Image Compression and Encryption Using Discrete Fractional Cosine Transform

A Thesis

Submitted in the partial fulfillment of requirement for the award of the degree of

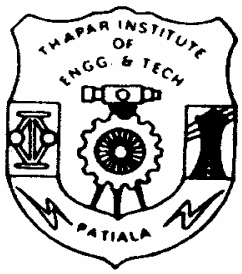
Master of Engineering
In
Electronics and Communication

by

Navdeep Singh
Regn. No. 8024115

Under the guidance of

Mr. Kulbir Singh
Lecturer
Department of Electronics &
Communication Engineering



Department of Electronics & Communication Engineering
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Deemed University)
PATIALA-147004

ACKNOWLEDGEMENT

The real sprit of achieving a goal is through the way of excellence and austerous discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

With deep sense of gratitude I express my sincere thanks to my esteemed and worthy supervisor, Mr. Kulbir Singh, Lecturer, Department of Electronics and Communication Engineering, for his valuable guidance in carrying out this work under his effective supervision, encouragement, enlightenment and cooperation.

I shall be failing in my duties if I do not express my deep sense of gratitude towards Dr. R.S. Kaler, Prof. & Head of the Deptt. Of Electronics & Communication Engineering Technology (Deemed University), Patiala who has been a constant source of inspiration for me throughout the thesis semester.

I am also thankful to all the staff members of the Electronics & Communication Engineering Department for their full cooperation and help.

The technical guidance and constant encouragement made it possible to tied over the numerous problems, which so ever came up during the study. My greatest thanks to all who wished me success. Above all I render my gratitude to the ALMIGHTY who bestowed self-confidence, ability and strength in me to complete this work.

(Navdeep Singh)

ABSTRACT

Image processing most frequently uses the transforms like Discrete Cosine Transform or Fourier transform for image compression. This involves the conversion of the source signal to frequency component. The application of Fractional version of cosine transform called Discrete Fractional Cosine Transform) to image compression is fairly new. DFRCT shares many useful properties of regular DCT & it has free parameter, its fraction. When it is 0 we get cosine modulated version of input signal. When it is unity we get conventional DCT. As the fraction changes from 0 to 1 we get different forms of signal which interpolate between cosine modulated form of signal & its DCT signal representation. Due to its free parameter its 'a' DRFCT may find its place in many applications like image encryption & image compression where DCT is found to be useful. DRFCT provides low Mean Square Error & high Peak Signal Noise Ratio as compare to DCT for image compression. Another key advantage of DRFCT is that it is used for image encryption using random phase masking. This technique provides more secure image encryption & low value of Mean Square Error, with the addition of extra degree of keys provided by the fractions of DRFCT.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Certificate	i
	Acknowledgement	ii
	Abstract	iii
	Contents	iv
	List of Tables	vi
	List of Figures	vii
	Abbreviations Used	x
1.	**CONCEPTS OF COMPRESSION**	
	1.1 Image Compression	1
	1.1.1 Principle from behind compression	3
	1.1.2 Asymmetries in compression algorithms	4
	1.1.3 Image Compression System	5
	1.1.4 Compression Schemes	6
	1.2 Encryption	11
	1.3 Objective	12
	1.4 Organization	13
2.	**DISCRETE COSINE TRANSFORM**	
	2.1 Basic DCT Concepts	14
	2.1.1 The One-Dimension DCT	14
	2.1.2 Two Dimensional DCT	18
	2.2 DCT Integer Representation & Quantization	20
	2.3 Mathematical Definition of DCT	21
	2.3.1 The One Dimensional DCT	21
	2.3.2 The Two Dimensional DCT	26
3.	***DISCRETE FRACTIONAL COSINE TRANSFORM***	
	3.1 Fractional Operations	31

3.2 Fractional Cosine Transforms 32
3.3 Calculation of DRFCT Kernel 34
3.3.1 Eigenvectors & Eigenvalues of Kernel Matrices 36
4. IMAGE COMPRESSION USING DRFCT
4.1 Image Compression Algorithm 41
4.2 Characteristics to judge compression algorithm 41
4.2.1 Compression Ratio 41
4.2.2 Compression speed 42
4.2.3 Mean Square Error 43
4.2.4 Peak Signal to Noise Ratio 43
5. IMAGE ENCRYPTION USING DRFCT
5.1 Keys in DFRCT 44
5.2 Algorithm for Image Encryption 45
5.2.1 Encryption 45
5.2.2 Decryption 46
6. SIMULATION RESULTS
6.1 Simulation Results of Image Compression. 47
6.2 Simulation Results of Image Encryption. 52
7. CONCLUSIONS 96
REFERENCES 97

LIST OF ABBREVIATIONS

'a':	Fractional Order
CR:	Compression Ratio
CT:	Cosine Transform
DCT:	Discrete Cosine Transform
DFRCT:	Discrete Fractional Cosine Transform
DFRFT:	Discrete Fractional Fourier Transform
DFT:	Discrete Fourier Transform
FRCT:	Fractional Cosine Transform
FRFT:	Fractional Fourier Transform
FFT:	Fast Fourier Transform
FT:	Fourier Transform
MSE:	Mean Square Error
PSNR:	Peak Signal to Noise Ratio

Chapter 1

INTRODUCTION

Introduction

Digital image processing techniques are used to enhance the quality of image and to reduce the data needed to represent the image without visually affecting the quality of image. The objective of compression is to reduce the data volume and achieve reproduction of the original data without any perceived loss in data quality. Applications of compression include digital imaging. Compression of medical images for efficient use of storage space and transmission bit rate has become a necessity.

Cryptographic techniques are used to scramble images so that an adversary could not the original image without knowing the secret key. The objective of encryption is to transform data into an unreadable form to ensure privacy. Encryption was widely used in military imaging for secure transmission of information. Today, however, encryption is widely used by everyone to transmit information such as commercial transactions (exchange of credit card information, etc.).

In the case of the former, the "compressed" data needs to be uncompressed to the original form while in the case of the latter; the "encrypted" data needs to be decrypted to the original form.

1.1 IMAGE COMPRESSION

Today transmitting multimedia material in uncompressed form is completely out of question. The only hope is that massive compression is possible. Fortunately a large body of

research over past few decades has led to many compression techniques and algorithms that make multimedia transmission feasible. There is a problem of storing, transmitting and manipulating digital image because of the size of image files involved, transmitting images will always consume large amount of bandwidth and storing images will require hefty resources as shown in Table 1

Table 1.1 Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required.

Multimedia Data	Size/Duration	Bits/Pixel or Bits/Sample	Uncompressed Size (B for bytes)	Transmission Bandwidth (b for bits)	Transmission Time (using a 28.8K Modem)
A page of text	11" x 8.5"	Varying resolution	4-8 KB	32-64 Kb/page	1.1 - 2.2 sec
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Grayscale Image	512 x 512	8 bps	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512 x 512	24 bps	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bps	5.16 MB	41.3 Mb/image	23 min 54 sec
SHD Image	2048 x 2048	24 bps	12.58 MB	100 Mb/image	58 min 15 sec
Full-motion Video	640 x 480, 1 min (30 frames/sec)	24 bps	1.66 GB	221 Mb/sec	5 days 8 hrs

The examples above clearly illustrate the need for sufficient storage space, large transmission bandwidth and long transmission time for image, audio, and video data [1].

