A Tutorial of Integer Transform

Po-Hung Wu

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, 10617,

E-mail: f98942124@ntu.edu.tw,

Chapter 1 Introduction

The integer transform (such as the Walsh transform, the Haar transform) is a discrete transform that all the entries in the transform matrix are integers. For software and hardware, the integer transform is much easier to be implemented because there is no real number multiplication, but performance is not guaranteed. By contrast, the non-integer transform (such as the DCT, the DFT) has higher performance, but the real number multiplication is needed. As we know, more computation time and implementing cost are required when there are more real number multiplications, and, therefore, this is the purpose why the integer transform has been researched. In this tutorial, I survey the development of integer transform since old days. The first time integer transform was mentioned is about the color space transform, and then the DCT approximation by prototype matrices was proposed. For example, (1.1) is the SIM transform whose entries are summation of 2^k, and so is its inverse transform shown in (1.2).

$$\begin{bmatrix} S \\ I \\ M \end{bmatrix} = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ -0.50 & 0.00 & 0.50 \\ -0.25 & 0.50 & -0.25 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{1.1}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} S \\ I \\ M \end{bmatrix}. \tag{1.2}$$

The SIM transform convert RGB color space into the SIM color space. Although the performance of the SIM color space is worse than RGB space, no floating-point processor is needed in the transform. Then the perfect reconstruction is reachable, and the cost and time spending is much less in operation. Moreover, in the end of Chapter 4, we propose a method generating a proximate integer transform whose NRMSE is smaller than 0.3% even better than the original one. Nowadays, many lifting-schemes based integer transforms have been researched rapidly. The Jacket transform, which is one of the most powerful integer transform, is a generalization of the Hadamard (Walsh) transform and useful in signal and image processing. All these integer transforms will be discussed in the following chapters.

Chapter 2 Integer Cosine and Sinusoidal Transform by Prototype Matrices

An integer cosine/sinusoidal transform (ICT/IST) can be implemented by using simple integer arithmetic with performance close to that of the discrete cosine transform (DCT). A simple method to eliminate floating point arithmetic is to approximate the real magnitudes of the DCT components by *M*-bit integers, so that the DCT can be computed by using integer additions and multiplications. In this section, we review how to convert the order-8 cosine transforms into a family of ICT by using the theory of dyadic symmetric [1]. This technique can be applied to any transform, but solutions are not guaranteed.

2.1 Dyadic Symmetry

Definition of dyadic symmetry [13]：

A vector of 2^m elements $[a_0, a_1, \ldots, a_{2^m-1}]$ is said to have the *i*th dyadic symmetry if and only if $a_j = s \bullet a_{j \oplus i}$, where $\oplus$ is the 'exclusive' operation, *j* lies in the range [0, 2^m-1] and I in the range [1, 2^m-1], s = 1 when the symmetry is even, s = -1 when the symmetry is odd.

Table 2.1：The seven vectors H_S having S^{th} even dyadic symmetry.

	Vector H_S
S	$[a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8]$
1	$[h_1\ h_1\ h_2\ h_2\ h_3\ h_3\ h_4\ h_4]$
2	$[h_1\ h_2\ h_1\ h_2\ h_3\ h_4\ h_3\ h_4]$
3	$[h_1\ h_2\ h_2\ h_1\ h_3\ h_4\ h_4\ h_3]$
4	$[h_1\ h_2\ h_3\ h_4\ h_1\ h_2\ h_4\ h_3]$
5	$[h_1\ h_2\ h_3\ h_4\ h_2\ h_1\ h_4\ h_3]$
6	$[h_1\ h_2\ h_3\ h_4\ h_3\ h_4\ h_1\ h_2]$
7	$[h_1\ h_2\ h_3\ h_4\ h_4\ h_3\ h_2\ h_1]$

For a vector of eight elements, there are seven possible dyadic symmetries. As an example, Table 2.1 shows vectors $H_s = [a_0\ \ a_1\ \ a_2\ \ a_3\ \ a_4\ \ a_5\ \ a_6\ \ a_7]$ which have the seven dyadic symmetries.

Theorem of orthogonality：Two vectors *U* and *V* are orthogonal if *U* and *V* have the same type of dyadic symmetry and one is even and another is odd.

2.2 Generation of the Integer Cosine Transforms

2.2.1 Generation of the Order-8 ICTs

Let [*T*] be the kernel of order-*n* DCT and $T_n(i, j)$ is the j^{th} component of the i^{th} DCT basis vector as below.

$$T_n(i,j)=\begin{cases}\dfrac{1}{\sqrt{n}}, & \text{for } i=0\\ \sqrt{\dfrac{2}{n}}\cos\left[\dfrac{i(j+0.5)\pi}{n}\right], & \text{for } 1\le i\le n-1\end{cases}, \quad 0\le j\le n-1 \tag{2.1}$$

There are three steps to convert the order-8 DCT kernel into ICT kernels.

Step 1：Express the order-8 DCT kernel [*T*] in the form of a matrix of variable. Then we have

$$[T]=\left[k_0J_0,\ k_1J_1,\ k_2J_2,\ k_3J_3,\ k_4J_4,\ k_5J_5,\ k_6J_6,\ k_7J_7\right]^t, \tag{2.2}$$

where J_i means i^{th} basis vector and k_i denotes a scaling constant then makes $|k_i\cdot J_i|=1$. Let $J(i,j)$ be the j^{th} element of J_i. From (1), we find that $T_8(1,0)=-T_8(1,7)=-T_8(3,2)$ $=T_8(3,5)=-T_8(5,1)=T_8(5,6)=-T_8(7,3)=T_8(7,4)$, and then represent the magnitudes of $J(1,0), J(1,7), J(3,2), J(3,5), J(5,1), J(5,6), J(7,3),$ and $J(7,4)$ by a single variable '*a*'. As shown in Table 2.2, all eight vectors can be expressed similarly as variables '*a*', '*b*', '*c*', '*d*', '*e*', and '*f*'.

Table 2.2：The 8 scaled basis vectors in $|J|$.

i	J_i							
0	1	1	1	1	1	1	1	1
1	*a*	*b*	*c*	*d*	*–d*	*–c*	*–b*	*–a*
2	*e*	*f*	*– f*	*–e*	*–e*	*– f*	*f*	*e*
3	*b*	*–d*	*–a*	*–c*	*c*	*a*	*d*	*–b*
4	1	–1	–1	1	1	–1	–1	1
5	*c*	*–a*	*d*	*b*	*–b*	*–d*	*a*	*–c*
6	*f*	*–e*	*e*	*– f*	*– f*	*e*	*–e*	*f*
7	*d*	*–c*	*b*	*–a*	*a*	*–b*	*c*	*–d*

Step 2：Find the conditions which J_i and J_j are orthogonal. From Table 2.3, we can know that J_0 has even 7^{th} dyadic symmetry and J_1 has odd 7^{th} dyadic symmetry. According to the theo-

rem of orthogonality, J_0 and J_1 are always orthogonal. For another example, J_0 and J_2 are always orthogonal because J_0 has even 3rd dyadic symmetry and J_2 has odd 3rd dyadic symmetry. Table 2.4 reveals that the only condition that the variables *a*, *b*, *c*, and *d* must satisfy to ensure that the transform [T] be orthogonal is

$$a \cdot b = a \cdot c + b \cdot d + c \cdot d. \tag{2.3}$$

Table 2.3：S^{th} dyadic symmetry type in each basis vector J_i.

i	Dyadic symmetry S in J_i						
0	*E*	*E*	*E*	*E*	*E*	*E*	*E*
1	–	–	–	–	–	–	*O*
2	–	–	*O*	–	–	–	*E*
3	–	–	–	–	–	–	*O*
4	*O*	*O*	*E*	*E*	*O*	*O*	*E*
5	–	–	–	–	–	–	*O*
6	–	–	*O*	–	–	–	*E*
7	–	–	–	–	–	–	*O*

Table 2.4：Conditions under which the i^{th} basis vector and the j^{th} basis vector are orthogonal.

			i				
1	2	3	4	5	6	7	j
*3	*2	*3	*2	*3	*2	*3	0
	*3	*1	*3	*1	*3	*4	1
		*3	*2	*3	*4	*3	2
			*3	*4	*3	*1	3
				*3	*2	*3	4
					*3	*1	5
						*3	6

*1: if $a \cdot b = a \cdot c + b \cdot d + c \cdot d.$
*2: must the orthogonal due to 3rd dyadic symmetry
*3: must the orthogonal due to 7th dyadic symmetry
*4: must the orthogonal due to dot product equals zero

From (3), we know that the equation with four variables has infinite number of solutions. This implies that an infinite number of new orthogonal transforms are generated from the

DCT.

Step 3：Set up boundary conditions and generate new transforms. (1) implies the following condition for the DCT.

$$a \geq b \geq c \geq d \text{ and } e \geq f. \tag{2.4}$$

To make the new orthogonal transforms resemble the DCT, (4) must be satisfied. For eliminating truncation error due to floating point, (1.6) must be satisfied.

$$a, b, c, d, e, \text{ and } f \text{ must be integers.} \tag{2.5}$$

When the kernel of order-8 DCT [*T*] is satisfied (1.4), (1.5), and (1.6), it can convert into the order-8 ICT. For example, ICT(5, 3, 2, 1, 3, 1) represents the order-8 ICT with *a*=5, *b*=3, *c*=2, *d*=1, *e*=3, and *f*=1.

2.2.2 Generation of the order-2*n* ICTs by Iteration

Any order-2*n* orthogonal transform $T_{2n}(i, j)$ can be generated form an order-*n* orthogonal transform $T_n(i, j)$ as follows:

1. The first *n* basis vectors of $T_{2n}(i, j)$：

$$T_{2n}(i,2j) = T_n(i, j) \text{ and } T_{2n}(i,2j+1) = T_n(i, j), \text{ for } 0 \leq j \leq n-1. \tag{2.6}$$

2. The last *n* basis vectors of $T_{2n}(i, j)$

i：$T_{2n}(i+n,2j) = T_n(i, j)$ *and* $T_{2n}(i+n,2j+1) = -T_n(i, j)$, for $j \in \{0,2,4,\ldots,n-2\}$. (2.7)

ii：$T_{2n}(i+n,2j) = -T_n(i, j)$ *and* $T_{2n}(i+n,2j+1) = T_n(i, j)$, for $j \in \{1,3,5,\ldots,n-1\}$. (2.8)

Any order-2^m ICT for *m*>3 can be generated from the order-8 ICT.

2.2.3 Generation of the order-16 ICTs by Dyadic Symmetry

In Section 2.2.2, we introduce that any order-2^m ICT for m>3 can be generated from the order-8 ICT. However, the performance of order-16 ICT generated by iteration is inferior to by dyadic symmetry due to the number of variable used in deriving. In iteration, there are only six variables as in the order-8 ICT, but there are more variables used in dyadic symmetry. Therefore, we briefly introduce how to generate an order-16 ICT by dyadic symmetry [2] in this section.

