations exceeds the maximum iteration limits, then the program stops.
   Criterion2. If the improvement $\triangle E$ for E_{min} is less than the improvement threshold ϵ, then the program stops.

Our iterative TS algorithm is summarized as follows:

Algorithm 5. *The optimization of PLUS factorization.*

```
Initialization: Randomly Choose an initial point $X$ in $\Omega$.
Set $X_{min} = X$, $E_{min} = E(X)$ and $i = 0$.
while no termination criteria are meet do
  $i = i + 1$
  Find $B(X)$ and $\forall X' \in B(X)$ calculate $E(X')$.
  Construct $C(X)$ and find $X_{next}$.
  if $X_{next}$ satisfies aspiration criteria then
    $X = X_{next}$.
    if $E(X) < E_{min}$ then
      $X_{min} = X, E_{min} = E(X)$.
    end if
  else if $X_{next}$ is not tabu-active, then
    $X = X_{next}$
    if $E(X) < E_{min}$ then
      $X_{min} = X, E_{min} = E(X)$.
    end if
```

```
        end if
        Update tabu list.
    end while
```

4. Experimental Results and Discussion

4.1. Examples of the Stable PLUS Factorization Algorithms

Example:

$$A = \begin{pmatrix} 4 & 3 & 2 & 0 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

The original general PLUS factorization algorithm stops in the first iteration due to $a_{14} = 0$.

With partial pivoting, the result of Algorithm 2 is:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 3 & -0.5 & 1 & 0 \\ 2 & -0.25 & 1.5 & 1 \end{pmatrix}$$

$$\cdot \begin{pmatrix} 1 & 1 & 0.33 & 4 \\ 0 & -1 & 0.67 & -16 \\ 0 & 0 & 1 & -18 \\ 0 & 0 & 0 & 18 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0.25 & 0.67 & 1 \end{pmatrix}$$

With complete pivoting, the result of Algorithm 4 is:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2.5 & 3.13 & 1 & 0 \\ 2 & 1.25 & 1.22 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2.5 & 1.97 & 4 \\ 0 & -1 & -0.94 & -8 \\ 0 & 0 & 1 & 18 \\ 0 & 0 & 0 & -18 \end{pmatrix}$$

$$\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & -0.38 & 0.007 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Remark: The differences between the two above factorizations are that the latter gives lower integer transform error, since complete pivoting results in elements with smaller magnitudes in factor matrices. E_2 for the first factorization is 17.03, and E_2 for the second factorization is 15.68.

4.2. Experiments for PLUS factorization optimization

DCT matrices are used in the experiments to exemplify the effectiveness of PLUS factorization optimization, due to the popularity of DCT in image and video coding. For each optimal PLUS factorization, 10000 randomly generated matrices are tested for the average transform error with Overall Mean Square Error (OMSE) and Overall Mean Error (OME):

$$OMSE = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{10000} e_k^2(i,j)}{n \times n \times 10000} \tag{9}$$

$$OME = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{10000} e_k(i,j)}{n \times n \times 10000} \tag{10}$$

where $e_k(i,j) = \hat{x}_k(i,j) - x_k(i,j)$, $x_k(i,j)$ are the coefficients of the randomly generated matrices after DCT transform, and $\hat{x}_k(i,j)$ are the coefficients after integer transform with the PLUS factor matrices of the DCT matrices.

Table 1: E(LUS), OMSE and OME of optimal factorizations for DCT matrices with exhaustive search

n	error metric	E(LUS)	OMSE	OME
	E_1	2.1907	0.3821	0.00015
2	E_2	1.7809	**0.1272**	**0.00011**
	E_∞	1.0205	0.3000	0.0003
	E_1	3.2134	0.3855	0.00015
4	E_2	2.8893	**0.1485**	**0.00011**
	E_∞	1.3205	0.3000	0.0003
	E_1	17.5834	0.4011	0.00016
8	E_2	4.6766	**0.1549**	**0.00011**
	E_∞	4.5655	0.3123	0.00049

n is the order of the DCT matrices.

The results in Table 1 reveal that:

- OMSE and OME of the global optimal factorizations found with error metrics defined using L_2 are less than those defined using L_1 or L_∞.

- Less E(LUS) is related to less OMSE and OME, and E(LUS) increases slowly with the orders of matrices.

Table 2: Some optimal factorizations for DCT found by exhaustive search

n	$\boldsymbol{P}_L$	$\boldsymbol{P}_R$	$\boldsymbol{u}$	E_2
2	1 2	1 2	1 −1	1.7809
	2 1	1 2	1 1	1.7809
4	2 0 4 3	1 2 3 4	1 1 1	2.8893
	4 3 2 1	2 4 1 3	−1 1 1	2.8893
	4 7 1 8 6 5 3 2	3 2 5 4 7 1 8 6	−1−1 1−1 1 1 1 1	4.6766
8	4 7 1 8 6 5 3 2	3 2 5 4 7 8 1 6	−1−11−1 1 1 1−1	4.6766
	2 3 1 4 8 5 76	1 4 6 3 5 2 7 8	1−1 1−1 1−1−1−1	4.6766
	2 3 1 4 8 5 7 6	1 4 6 3 5 7 2 8	1−1 1−1 1−1−1 1	4.6766

Table 3: Transform error E_2 of optimal factorizations found by TS

		t					
n	k	5	10	15	20	E_{ae}	E_{mse}
	4	**2.89**	**2.89**	**2.89**	**2.89**		
4	8	**2.89**	**2.89**	**2.89**	**2.89**	0	0
	11	**2.89**	**2.89**	**2.89**	**2.89**		
	4	**4.68**	4.81	4.82	4.78		
8	6	4.76	4.76	4.76	4.76	0.08	0.0016
	8	4.76	4.76	4.71	4.76		
	5	8.35	8.14	8.31	**7.94**		
16	8	8.19	8.23	8.12	8.23	—	0.011
	11	8.28	8.28	8.28	8.28		

k is the size of candidate list;
t is tabu tenure;
$E_{ae} = \frac{1}{N}\sum_{i=1}^{N} E(i) - E_{min}$;
E_{min} is E_2 in Table 2;
$E_{mse} = \frac{1}{N}\sum_{i=1}^{N}(E(i) - \overline{E})^2$;
$\overline{E}$ is the average error.