1.1.1 Principal behind Compression

In images the neighboring pixels are correlated and therefore contain redundant information [2]. The fundamental components of compression are redundancy and irrelevancy reduction. Redundancy means duplication and Irrelevancy means the parts of signal that will not be noticed by the signal receiver, which is the Human Visual System .

There are three types of redundancy can be identified:

- Spatial Redundancy i.e. correlation between neighboring pixel values.
- Spectral Redundancy i.e. correlation between different color planes or spectral bands.
- Temporal Redundancy i.e. correlation between adjacent frames in a sequence of images (in video applications).

Image compression focuses on reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies. The audio, video and image signals are available to be compressed, because there is a considerable redundancy in these signals. Since this thesis is about still image compression, therefore temporal redundancy is not relevant.

1.1.2 Asymmetries in compression Algorithms

There are certain asymmetries in compression algorithms that are important to understand:-

All compression systems require two algorithms, one for compressing the data at the source, and another for decompressing it at the destination. These algorithms are also referred as encoding and decoding algorithm. They have certain asymmetry like, for many applications a multimedia document, say, a movie only be encoded once but will be decoded thousands of time. This asymmetry means that it is acceptable for encoding algorithm to be slow and require

expensive hardware provided that decoding algorithm is fast and does not require expensive hardware. After all, the operator of a multimedia server might be quite willing to rent a parallel supercomputer for a few weeks to encode its entire video library, but requiring consumers to rent a supercomputer for 2 hours to view a video is not likely to be a big success. Many practical compression systems go to great lengths to make decoding fast and simple, even at the price of making encoding slow and complicated.

On the other hand, for real time multimedia, such as video conferencing, slow encoding is unacceptable. Encoding must happen on the fly, in real time. So, real-time multimedia uses different algorithms or parameters than storing videos on disk, often with appreciably less compression.

A second asymmetry is that the encode/decode process need not to invertible. That is, when compressing a file, transmitting it, and then decompressing it, the user expects to get the original back, accurate down to the last bit. With multimedia, this requirement does not exist. It is usually acceptable to have the video signal after encoding and then decoding is slightly different than the original. When the decoded output is not exactly equal to the original input, the system is said to be lossy. If the input and output are identical, the system is lossless. Lossy systems are important because accepting a small amount of information loss can give a huge payoff in terms of the compression ratio possible.

1.1.3 Image Compression System

Some of the most successful image compression techniques (both lossless and lossy) involve the following three major stages: image transformation, quantization

(lossy compression only) and encoding. Fig 1.1 shows the block diagram of simple compression system.

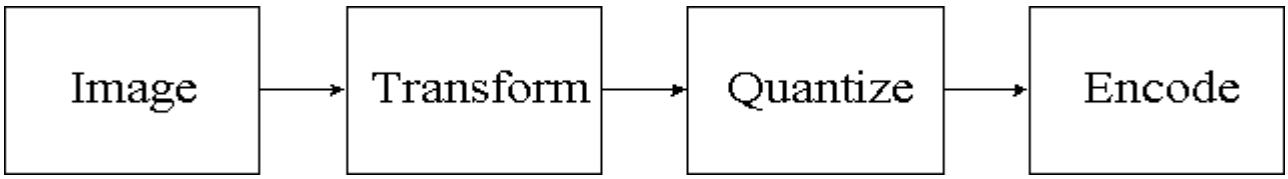


Fig 1.1: Block diagram of Simple compression System

(a) Transform

Image transform, sometimes referred to as decorrelation, is used to reduce the dynamic range of the signal, to eliminate redundant information, and to provide a suitable representation for entropy encoding.

A transform should satisfy three conditions [3]:

(i) All transform coefficients become statistically independent.

(ii) Energy of the transformed image is compacted into a minimum number of coefficients.

(iii) The transform coefficients are concentrated in minimum frequency or transform Scale regions

(b) Quantizer

A Quantizer reduces the precision of the values generated by the transform and therefore reduces the number of bits required to save the transform co-coefficients. Quantization is fundamentally lossy,the loss in decompressed image is due to this section.

(c) Entropy Encoder

An entropy encoder does further compress the quantized values. This is done to achieve even better overall compression. The various commonly used entropy encoders are the Huffman encoder, arithmetic encoder, and simple run-length encoder. For better performance with compression, it's important to have the best of all the three components.

1.1.4 Compression Schemes

Compression schemes can be divided into two general categories: entropy encoding and source encoding [4].

(a) Entropy Encoding

Entropy encoding just manipulates bit streams without regard to what the bits mean. It is a general, lossless, fully reversible technique, applicable to all data. .

The first type of entropy encoding is run-length encoding. In many kinds of data, strings of repeated symbols (bits, numbers etc.) are common. These can be replaced by a special marker not otherwise allowed in the data, followed by the symbol comprising the run, followed by how many times it occurred. If the special marker itself occurs in the data, it is duplicated (as in character stuffing). For example, consider the following string of decimal digits.

315000000000000845871111111111111635467000000000000065

If we now introduce A as the marker and use two digit numbers for the repetition count, we can encode the above digit string as

315A01284587A11316354674A02265

Here run length encoding has cut the data string in half.

Runs are common in multimedia. In audio, silence is often represented by runs of zeros. In video, runs of the same color occur in shots of the sky, walls, and many flat surfaces. All of these runs can be greatly compressed.

The second type of entropy encoding is statistical encoding. In this we use a short code to represent common symbols and long ones to represent infrequent ones. Morse code uses this principle, with E being and Q being… and so on. Huffman coding and the Ziv-Lempel algorithm used by the UNIX compress program also use statistical encoding.

The third type of entropy encoding is CLUT (Color Look up Table) encoding. Consider an image using RGB encoding with 3 bytes/pixel. In theory, the image might contain as many as 2^{24} different color values. In practice, it will normally contain many fewer values, especially if the image is a cartoon or computer generated drawing, rather than a photograph. Suppose that only 256 color values are actually used. A factor of almost three compressions can be achieved by building a 768 byte table listing the RGB values of the 256 colors actually used, and then representing each pixel by the index of its RGB value in the table. This is a clear example where encoding is slower than decoding because encoding requires searching the table whereas decoding can be done with a single indexing operation.

(b) Source Encoding

Source encoding, takes advantage of properties of the data to produce more (usually lossy) compression. Some examples of source encoding are, the first example is differential encoding, in which a sequence of values (e.g. audio samples) are encoded by representing each one as the difference from the previous value. Differential pulse code modulation is an example

of this technique. It is lossy because the signal might jump so much between two consecutive values that the difference does not fit in the field provided for expressing differences, so at least one incorrect value will be recorded and some information lost.