Similarly, as shown in Section 2.1, the definition of dyadic symmetry is used to generate the order-16 even dyadic symmetry matrix shown in Table 2.5.

Table2.1 The fifteen vectors H_s having S^{th} even dyadic symmetry.

S	Vector H_s															
	$[a_0$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	$a_{15}]$
1	h_1	h_1	h_2	h_2	h_3	h_3	h_4	h_4	h_5	h_5	h_6	h_6	h_7	h_7	h_8	h_8
2	h_1	h_2	h_1	h_2	h_3	h_4	h_3	h_4	h_5	h_6	h_5	h_6	h_7	h_8	h_7	h_8
3	h_1	h_2	h_2	h_1	h_3	h_4	h_4	h_3	h_5	h_6	h_6	h_5	h_7	h_8	h_8	h_7
4	h_1	h_2	h_3	h_4	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_5	h_6	h_7	h_8
5	h_1	h_2	h_3	h_4	h_2	h_1	h_4	h_3	h_5	h_6	h_7	h_8	h_6	h_5	h_8	h_7
6	h_1	h_2	h_3	h_4	h_3	h_4	h_1	h_2	h_5	h_6	h_7	h_8	h_7	h_8	h_5	h_6
7	h_1	h_2	h_3	h_4	h_4	h_3	h_2	h_1	h_5	h_6	h_7	h_8	h_8	h_7	h_6	h_5
8	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
9	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_2	h_1	h_4	h_3	h_6	h_5	h_8	h_7
10	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_3	h_4	h_1	h_2	h_7	h_8	h_5	h_6
11	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_4	h_3	h_2	h_1	h_8	h_7	h_6	h_5
12	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_5	h_6	h_7	h_8	h_1	h_2	h_3	h_4
13	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_6	h_5	h_8	h_7	h_2	h_1	h_4	h_3
14	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_7	h_8	h_5	h_6	h_3	h_4	h_1	h_2
15	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_8	h_7	h_6	h_5	h_4	h_3	h_2	h_1

To obtain the order-16 odd dyadic symmetry matrix, one only needs to change the signs of

appropriate elements of H_s according to the definition of dyadic symmetry given in section 2.1. In the same way, two vectors U and V are orthogonal if U and V have the same type of dyadic symmetry and one is even and the other is odd.

Let $[T]$ be the kernel of order-16 DCT and $T_{16}(i, j)$ is the jth component of the ith DCT basis vector as below：

$$T_{16}(i,j)=\begin{cases}\dfrac{1}{\sqrt{16}}, & \text{for } i=0\\ \sqrt{\dfrac{2}{16}}\cos\left[\dfrac{i(j+0.5)\pi}{16}\right], & \text{for } 1\le i\le 15\end{cases},\ 0\le j\le 15. \tag{2.9}$$

The steps are described in the following that transforms the order-16 DCT kernels into ICT kernels.

Step 1：Express the order-16 DCT kernel $[T]$ in the form of a matrix of variable. Then we have

$$[T]=[k_0J_0,\ k_1J_1,\ k_2J_2,\ k_3J_3,\ k_4J_4,\ k_5J_5,\ k_6J_6,\ k_7J_7,$$

$$k_8J_8\ k_9J_9,\ k_{10}J_{10},\ k_{11}J_{11},\ k_{12}J_{12}\ k_{13}J_{13}\ k_{14}J_{14}]^t, \tag{2.10}$$

where J_i means ith basis vector and k_i denotes a scaling constant then makes $|k_i \cdot J_i|=1$. Let $J(i, j)$ be the jth element of J_i. In Table 2.2, all sixteen vectors can be expressed as variables ‘*a*’, ‘*b*’, ‘*c*’, …, ‘*m*’ and ‘*n*’ in the same manner.

Table 2.2 The 16 scaled basis vectors in $|J|$.

i	J_i															
0	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l
1	a	b	c	d	e	f	g	h	$-h$	$-g$	$-f$	$-e$	$-d$	$-c$	$-b$	$-a$
2	i	j	k	l	$-l$	$-k$	$-j$	$-i$	$-i$	$-j$	$-k$	$-l$	l	k	j	i
3	b	e	h	$-f$	$-c$	$-a$	$-d$	$-g$	g	d	a	c	f	$-h$	$-e$	$-b$
4	m	n	$-n$	$-m$	$-m$	$-n$	n	m	m	n	$-n$	$-m$	$-m$	$-n$	n	m
5	c	h	$-d$	$-b$	$-g$	e	a	f	$-f$	$-a$	$-e$	g	b	d	$-h$	$-c$
6	j	$-l$	$-i$	$-k$	k	i	l	$-j$	$-j$	l	i	k	$-k$	$-i$	$-l$	j
7	d	$-f$	$-b$	$-h$	a	g	$-c$	$-e$	e	c	$-g$	$-a$	h	b	f	$-d$
8	l	$-l$	$-l$	l	l	$-l$	$-l$	l	l	$-l$	$-l$	l	l	$-l$	$-l$	l
9	e	$-c$	$-g$	a	$-h$	$-b$	f	d	$-d$	$-f$	b	h	$-a$	g	c	$-e$
10	k	$-i$	l	j	$-j$	$-l$	i	$-k$	$-k$	i	$-l$	$-j$	j	l	$-i$	k
11	f	$-a$	e	g	$-b$	d	h	$-c$	c	$-h$	$-d$	b	$-g$	$-e$	a	$-f$
12	n	$-m$	m	$-n$	$-n$	m	$-m$	n	n	$-m$	m	$-n$	$-n$	m	$-m$	n
13	g	$-d$	a	$-c$	f	h	$-e$	b	$-b$	e	$-h$	$-f$	c	$-a$	d	$-g$
14	l	$-k$	j	$-i$	i	$-j$	k	$-l$	$-l$	k	$-j$	i	$-i$	j	$-k$	l
15	h	$-g$	f	$-e$	d	$-c$	b	$-a$	a	$-b$	c	$-d$	e	$-f$	g	$-h$

Step 2：Find the conditions which J_i and J_j are orthogonal.

Table 2.3 Dyadic symmetry types in each basis vector J_i.

	Dyadic Symmetry S in J_i														
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*	*E*
1															*O*
2							*O*	*O*							*E*,*O*
3															*O*
4			*O*	*O*			*E*	*E*			*O*	*O*			*E*
5															*O*
6							*O*	*O*							*E*
7															*O*
8	*O*	*O*	*E*	*E*	*O*	*O*	*E*	*E*	*O*	*O*	*E*	*E*	*O*	*O*	*E*
9															*O*
10							*O*	*O*							*E*,*O*
11															*O*
12			*O*	*O*			*E*	*E*			*O*	*O*			*E*
13															*O*
14							*O*	*O*							*E*
15															*O*

To make the *i*th basis vector J_i and the *j*th basis vector J_j are orthogonal for all *i*, *j*, (2.11)-(2.14) must be satisfied.

$$a\,b+\ b\,e-\ c\,h+\ d+f\ \ e-e\ \ d+g\ \ +c \tag{2.11}$$

$$a\,c+\ b\,h+\ e\,f-\ a+g\ \ f-h\ \ e+d\ \ -b, \tag{2.12}$$

$$a\,d+\ d\,h+\ a+e\ \ f+g\ \ b+f\ \ -b\,c\ \ +c, \tag{2.13}$$

$$i\,j=\ j\,k+\ k+1\ , \tag{2.14}$$

Step 3：Set up boundary conditions and generate new transforms. From (2.9) and Table 2.2, the relationship of the magnitude of the variables *a*, *b*, *c*, *d*, *e*, and *f* is in the following：

$$a > i > b > m > c > j > d > e > k > f > n > g, \tag{2.15}$$

To relax the condition in (2.14), we can get

$$a \geq b \geq c \geq d \geq e \geq f \geq g \geq 0 \tag{2.16}$$

$$i \geq j \geq h \geq k \geq 0 \tag{2.17}$$

$$m \geq n. \tag{2.18}$$

For eliminating truncation error, the (2.19) also must be satisfied.

$$a,\ b,\ c,\ d,\ e,\ \ldots,\ m,\ \text{and}\ n \tag{2.19}$$

When the kernel of order-16 DCT [*T*] satisfies (2.11)-(2.14) and (2.16)-(2.18), it can convert into the order-16 ICT.

2.3 Generation of the Integer Sinusoidal Transforms

The sine and cosine transform are members of sinusoidal transforms family. Each member of the sinusoidal transform family is the optimal Karhunen-Loeve transform of a Markov process. In [3], the authors derive order-8 sinusoidal transforms that can be implemented by using the integer arithmetic. The even sine-1 and even sine-2 transforms of order 8 and 16 are the most popular sinusoidal transforms for image coding.

Jain (1976) suggested that the KLT, DCT, even sine-1 and even sine-2 transforms can all be considered as members of a sinusoidal family by means of a parametric matrix $[J(k_1,k_2,k_3,k_4)]$ where $[\Phi]=[\phi_1\ \phi_2\ \ldots\ \phi_{n-1}]^t$ is a member of the family if ϕ_ms are eigenvectsor of a *J* matrix of order n. In other words, transform members of the sinusoidal family can be constructed from eigenvectors of *J* matrix. Table 2.4 shows some members of the sinusoidal family generated by $J(k_1,k_2,0,0)$.

k_1	k_2	$[\Phi]$
0	0	Even sine-1
0	-1	Odd sine-2
0	1	Odd sine-1
-1	0	Odd sine-3
-1	-1	Even sine-2
-1	1	Even sine-3
1	0	Odd cosine-3
1	-1	Even cosine-2
1	1	Even cosine-1

Table 2.4 Some members of the sinusoidal family generated by $J(k_1, k_2, 0, 0)$.

$$[J(k_1,k_2,k_3,k_4)] = \begin{bmatrix} 1-k_1 & -\alpha & 0 & \vdots & \vdots & k_3\alpha \\ -\alpha & 1 & -\alpha & \vdots & \vdots & 0 \\ 0 & -\alpha & 1 & -\alpha & \vdots & \vdots \\ \vdots & 0 & -\alpha & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & -\alpha & 1 & -\alpha \\ k_4\alpha & \vdots & \vdots & 0 & -\alpha & 1-k_2 \end{bmatrix}$$

2.3.1 Order-8 Integer Even Sine-1 Transform

Because there are many types of integer sine transforms, we only take the order-8 even sine-1 integer transform as an example. Let [*T*] be the kernel of order-8 even sine-1 transform and $T_n(i, j)$ is the j^{th} component of the i^{th} even sine-1 transform basis vector as below.

$$T_n(i,j) = \sqrt{\frac{2}{n+1}} \sin\left[\frac{(i+1)(j+1)\pi}{2n}\right], \text{ for } 0 \le i, j \le n-1 \qquad (2.20)$$

Consider the order-8 even sine-1 transform, we can get its kernel ：

$$[T] = \left[k_0 J_0 \; k_1 J_1 \; k_2 J_2 \; k_3 J_3 \; k_4 J_4 \; k_5 J_5 \; k_6 J_6\right]^t \qquad (2.21)$$

where J_i means i^{th} basis vector and k_i denotes a scaling constant then makes $|k_i \cdot J_i| = 1$.