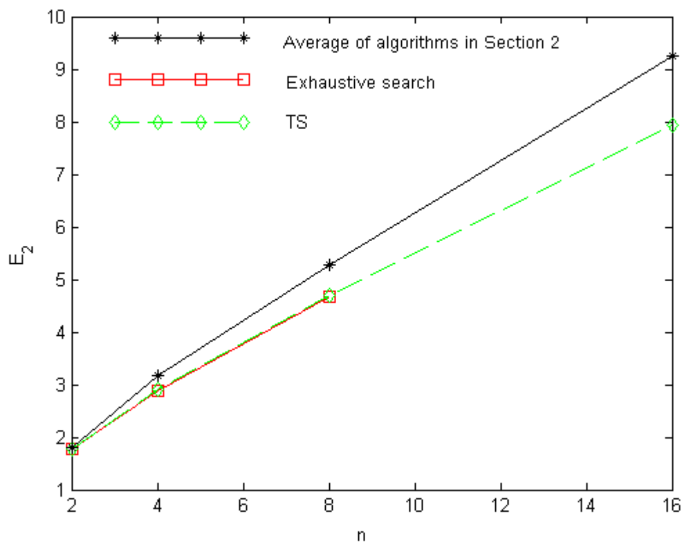


Figure 1: E_2 comparison between the factorizations found by three algorithms

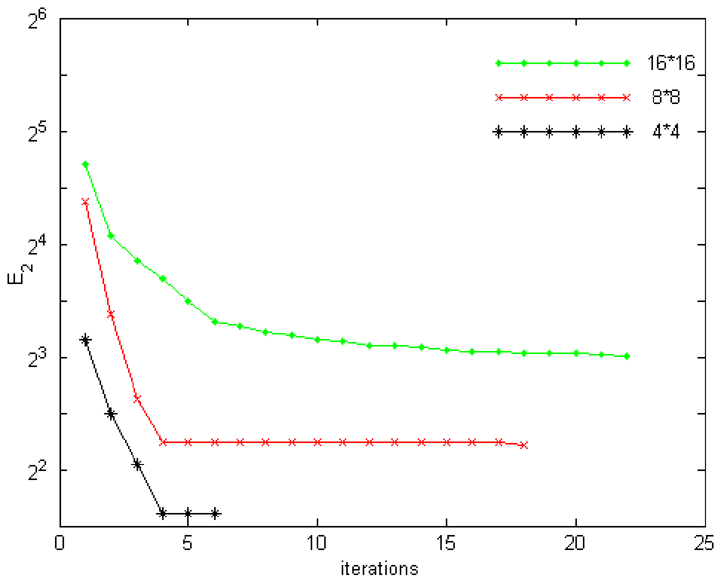


Figure 2: Convergence speed of optimization algorithm using TS

Based on the results in Table 3, Fig. 1 and Fig. 2, some discussions are given as follows:

- For $n = 4$, when the solution space of optimization is relatively small, our algorithm can always find the optima.
- For $n = 8$, when the solution space of optimization is expanded, our algorithm can find the global optima and other sub-optimal solutions. The transform errors of sub-optimal solutions are very close to that of the global optima.
- For $n = 16$, when the solution space of optimization is very large, our algorithm can find the sub-optimal solutions with little fluctuation. The sub-optimal solutions found with our fast TS method are all much better than randomly found ones and those found by any PLUS factorization algorithm.
- The convergence speed of our optimization algorithm using Tabu Search is very fast. In contrast to exhaustive search for weeks, each iteration for factorization of the 8×8 DCT matrix only costs 0.2ms on a PC with 0.7G CPU and 128M memory, and the sub-optimal solutions can be found in only a few iterations.
- When n is small, the PLUS factorizations obtained from algorithms in Section 2 are with little larger transform error. Thus, these algorithms are practical in the applications when n is small.

4.3. Optimal PLUS Factorization for Image Coding

In this section, we apply optimal PLUS factorization to lossless/lossy image coding. The integer pixel values are transformed into integer coefficients with the integer transforms under study. The integer transform implemented with PLUS factorization is illustrated in Fig. 4.

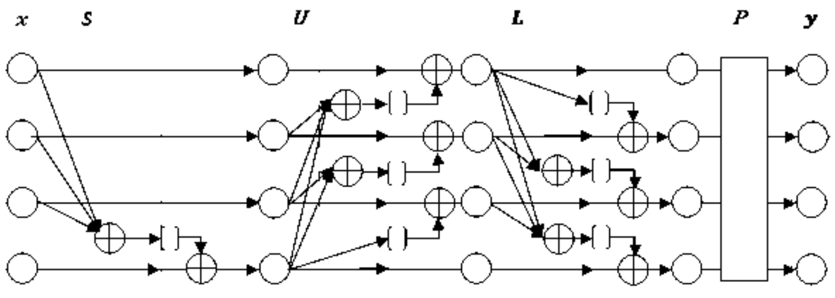


Figure 3: Flowchart of 4-point integer DCT implemented with PLUS. In the figure, [ ] denotes a round-off operation.

We compare the lossless coding performance of the integer transforms in terms of entropy $H(C)$, which is similar to coding data rate in unit of bits per pixel (bpp). (Note that the distortion caused by lossless coding is zero.) $H(C)$ is the average entropy of transform coefficients, given as below:

$$H(C) = \frac{1}{s}\sum_{i=1}^{s} H(C_i) \tag{11}$$

where $H(C_i)$ is the entropy of the transform coefficients of the i-th subband, and s is the total number of the subbands. For n-point 2 dimensional DCT, $s = n \times n$. Here we use entropy of transform coefficients instead of implementing an entropy coding scheme (e.g., Huffman code or arithmetic code) to code the resulting transform coefficients for two reasons. First, this paper focuses on matrix factorization for integer transform implementation; hence, we should compare the decorrelation performance of the resulting integer transforms, which is usually characterized by the average entropy defined in Eq. (11). Second, entropy coding for specific integer transforms is out of the scope of this paper; we will leave this for future study.

To compare the lossy coding performance of the integer transforms, we use bpp vs. Peak Signal-to-Noise Ratio (PSNR). In the lossy encoder that we implement, an input image is first transformed by an integer transform; then the integer transform coefficients are uniformly quantized; the quantized coefficients are coded with fixed length coding instead of variable length coding (or entropy coding), since entropy coding is out of the scope of this paper. Finally, the decoder reconstructs the image.