Differential encoding is a kind of source encoding because it takes advantage of the property that large jumps between consecutive data points are unlikely .Not all the sequence of numbers have this property

The other type of source encoding consists of transformations .By transforming signals from one domain to another, compression may become easier .If we take the Fourier transform of 1-D signal in which we represent a function of time as a list of amplitudes now if we give the exact value of all amplitudes, the original function can be reconstructed perfectly .However by only giving the values of, say first 8 amplitudes rounded to first two decimal places, the signal can still reconstructed so well that the listener can't tell that some information has been lost. The gain is that transmitting 8 amplitudes require many fewer bits than transmitting the sampled waveform.

Transformations are also applicable to 2-D image data. Suppose that 4 X 4 matrix given in Fig. 1.2(a) represents the gray scale value of monochrome image. Transform these data by subtracting the value in the upper left hand corner from all elements except itself, as shown in Fig. 1.2(b). This transformation might be useful if variable-length encoding is used. For example, values between -7 and +7 could be encoded with 4-bit numbers and values between 0 and 255 could be encoded as a special 4-bit code (-8) followed by an 8-bit number.

160	160	161	160
161	165	166	158
160	167	165	161
159	160	160	160

160	0	1	0
1	5	6	−2
0	7	5	1
−1	0	1	0

Fig. 1.2(a) Fig. 1.2(b)

Although this simple transformation is lossless, other, more useful ones are not. An especially important two-dimensional spatial transformation is the DCT (Discrete Cosine Transformation) [5-7]. This transformation has the property that for images without sharp discontinuities, most of the spectral power is in the first few terms, allowing the later ones to be ignored without much information loss. In this thesis fractional version of DCT called Discrete Fractional Cosine Transform (DFRCT) is used as transformation.

The third example of source encoding is vector quantization, which is also directly applicable to image data. Here, the image is divided up into fixed-size rectangles.

In addition to the image itself, we also need a table of rectangles of the same size as the image rectangles (possibly constructed from the image). This table is called the code book. Each rectangle is transmitted by looking it up in the code book and just sending the index instead of the rectangle. If the code book is created dynamically (i.e. per image), it must be transmitted, too. Clearly, if a small number of rectangles dominate the image, large savings in bandwidth are possible here.

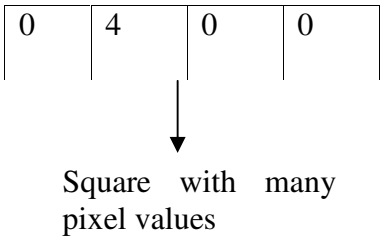
An example of vector quantization is shown in Fig.1.3. In Fig. 1.3(a) we have a grid of rectangles of unspecified size. In Fig. 1.3(b) we have the code book. The output stream is just the list of integers 001022032200400 shown in Fig. 1.3(c) each one represents an entry from the code book.

0	0	1	0
2	2	2	0
3	2	2	0

0
1
2

0	0	1	0	2	2	0	3	2	2	0	0	4	0	0

→ Encoded Image



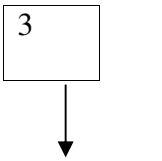


Fig. 1.3(a) Fig. 1.3(b) Fig. 1.3(c)

In a sense, vector quantization is just a two-dimensional generalization of CLUT. The real difference is what happens if no match can be found. Three strategies are possible. The first one is to use the best match, and append some information about how to second one is to use the best match, and append some information about how to improve the match. The first two strategies are lossy but exhibit high compression. The third is lossless but less effective as a compression algorithm. By this, it is seen that encoding (pattern matching) is far more time consuming than decoding (indexing into a table).

1.2 ENCRYPTION

The explosive growth in the use of computers for storing information and e-mail for transmitting it, and the arrival of e- commerce, has led to cryptography becoming an essential feature of modern communications and data storage. Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication.

Encryption is an area of cryptography involving the transformation of information

into some gibberish form, thus ensuring privacy by keeping the information hidden from anyone for whom it is not intended - one may wish to encrypt files on a hard disk to prevent an intruder from reading them, or in a multi-user setting, encryption allows secure communication over an insecure channel. An example of this: A wishes to send a message to B so that no one else besides B can read it. A encrypts the message (plaintext) with an encryption key; the encrypted message (cipher text) is sent to B. B decrypts the cipher text with the decryption key and reads the message. An attacker, C, may either try to obtain the secret key or to recover the plaintext without using the secret key. In a secure cryptosystem, the plaintext cannot be recovered from the cipher text except by using the decryption key.

Till today the most successful image encryption scheme is random phase encoding in fractional domain. In this thesis DFRCT with random phase masking is used as basic cryptographic tool. This double phase encoding scheme encrypts the image into different fractional orders and random distribution that the unauthorized person by

no means access the image without keys. The significant feature of image encryption benefits from its extra degree of freedom that is provided by fractional orders.

1.3 OBJECTIVE

In this thesis Fractional version of Discrete Cosine Transform (DCT) called Discrete Fractional Cosine Transform (DFRCT) is used as basic tool in image processing for applications like image compression and image encryption. It is seen that due to its fractional order DFRCT may find its place in many applications where DCT is found to be useful. The objectives of thesis are

1. To form 2-D DFRCT from 1-D DFRCT by applying 1-D DFRCT to each row of matrix and then to each column of result.
2. To implement DFRCT on four natural gray scale still images and study the effects of variations of cutt off, Compression Ratio (CR) and fractional order.
3. To prove DFRCT due to its free parameter its fraction ‘a’ provides better result than DCT in the field of image compression

4. To encrypt the image using fractional orders of DFRCT along with the random phase masking and to show this encrypto system gives nearly zero MSE between original and decrypted image.

1.4 ORGANIZATION

The thesis is organized in six chapters:

Chapter 2 describes the basic concept of Discrete Cosine Transform. This chapter shows how the image is decompressed into its underlying frequency coefficients. In end this chapter gives the mathematical definition for 1-D & 2-D Discrete Fractional Cosine Transform.

Chapter 3 describes the Fractional operators, need of discretizatio, in end calculation of DRFCT kernel from Hermite eigenvectors and eigenvalues.

Chapter 4 gives the various steps to implement 2-D Discrete Fractional Cosine Transform on image.

Chapter 5 gives the image encryption algorithm using Fractional orders of Fractional Discrete Transforms with random phase masking.

Chapter 6 gives the results of implementation of DRFCT on four natural gray scale cons, cameraman, Lena & Rice images for applications like image compression & image encryption.

In the end of thesis the whole work is concluded.

Chapter 1

INTRODUCTION

Introduction

Digital image processing techniques are used to enhance the quality of image and to reduce the data needed to represent the image without visually affecting the quality of image. The objective of compression is to reduce the data volume and achieve reproduction of the original data without any perceived loss in data quality. Applications of compression include digital imaging. Compression of medical images for efficient use of storage space and transmission bit rate has become a necessity.