Therefore, all eight vectors can be expressed as variables '*a*', '*b*', '*c*', and '*d*' in (2.22).

$$[T]=\begin{bmatrix} k_0(a & b & c & d & d & c & b & a) \\ k_1(b & d & c & a & -a & -c & -d & -b) \\ k_2(c & c & 0 & -c & -c & 0 & c & c) \\ k_3(d & a & -c & -b & b & c & -a & -d) \\ k_4(d & -a & -c & b & b & -c & -a & d) \\ k_5(c & -c & 0 & c & -c & 0 & c & -c) \\ k_6(b & -d & c & -a & -a & c & -d & b) \\ k_7(a & -b & c & -d & d & -c & b & -a) \end{bmatrix} \tag{2.22}$$

Like what we discuss in the integer cosine transform, the variable *a*, *b*, *c*, and *d* must be satisfied the condition to make the kernel [*T*] be orthogonal:

$$c(a+b-d)=0, \tag{2.23}$$

$$a(d-b)-c^2+bd=0. \tag{2.24}$$

To simplify (2.23) and (2.24), we can get

$$d=a+b\quad(c>0) \tag{2.25}$$

$$c=\sqrt{a^2+b^2+ab}. \tag{2.26}$$

From (2.20) and (2.22), the relationship of the magnitude of the variables *a*, *b*, *c*, and *d* is for the order-8 even sine-1 transform:

$$d>c>b>a>0 \tag{2.27}$$

The other integer sinusoidal transforms can be derived as well as section 2.2 and are summarized in Table 2.5.

The integer sinusoidal transform	**The basis vector of the transform $T_n(i,j)$ kernel and the transform kernel $[T]$**	**The orthogonal condition and relationship of the magnitude**
Even sine-1 transform	$T_n(i,j)=$ $\sqrt{\frac{2}{n+1}}\sin\left[\frac{(i+1)(j+1)\pi}{2n}\right]$, for $0\le i,j\le n-1$	$d=a+b\ (c>0)$ $c=\sqrt{a^2+b^2+ab}$ $d>c>b>a>0.$

	$[T] =$ $\begin{bmatrix} k_0(a & b & c & d & d & c & b & a) \\ k_1(b & d & c & a & -a & -c & -d & -b) \\ k_2(c & c & 0 & -c & -c & 0 & c & c) \\ k_3(d & a & -c & -b & b & c & -a & -d) \\ k_4(d & -a & -c & b & b & -c & -a & d) \\ k_5(c & -c & 0 & c & -c & 0 & c & -c) \\ k_6(b & -d & c & -a & -a & c & -d & b) \\ k_7(a & -b & c & -d & d & -c & b & -a) \end{bmatrix}$	
Odd sine-2 transform	$T_n(i,j) =$ $\frac{2}{\sqrt{2n+1}} \sin\left[\frac{2(i+1)(j+1)\pi}{2n+1}\right],$ for $0 \le i, j \le n-1$ $[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(b & d & f & h & -g & -e & -c & -a) \\ k_2(c & f & -h & -e & -b & a & d & g) \\ k_3(d & h & -e & -a & c & g & -f & -b) \\ k_4(e & -g & -b & c & h & -d & a & f) \\ k_5(f & -e & a & g & -d & b & h & -c) \\ k_6(g & -c & d & -f & a & h & -b & e) \\ k_7(h & -a & g & -b & f & -c & e & -d) \end{bmatrix}$	$ab + bd + cf + dh -$ $eg - fe - gc - ha = 0$ $ac + bf - ch - de -$ $eb + fa + gd + hg = 0$ $ae - bg - cb + dc +$ $eh - fd + ga + hf = 0$ $d > e > c > f >$ $b > g > a > h > 0$
Odd sine-1 transform	$T_n(i,j) =$ $\frac{2}{\sqrt{n+1}} \sin\left[\frac{(2i+1)(j+1)\pi}{2n+1}\right],$ for $0 \le i, j \le n-1$ $[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(c & f & h & e & b & -a & -d & -g) \\ k_2(e & g & b & -c & -h & -d & a & f) \\ k_3(g & c & -d & -f & a & h & b & -e) \\ k_4(h & -a & -g & b & f & -c & -e & d) \\ k_5(f & -e & -a & g & -d & -b & h & -c) \\ k_6(d & -h & e & -a & -c & g & -f & b) \\ k_7(b & -d & f & -h & g & -e & c & -a) \end{bmatrix}$	$a(c-f)+b(e+f)$ $-d(g-e)-h(g-c)$ $= 0$ $a(e+g)+b(c+g)$ $-d(c+f)+h(f-e)$ $= 0$ $a(h-b)-c(f+g)$ $+d(b+h)+e(f-g)$ $= 0$ $h > g > f > e >$ $d > c > b > a$
Odd sine-3 transform	$T_n(i,j) =$ $\frac{2}{\sqrt{2n+1}} \sin\left[\frac{(i+1)(2j+1)\pi}{2n+1}\right],$ for $0 \le i, j \le n-1$	$ah + bf + cd + db -$ $ea - fc - ge - hg = 0$

	$[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(h & f & d & b & -a & -c & -e & -g) \\ k_2(b & e & h & -g & -d & -a & c & f) \\ k_3(g & c & -b & -f & h & d & -a & -e) \\ k_4(c & h & -e & a & f & -g & -b & d) \\ k_5(f & -a & -g & e & -b & -h & d & -c) \\ k_6(d & -g & a & h & -c & e & -f & b) \\ k_7(e & -d & f & -c & g & -b & h & -a) \end{bmatrix}$	$ab+be+ch-dg-$ $ed-fa+gc+hf=0$ $ac+bh-ce+da+$ $ef-fg-gb+hd=0$ $e>d>f>c>$ $g>b>h>a>0$
Even sine-2 transform	$T_n(i,j) =$ $\begin{cases} \sqrt{\frac{2}{n}} \sin\left[\frac{(i+1)(2j+1)\pi}{2n}\right], \\ \text{for } 0 \le i, j \le n-1 \text{and } i \ne n-1 \\ \frac{(-1)^j}{\sqrt{n}}, \\ \text{for } 0 \le j \le n-1 \text{and } i = n-1 \end{cases}$ $[T] =$ $\begin{bmatrix} k_0(a & b & c & d & d & c & b & a) \\ k_1(e & f & f & e & -e & -f & -f & -e) \\ k_2(b & d & a & -c & -c & a & d & b) \\ k_3(g & g & -g & -g & g & g & -g & -g) \\ k_4(c & a & -d & b & b & -d & a & c) \\ k_5(f & -e & -e & f & -f & e & e & -f) \\ k_6(d & -c & b & -a & -a & b & -c & d) \\ k_7(g & -g & g & -g & g & -g & g & -g) \end{bmatrix}$	$ab+bd+ac-cd=0$ $d>c>b>a>0$ $f>e$
Even sine-3 transform	$T_n(i,j) =$ $\sqrt{\frac{2}{n}} \sin\left[\frac{(2i+1)(2j+1)\pi}{4n}\right],$ for $0 \le i, j \le n-1$	$ab+be+ch+df+$ $ec-fa-gd-hg=0$ $ac+bh+cd-db-$ $eg-fe+ga+hf=0$ $ad+bf-cb-dh-$ $ea+fg+gc-he=0$ $h>g>f>e>$ $d>c>b>a>0$

	$[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(b & e & h & f & c & -a & -d & -g) \\ k_2(c & h & d & -b & -g & -e & a & f) \\ k_3(d & f & -b & -h & -a & g & c & -e) \\ k_4(e & c & -g & -a & h & -b & -f & d) \\ k_5(f & -a & -e & g & -b & -d & h & -c) \\ k_6(g & -d & a & c & -f & h & -e & b) \\ k_7(h & -g & f & -e & d & -c & b & -a) \end{bmatrix}$	
Odd cosine-1 transform	$T_n(i,j) =$ $\frac{2}{\sqrt{4n+1}} \cos\left[\frac{(2i+1)(2j+1)\pi}{4n+2}\right],$ for $0 \le i, j \le n-1$ $[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(b & e & h & -g & -d & -a & -c & -f) \\ k_2(c & h & -e & -a & -f & g & b & d) \\ k_3(d & -g & -a & -h & c & e & -f & -b) \\ k_4(e & -d & -f & c & g & -b & -h & a) \\ k_5(f & -a & g & e & -b & h & d & -c) \\ k_6(g & -c & b & -f & -h & d & -a & e) \\ k_7(h & -f & d & -b & a & -c & e & -g) \end{bmatrix}$	$ab+be+ch-dg-$ $ed-fa-gc-hf=0$ $ac+bh-cf-ad-$ $ef+fg+bg+dh=0$ $ae-bd-cf+cd+$ $eg-bf-gh+ha=0$ $a>b>c>d>$ $e>f>g>h>0$
Even cosine-2 transform	$T_n(i,j) =$ $\sqrt{\frac{2}{n}} \cos\left[\frac{(2i+1)(2j+1)\pi}{4n}\right],$ for $0 \le i, j \le n-1$ $[T] =$ $\begin{bmatrix} k_0(a & b & c & d & e & f & g & h) \\ k_1(b & e & h & -f & -c & -a & -d & -g) \\ k_2(c & h & -d & -b & -g & e & a & f) \\ k_3(d & -f & -b & h & a & g & -c & -e) \\ k_4(e & -c & -g & a & -h & -b & f & d) \\ k_5(f & -a & e & g & -b & d & h & -c) \\ k_6(g & -d & a & -c & f & h & -e & b) \\ k_7(h & -g & f & -e & d & -c & b & -a) \end{bmatrix}$	$ab+be+ch-df-$ $ec-fa-gd-hg=0$ $ac+bh-cd-db-$ $eg+fe+ga+hf=0$ $ad-bf-cb+dh+$ $ea+fg-gc-he=0$ $a>b>c>d>$ $e>f>g>h>0$

Even co-sine-1 transform (introduced in section 1.3)	$T_n(i,j)=\begin{cases}\dfrac{1}{\sqrt{n}}, & \text{for } i=0,\ 0\le j\le n-1\\ \sqrt{\dfrac{2}{n}}\cos\left[\dfrac{i(j+0.5)\pi}{n}\right], & \text{for } 0\le i,j\le n-1 \text{ and } i\ne 0\end{cases}$ $[T]=$ $\begin{bmatrix} k_0(1 & 1 & 1 & 1 & 1 & 1 & 1 & 1) \\ k_1(a & b & c & d & -d & -c & -b & -a) \\ k_2(e & f & -f & -e & -e & -f & f & e) \\ k_3(b & -d & -a & -c & c & a & d & -b) \\ k_4(1 & -1 & -1 & 1 & 1 & -1 & -1 & 1) \\ k_5(c & -a & d & b & -b & -d & a & -c) \\ k_6(f & -e & e & -f & -f & e & -e & f) \\ k_7(d & -c & b & -a & a & -b & c & -d) \end{bmatrix}$	$ab=ac+bd+dc$ $a>b>c>d$ $e>f$

Table 2.5 The integer sinusoidal transforms.