4.3.1. Integer DCT with PLUS Factorization

We apply the optimal PLUS factorization of DCT found by our proposed algorithm to lossless image coding. The transform matrices are 2-point, 3-point and 4-point DCTs. Their integer reversible implementation by optimal PLUS factorization are denoted as 'Opt2', 'Opt3' and 'Opt4' respectively. Their integer reversible implementation by expansion factors [29] are denoted as 'IntDCT2', 'IntDCT3' and 'IntDCT4'. Table 4 shows the entropy of transform coefficients obtained by our proposed optimal PLUS factorization schemes and integer DCT with expansion factors. The entropy obtained by our algorithm is less than that obtained by integer DCT with expansion factors for all test images.

Table 4: Entropy comparison of Integer DCT with expansion factors [29] and optimal PLUS factorization

Image	IntDCT2	Opt2	IntDCT3	Opt3	IntDCT4	Opt4
Barbara	6.94	5.95	8.02	6.65	6.92	5.57
Lena	6.37	5.38	7.43	5.97	6.36	5.03
Boat	6.41	5.42	7.46	6.06	6.39	5.12
Jet	5.89	5.11	6.89	5.72	5.87	4.95
Mandrill	7.59	6.59	8.66	7.00	7.57	6.62
Goldhill	6.69	5.70	7.78	6.23	6.68	5.43
Average	6.65	5.69	7.71	6.27	6.63	5.45

The corresponding optimal PLUS factorizations can be found in Appendix C.

4.3.2. Integer Lapped Transform with PLUS Factorization

We also apply optimal PLUS factorization to make Lapped Transform [31, 32, 33, 34, 37] integer reversible. The Photo Core Transform (PCT) and Photo Overlap Transform (POT) are defined in [30, 35, 36]. We obtain the optimal PLUS factorization for 4-point POT and 4-point DCT, and apply it to lossy/lossless image coding, which is denoted as 'PLUS_1'. We also apply optimal PLUS factorization to 4-point POT and 4-point PCT which is an approximation of DCT. This coding scheme is denoted as 'PLUS_2'. The optimal PLUS factorization scheme 'PLUS_1' and 'PLUS_2' can be found in

Appendix D. The lifting factorization scheme in JPEG-XR [30, 36] is denoted as 'JPEG-XR'. The entropy performance, as well as lossy performance of 'PLUS_1' and 'PLUS_2' is better than that of 'JPEG-XR', as shown in the Table 5, Table 6 and Fig. 5. For example, for image Lena, at the bit rate of 0.25 bpp, 'PLUS_1' achieves 2.5dB gain in PSNR than 'JPEG-XR'. In addition, the performance of 'PLUS_1' is better than that of 'PLUS_2', which means that the DCT implemented in 'PLUS_1' has higher decorrelation ability than its approximation in 'PLUS_2'.

Table 5: Entropy comparison of Integer Lapped Transforms with JPEG-XR, PLUS_1 and PLUS_2

Image	JPEG-XR	PLUS_1	PLUS_2
Lena	4.93	4.48	4.61
Baboon	6.34	6.11	6.22
Barbara	5.64	4.98	5.29
Boat	5.10	4.54	4.71
Goldhill	5.27	4.92	5.01
Peppers	5.09	4.76	4.83
Average	5.40	4.97	5.11

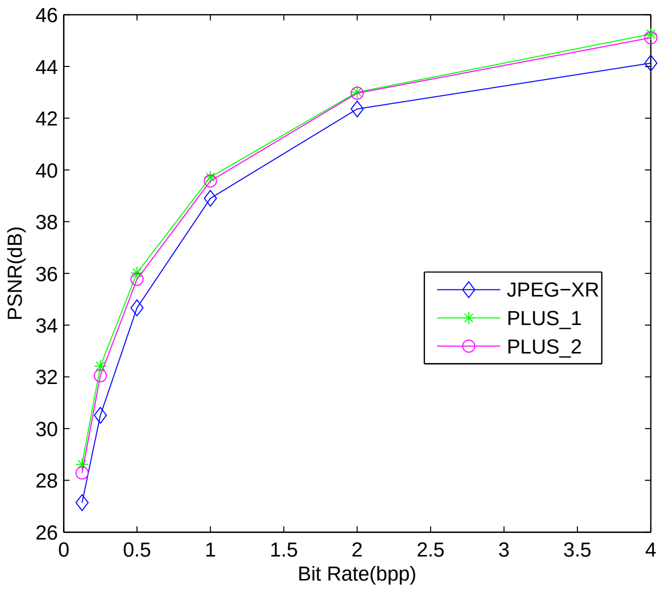
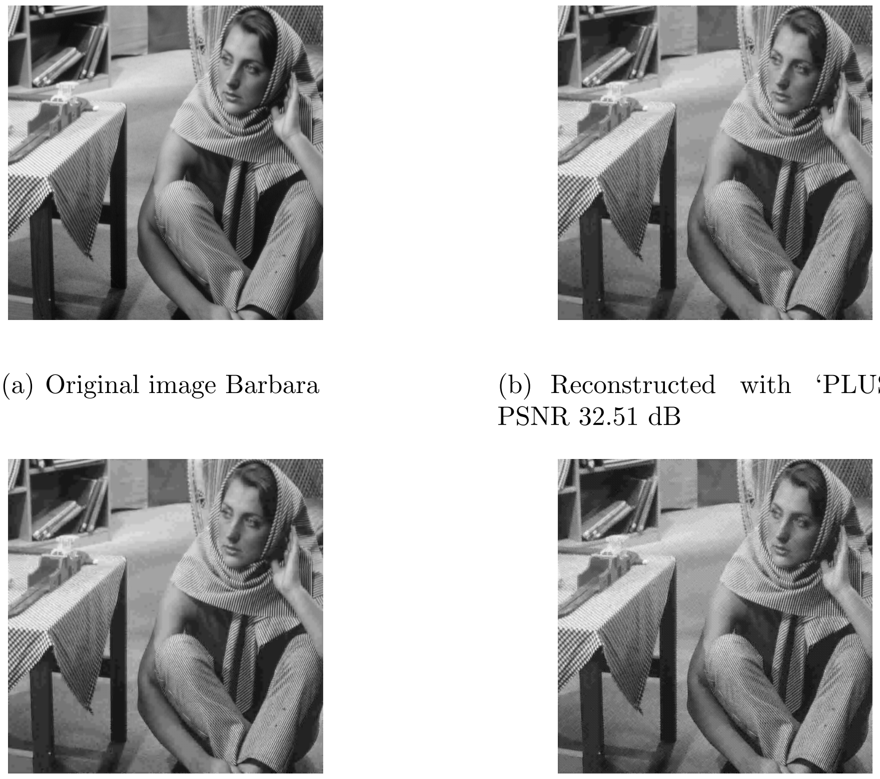