Cryptographic techniques are used to scramble images so that an adversary could not the original image without knowing the secret key. The objective of encryption is to transform data into an unreadable form to ensure privacy. Encryption was widely used in military imaging for secure transmission of information. Today, however, encryption is widely

used by everyone to transmit information such as commercial transactions (exchange of credit card information, etc.).

In the case of the former, the "compressed" data needs to be uncompressed to the original form while in the case of the latter; the "encrypted" data needs to be decrypted to the original form.

1.1 IMAGE COMPRESSION

Today transmitting multimedia material in uncompressed form is completely out of question. The only hope is that massive compression is possible. Fortunately a large body of research over past few decades has led to many compression techniques and algorithms that make multimedia transmission feasible. There is a problem of storing, transmitting and manipulating digital image because of the size of image files involved, transmitting images will always consume large amount of bandwidth and storing images will require hefty resources as shown in Table 1

Table 1.1 Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required.

Multimedia Data	Size/Duration	Bits/Pixel or Bits/Sample	Uncompressed Size (B for bytes)	Transmission Bandwidth (b for bits)	Transmission Time (using a 28.8K Modem)
A page of text	11" x 8.5"	Varying resolution	4-8 KB	32-64 Kb/page	1.1 - 2.2 sec
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Grayscale Image	512 x 512	8 bps	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512 x 512	24 bps	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bps	5.16 MB	41.3	23 min 54 sec

				Mb/image	
SHD Image	2048 x 2048	24 bps	12.58 MB	100 Mb/image	58 min 15 sec
Full-motion Video	640 x 480, 1 min (30 frames/sec)	24 bps	1.66 GB	221 Mb/sec	5 days 8 hrs

The examples above clearly illustrate the need for sufficient storage space, large transmission bandwidth and long transmission time for image, audio, and video data [1].

1.1.1 Principal behind Compression

In images the neighboring pixels are correlated and therefore contain redundant information [2]. The fundamental components of compression are redundancy and irrelevancy reduction. Redundancy means duplication and Irrelevancy means the parts of signal that will not be noticed by the signal receiver, which is the Human Visual System .

There are three types of redundancy can be identified:

- Spatial Redundancy i.e. correlation between neighboring pixel values.
- Spectral Redundancy i.e. correlation between different color planes or spectral bands.
- Temporal Redundancy i.e. correlation between adjacent frames in a sequence of images (in video applications).

Image compression focuses on reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies. The audio, video and image signals are

available to be compressed, because there is a considerable redundancy in these signals. Since this thesis is about still image compression, therefore temporal redundancy is not relevant.

1.1.2 Asymmetries in compression Algorithms

There are certain asymmetries in compression algorithms that are important to understand:-

All compression systems require two algorithms, one for compressing the data at the source, and another for decompressing it at the destination. These algorithms are also referred as encoding and decoding algorithm. They have certain asymmetry like, for many applications a multimedia document, say, a movie only be encoded once but will be decoded thousands of time. This asymmetry means that it is acceptable for encoding algorithm to be slow and require expensive hardware provided that decoding algorithm is fast and does not require expensive hardware. After all, the operator of a multimedia server might be quite willing to rent a parallel supercomputer for a few weeks to encode its entire video library, but requiring consumers to rent a supercomputer for 2 hours to view a video is not likely to be a big success. Many practical compression systems go to great lengths to make decoding fast and simple, even at the price of making encoding slow and complicated.

On the other hand, for real time multimedia, such as video conferencing, slow encoding is unacceptable. Encoding must happen on the fly, in real time. So, real-time multimedia uses different algorithms or parameters than storing videos on disk, often with appreciably less compression.

A second asymmetry is that the encode/decode process need not to invertible. That is, when compressing a file, transmitting it, and then decompressing it, the user expects to get the

original back, accurate down to the last bit. With multimedia, this requirement does not exist. It is usually acceptable to have the video signal after encoding and then decoding is slightly different than the original. When the decoded output is not exactly equal to the original input, the system is said to be lossy. If the input and output are identical, the system is lossless. Lossy systems are important because accepting a small amount of information loss can give a huge payoff in terms of the compression ratio possible.

1.1.3 Image Compression System

Some of the most successful image compression techniques (both lossless and lossy) involve the following three major stages: image transformation, quantization (lossy compression only) and encoding. Fig 1.1 shows the block diagram of simple compression system.

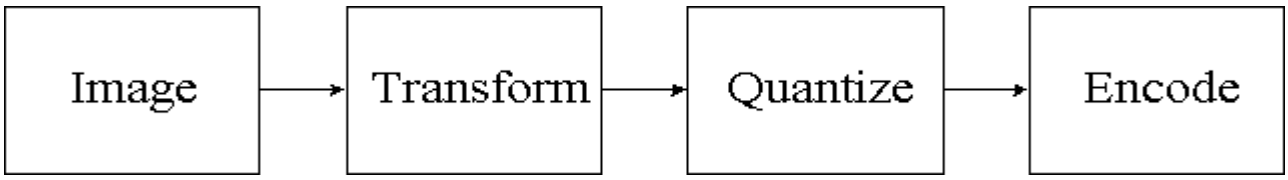


Fig 1.1: Block diagram of Simple compression System

(a) Transform

Image transform, sometimes referred to as decorrelation, is used to reduce the dynamic range of the signal, to eliminate redundant information, and to provide a suitable representation for entropy encoding.

A transform should satisfy three conditions [3]:

(i) All transform coefficients become statistically independent.

(ii) Energy of the transformed image is compacted into a minimum number of coefficients.

(iii) The transform coefficients are concentrated in minimum frequency or transform Scale regions

(b) Quantizer

A Quantizer reduces the precision of the values generated by the transform and therefore reduces the number of bits required to save the transform co-coefficients. Quantization is fundamentally lossy,the loss in decompressed image is due to this section.

(c) Entropy Encoder

An entropy encoder does further compress the quantized values. This is done to achieve even better overall compression. The various commonly used entropy encoders are the Huffman encoder, arithmetic encoder, and simple run-length encoder. For better performance with compression, it's important to have the best of all the three components.

1.1.4 Compression Schemes

Compression schemes can be divided into two general categories: entropy encoding and source encoding [4].

(a) Entropy Encoding

Entropy encoding just manipulates bit streams without regard to what the bits mean. It is a general, lossless, fully reversible technique, applicable to all data. .

The first type of entropy encoding is run-length encoding. In many kinds of data, strings of repeated symbols (bits, numbers etc.) are common. These can be replaced by a special marker not otherwise allowed in the data, followed by the symbol comprising the run, followed by how many times it occurred. If the special marker itself occurs in the data, it is duplicated (as in character stuffing). For example, consider the following string of decimal digits.

315000000000000845871111111111116354670000000000000065

If we now introduce A as the marker and use two digit numbers for the repetition count, we can encode the above digit string as

315A01284587A11316354674A02265

Here run length encoding has cut the data string in half.