2.4 Performance of the order-*n* ICTs

2.4.1 Transform Efficiency Performance

In transform coding in image, orthogonal transforms are used to convert highly correlated signals into coefficients of low correlation. The ability of decorrelation may be measured by the transform efficiency, which is defined on the first-order Markov process of adjacent element correlation ρ. The Karhunen-Loeve transform (KLT) is the optimal transform that converts signals into completely uncorrelated coefficients, and the KLT has a transform efficiency equal to 100% for all ρ.

Consider *n*-dimensional vector *X*, which is a sample from one-dimensional, zero-mean, unit-variance first-order Markov process with adjacent element correlation ρ, and *Y* is a transformed matrix.

$$Y = [T]X \tag{2.28}$$

$$\begin{aligned}[C_Y] &= E\left[Y \cdot Y^t\right] \\ &= [T][C_X][T]^t \\ &= \begin{bmatrix} s_{11} & \cdots & \cdots & s_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ s_{n1} & \cdots & \cdots & s_{nn} \end{bmatrix}\end{aligned} \tag{2.29}$$

where $[C_X]$ is covariance matrix of the *n*-dimensional vector *X* then the *(i*, j)th element of $[C_X]$ is $\rho^{|i-j|}$ and $[C_Y]$ is covariance matrix of the transformed matrix. The efficiency of the kernel [*T*] is defined on the transformed domain:

$$\text{Efficiency } \eta = \frac{\sum_{i=1}^{n} |s_{ii}|}{\sum_{p=1}^{n}\sum_{q=1}^{n} |s_{pq}|}. \tag{2.30}$$

2.4.2 Basis Restriction Mean-Square-Error Performance

The data compression ability of a transform can be measured by means of the basis restriction mean-square-error [14]. Consider a two-dimensional zero-mean unit-variance non-separable isotropic Markov process with covariance function as following:

$$\begin{aligned}C_x(i, j; p, q) &= E\left[x_{i,j} \cdot x_{p,q}\right] \\ &= \rho^{\sqrt{(i-p)^2+(j-q)^2}}\end{aligned} \tag{2.31}$$

where ρ is the adjacent element correlation in the vertical and horizontal directions. Let the *n* ×*n* matrix [*X*] be a sample of the Markov process. If [*X*] is transformed into [*C*] by transform [*T*], we can get

$$[C] = [T][X][T]^t, \tag{2.32}$$

where the elements of [X] and [C] are $x_{i,j}$ and $c_{u,v}$, respectively. The covariance function of [C] is

$$\begin{aligned} C_C(u,v;r,s) &= E\left[c_{u,v}\cdot c_{r,s}\right] \\ &= \sum_i \sum_j \sum_p \sum_q C_X(i,j;p,q) T(u,i) T(u,j) T(r,p) T(s,q). \end{aligned} \tag{2.33}$$

Therefore, the variance of $c_{u,v}$ is equal to

$$\sigma^2(u,v) = C_C(u,v;u,v). \tag{2.34}$$

Suppose Ω be the set containing M index pairs (u, v) corresponding to the largest M $\sigma_C(u,v)$ then the basis restriction mean square error is defined as:

$$e(M) = 1 - \frac{\sum_{u,v\in\Omega} \sum \sigma_C(u,v)^2}{\sum_u \sum_v \sigma_C(u,v)^2}. \tag{2.35}$$

We can use the basis restriction mean-square-error to measure the data compression ability of a transform. Comparing the basis restriction mean-square-errors of the ICTs with various transforms when $\rho = 0.95$. We can find that the relationship of the basis restriction mean square error:

$$\text{WT} > \text{CMT} > \text{ICT} > \text{DCT} > \text{KLT}, \tag{2.36}$$

where WT and CMT are Walsh transform and C-matrix transform respectively.

2.5 Implementation

[T] is an ICT. From (2), we have

$$[T] = [K][J], \tag{2.37}$$

$$[T]^{-1} = [T]^t = [J]^t [K], \tag{2.38}$$

where [*K*] is a diagonal matrix whose (*i*, *i*)th element is k_i and [*J*] is a matrix whose *i*th row is shown in Table 2. [*J*] contains only integer elements because it is an ICT. The adaptive 1-D transform coding system utilizes an order-8 ICT shown in Fig. 2.1. The upper part of Fig. 2.1 is the transmitter and the lower part is the receiver.

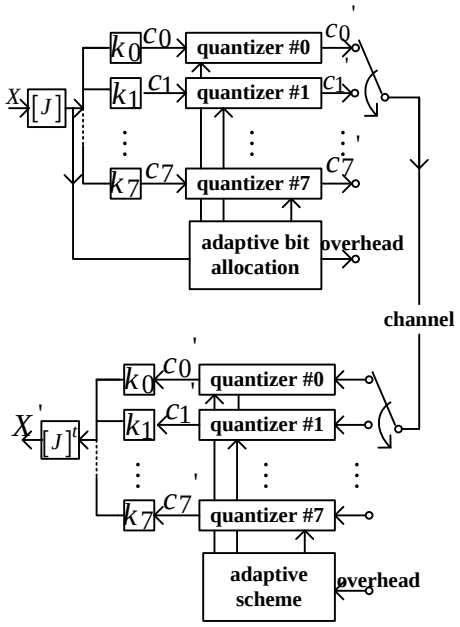

Fig. 2.1 The Adaptive 1-D transform coding system using an order-8 ICT.

The fast algorithm of the ICT is like the DCT and only requires 4 iterations shown in Fig. 2.2.

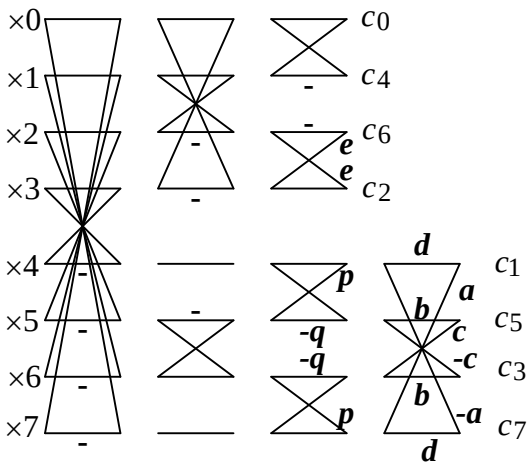

Fig. 2.2 The fast algorithm of the order-8 ICT where $p = (b + c)/2a$ and $q = (a - d)/2c$.

Chapter 3

Integer Transform by Lifting Scheme

(這一章要強調 lifting scheme 可以將所有的 non-singular matrix 變成 reversible integer transform，這也是 lifting scheme 可以逐漸取代 prototype matrix method 的主因)

To compress digital signals without distortion, the transform must map integers to integers and be perfectly invertible. In Chapter 2, the DCT derived from prototype satisfies mapping integers to integers, but the inverse does not. As the result, the perfect reconstruction is not available. In [4], an optimal solution of reversible integer implementation for general linear transforms is presented. It is a generalized lifting scheme and can be applied to any invertible linear transform besides discrete wavelet transforms. After the factorization method proposed by Hao in [7], any non-singular matrix can be converted into a reversible integer transform.

An affine transform in finite-dimensional space can be expressed in a matrix form $y = Ax + b$, which becomes a linear transform when b=0. Perfect inversion of a transform demands that both the forward transform and its reverse are integer reversible. Analogously, if the transformed values are changed due to quantization or some other operation, we still anticipate that the values can be reversely transformed into integers. Then, the sufficient integer reversible condition is $\| A \|_\infty = \| A^{-1} \|_\infty = 1$. A transform matrix does not always satisfy this condition, but those called the elementary reversible matrices (ERMs) can.

3.1 Elementary Reversible Matrix

Many matrix computation algorithms map number x to number y, such as those for a linear

transform $y = ax + b$. If variable x can be assigned to an arbitrary integer, the condition of integer output y is that both number a and number b are integers [16]. Since there might be some processing procedures in the transformation domain, the range of y is considered to be a set of integers without restrictions. In the case of an arbitrary integer output, the condition for the inverse transform that maps integers to integers is that both $1/a$ and b are integers. Therefore, the perfectly integer-invertible condition of the transform is that a is an integer factor and b is an integer, where an ***integer factor*** is defined as a multiplier of integers that does not change their magnitude, which is denoted by j.

If the computational ordering is properly arranged in $y = Ax$, we can find some elementary reversible structures for perfectly invertible integer implementation. A corresponding matrix is defined as an ***elementary reversible matrix (ERM)***. An upper or lower triangular transform matrix whose diagonal elements are integer factors is a kind of ERM, which is called a ***triangular ERM (TERM)***. If all the diagonal elements of a TERM are equal to 1, the TERM will be a unit triangular matrix. A TERM has the following two important properties.

- The product of two upper TERMs is also an upper TERM, and the product of two lower TERMs also makes a lower TERM.
- The determinant of a TERM is an integer factor.

If $A = \{a_{mn}\}$ is an upper TERM, the computational ordering of linear transform $y = Ax$ can be arranged to be top-down:

$$\begin{cases} y_m = j_m x_m + \left[\sum_{n=m+1}^{N} a_{mn} x_n \right] = j_m x_m + [b_m], \\ y_N = j_N x_N \end{cases} \tag{3.1}$$

where m = 1, 2, …, N-1.

Its inverse ordering is reversed:

$$\begin{cases} x_N = y_N / j_N \\ x_m = (1/j_m)\cdot(y_m - \left[\sum_{n=m+1}^{N} a_{mn}x_n\right]), \\ \quad = (1/j_m)\cdot(y_m - [b_m]) \end{cases} \tag{3.2}$$

where m = N, N-1, …, 1.

If A is a lower TERM, the computational ordering of linear transform y = Ax can be arranged to be bottom-up and its inversion to be top-down:

$$\begin{cases} y_m = j_m x_m + \left[\sum_{n=1}^{m-1} a_{mn}x_n\right] = j_m x_m + [b_m] \\ y_1 = j_1 x_1 \end{cases}, \tag{3.3}$$

where m = N, N-1, …, 2.

$$\begin{cases} x_1 = y_1 / j_1 \\ x_m = (1/j_m)\cdot(y_m - \left[\sum_{n=1}^{m-1} a_{mn}x_n\right]), \\ \quad = (1/j_m)\cdot(y_m - [b_m]) \end{cases} \tag{3.4}$$

where m = 2, 3, …, N.