Figure 4: Average bpp vs. PSNR with integer transforms for test images

Table 6: Bpp vs. PSNR comparison of Integer Lapped Transforms with JPEG-XR, PLUS_1 and PLUS_2

Image	bpp	JPEG-XR	PLUS_1	PLUS_2
Lena	4	44.14	45.17	45.03
	2	42.35	42.97	42.93
	1	38.90	39.81	39.69
	0.5	35.16	36.95	36.69
	0.25	31.45	33.94	33.52
	0.125	28.46	30.10	29.78
Barbara	4	44.13	45.20	45.06
	2	42.36	42.97	42.93
	1	38.89	39.96	39.73
	0.5	34.68	36.54	36.02
	0.25	30.34	32.51	31.78
	0.125	26.43	28.14	27.42
Baboon	4	44.14	45.42	45.30
	2	42.39	43.12	43.06
	1	38.85	39.28	39.21
	0.5	33.89	34.31	34.24
	0.25	28.85	29.52	29.30
	0.125	24.46	25.39	25.06
Boat	4	44.09	45.14	44.97
	2	42.31	42.92	42.91
	1	39.02	40.22	40.05
	0.5	35.26	37.01	36.67
	0.25	30.91	33.14	32.74
	0.125	27.40	29.12	28.85
Goldhill	4	44.13	45.24	45.09
	2	42.37	43.01	42.96
	1	38.88	39.55	39.41
	0.5	34.42	35.64	35.45
	0.25	30.38	32.11	31.86
	0.125	27.51	28.79	28.63
Peppers	4	44.15	45.33	45.17
	2	42.35	43.03	43.00
	1	38.87	39.48	39.33
	0.5	34.59	35.67	35.57
	0.25	31.17	33.27	33.09
	0.125	28.57	30.19	29.99

We also compare the subjective performance of 'JPEG-XR', 'PLUS_1' and 'PLUS_2' in terms of the visual quality of the reconstructed image 'Barbara' in Fig. 5. The 'Barbara' reconstructed by 'PLUS_1' has the best visual quality, and then the 'PLUS_2', and the worst is 'JPEG-XR'.



(a) Original image Barbara

(b) Reconstructed with 'PLUS_1' PSNR 32.51 dB

(c) econstructed with 'PLUS_2' PSNR 31.78 dB

(d) Reconstructed with 'JPEG-XR' PSNR 30.33 dB

Figure 5: Lossy subjective performance comparison of 'JPEG-XR', 'PLUS_1' and 'PLUS_2' at 0.25 bpp

5. Conclusion

In this paper, we have addressed two problems of PLUS factorization: stabilization and optimization. We proposed stable algorithms for PLUS factorization by pivoting and back tracing strategies, and presented the stability theorem. We proposed a fast optimization algorithm for PLUS factorization with Tabu Search to find the optimal PLUS factorizations with the least transform error. The experiments in lossy and lossless image coding demonstrated that the optimal PLUS factorization achieves superior performance in implementing integer reversible transforms for transform matrices of any size. The optimal PLUS factorization for progressive lossy transform coding and new integer transform design will be explored in our future work.

Appendix A. The Proof of Theorem 2

Proof: (by induction)
We just need to prove that: if $\boldsymbol{P}^T\boldsymbol{A} = \boldsymbol{LUS} = \boldsymbol{L}^{'}\boldsymbol{U}^{'}\boldsymbol{S}^{'}$, then $\boldsymbol{L} = \boldsymbol{L}^{'}, \boldsymbol{U} = \boldsymbol{U}^{'}, \boldsymbol{S} = \boldsymbol{S}^{'}$ ($\boldsymbol{S}$ is the single-row elementary reversible matrix with nonzeros in the last row, s_i is the ith entry in the last row of $\boldsymbol{S}$, $\boldsymbol{S}$ with other patterns could be proved similarly).

1. Firstly, we prove $\boldsymbol{S} = \boldsymbol{S}^{'}$
   Because $\det((\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{S}^{-1})^{(k)}) = \det((\boldsymbol{P}^T\boldsymbol{A}\ \boldsymbol{S}^{'-1})^{(k)}) = \det((\boldsymbol{LU})^{(k)}) = \prod_{i=1}^{k} u_i,\ k = 1, \cdots, n-1$, we use induction to prove $s_i = s_i^{'}, i = 1, \cdots, n-1$.
   1° When i=1, $a_{11} - s_1 a_{1n} = a_{11} - s_1^{'} a_{1n}$ and $a_{1n} \neq 0$, so $s_1 = s_1^{'}$.
   2° Assume that $s_i = s_i^{'}$ holds, when $i < t$. When $i = t$,
   $\det((\boldsymbol{P}^T[\boldsymbol{a}_1 - s_1\boldsymbol{a}_n, \boldsymbol{a}_2 - s_2\boldsymbol{a}_n, \cdots, \boldsymbol{a}_{t-1} - s_{t-1}\boldsymbol{a}_n, \boldsymbol{a}_t])^{(t)}) - s_t\det((\boldsymbol{P}^T[\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_{t-1}, \boldsymbol{a}_n])^{(t)}) = \det((\boldsymbol{P}^T[\boldsymbol{a}_1 - s_1\boldsymbol{a}_n, \boldsymbol{a}_2 - s_2\boldsymbol{a}_n, \cdots, \boldsymbol{a}_{t-1} - s_{t-1}\boldsymbol{a}_n, \boldsymbol{a}_t])^{(t)}) - s_t^{'}\det((\boldsymbol{P}^T\ [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_{t-1}, \boldsymbol{a}_n])^{(t)})$, so $s_t = s_t^{'}$.
   3° To sum up, $s_i = s_i^{'}, i = 1, \cdots, n-1$ hold, i.e. $\boldsymbol{S} = \boldsymbol{S}^{'}$.
2. Then we prove $\boldsymbol{L} = \boldsymbol{L}^{'}$, $\boldsymbol{U} = \boldsymbol{U}^{'}$.
   Because $\boldsymbol{LU} = \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{S}^{-1} = \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{S}^{'-1} = \boldsymbol{L}^{'}\boldsymbol{U}^{'}$, we can have $\boldsymbol{L}^{'-1}\boldsymbol{L} = \boldsymbol{U}^{'}\boldsymbol{U}^{-1}$. Because $\boldsymbol{L}^{'-1}\boldsymbol{L}$ is a lower triangular matrix, $\boldsymbol{U}^{'}\boldsymbol{U}^{-1}$ is a upper triangular matrix, we have $\boldsymbol{L}^{'-1}\boldsymbol{L} = \boldsymbol{U}^{'}\boldsymbol{U}^{-1} = \boldsymbol{I}$. Finally, we have $\boldsymbol{L}^{'} = \boldsymbol{L}$ and $\boldsymbol{U}^{'} = \boldsymbol{U}$. ■