Runs are common in multimedia. In audio, silence is often represented by runs of zeros. In video, runs of the same color occur in shots of the sky, walls, and many flat surfaces. All of these runs can be greatly compressed.

The second type of entropy encoding is statistical encoding. In this we use a short code to represent common symbols and long ones to represent infrequent ones. Morse code uses this principle, with E being and Q being… and so on. Huffman coding and the Ziv-Lempel algorithm used by the UNIX compress program also use statistical encoding.

The third type of entropy encoding is CLUT (Color Look up Table) encoding. Consider an image using RGB encoding with 3 bytes/pixel. In theory, the image might contain as many as 2^{24} different color values. In practice, it will normally contain many fewer values, especially if the image is a cartoon or computer generated drawing, rather than a photograph. Suppose that only 256 color values are actually used. A factor of almost three compressions can be achieved by building a 768 byte table listing the RGB values of the 256 colors actually used, and then representing each pixel by the index of its RGB value in the table. This is a clear example where encoding is slower than decoding because encoding requires searching the table whereas decoding can be done with a single indexing operation.

(b) Source Encoding

Source encoding, takes advantage of properties of the data to produce more (usually lossy) compression. Some examples of source encoding are, the first example is differential encoding, in which a sequence of values (e.g. audio samples) are encoded by representing each one as the difference from the previous value. Differential pulse code modulation is an example of this technique. It is lossy because the signal might jump so much between two consecutive values that the difference does not fit in the field provided for expressing differences, so at least one incorrect value will be recorded and some information lost.

Differential encoding is a kind of source encoding because it takes advantage of the property that large jumps between consecutive data points are unlikely .Not all the sequence of numbers have this property

The other type of source encoding consists of transformations .By transforming signals from one domain to another, compression may become easier .If we take the Fourier transform of 1-D signal in which we represent a function of time as a list of amplitudes now if we give the

exact value of all amplitudes, the original function can be reconstructed perfectly .However by only giving the values of, say first 8 amplitudes rounded to first two decimal places, the signal can still reconstructed so well that the listener can't tell that some information has been lost. The gain is that transmitting 8 amplitudes require many fewer bits than transmitting the sampled waveform.

Transformations are also applicable to 2-D image data. Suppose that 4 X 4 matrix given in Fig. 1.2(a) represents the gray scale value of monochrome image. Transform these data by subtracting the value in the upper left hand corner from all elements except itself, as shown in Fig. 1.2(b). This transformation might be useful if variable-length encoding is used. For example, values between -7 and +7 could be encoded with 4-bit numbers and values between 0 and 255 could be encoded as a special 4-bit code (-8) followed by an 8-bit number.

160	160	161	160
161	165	166	158
160	167	165	161
159	160	160	160

Fig. 1.2(a)

160	0	1	0
1	5	6	−2
0	7	5	1
−1	0	1	0

Fig. 1.2(b)

Although this simple transformation is lossless, other, more useful ones are not. An especially important two-dimensional spatial transformation is the DCT (Discrete Cosine Transformation) [5-7]. This transformation has the property that for images without sharp discontinuities, most of the spectral power is in the first few terms, allowing the later ones to be ignored without much information loss. In this thesis fractional version of DCT called Discrete Fractional Cosine Transform (DFRCT) is used as transformation.

The third example of source encoding is vector quantization, which is also directly applicable to image data. Here, the image is divided up into fixed-size rectangles.

In addition to the image itself, we also need a table of rectangles of the same size as the image rectangles (possibly constructed from the image). This table is called the code book. Each rectangle is transmitted by looking it up in the code book and just sending the index instead of the rectangle. If the code book is created dynamically (i.e. per image), it must be transmitted, too. Clearly, if a small number of rectangles dominate the image, large savings in bandwidth are possible here.

An example of vector quantization is shown in Fig.1.3. In Fig. 1.3(a) we have a grid of rectangles of unspecified size. In Fig. 1.3(b) we have the code book. The output stream is just the list of integers 001022032200400 shown in Fig. 1.3(c) each one represents an entry from the code book.

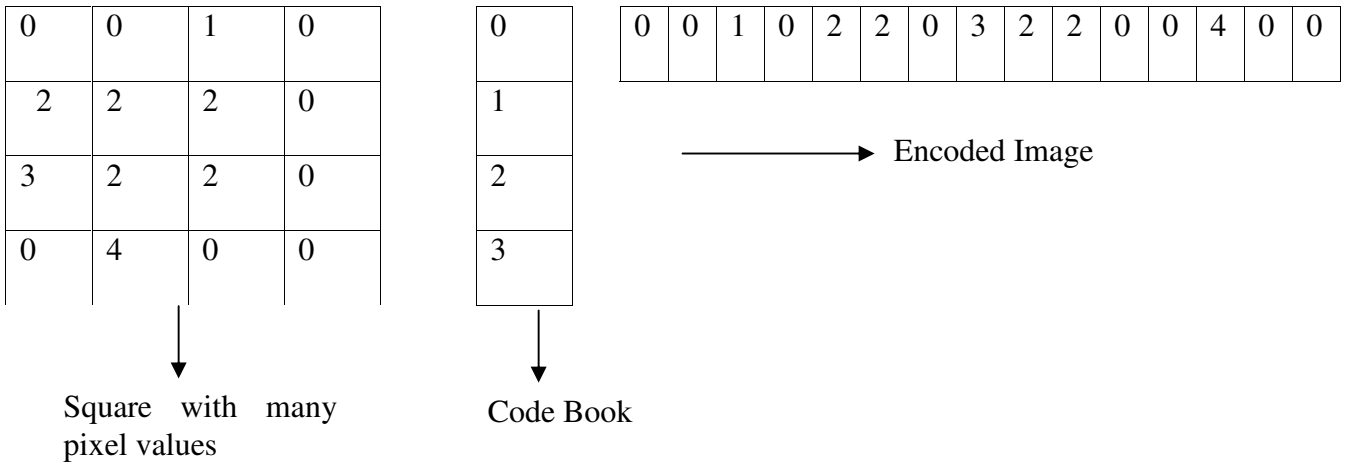


Fig. 1.3(a) Fig. 1.3(b) Fig. 1.3(c)

In a sense, vector quantization is just a two-dimensional generalization of CLUT. The real difference is what happens if no match can be found. Three strategies are possible. The first one is to use the best match, and append some information about how to second one is to use the best match, and append some information about how to improve the match. The first two

strategies are lossy but exhibit high compression. The third is lossless but less effective as a compression algorithm. By this, it is seen that encoding (pattern matching) is far more time consuming than decoding (indexing into a table).

1.2 ENCRYPTION

The explosive growth in the use of computers for storing information and e-mail for transmitting it, and the arrival of e- commerce, has led to cryptography becoming an essential feature of modern communications and data storage. Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication.