Obviously, if a matrix can be converted into a TERM using row and column permutation only, it is also an ERM. Then, we can find another feasible ERM form known as ***single-row ERM (SERM)*** with integer factors on the diagonal and only one row of off-diagonal elements that are not all zeros.

3.2 Matrix Factorizations

In [4] and [5], it has been proved that a nonsingular matrix can be factorized into a product of at most three TERMs or a series of SERMs. An algorithm of matrix factorization for reversible integer mapping is shown in the following.

If

$$A=\begin{bmatrix} a_{1,1}^{(0)} & a_{1,2}^{(0)} & \cdots & a_{1,N}^{(0)} \\ a_{2,1}^{(0)} & a_{2,2}^{(0)} & \cdots & a_{2,N}^{(0)} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N,1}^{(0)} & a_{N,2}^{(0)} & \cdots & a_{N,N}^{(0)} \end{bmatrix}, \tag{3.5}$$

there must be a permutation matrix P_1 for row interchanges such that

$$P_1A=\begin{bmatrix} p_{1,1}^{(1)} & p_{1,2}^{(1)} & \cdots & p_{1,N}^{(1)} \\ p_{2,1}^{(1)} & p_{2,2}^{(1)} & \cdots & p_{2,N}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N,1}^{(1)} & p_{N,2}^{(1)} & \cdots & p_{N,N}^{(1)} \end{bmatrix}, \tag{3.6}$$

and $p_{1,N}^{(1)} \neq 0$. Therefore, there must exist a number s_1 such that $p_{1,1}^{(1)} - s_1 p_{1,N}^{(1)} = 1$, and then s_1 would be $s_1 = (p_{1,1}^{(1)} - 1) / p_{1,N}^{(1)}$.

$$\begin{aligned} P_1AS_{0,1} &= P_1A\begin{bmatrix} 1 & & \\ & I & \\ -s_1 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & p_{1,2}^{(1)} & \cdots & p_{1,N}^{(1)} \\ p_{2,1}^{(1)} - s_1 p_{2,N}^{(1)} & p_{2,2}^{(1)} & \cdots & p_{2,N}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N,1}^{(1)} - s_1 p_{N,N}^{(1)} & p_{N,2}^{(1)} & \cdots & p_{N,N}^{(1)} \end{bmatrix}. \end{aligned} \tag{3.7}$$

Using the forward elimination of the first column can be achieved by multiplying an elementary Gauss matrix L_1.

$$\begin{aligned} L_1P_1AS_{0,1} &= \begin{bmatrix} 1 & & \\ s_1p_{2,N}^{(1)} - p_{2,1}^{(1)} & 1 & \\ \cdots & & I \\ s_1p_{N,N}^{(1)} - p_{N,1}^{(1)} & & 1 \end{bmatrix} P_1AS_{0,1} \\ &= \begin{bmatrix} 1 & a_{1,2}^{(2)} & \cdots & a_{1,N}^{(2)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,N}^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & a_{N,2}^{(2)} & \cdots & a_{N,N}^{(2)} \end{bmatrix}. \end{aligned} \tag{3.8}$$

Iterating the permutation and forward elimination for k = 2, 3, …, *N*-1. P_k is used to interchange the rows among the *k*th through the *N*th rows to guarantee that the k^{th} element in the

*N*th column is not zero. If there were no such P_k, *A* would be singular. $S_{0,k}$ (SERMs) convert $a_{k,k}^{(k)}$ into 1s, and $s_k = (p_{k,k}^{(k)} - 1) / p_{k,N}^{(k)}$. L_k record the row multipliers used for the Gaussian elimination of column *k*. Therefore, we derive

$$\begin{aligned} & L_{N-1}P_{N-1}\cdots L_2P_2L_1P_1AS_{0,1}S_{0,2}\cdots S_{0,N-1} \\ &= \begin{bmatrix} 1 & a_{1,2}^{(N-1)} & \cdots & a_{1,N}^{(N-1)} \\ 0 & 1 & \cdots & a_{2,N}^{(N-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{N,N}^{(N-1)} \end{bmatrix} \\ &= D_R U, \end{aligned} \tag{3.9}$$

where $a_{N,N}^{(N-1)} = e^{i\theta}$, $D_R = diag(1,1,...,1,e^{i\theta})$, and

$$U = \begin{bmatrix} 1 & a_{1,2}^{(N-1)} & \cdots & a_{1,N}^{(N-1)} \\ 0 & 1 & \cdots & a_{2,N}^{(N-1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{3.10}$$

Now, multiplying all the SERMs ($S_{0,k}$) together, all the permutation matrices (P_k) together, and all the unit lower triangular matrices (L_k) together, we get

$$S_{0,1}S_{0,2}\cdots S_{0,N-1} = \begin{bmatrix} 1 & & & \\ & I & & \\ & & 1 & \\ -s_1 & \cdots & -s_{N-1} & 1 \end{bmatrix} = S_0^{-1}. \tag{3.11}$$

$$\begin{aligned} & L_{N-1}P_{N-1}\cdots L_2P_2L_1P_1 \\ &= L_{N-1}(P_{N-1}L_{N-2}P_{N-1}^T)\cdots(P_{N-1}\cdots P_2L_1P_2^T\cdots P_{N-1}^T)(P_{N-1}\cdots P_2P_1) \\ &= L^{-1}P^T. \end{aligned} \tag{3.12}$$

where $L^{-1} = L_{N-1}(P_{N-1}L_{N-2}P_{N-1}^T)\cdots(P_{N-1}\cdots P_2L_1P_2^T\cdots P_{N-1}^T)$, and $P^T = P_{N-1}\cdots P_2P_1$.

Therefore, we derive $L^{-1}P^TAS_0^{-1} = D_RU$ or $A = PLD_RUS_0$. Having once gotten a TERM factorization, we can easily obtain the corresponding SERM factorization. In [7], the authors prove that a matrix **A** has a TERM factorization of **A** = **PLUS**$_0$ if and only if det(**A**) = ±1, where L,

U and $\mathbf{S}_0$ are unit lower TERM, unit upper TERM and a SERM (also a TERM). In other word, a matrix can be factorized into at most three TERMs besides a possible permutation matrix.

Furthermore, an matrix can be factorized into two unit SERMs if and only if all the minors of leading principal submatrices are 1s. Then, we have

$$A=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}=\begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & A_{22}-A_{21}A_{11}^{-1}A_{12} \end{bmatrix}\begin{bmatrix} A_{11} & A_{12} \\ 0 & I \end{bmatrix}. \tag{3.13}$$

where $\det(\mathbf{A}_{11}) = 1$, and the two factorized block matrices do not change the minors of each other's leading principal submatrices. Hence, the following factorization can be carried through：

$$\begin{aligned} A &= LU = (LUS_1^{-1})S_1 = (LUS_1^{-1}S_2^{-1})S_2S_1 \\ &= \cdots = (LUS_1^{-1}S_2^{-1}\cdots S_{N-1}^{-1})S_{N-1}\cdots S_2S_1 = S_N\cdots S_2S_1. \end{aligned} \tag{3.14}$$

Therefore, matrix **A** has a SERM factorization of $\mathbf{A}=\mathbf{P}\mathbf{S}_N\mathbf{S}_{N-1}\ldots\mathbf{S}_1\mathbf{S}_0$ if and only if $\det(\mathbf{A}) = \det(\mathbf{P}) = \pm 1$, where $S_m(m=0, 1, \ldots, N)$ are unit SERMs.

The transform matrix **A** of RGB to YC_bC_r is taken as an example here. As step in (3.6)-(3.12), we have

$$\mathbf{A}=\begin{bmatrix} 0.229 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix}. \tag{3.15}$$

$$\begin{aligned} \mathbf{S_1}&=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 6.7632 & 0 & 1 \end{bmatrix}, \qquad \mathbf{S_2}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 24.0419 & 1 \end{bmatrix} \\ \mathbf{L_1}&=\begin{bmatrix} 1 & 0 & 0 \\ -3.2126 & 1 & 0 \\ 0.0478 & 0 & 1 \end{bmatrix}, \qquad \mathbf{L_1}=\begin{bmatrix} 1 & 0 & 0 \\ -3.2126 & 1 & 0 \\ 0.0478 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.16}$$

Then, we got

$$\mathbf{L_2L_1AS_1S_2} = \begin{bmatrix} 1 & 3.3278 & 0.114 \\ 0 & 0.9991 & 0.1338 \\ 0 & -0.0019 & 0.2197 \end{bmatrix} \cong \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.2197 \end{bmatrix} \begin{bmatrix} 1 & 3.3278 & 0.1140 \\ 0 & 1 & 0.1338 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{DU}. \tag{3.17}$$

Therefore, the matrix **A** can be represented as a production of a lower TERM, an upper TERM, a diagonal matrix (if det(**A**) is not 1) and a SERM.

$$\begin{aligned} \mathbf{A} &= (\mathbf{L_2L_1})^{-1}\mathbf{DU}(\mathbf{S_1S_2})^{-1} \\ &\cong \begin{bmatrix} 1 & 0 & 0 \\ 3.2126 & 1 & 0 \\ -7.139 & -2.2073 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.2197 \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & 3.3278 & 0.1140 \\ 0 & 1 & 0.1338 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -6.7632 & -24.0419 & 1 \end{bmatrix} = \mathbf{LDUS_0}. \end{aligned} \tag{3.18}$$

At last, we use (3.14) to convert **A** into a series of SERMs.

$$\begin{aligned} \mathbf{D^{-1}A} &\cong \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.0478 & -2.2073 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 3.2126 & 1 & 0.1338 \\ 0 & 0 & 1 \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & 3.3278 & 0.1140 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -6.7632 & -24.0419 & 1 \end{bmatrix} = \mathbf{S_3S_2S_1S_0}. \end{aligned} \tag{3.19}$$

Chapter 4

Integer Color Transform

In [6], this is the first paper discussing “reversible,” so there is necessity to review color space choice by integer color transform. The color transform is used to re-represent the pic-

ture data in a form more suitable for the following compression. The actual compression is then performed in other representation. After decompression, the transformed colors are transferred back to the original color space. Since R, G and B data of ordinary images are highly correlated, it is reasonable to try to exploit this preliminary Knowledge. This can be done by compression scheme that take into account this correlation. However, the use of any transform inevitably bring along disadvantages too：image quality degradation in all practical cases, time loss and additional complexity.

The optimal choice for color transform would satisfy the following conditions：

- Decorrelate the data as much as possible.
- Minimize the total number of bits in the data path needed for a certain quality level.
- Need no calculation with critical accuracy.