Appendix B. Perturbation Analysis

Assume that a nonsingular matrix $\boldsymbol{A} \in R^{n\times n}$ has unique PLUS factorization, $\boldsymbol{A} = \boldsymbol{LUS}$. Let $\Delta\boldsymbol{A} \in R^{n\times n}$ be a perturbation such that $\boldsymbol{A} + \Delta\boldsymbol{A}$ also has unique PLUS factorization:

$$\boldsymbol{A} + \Delta\boldsymbol{A} = (\boldsymbol{L} + \Delta\boldsymbol{L})(\boldsymbol{U} + \Delta\boldsymbol{U})(\boldsymbol{S} + \Delta\boldsymbol{S}) \tag{12}$$

Note that $\boldsymbol{P}$ is not considered, because the analysis of the sensitivity of general PLUS factorization algorithm is simpler and without much loss of generality. The measure of $\Delta\boldsymbol{L}$, $\Delta\boldsymbol{U}$ and $\Delta\boldsymbol{S}$ is deduced as following.

We use the matrix-vector equation approach [22], to present perturbation analysis for PLUS factorization. Compared with Chang's perturbation analysis for LU factorization,

1. The pattern of $\Delta\boldsymbol{U}$ in PLUS factorization is different from that of LU factorization.
2. $\Delta\boldsymbol{S}$ increases the analysis complexity.

We use the following notations for the perturbation analysis. $\boldsymbol{A}(t)$ represents matrix function, and $\dot{\boldsymbol{A}}(t)$ represents its derivative. For any matrix $\boldsymbol{A} \in R^{n\times n}$, $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n]$, $\boldsymbol{a}_i$ is the ith column vector of $\boldsymbol{A}$, and $vec(\boldsymbol{A}) \in R^{n^2\times 1}$ is the vector form of $\boldsymbol{A}$ with $\boldsymbol{a}_i$ in succession. $\widetilde{\boldsymbol{L}} \in R^{n(n-1)/2\times 1}$ is composed of elements in nonzero locations of $\dot{\boldsymbol{L}}(0)$, $\widetilde{\boldsymbol{U}} \in R^{(n^2-n+2)/2\times 1}$ is composed of elements in nonzero locations of $\dot{\boldsymbol{U}}(0)$, and $\widetilde{\boldsymbol{S}} \in R^{(n-1)\times 1}$ is composed of elements in nonzero locations of $\dot{\boldsymbol{S}}(0)$.

Lemma 1. *Assume that a nonsingular matrix $\boldsymbol{A} \in R^{n\times n}$ has unique PLUS factorization, $\boldsymbol{A} = \boldsymbol{LUS}$. Let $\Delta\boldsymbol{A} \in R^{n\times n}$, $\Delta\boldsymbol{A} = \varepsilon\boldsymbol{E}$, where ε is small enough, such that $\boldsymbol{A} + t\boldsymbol{E}$ satisfies Condition 1 for all $|t| \leq \varepsilon$, then $\boldsymbol{A} + t\boldsymbol{E}$ has unique PLUS factorization:*

$$\boldsymbol{A} + t\boldsymbol{E} = \boldsymbol{L}(t)\boldsymbol{U}(t)\boldsymbol{S}(t), |t| \leq \varepsilon, \tag{13}$$

which leads to

$$\dot{\boldsymbol{L}}(0)\boldsymbol{US} + \boldsymbol{L}\dot{\boldsymbol{U}}(0)\boldsymbol{S} + \boldsymbol{LU}\dot{\boldsymbol{S}}(0) = \boldsymbol{E}. \tag{14}$$

For $t = \varepsilon$, we obtain $\boldsymbol{A} + \Delta\boldsymbol{A}$ with the unique PLUS factorization

$$\boldsymbol{A} + \Delta\boldsymbol{A} = (\boldsymbol{L} + \Delta\boldsymbol{L})(\boldsymbol{U} + \Delta\boldsymbol{U})(\boldsymbol{S} + \Delta\boldsymbol{S}), \tag{15}$$

where $\Delta\boldsymbol{L}$, $\Delta\boldsymbol{U}$ and $\Delta\boldsymbol{S}$ satisfy

$$\Delta\boldsymbol{L} = \varepsilon\dot{\boldsymbol{L}}(0) + O(\varepsilon^2) \tag{16a}$$

$$\Delta\boldsymbol{U} = \varepsilon\dot{\boldsymbol{U}}(0) + O(\varepsilon^2) \tag{16b}$$

$$\Delta\boldsymbol{S} = \varepsilon\dot{\boldsymbol{S}}(0) + O(\varepsilon^2) \tag{16c}$$

It can be easily proved using Taylor expansion in the similar way as in [22]. Note that $\boldsymbol{L}(0) = \boldsymbol{L}$, $\boldsymbol{L}(\varepsilon) = \boldsymbol{L} + \Delta\boldsymbol{L}$, $\boldsymbol{U}(0) = \boldsymbol{U}$, $\boldsymbol{U}(\varepsilon) = \boldsymbol{U} + \Delta\boldsymbol{U}$, $\boldsymbol{S}(0) = \boldsymbol{S}$, and $\boldsymbol{S}(\varepsilon) = \boldsymbol{S} + \Delta\boldsymbol{S}$.