Encryption is an area of cryptography involving the transformation of information into some gibberish form, thus ensuring privacy by keeping the information hidden from anyone for whom it is not intended - one may wish to encrypt files on a hard disk to prevent an intruder from reading them, or in a multi-user setting, encryption allows secure communication over an insecure channel. An example of this: A wishes to send a message to B so that no one else besides B can read it. A encrypts the message (plaintext) with an encryption key; the encrypted message (cipher text) is sent to B. B

decrypts the cipher text with the decryption key and reads the message. An attacker, C, may either try to obtain the secret key or to recover the plaintext without using the secret key. In a secure cryptosystem, the plaintext cannot be recovered from the cipher text except by using the decryption key.

Till today the most successful image encryption scheme is random phase encoding in fractional domain. In this thesis DFRCT with random phase masking is used as basic cryptographic tool. This double phase encoding scheme encrypts the image into different fractional orders and random distribution that the unauthorized person by no means access the image without keys. The significant feature of image encryption benefits from its extra degree of freedom that is provided by fractional orders.

1.3 OBJECTIVE

In this thesis Fractional version of Discrete Cosine Transform (DCT) called Discrete Fractional Cosine Transform (DFRCT) is used as basic tool in image processing for applications like image compression and image encryption. It is seen that

due to its fractional order DFRCT may find its place in many applications where DCT is found to be useful. The objectives of thesis are

1. To form 2-D DFRCT from 1-D DFRCT by applying 1-D DFRCT to each row of matrix and then to each column of result.
2. To implement DFRCT on four natural gray scale still images and study the effects of variations of cutt off, Compression Ratio (CR) and fractional order.
3. To prove DFRCT due to its free parameter its fraction ‘a’ provides better result than DCT in the field of image compression
4. To encrypt the image using fractional orders of DFRCT along with the random phase masking and to show this encrypto system gives nearly zero MSE between original and decrypted image.

1.4 ORGANIZATION

The thesis is organized in six chapters:

Chapter 2 describes the basic concept of Discrete Cosine Transform. This chapter shows how the image is decompressed into its underlying frequency coefficients. In end this chapter gives the mathematical definition for 1-D & 2-D Discrete Fractional Cosine Transform.

Chapter 3 describes the Fractional operators, need of discretizatio, in end calculation of DRFCT kernel from Hermite eigenvectors and eigenvalues.

Chapter 4 gives the various steps to implement 2-D Discrete Fractional Cosine Transform on image.

Chapter 5 gives the image encryption algorithm using Fractional orders of Fractional Discrete Transforms with random phase masking.

Chapter 6 gives the results of implementation of DRFCT on four natural gray scale cons, cameraman, Lena & Rice images for applications like image compression & image encryption.

In the end of thesis the whole work is concluded.

Chapter 2

DISCRETE COSINE TRANSFORM

Introduction

In this chapter an especially important two-dimensional spatial Discrete Cosine transformation (DCT) is discussed. This transformation has the property that for images without sharp discontinuities, most of the spectral power is in the first few terms, allowing the later ones to be ignored without much information loss. Due to it's this property it is widely used in the field of image compression.

2.1 BASIC DCT CONCEPTS

It is seen that the human visual system response is very dependent on spatial frequency. If we could somehow decompose the image into a set of waveforms, each with a particular spatial frequency, we might be able to separate the image structure the eye can see from the structure that is imperceptible. The DCT can provide a good approximation to this decomposition [8].

2.1.1 The one-dimensional DCT

To understand how an image can be decomposed into its underlying spatial frequencies, first consider a one-dimensional (1-D) case. Start with a set of eight arbitrary grayscale samples such as is shown in Fig. 2.1.

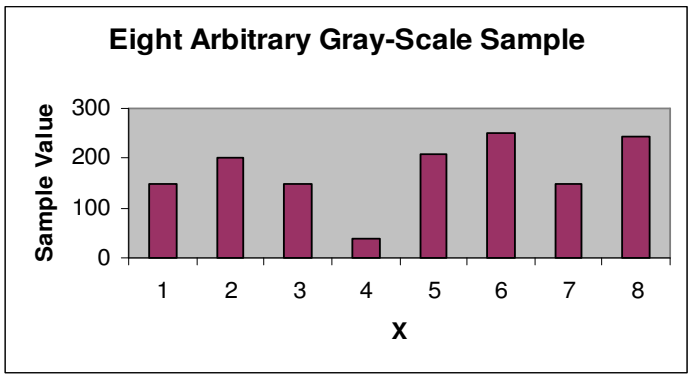


Fig. 2.1(a)

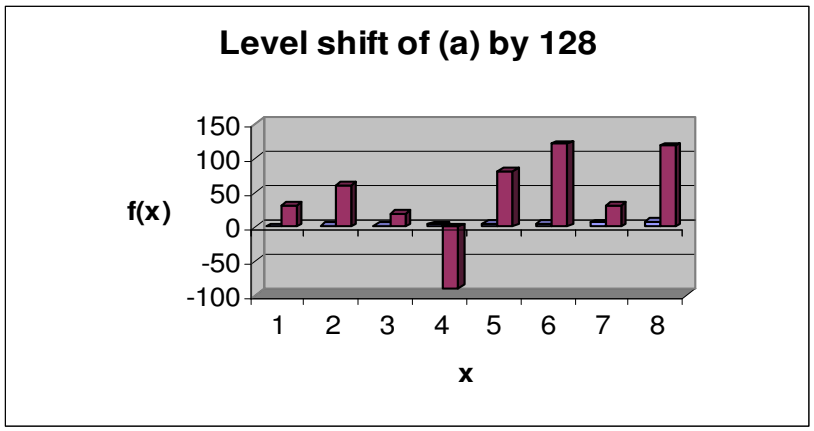


Fig. 2.1(b)

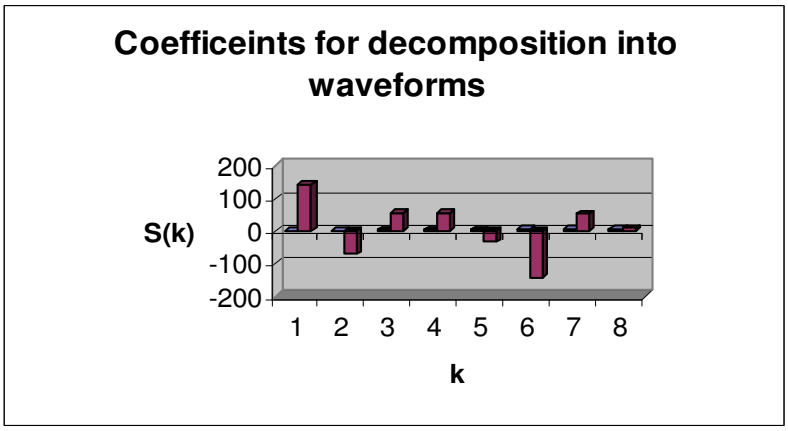


Fig. 2.1(c)

The samples have values in the range 0 to 255, but after a level shift by 128 (as is done by JPEG), we get the values $f(x)$ in Fig. 2.1(b). To decompose these eight sample values into a set of waveforms of different spatial frequencies.