There are many popular color spaces, such as Atd, YIQ and YC_bC_r shown in (40) [17]. There are even a linear transform that completely removes global statistical dependence between its components. This orthogonal transform is known as the Karhunen-Loeve Transform (KLT), also called the Hotelling Transform shown in (41). In this tutorial, we focus on "integer" and "reversible," so we do not discuss the de-correlation here. In the following, reversibility and word-length for transformation coefficients are discussed.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.334 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.1}$$

$$\begin{bmatrix} L \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0.54933 & 0.60238 & 0.57912 \\ 0.80429 & -0.19322 & -0.56194 \\ 0.22661 & -0.7747 & -0.59063 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.2}$$

4.1 Reversibility

A choice of color spaces is based on the finite register length that effects the reversibility, i.e. the differences RGB - RGB_{rec}.

The factors in reversibility is consisted of two part：

- The determination of the number of bits needed to represent the data in a particular color space.
- The determination of the number of bits needed to represent the coefficients (for the multiplier) and the intermediate results (between the multiplier and adder).

Saturation can only occur in the reverse transformation because of the possibly large rounding errors in the compression color space. In order words, the original color space cannot be perfected reconstructed due to the limit of bit length. There are detail experiments about how many bits are enough in [], and it highly depends on the human vision and the types of image.

4.2 Word-Length for Transformation Coefficients

4.2.1 Quantization of Transform Coefficients

In this subsection, we discuss the reconstruction errors. First, we have:

$$v = T \bullet R_{ori}, \tag{4.3}$$

$$R_{rec} = T^{-1} \bullet v, \tag{4.4}$$

with R_{ori} and R_{rec} the original and reconstructed RGB values, T and T^{-1} the forward and reverse transformation matrix, and v the color values in the compression color space.

This implies：

$$\begin{aligned} R_{rec} - R_{ori} &= ((T^{-1} \bullet T) - I) \bullet R_{ori} \\ &= 0 \end{aligned}, \tag{4.5}$$

with I the unity matrix.

Now, both the matrix T and T^{-1} will be replaced by a quantized version.

$$v = Q_f(T) \bullet R_{ori}, \tag{4.6}$$

$$R_{rec} = Q_r(Q_f(T)^{-1}) \bullet v. \tag{4.7}$$

This results in a reconstruction error：

$$\begin{aligned} R_{rec} - R_{ori} &= ((Q_r(Q_f(T)^{-1}) \bullet Q_f(T)) - I) \bullet R_{ori} \\ &= E \bullet R_{ori}. \end{aligned} \tag{4.8}$$

From this last expression we can conclude that：

- The reconstruction error is proportional to the original R_{ori}.
- The quantization of the forward transformation alone has no effect on the reconstruction error. Perturbations are compensated in the inverse matrix.
- The quantization of the reverse transformation is not compensated for in any way.

4.2.2 The Criterion of the Determinant of the Transform

In this subsection, authors try to derive a condition under the inverse matrix quantized with Q_1 bits right of the binary point representable without errors by using coefficients with Q_2 bits for the fractional part.

Let F be the forward transformation matrix with Q_1 bits right of the binary point and R the reverse transformation matrix with Q_2 for its fractional part.

$$F \xrightarrow{Inv} R \tag{4.9}$$

$$\frac{1}{2^{Q_1}} F_{\text{int}} \xrightarrow{Inv} \frac{1}{2^{Q_2}} R_{\text{int}} \tag{4.10}$$

$$\frac{1}{2^{Q_1+Q_2}} F_{\text{int}} \xrightarrow{Inv} R_{\text{int}} \tag{4.11}$$

F_{int} and R_{int} are matrices with integer coefficients. An element of R_{int} can be explicit as follows：

$$r_{\text{int}} = \frac{\text{minor}_{ij}(\frac{1}{2^{Q_1+Q_2}} F_{\text{int}})}{\det(\frac{1}{2^{Q_1+Q_2}} F_{\text{int}})} = \frac{2^{Q_1+Q_2} \bullet \text{minor}_{ij}(F_{\text{int}})}{\det(F_{\text{int}})}. \tag{4.12}$$

Therefore, the necessary condition for the inverse of F to be exactly representable with Q_2 bits right of the binary point, is that for all i, j

$$\frac{2^{Q_1+Q_2} \bullet \text{minor}_{ij}(F_{\text{int}})}{\det(F_{\text{int}})}, \tag{4.13}$$

is integer, or $\det(F_{int})$ is a divisor of $2^{Q_1+Q_2} \bullet \text{minor}_{ij}(F_{\text{int}})$ for all i, j.

A sufficient condition is that $\det(F_{int})$ is a divisor of $2^{Q_1+Q_2}$. This means that

$$\det(F_{\text{int}}) = \pm 2^i \quad \text{with } 0 \le i \le Q_1 + Q_2, \tag{4.14}$$

$$\det(F) = \pm \frac{2^i}{2^{NQ_1}} \quad \text{with } N = \text{matrix dimention.} \tag{4.15}$$

Consequently, under this criterion, we can assure that the inverse matrix is representable with Q_1 and Q_2 bits right of the binary point respectively.

4.3 Simple Transform

Under the criterion above, a simple transform is constructed without multiplication. We call it SIM (from SIMple)：

$$\begin{bmatrix} S \\ I \\ M \end{bmatrix} = \begin{bmatrix} 0.25 & 0.50 & 0.25 \\ -0.50 & 0.00 & 0.50 \\ -0.25 & 0.50 & -0.25 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{4.16}$$

Its inverse transform is equally simple：

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} S \\ I \\ M \end{bmatrix}. \tag{4.17}$$

The SIM transform convert RGB color space into the SIM color space. Although the performance of the SIM color space is worse than RGB space, no floating-point processor is needed in the transform. Then the perfect reconstruction is reachable, and the cost and time spending is much less in operation.

For another example, the color transform used in the JPEG-2000 is shown in (4.18), and (4.19) is the inverse [21]. This transform is a lossless color transform with takes N-bit RGB components in and produces N-bit Y component and N+1 bit Ur and Vr components. The components can be inverted to the oringal N-bit RGB values without loss.

$$\begin{cases} Y_r = \left\lfloor \dfrac{R + 2G + B}{4} \right\rfloor \\ U_r = R - G \\ V_r = B - G \end{cases}, \tag{4.18}$$

$$\begin{cases} G = Y_r - \left\lfloor \dfrac{U_r + V_r}{4} \right\rfloor \\ R = U_r + G \\ B = V_r + G \end{cases}. \tag{4.19}$$

4.4 Reversible Integer Color Transform

In Section 4.2, however, the criterion is coarse that still causes strong distortion. In [7], the authors introduce a systematic algorithm that can convert any 3 by 3 color transform into a reversible integer-to-integer transform and also discuss the ways to improve accuracy and reduce implementation complexity. In this section, we briefly discuss how to generate reversible integer color transform based on matrix decomposition [18]-[19]. There are five goals for the reversible integer color transform.

- The integer color transform should be reversible.
- No floating-point processor is required for both the forward and the inverse transforms.
- The bit-length of the output should be constrained.
- Less complexity for implementation
- Accuracy: The integer transforms should well approximate the original transform.

There are five advantages in the integer color transform：1) the ways to save the number of time cycles for implementation. 2) The method to analyze the accuracy by normalized root mean square error are proposed. 3) The closed-form solutions of coefficients are shown. 4) Instead of ± 1, the entries of diagonal matrices can be $\pm 2^k$, so it is more general and accuracy. 5) The derived integer transforms that are optimal in accuracy successfully.

4.4.1 Decomposition and Integerization

Step 1：Normalize the original 3×3 color transform A0 as A such that det(A)= ± 1.

$$A = \sigma A_0, \quad \text{where } \sigma = |\det(A_0)|^{-1/3}. \tag{4.20}$$

Step 2：Perform permutation, scaling, and sign changing operations for **A**.

$$C = D_1^{-1} P_1 A P_2 D_2^{-1}, \tag{4.21}$$

where $\mathbf{P_1}$ is a row-permuting matrix and $\mathbf{P_2}$ is a column-permuting matrix. $\mathbf{D_1}$ and $\mathbf{D_2}$ are diagonal matrices.

$$\begin{aligned} &D_1[m,m] = \pm 2^{-km}. \\ &D_2[m,m] = \pm 2^{km}. (m = 1, 2, \ and \ 3) \end{aligned} \tag{4.22}$$

$$D_{1 \ or \ 2}[m,n] = 0, \ when \ m \neq n. \tag{4.23}$$

$\mathbf{D_1}$ and $\mathbf{D_2}$ have the effects of scaling and sign changing. Note that there are 3! choices for $\mathbf{P_1}$ and $\mathbf{P_2}$, and 2^6 choices for the signs of the diagonal entries of $D_{1 \text{ or } 2}$[m, m]. If we do not consider the case where det(**C**) = -1 and constrain that

$$0 \leq k_m \leq p_k, \tag{4.24}$$

then, in sum, there are

$$(3!)^2 (p_k + 1)^3 2^6 / 2 = 1152 (p_k + 1)^3 \tag{4.25}$$

choices for **C**.

Step 3：Applying the lifting scheme, the single-row elementary reversible matrix scheme [4], [20] and several modification to decompose **C** into four one-row matrices.

$$C = T_4 T_3 T_2 T_1. \tag{4.26}$$

$$T_1 = \begin{bmatrix} 1 & t_1 & t_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ where \ t_1 = (c_{22} - 1) / c_{21}. \tag{4.27}$$

$$t_2 = -(t_1 z_2 + z_1). \tag{4.28}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}^{-1} \begin{bmatrix} -c_{23} \\ 1 - c_{33} \end{bmatrix}. \tag{4.29}$$

where c_{mn} are the entries of **C**.

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ t_3 & 1 & t_4 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.30}$$

where $t_3 = c_{21}$ and $t_4 = -z_2$.

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_5 & t_6 & 1 \end{bmatrix}, \tag{4.31}$$

where $t_5 = c_{22}c_{31} - c_{21}c_{32}$ and $t_6 = c_{32} - tc_{31}$.

$$T_4 = \begin{bmatrix} 1 & t_7 & t_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.32}$$

where $t_7 = c_{12} - t_1c_{11} - t_6t_8$ and $t_8 = c_{13} + z_1c_{11} + z_2c_{12}$.