From Equation (14), we obtain

$$\boldsymbol{e}_i = \dot{\boldsymbol{L}}(0)\boldsymbol{U}\boldsymbol{s}_i + \boldsymbol{L}\dot{\boldsymbol{U}}(0)\boldsymbol{s}_i + \boldsymbol{L}\boldsymbol{U}\dot{\boldsymbol{s}}(0)_i, i = 1, \cdots, n. \tag{17}$$

By rearranging equation (17), we obtain a matrix-vector equation:

$$\begin{pmatrix} \boldsymbol{W}_L & \boldsymbol{W}_U & \boldsymbol{W}_S \end{pmatrix} \begin{pmatrix} \widetilde{\boldsymbol{L}} \\ \widetilde{\boldsymbol{U}} \\ \widetilde{\boldsymbol{S}} \end{pmatrix} = vec(\boldsymbol{E}) \tag{18}$$

where $\boldsymbol{W}_L \in R^{n^2 \times n(n-1)/2}$ is composed of $n \times (n-1)$ sub-matrices with following pattern:

$$\begin{pmatrix} \boldsymbol{W}_{L_{1,1}} & \boldsymbol{W}_{L_{1,2}} & \cdots & \boldsymbol{W}_{L_{1,n-2}} & \boldsymbol{W}_{L_{1,n-1}} \\ \boldsymbol{W}_{L_{2,1}} & \boldsymbol{W}_{L_{2,2}} & \cdots & \boldsymbol{W}_{L_{2,n-2}} & \boldsymbol{W}_{L_{2,n-1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{W}_{L_{n-1,1}} & \boldsymbol{W}_{L_{n-1,2}} & \cdots & \boldsymbol{W}_{L_{n-1,n-2}} & \boldsymbol{W}_{L_{n-1,n-1}} \\ \boldsymbol{W}_{L_{n,1}} & \boldsymbol{W}_{L_{n,2}} & \cdots & \boldsymbol{W}_{L_{n,n-2}} & \boldsymbol{W}_{L_{n,n-1}} \end{pmatrix}$$

$\boldsymbol{W}_{L_{ij}} \in R^{n \times (n-j)}, 1 \leq i, j \leq n,$

$$\boldsymbol{W}_{L_{ij}} = \begin{cases} (s_i u_{jn} + u_{ji}) \begin{pmatrix} 0 \\ \boldsymbol{I}_{n-j} \end{pmatrix}, i < n & \text{(19a)} \\ u_{jn} \begin{pmatrix} 0 \\ \boldsymbol{I}_{n-j} \end{pmatrix}, \qquad i = n & \text{(19b)} \end{cases}$$

$\boldsymbol{W}_U \in R^{n^2 \times (n^2-n+2)/2}$ is composed of $n \times (n-1)$ sub-matrices. Each sub-matrix has n rows, and the sub-matrices in the ith ($i = 1, \cdots, n-2$) column block have i columns, the sub-matrices in the $(n-1)$th column block have n columns, with the following pattern:

$$\begin{pmatrix} \boldsymbol{0}_{n\times 1} & & & & s_1\boldsymbol{L} \\ \boldsymbol{L}\boldsymbol{1}_{n\times 1} & & & & s_2\boldsymbol{L} \\ & \boldsymbol{L}\boldsymbol{1}_{n\times 2} & & & s_3\boldsymbol{L} \\ & & \ddots & & \vdots \\ & & & \boldsymbol{L}\boldsymbol{1}_{n\times(n-2)} & s_{n-1}\boldsymbol{L} \\ & & & & \boldsymbol{L} \end{pmatrix}$$

$\boldsymbol{W}_S \in R^{n^2\times(n-1)}$ is a diagonal block matrix, composed of the same $n \times (n-1)$ sub-matrices, $\boldsymbol{Lu}_n \in R^{n\times 1}$, with the following pattern:

$$\begin{pmatrix} \boldsymbol{Lu}_n & & & \\ & \boldsymbol{Lu}_n & & \\ & & \ddots & \\ & & & \boldsymbol{Lu}_n \end{pmatrix}$$

Obviously, by fundamental matrix transform, $[\boldsymbol{W}_L, \boldsymbol{W}_U, \boldsymbol{W}_S]$ can be transformed into a lower triangular matrix with $(\underbrace{a_{1n}, u_1, \cdots, u_1}_{n}, \underbrace{1, 1, u_2, \cdots, u_2}_{n}, \underbrace{1, 1, 1, u_3, \cdots, u_3}_{n}, \cdots, \underbrace{1, \cdots, 1, u_{n-1}}_{n}, \underbrace{1, \cdots, 1}_{n})$ as the diagonal entries. Therefore, $\boldsymbol{W} = [\boldsymbol{W}_L, \boldsymbol{W}_U, \boldsymbol{W}_S]$ is invertible. Let

$$\boldsymbol{W}^{-1} = \begin{pmatrix} \boldsymbol{Y}_L \\ \boldsymbol{Y}_U \\ \boldsymbol{Y}_S \end{pmatrix},$$

then we obtain

$$\begin{pmatrix} \widetilde{\boldsymbol{L}} \\ \widetilde{\boldsymbol{U}} \\ \widetilde{\boldsymbol{S}} \end{pmatrix} = \boldsymbol{W}^{-1} vec(\boldsymbol{E}) = \begin{pmatrix} \boldsymbol{Y}_L \\ \boldsymbol{Y}_U \\ \boldsymbol{Y}_S \end{pmatrix} vec(\boldsymbol{E}) \tag{20}$$

$$\|\dot{\boldsymbol{L}}(0)\|_F = \|\widetilde{\boldsymbol{L}}\|_2 \leq \|\boldsymbol{Y}_L\|_F \|vec(\boldsymbol{E})\|_2 = \|\boldsymbol{Y}_L\|_F \; \|\boldsymbol{E}\|_F \tag{21a}$$

$$\|\dot{\boldsymbol{U}}(0)\|_F = \|\widetilde{\boldsymbol{U}}\|_2 \leq \|\boldsymbol{Y}_U\|_F \|vec(\boldsymbol{E})\|_2 = \|\boldsymbol{Y}_U\|_F \; \|\boldsymbol{E}\|_F \tag{21b}$$

$$\|\dot{\boldsymbol{S}}(0)\|_F = \|\widetilde{\boldsymbol{S}}\|_2 \leq \|\boldsymbol{Y}_S\|_F \|vec(\boldsymbol{E})\|_2 = \|\boldsymbol{Y}_S\|_F \; \|\boldsymbol{E}\|_F \tag{21c}$$

$$\frac{\|\Delta \boldsymbol{L}\|_F}{\| \boldsymbol{L} \|_F} \leq \frac{\|\boldsymbol{Y}_L\|_F \|\boldsymbol{E}\|_F}{\| \boldsymbol{L} \|_F} \varepsilon + O(\varepsilon^2) = \kappa_L(\boldsymbol{A}) \frac{\| \Delta \boldsymbol{A} \|_F}{\| \boldsymbol{A} \|_F} + O(\varepsilon^2) \tag{22a}$$