Fig. 2.2 shows a set of eight different cosine waveforms of uniform amplitude, each sampled at eight points. The top-left waveform $(k = 0)$ is simply a constant, whereas the other seven waveforms $(k = 1,.....,7)$ show an alternating behavior at progressively higher frequencies

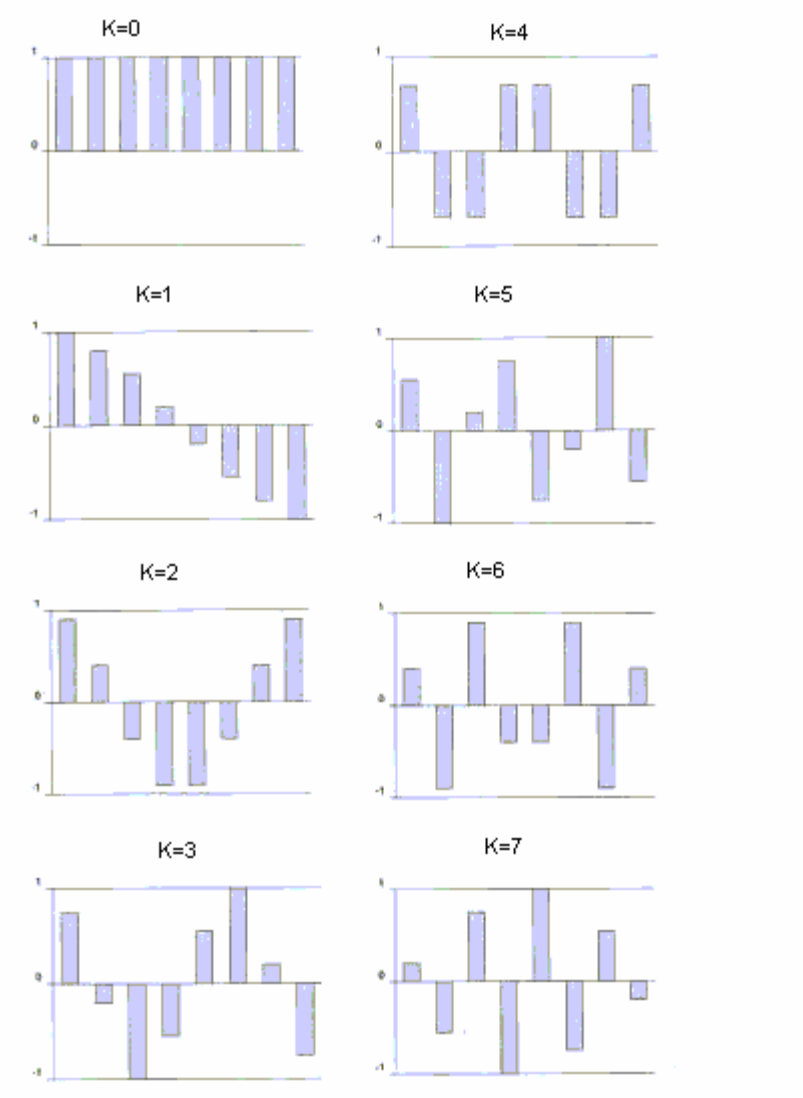


Fig. 2.2 Cosine Basis Waveforms

These waveforms (which are called cosine basis functions) are said to be orthogonal. A set of waveforms is orthogonal if it has the following interesting property: if the product of any two waveforms in the set at each sampling point, and sum these products over all sampling points, the result is zero. If the waveform is multiplied by itself and summed, the result is a constant. For example, the product of waveform 0 and waveform 1, is taken and the sum is calculated over all the sample points, the result is zero. On the other hand, the product of waveform 1 with itself is taken, the product at each sample point is the square of the waveform value; therefore, the sum of the products over all sample points is a positive constant (which is used to define a scale factor for the waveforms).

Orthogonal waveforms are independent. That is, there is no way that a given waveform can be represented by any combination of the other waveforms. However, the complete ser of eight waveforms when scaled by numbers called coefficients and added together, can be used to represent any eight sample values such as those in Fig. 2.1(b). The coefficients $s(k)$ are plotted in Fig. 2.1(c)

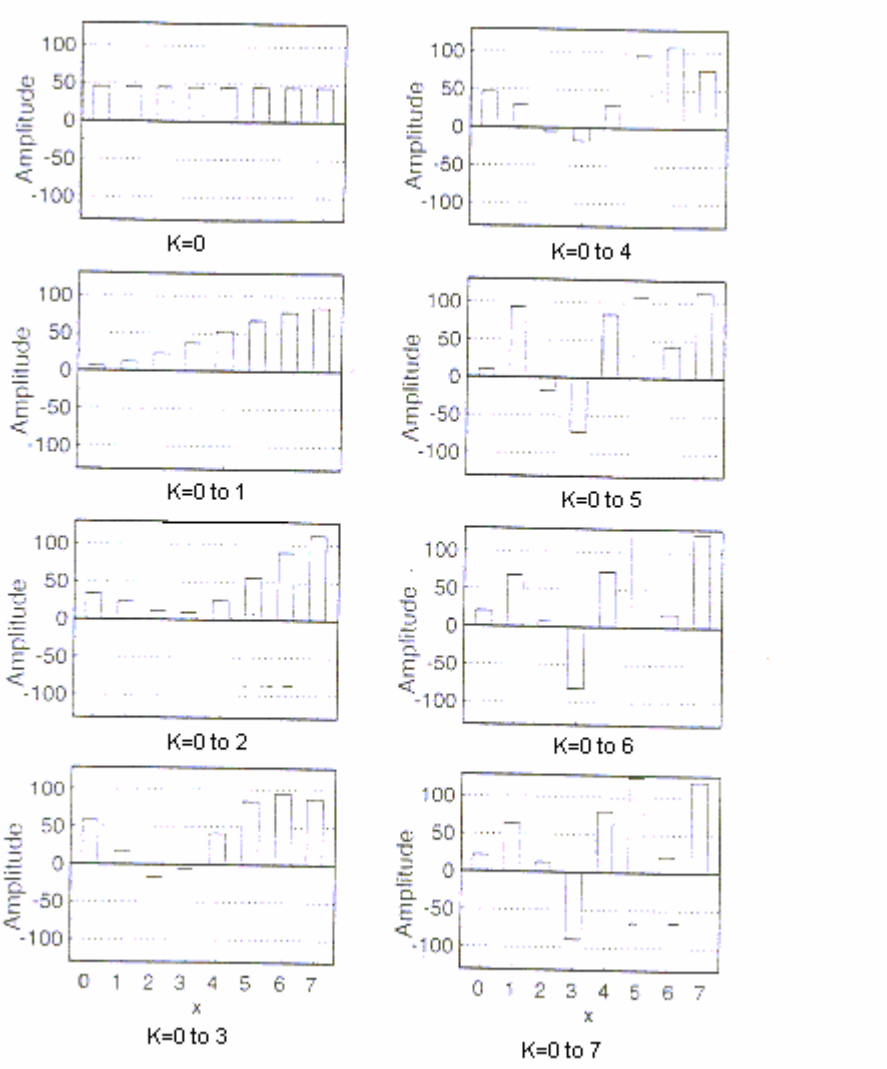


Fig. 2.3 Progressively summed waveforms

Fig. 2.3 shows a sequence in which the eight scaled waveforms are progressively summed, starting with the lowest frequency (adding one more each time), until finally the original set of samples in reconstructed. The coefficients plotted in Fig. 2.1(c) are the output of an 8-point DCT for the sample valued in Fig. 2.1(b).