Step 4：After truncation operation Q_r, we derive the reversible integer transform **B** that approximates **A** from

$$B = P_1^T D_1 V_4 V_3 V_2 V_1 D_2 P_2^T, \tag{4.33}$$

where

$$\begin{aligned} V_1 &= \begin{bmatrix} 1 & g_1 & g_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad V_2 = \begin{bmatrix} 1 & 0 & 0 \\ g_3 & 1 & g_4 \\ 0 & 0 & 1 \end{bmatrix}, \\ V_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ g_5 & g_6 & 1 \end{bmatrix}, \quad V_4 = \begin{bmatrix} 1 & g_7 & g_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{4.34}$$

where

$$g_n = Q_b(t_n), \quad n = 1,2,...8 \tag{4.35}$$

$$Q_b\left[\pm \sum_{n=-\infty}^{\infty} d_n 2^{-n}\right] = \pm \sum_{n=-\infty}^{b} d_n 2^{-n}, \ (d_n = 0 \ or \ 1). \tag{4.36}$$

where b is an integer.

$$V_1 = \begin{bmatrix} 1 & g_1 & g_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad V_2 = \begin{bmatrix} 1 & 0 & 0 \\ g_3 & 1 & g_4 \\ 0 & 0 & 1 \end{bmatrix},$$
$$V_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ g_5 & g_6 & 1 \end{bmatrix}, \quad V_4 = \begin{bmatrix} 1 & g_7 & g_8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.37}$$

Therefore, **B** is a binary matrix that approximates **A**, and its inverse $\mathbf{B}^{-1}$ is shown in (4.34). Obviously, $\mathbf{B}\mathbf{B}^{-1}=1$.

$$B^{-1} = P_2 D_2^{-1} V_1 V_2 V_3 V_4 D_1^{-1} P_1. \tag{4.38}$$

4.4.2 Bit Constraint, Time Cycle Problem and Accuracy Analysis

Now, we try to satisfy Goal 3-5. If the least significant bit of g_n is 2^{-b}, and 2^{-a} and 2^{-c} are the least significant bits of x and z ($z = \mathbf{B}x$), then

$$c = a + 4b. \tag{4.39}$$

Thus, the bit-length of z is much longer than that of x. To solve this problem, we can convert $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$ and $\mathbf{V}_4$ in (4.33) into addition-truncation operations. For example, we can convert $x_2=\mathbf{V}_1 x1$ into

$$\begin{aligned} x_2[1] &= x_1[1] + Q_r\{g_1 x_1[2] + g_2 x_1[3]\}, \\ x_2[2] &= x_1[2],\ x_2[3] = x_1[3]. \end{aligned} \tag{4.40}$$

where Q_r is truncation operation. Therefore, the bit length is constrained ,and the reversibility is preserved.

There is another problem for implementing the integer color transform. That is, too many time cycles are required. Suppose that in each time cycle we can only do one addition and one multiplication in each entry. Quantization (just throwing bits), permuting (just twisting circuit in hardware), and multiplying 2^{-k} (just doing bit shifting) does not require any time

cycle. Then, each of $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, and $\mathbf{V}_4$ requires three time cycles. However, from the facts that 1) two adjacent one-row matrices can be implemented together and 2) if the least significant bit of x_1 is larger than one of $g_1 x_1[2] + g_2 x_1[3]$, then (4.36) is equivalent to

$$x_2[1] = Q_r\{x_1[1] + g_1 x_1[2] + g_2 x_1[3]\}. \tag{4.41}$$

Therefore, we can reduce the number of time cycles into 5. Due to the proof is trivial, readers can refer to the original paper.

Finally, we discuss the problem of accuracy, i.e., Goal 5. Note that, in Steps 2)–5), the truncation operation Q_r^{-k} is equivalent to adding a small number

$$Q_{r-k_m}\{a\} = a + \tau, \quad where \ -2^{-r+k_m-1} \le \tau \le 2^{-r+k_m-1}, \tag{4.42}$$

and

$$E[\tau] = 0, \quad E[\tau^2] = 4^{-r+k_m}/12. \tag{4.43}$$

Then, due to (4.32), we have

$$z = P_1^T D_1(V_4\{V_3[V_2(V_1 D_2 P_2^T x + \Delta_1) + \Delta_2] + \Delta_3\} + \Delta_4). \tag{4.44}$$

Therefore

$$\begin{aligned} z - y = {} & P_1^T D_1 V_4 V_3 V_2 \Delta_1 + P_1^T D_1 V_4 V_3 \Delta_2 + P_1^T D_1 V_4 \Delta_3 \\ & + P_1^T D_1 \Delta_4 + P_1^T D_1 (V_4 V_3 V_2 V_1 - T_4 T_3 T_2 T_1)_1 D_2 P_2^T. \end{aligned} \tag{4.45}$$

Suppose that b in (4.31) is large enough such that $\mathbf{V}_n = \mathbf{T}_n + \bigtriangledown_n$, where the entries of $\bigtriangledown_n$ is very small compared with those of T_n.

$$\begin{aligned} V_4 V_3 V_2 V_1 - T_4 T_3 T_2 T_1 &= (T_4 + \nabla_4)(T_3 + \nabla_3)(T_2 + \nabla_2)(T_1 + \nabla_1) - T_4 T_3 T_2 T \\ &= T_4 T_3 T_2 \nabla_1 + T_4 T_3 \nabla_2 T + T_4 \nabla_3 T_2 T + \nabla_4 T_3 T_2 T. \end{aligned} \tag{4.46}$$

If b is very large, $\nabla_1 \nabla_2 \nabla_3$ and ∇_4 are very small, and there are four terms could be ignored.

$$z - y \approx P_1^T D_1 T_4 T_3 T_2 \Delta_1 + P_1^T D_1 T_4 T_3 \Delta_2 + P_1^T D_1 T_4 \Delta_3 + P_1^T D_1 \Delta_4. \tag{4.47}$$

For example, for the case of the integer KLA transform, when $r = 0$ and $b = 8$, the last four

terms affect only 8.05% of error. When r = 0 and b = 12, these terms contribute only 0.07% of error.

4.4.3 Experiment Results

In the following, we try to convert the ten color transforms into reversible integer transforms, and shown in Table 4.1 and Table 4.2, where g_k (k=1~8), $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{P}_1$ and $\mathbf{P}_2$ are the parameters, diagonal matrices, and permuting matrices using in the processes.

(1) RGB to KLA

$$\begin{bmatrix} 0.8185 & 0.8975 & 0.8629 \\ 1.19847 & -0.2879 & -0.8373 \\ 0.3376 & -1.1539 & -0.8800 \end{bmatrix}$$

(2) RGB to IV_1V_2

$$\begin{bmatrix} 1 & 1 & 1 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 1/\sqrt{6} & -1/\sqrt{6} & 0 \end{bmatrix}$$

(3) RGB to XYZ

$$\begin{bmatrix} 0.8706 & 0.2496 & 0.2868 \\ 0.4288 & 0.8419 & 0.1635 \\ 0 & 0.0947 & 1.6606 \end{bmatrix}$$

(4) RGB to UVW

$$\begin{bmatrix} 0.8387 & 0.2402 & 0.2754 \\ 0.6192 & 1.2156 & 0.2361 \\ 0.3003 & 1.7125 & 1.2984 \end{bmatrix}$$

(5) RGB to YIQ

$$\begin{bmatrix} 0.4722 & 0.9270 & 0.1800 \\ 0.9412 & -0.4327 & -0.5058 \\ 0.3332 & -0.8259 & 0.4927 \end{bmatrix}$$

(6) RGB to DCT

$$\begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \\ 0.7071 & 0 & -0.7071 \\ 0.4082 & -0.8165 & 0.4082 \end{bmatrix}$$

(7) RGB to YC_bC_r

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.500 & -0.419 & -0.081 \\ -0.169 & -0.331 & -0.500 \end{bmatrix}$$

(8) RGB to YUV

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.1 \end{bmatrix}$$

(9) RGB to $R_cG_cB_c$

$$\begin{bmatrix} 1.609 & -0.447 & -0.104 \\ -0.058 & 0.977 & 0.051 \\ -0.025 & -0.037 & 1.162 \end{bmatrix}$$

(10) RGB to $R_sG_sB_s$

$$\begin{bmatrix} 1.167 & -0.146 & -0.151 \\ 0.114 & 0.753 & 0.159 \\ -0.001 & 0.059 & 1.128 \end{bmatrix}$$

Table 4.1 Parameters of the reversible integer color transforms (part 1)

	RGB to KLA	RGB to IV_1V_2	RGB to XYZ	RGB to UVW	RGB to YIQ
g_1	13/32	0	-175/128	-75/256	35/256
g_2	641/256	-47/128	-89/256	313/256	227/256
g_3	-115/256	1/2	205/256	141/256	-111/128
g_4	-77/128	-209/256	71/256	-25/128	-81/256
g_5	21/64	-157/256	95/256	7/256	-81/256
g_6	73/128	209/256	-27/256	97/256	57/256
g_7	-57/256	-101/128	-349/256	57/128	51/128
g_8	193/256	247/256	57/64	29/128	-31/128
$\mathbf{P}_1^{\mathrm{T}}\mathrm{x}\mathbf{D}^1$	$\begin{bmatrix} 0 & 2^{-1} & 0 \\ 0 & 0 & 1 \\ 2^{-2} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2^{-1} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 2^{-1} \\ 2^{-1} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 2^{-1} & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2^{-1} & 0 & 0 \\ 0 & 2^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$\mathbf{D}^2\mathrm{x}\mathbf{P}_2^{\mathrm{T}}$	$\begin{bmatrix} 0 & -4 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
NRMSE	0.1609%	0.1842%	0.2156%	0.1291%	0.2703%

Table 4.2 Parameters of the the reversible integer color transforms (part 2)

	RGB to DCT	RGB to YC_bC_r	RGB to YUV	RGB to $R_cG_cB_c$	RGB to $R_sG_sB_s$
g_1	-215/1024	115/256	289/1024	53/128	139/1024
g_2	1313/1024	47/256	343/1024	325/1024	227/512
g_3	-107/512	-341/1024	99/1024	115/512	-81/256
g_4	221/256	-209/512	-347/512	-181/256	-443/1024
g_5	-857/1024	1	-137/256	-209/512	-423/512

g_6	1047/1024	141/256	-125/1024	-254/1024	57/128
g_7	-149/1024	119/1024	201/1024	53/128	205/512
g_8	-7/1024	-557/1024	-79/512	325/1024	-31/256
$\mathbf{P}_1^{\mathrm{T}}\mathrm{x}\mathbf{D}^1$	$\begin{bmatrix} 2^{-1} & 0 & 0 \\ 0 & 2^{-1} & 1 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2^{-1} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 2^{-1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 2^{-1} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 2^{-1} & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2^{-1} \end{bmatrix}$
$\mathbf{D}^2\mathrm{x}\mathbf{P}_2^{\mathrm{T}}$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & -2 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 2 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$
NRMSE	0.1609%	0.1842%	0.2156%	0.1291%	0.2703%

Chapter 5 Reversible Jacket Transform

In this chapter, we survey the class of Reverse Jacket transform (RJT), where the Walsh-Hadamard transform (WHT) and weighted Hadamard transform are the special cases. The WHT and discrete Fourier transform are widely used in the field of signal processing [8], [9]. The weighted Hadamard transform is a special case of the WHT, where the partial entries should not have to be ± 1. As our two side jacket is an inside and outside compatible, at least two positions of a Reverse Jacket matrix are replaced by their inverse; this elements are changed and their position are moved, for example from inside of the middle circle to outside or from to inside without loss of signs [10]. In multidimensional subsampling, the Reversible Jacket transform can be used in Quincunx and various polygonal subsampling, PCI signal processing, and information theory.