$$\frac{\|\Delta \boldsymbol{U}\|_F}{\| \boldsymbol{U} \|_F} \leq \frac{\|\boldsymbol{Y}_U\|_F \|\boldsymbol{E}\|_F}{\| \boldsymbol{U} \|_F} \varepsilon + O(\varepsilon^2) = \kappa_U(\boldsymbol{A}) \frac{\| \Delta \boldsymbol{A} \|_F}{\| \boldsymbol{A} \|_F} + O(\varepsilon^2) \tag{22b}$$

$$\frac{\|\Delta \boldsymbol{S}\|_F}{\| \boldsymbol{S} \|_F} \leq \frac{\|\boldsymbol{Y}_S\|_F \|\boldsymbol{E}\|_F}{\| \boldsymbol{S} \|_F} \varepsilon + O(\varepsilon^2) = \kappa_S(\boldsymbol{A}) \frac{\| \Delta \boldsymbol{A} \|_F}{\| \boldsymbol{A} \|_F} + O(\varepsilon^2) \tag{22c}$$

where

$$\kappa_L(\boldsymbol{A}) = \| \boldsymbol{Y}_L \|_F \| \boldsymbol{A} \|_F \, / \, \| \boldsymbol{L} \|_F$$
$$\kappa_U(\boldsymbol{A}) = \| \boldsymbol{Y}_U \|_F \| \boldsymbol{A} \|_F \, / \, \| \boldsymbol{U} \|_F$$
$$\kappa_S(\boldsymbol{A}) = \| \boldsymbol{Y}_S \|_F \| \boldsymbol{A} \|_F \, / \, \| \boldsymbol{S} \|_F$$

Theorem 3. *Perturbation analysis II.*
Assume that $\boldsymbol{A}$ *and* $\boldsymbol{A}+\Delta\boldsymbol{A}$ *are both nonsingular and their PLUS factorizations exist:* $\boldsymbol{A}=\boldsymbol{LUS}$ *and* $\boldsymbol{A}+\Delta\boldsymbol{A}=(\boldsymbol{L}+\Delta\boldsymbol{L})(\boldsymbol{U}+\Delta\boldsymbol{U})(\boldsymbol{S}+\Delta\boldsymbol{S})$, *then Equations (20) (21a) (21b) (21c) (22a) (22b) (22c) hold.*

Perturbation analysis example:

The original PLUS factorization is:

$$\begin{aligned}\boldsymbol{A}&=\boldsymbol{LUS}\\ &=\begin{pmatrix}0.8913 & 0.4565\\ 0.7621 & 0.0185\end{pmatrix}\\ &=\begin{pmatrix}1 & 0\\ 0.7665 & 1\end{pmatrix}\begin{pmatrix}1 & 0.4565\\ 0 & -0.3314\end{pmatrix}\begin{pmatrix}1 & 0\\ -0.2381 & 1\end{pmatrix}\end{aligned}$$

With a perturbation $\Delta\boldsymbol{A}$, the PLUS factorization is:

$$\begin{aligned}\boldsymbol{A}+\Delta\boldsymbol{A}&=(\boldsymbol{L}+\Delta\boldsymbol{L})(\boldsymbol{U}+\Delta\boldsymbol{U})(\boldsymbol{S}+\Delta\boldsymbol{S})\\ &=\begin{pmatrix}0.8913 & 0.4565\\ 0.7621 & 0.0185\end{pmatrix}+\begin{pmatrix}0.009 & 0.001\\ 0.007 & 0.004\end{pmatrix}\\ &=\begin{pmatrix}1 & 0\\ 0.7746 & 1\end{pmatrix}\begin{pmatrix}1 & 0.4575\\ 0 & -0.3316\end{pmatrix}\begin{pmatrix}1 & 0\\ -0.2179 & 1\end{pmatrix}\end{aligned}$$

Therefore, the perturbation error is limited.

Appendix C. The optiaml PLUS factorizations for DCT

The PLUS factorization 'Opt2' for 2-point DCT C_2^{II}:

$$\boldsymbol{C}_2^{II}=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\ 0.4142 & 1\end{pmatrix}\begin{pmatrix}1 & -0.7071\\ 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\ 0.4142 & 1\end{pmatrix}$$

The PLUS factorization 'Opt3' for 3-point DCT C_3^{II}:

$$\begin{aligned}\boldsymbol{C}_3^{II}=&\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0.3382 & 1 & 0\\ 0.2391 & -0.5176 & 1\end{pmatrix}\\ &\begin{pmatrix}1 & -0.3660 & -0.7071\\ 0 & 1 & 0.8165\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0.4142 & -0.5176 & 1\end{pmatrix}\end{aligned}$$

The PLUS factorization 'Opt4' for 4-point DCT C_4^{II}:

$$\boldsymbol{C}_4^{II} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.3827 & 1 & 0 & 0 \\ -0.9239 & -0.6682 & 1 & 0 \\ 0 & 0.3318 & 0.6934 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -0.3318 & 0.3318 & -0.5 \\ 0 & 1 & -0.0761 & -0.4619 \\ 0 & 0 & 1 & -0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0.3364 & -0.3364 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Appendix D. The optimal PLUS factorizations for Lapped Transform

For scheme 'PLUS_1', it is composed of optimal PLUS factorization of POT and optimal PLUS factorization of DCT. For scheme 'PLUS_2', it is composed of optimal PLUS factorization of POT and optimal PLUS factorization of the approximation of DCT.

The optimal PLUS factorization of POT is:

$$\begin{pmatrix} -0.1448 & 0.2313 & -0.2313 & 0.9720 \\ 0.2313 & 0.9720 & -0.1448 & -0.2313 \\ -0.2313 & -0.1448 & 0.9720 & 0.2313 \\ 0.9720 & -0.2313 & 0.2313 & -0.1448 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.276 & 1 & 0 & 0 \\ -0.276 & -0.173 & 1 & 0 \\ -0.333 & 0.328 & -0.085 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -0.2585 & 0.2311 & -0.1448 \\ 0 & 1 & -0.2089 & -0.1913 \\ 0 & 0 & 1 & 0.1583 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0.3364 & -0.3364 & 1 \end{pmatrix}$$

The optimal PLUS factorization of DCT is:

$$\begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2346 & 1 & 0 & 0 \\ 0.4142 & -0.7654 & 1 & 0 \\ 0.2346 & 0 & -0.6934 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -0.2929 & -0.0137 & -0.6533 \\ 0 & 1 & 0.3066 & 0.6533 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5307 & -0.8626 & 0.3933 & 1 \end{pmatrix}$$

The optimal PLUS factorization of the approximation of DCT is:

$$\begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.7071 & 0 & 0 & -0.7071 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0 & 0.7071 & -0.7071 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2929 & 1 & 0 & 0 \\ 0.2929 & 0 & 1 & 0 \\ 0 & 0.7071 & -2.1213 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -0.5 & -1.5 & -0.7071 \\ 0 & 1 & 2 & 0.7071 \\ 0 & 0 & 1 & 0.7071 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.4142 & -0.7071 & -2.1213 & 1 \end{pmatrix}$$

References

[1] http://www.jpeg.org/

[2] http://www.mpeg.org/

[3] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. on Circuits and Systems for Video Technology, 13(7)(2003), 560-576.

[4] N. Ahmed, T. Natarajan, K. R. Rao, Discrete cosine transform, IEEE Trans. Computers, C-23(1974), 90-93.

[5] S.Mallat, A wavelet tour of signal processing, Second Edition, Academic Press, Sep. 15, 1999.

[6] M.D. Adams, Generalized reversible integer-to-Integer transform framework, in: Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, Aug. 2003, pp. 569-572.

[7] P. Hao, Customizable triangular factorizations of matrices, Linear Algebra Appl., 382(2004), 135-154.

[8] P. Hao, Q. Shi, Matrix factorizations for reversible integer mapping, IEEE Trans. Signal Processing, 49(1)(2000), 2314-2324.

[9] F. Sheng, A. Bilgin, P.J. Sementilli, M.W. Marcellin, Lossy and lossless image compression using reversible integer wavelet transform, in: Proc. of IEEE International Conference on Image Processing 1998, 3(1998), pp. 876-880.

[10] M.Iwahashi, S. Fukuma, S. Chokchaitam, N. Kambayashi, Lossless/Lossy progressive coding based on reversible wavelet and lossless multichannel prediction, In: Proc. of IEEE IEEE International Conference on Image Processing 1999, 1(1999), pp. 430-434.

[11] G. Strang, T. Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press, 1997.

[12] Ingrid Daubechies, Wim Sweldens, Factoring wavelet transforms into lifting steps, J. Fourier Anal. Appl, 4(1998), 247-269, .

[13] B. Chen, A. Kaufman, 3D volume rotation using shear transformations, Graphical Models, 62(2000), 308-322.

[14] Y. Chen, P. Hao, C. Zhang, Shear-resize factorizations for fast image registration, in: Proc. of IEEE International Conference on Image Processing 2005, 3(2005), pp. 1120-1123.

[15] L. Yang, P. Hao, Infinity-Norm Rotation for Reversible Data Hiding, In: Proc. of IEEE International Conference on Image Processing 2007, 3(2007), pp. III245-III248.

[16] L. Condat, H.Z. Munchen, D. Van De Ville, Fully reversible image rotation by 1-D filtering, In: Proc. of IEEE International Conference on Image Processing 2008, 1(2008), pp. 913-916.

[17] Y. She, P. Hao, On the necessity and sufficiency of PLUS factorizations, Linear Algebra Appl., 400(2005), 193-202.

[18] Y. She, P. Hao, Y. Paker, Matrix factorizations for parallel integer transforms, IEEE Transactions on Signal Processing, 54(12)(2006), 4675-4684.

[19] Y. She, P. Hao, Block TERM factorization of uniform block matrices, Science in China (Series F), 47(4)(2004), 421-436.

[20] G. Strang, Every unit matrix is a LULU, Linear Algebra Appl., 265(1997), 165-172.

[21] T. Toffoli, Almost every unit matrix is a ULU, Linear Algebra Appl., 259(1997), 31-38.

[22] X.W. Chang, C.C. Paige, On the sensitivity of the LU factorization, BIT Numerical Mathematics, Springer Netherlands, 38(1998), 486-501.

[23] F. Glover, Tabu search: a tutorial, CAAI Report, University of Colorado, Boulder, 1990, pp. 1-47.

[24] A. Galantai, Perturbations of triangular matrix factorizations, Linear and Multilinear Algebra, 51(2)(2003), 175-198.

[25] L. Wang, Intelligent Optimization Algorithms with Applications, ed. 1, TUP & Springer, Beijing, 2001.

[26] G.W. Steward, J.-G. Sun, Matrix perturbation theory, ed. 2, Academic Press, San Diego, 1990.

[27] G.H. Golub, C.F. Van Loan, Matrix Computations, The Johns Hopkins University Press, London, 1996.

[28] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.

[29] G.Plonka, A global method for invertible integer DCT and integer wavelet algorithms, Appl. Comput. Harmon. Anal., 16(2004), 90-110.

[30] G.J. Sullivan, S. Regunathan, A. Gill, Text of ISO/IEC FCD 29199-2 (JPEG XR image coding - Specification), ISO/IEC JTC 1/SC 29/WG1 N 4739, 14 September 2008.

[31] H.S. Malvar, D.H. Staelin, The LOT:transform coding without blocking effects, IEEE Trans. on Acoustics, Speech, and Signal Processing, 37(1989), 553-559.

[32] H.S. Malvar, Lapped biorthogonal transforms for transform coding with reduced blocking and ringing artifacts, in: Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1997, 3(1997), pp. 2421-2424.

[33] P.M. Cassereau, D.H. Staelin, G. de Jager, Encoding of images based on a lapped orthogonal transform, IEEE Trans. Comm., 37(1989), 189-193.

[34] T.D. Tran, J. Liang, C. Tu, Lapped transform via time-domain pre- and post-filtering, IEEE Trans. on Signal Processing, 51(6)(2003), 1557-1571.

[35] T.D. Tran, The liftLT: fast-lapped transforms via lifting steps", IEEE trans. Signal Processing Letters, 7(6)(2000), 145-148.

[36] C. Tu, S. Srinivasan, G.J. Sullivan, S. Regunathan, H.S. Marvar, Low-compelxity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR/HD Photo, Proc. of SPIE Applications of Digital Image Processing XXXI, 7073(70730C)(2008), 1-12.

[37] H.S. Malvar, Signal Processing with Lapped Transforms, Norwood, MA: Artech House, 1992.