5.1 The Walsh-Hadamard Matrix

A Walsh-Hadamard matrix of order n is an $n \times n$ matrix of +1's and -1's such that $H_n H_n^T = nI$; the set of all Walsh-Hadamard matrices of order n is denoted by H_n.

The 2×2 Walsh-Hadamard matrix is defined as

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{5.1}$$

The $2^n \times 2^n$ Walsh-Hadamard matrix can be defined recursively using (5.1) as follows:

$$\begin{aligned} H_{2^n} &= H_2 \otimes H_{2^{n-1}} = H_{2^{n-1}} \otimes H_2 \\ &= H_2 \otimes H_2 \otimes \cdots \otimes H_2 \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \end{aligned} \tag{5.2}$$

The symbol $\otimes$ represents the Kronecker product.

Because of the orthogonality of Hadamard matrices, there are eight different Hadamard matrices of order 2:

$$\begin{aligned} H_2^{(1)} &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_2^{(2)} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \\ H_2^{(3)} &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad H_2^{(4)} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}, \\ H_2^{(5)} &= \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}, \quad H_2^{(6)} = \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix}, \\ H_2^{(7)} &= \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}, \quad H_2^{(8)} = \begin{bmatrix} 1 & -1 \\ -1 & -1 \end{bmatrix}. \end{aligned} \tag{5.3}$$

The Hadamard transform is an orthogonal transform and is highly practical in signal processing, especially in data compression. The reason for the practicality of the WHT is the fact that the elements of the Walsh-Hadamard matrix are either +1 or −1, and thus there are only additions and subtractions in the computation. Without multiplication, computation time is shortened and hardware cost is reduced.

5.2 The Weighted Hadamard Matrix

The Walsh-Hadamard matrix is an orthogonal matrix but the weighted Hadamard matrix is a nonorthogonal symmetric matrix and slightly different to the Walsh-Hadamard matrix. The weighted Hadamard matrix is generated by weighting center portion of the original Hadamard matrix. The lowest order weighted Hadamard matrix is of size 4 × 4 and is defined as follows：

$$[W]_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -w & w & -1 \\ 1 & w & -w & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{5.4}$$

The inverse of (5.4) is

$$[W]_4^{-1} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -w^{-1} & w^{-1} & -1 \\ 1 & w^{-1} & -w^{-1} & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{5.5}$$

The weighted Hadamard transform of an $N \times 1$ vector $[f]$ and $N \times N$ (image) matrix $[g]$ are given by

$$[F] = [W]_N [f]. \tag{5.6}$$

$$[G] = [W]_N [g][W]_N. \tag{5.7}$$

As with the Hadamard matrix, the recursive relation also can be applied to the weighted Hadamard matrix to generate higher order one (5.8).

$$[W]_N = [W]_{N/2} \otimes [H]_2, \tag{5.8}$$

where $[H]_2$ is the lowest Walsh Hadamard matrix given by (5.1).

The fast algorithm of the weighted Hadamard transform is related to the fast Hadamard transform. It can be derived by decomposing $[H]_N$ into a product of k sparse matrices, each

having rows/columns with only two nonzero elements [11].

$$[RC]_N = [H]_N [WH]_N , \tag{5.9}$$

where $[RC]_N$ is a coefficient matrix. Since $[H]_N^{-1} = 1/N[H]_N$, we have

$$[WH]_N = \frac{1}{N}[H]_N [RC]_N . \tag{5.10}$$

5.3 The Reverse Jacket Matrix

The Reverse Jacket matrix is a generalized weighted Hadamard matrix and has recursive structure and symmetric characteristics. A $2^k \times 2^k$ matrix $[RJ]_{2^k}$ is said to be a Reverse Jacket matrix with respect to the Hadamard matrix H_{2^k} if

$$[RJ]_{2^k} = H_{2^k}^{-1}[RJ]_{2^k} H_{2^k} . \tag{5.11}$$

The general form of the Reverse Jacket matrices is derived in [10] in follows：

$$[RJ]_2 = \begin{bmatrix} a & b \\ b & -c \end{bmatrix}, \tag{5.12}$$

$$[RJ]_4 = \begin{bmatrix} a & b & b & a \\ b & -c & c & -b \\ b & c & -c & -b \\ a & -b & -b & a \end{bmatrix}. \tag{5.13}$$

The inverse matrices of the Reverse Jacket transforms are

$$[RJ]_2^{-1} = signum(a \bullet c) \begin{bmatrix} a & b \\ b & -c \end{bmatrix}, \tag{5.14}$$

$$[RJ]_4^{-1} = lcm(a,b,c) \begin{bmatrix} 1/a & 1/b & 1/b & 1/a \\ 1/b & -1/c & 1/c & -1/b \\ 1/b & 1/c & -1/c & -1/b \\ 1/a & -1/b & -1/b & 1/a \end{bmatrix}, \tag{5.15}$$

where lcm is the least common multiple. The interesting phenomenon is that the element positions of the matrix can be replaced by its inverse matrix and the signs of them are not changed.

For the higher order reversible Jacket matrix, the recursive method still works out.

$$[RJ]_{2^{k+1}} = H_2 \otimes [RJ]_{2^k} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes [RJ]_{2^k},\ k \geq 2. \tag{5.16}$$

The fast algorithm for the reversible Jacket transform is similar to the weighted Hamadard transform. We first define a coefficient matrix as follows

$$[RC]_{2^k} = H_{2^k}[RJ]_{2^k},\ k \geq 2. \tag{5.17}$$

Then the inverse matrix of the Reverse Jacket transform can be written as

$$\begin{aligned} [RC]_{2^{k+1}} &= H_{2^{k+1}}[RJ]_{2^{k+1}} \\ &= H_{2^{k+1}}(H_2 \otimes [RJ]_{2^k}) \\ &= (H_2 \otimes H_{2^k})(H_2 \otimes [RJ]_{2^k}) \\ &= 2I_2(H_{2^k} \otimes [RJ]_{2^k}) \\ &= 2I_2 \otimes [RC]_{2^k}. \end{aligned} \tag{5.18}$$

Chapter 6 Conclusion

For traditional discrete transforms, floating point processors are required. So they cost more to implement and cannot be reconstructed perfectly. A well designed integer transform can solve all the problems. There is a trade-off between bit constrain and performance for the integer transform. Based on the lifting scheme, perfect reconstruction can be achieved practically, and the integer transform progresses rapidly. Moreover, the Reverse Jacket transform,

which includes the Walsh-Hadamard transform and weighted Hadamard transform, is also an integer transform developed recently, and can be applied to many field, such as pattern recognition, CDMA, and information encryption.

References

Based on prototype：

[1] W. K. Cham, "Development of integer cosine transform by the principles of dynamic symmetry," *Proc. Inst. Elect. Eng.*, pt. 1, vol. 136, no. 4, pp. 276-282, Aug. 1989.

[2] S. N. Koh, S. J. Huang, and H. K. TANG, "Development of order-16 integer transforms," Signal Processing, vol. 24, Issue 3, September 1991, pp. 283- 289.

[3] W. K. Cham and P. P. C. YIPS, "Integer sinusoidal transforms for image processing," *Int. J. Electrics*, vol. 70, no. 6, 1991, pp.1015-1030.

[4] W. K. Cham, and R. J. Clarke,: 'Dyadic symmetry and Walsh matrices', *1EE Proc. F, Commun., Radar* & *Signal Process.,* vol. 134, no. 2, pp. 141-144, 1987

[5] W. K. Cham and Y. T. Chan, "Integer discrete cosine transforms", ISSPA 87, Australia, August 1987, pp.674-676, Aug. 1987.

[6] W. K. Cham and R. J. Clarke, Application of the principle of dyadic symmetry to the generation of the orthogonal transforms. *Proceedings of the Institution of Electrical Engineers,* Pt F, 133, 264-270, 1986.

Based on lifting scheme and matrix factorization：

[7] P. Hao and Q. Shi., "Matrix Factorizations for Reversible Integer Mapping", *IEEE Trans. Signal Processing*, vol. 49, no. 10, p 2314-2324, Oct. 2001.

[8] Q. Shi, "Biorthogonal wavelet theory and techniques for image coding," *Proc. SPIE,* pp. 24–32, Oct. 1998.

[9] S. C. Pei and J. J. Ding, “Reversible integer color transform,” *IEEE Trans. Image Processing*, vol. 16, no. 6, pp. 1686-1690, June 2007.

[10] F. A. M. L. Bruekers and A. W. M. van den Enden, “New networks for perfect inversion and perfect reconstruction,” *IEEE J. Select. Areas Commun.*, vol. 10, pp. 130–137, Jan. 1992.

[11] Y. Chen and P. Hao, “Integer reversible transformation to make JPEG lossless,” in *Proc. Int. Conf. Signal Processing*, pp. 835–838, 2004.

[12] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, “Integer fast Fourier transform,”*IEEE Trans. Signal Processing*, vol. 50, no. 3, pp. 607–618, Mar. 2002.

[13] P. Hao and Q. Shi, “Comparative study of color transforms for image coding and derivation of integer reversible color transform,” in *Proc. Int. Conf. Pattern Recognition*, vol. 3, pp. 224–227, Sep. 2000.

Jacket Transform：

[14] K. G. Beauchamp, *Walsh Functions and Their Applications*. New York: Academic, 1975.

[15] M. H. Lee, “On the reverse Jacket matrix for weighted Hadamard transform,” *IEEE Trans.Circuit Syst.—Part II*, vol. 45, no. 3, Mar 1998.

[16] M. H. Lee and M. Kaveh, “Fast Hadamard transform based on a simple matrix factorization,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1666–1667, 1986.

[17] M. H. Lee, J. Y. Park, M. W. Kwon, and S.-R. Lee, “The reverse Jacket matrix of weighted Hadamard transform for multidimensional signal processing,” in *Proc. PIMRC'96*, Taipei, Taiwan, R.O.C., pp. 482–486, 1996.

Others：

[18] G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data*

and Formulae, John Wiley & Sons, Inc., 1982.

[19] B. Deknuydt, J. Smolders, L. Van Eycken, and A. Oosterlinck, "Color Space Choice for Nearly Reversible Image Compression," in Visual *Communications and Image Processing*, Proc. SPIE, vol. 1818, 1992, pp. 1300-1311.

[20] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. New York: Springer-Verlag, 1975

[21] M. J. Gormish, E. L. Schwartz, A. Keith, et al, "Lossless and nearly lossless compression for high quality images," *Proceeding of the SPIE / IS&T Conference on Very High Resolution and Imaging II*, vol. 3025, pp. 62-70, Feb. 1997.