The coefficient that scales the constant basis function ($k = 0$) is called the DC coefficient. The other coefficients are called AC coefficients. These names are derived from the historical use of the DCT for analyzing electrical currents that had both direct-and alternating-current (DC and AC) terms. The DC term gives the average over the set of samples.

The process of decomposing a set of samples into a scaled set of cosine basis functions is called the Forward Discrete Cosine Transform (FDCT). The process of reconstruction the set of samples from the scaled set of cosine basis functions is called the Inverse Discrete Cosine Transform (IDCT). If the sample sequence is longer than eight samples, it can be divided into eight-sample groups and DCT can be computed independently for each group. Because the cosine basis functions always have the same set of values at each of the discrete sampling points, only the coefficient values change from the one group of samples to the next.

2.1.2 The two-dimensional DCT

The 1-D DCT can be extended to apply to 2-D image arrays. Fig. 2.4 shows a set of 64 2-D cosine basis functions that are created by multiplying a horizontally oriented set of 1-D 8-point basis functions (shown in Fig. 1.2) by a vertically oriented set of the same functions.

The horizontally oriented set of basis functions represents horizontal frequencies and the other set of basis functions represents vertical frequencies. By convention, the DC term of the horizontal basis functions is to the left, and the DC term for the vertical basis functions is at the top. So, the top row and the left column have 1-D intensity variations, which, if plotted, would be the same as in Fig. 2.2 (neutral gray represents zero in these figures, white represents positive amplitudes, and black represents negative amplitudes.)

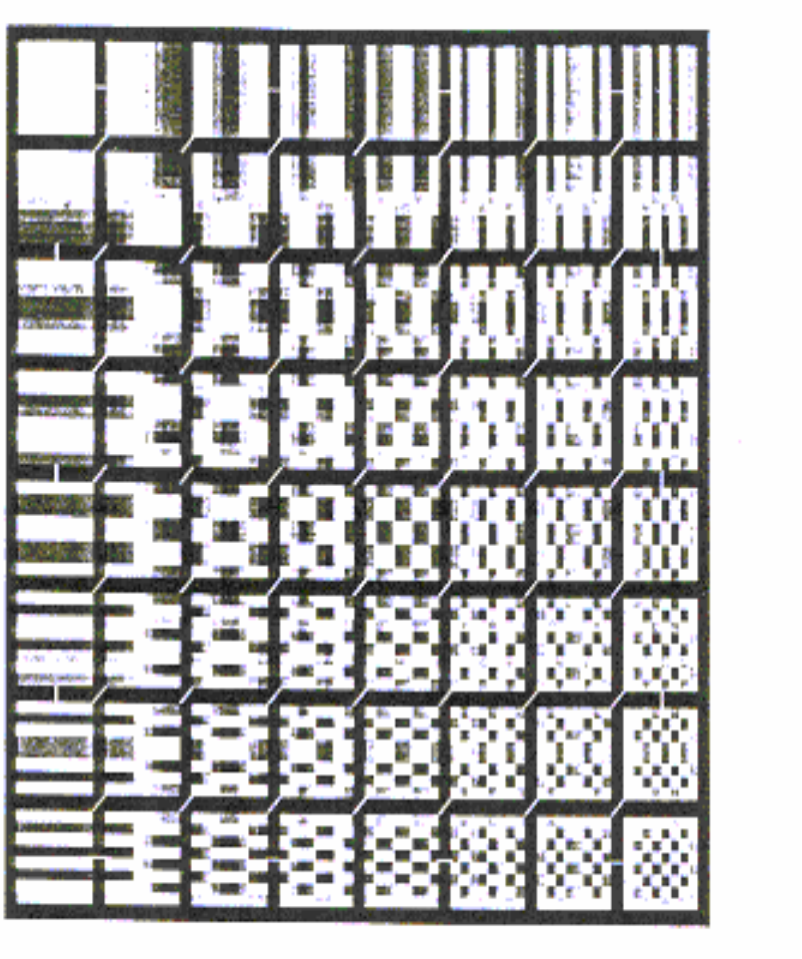


Fig. 2.4 2-D Cosine Basis

When scaled by an appropriate set of 64 coefficients, these 64 basis functions can be used to represent any 64 sample values such as the 8x8 block of samples shown at the upper left corner of the Fig. 2.5.

Fig. 2.5 also shows the sequence in which these 64 basis functions are progressively summed following the zigzag sequence shown by the white connecting lines. The zigzag pattern, which is used in the JPEG algorithms, approximately orders the basis functions from low to high spatial frequencies.

Because the 2-D DCT basis functions are products of two 1-D DCT basis functions, the only constant basis function is in the upper left corner of the array; the coefficient for this basis function is called the DC coefficient, whereas the rest of the coefficients are called AC coefficients.

Fig. 2.5 8x8 Block of Samples

2.2 DCT INTEGER QUANTIZATION

Quantization allows us to reduce the accuracy with which the DCT coefficients are represented when converting the DCT to an integer representation. This can be very important in image compression, as it tends to make many coefficients zero-especially those for high spatial frequencies.

The quantization values can be set individually for each DCT coefficient, using criteria based on visibility of the basis functions. By measuring the threshold for visibility of a given basis function - the coefficients amplitude that is just detectable by the human eye. The

coefficients are divided by that value (with appropriate rounding to integer values). By multiplying (dequantize) the scaled-down coefficients by that value before reconstructing, a condition is created in which the eye should not be able to detect any difference between quantized and unquantized DCT coefficient. To tolerate some visible artifacts in the reconstructed image, the larger value of the visibility threshold can be used..

This process of scaling the DCT coefficients and truncating them to integer values is called quantization, and the rescaling to restore approximately the original DCT coefficient magnitude is called dequantization.

2.3 MATHEMATICAL DEFINITION OF DCT

2.3.1 The One Dimensional Discrete Cosine Transform

Let $x(n)$ denotes a N point sequence that is zero outside $0 \le n \le N-1$. Among several variation consider one variation known as even symmetrical DCT , which is most often used in signal coding application .To derive DCT relationship, it is convenient to relate N point sequence $x(n)$ to derive new $2N$ point sequence $y(n)$, which is then related to its $2N$ point DFT $y(k)$. Then relate $y(k)$ to $C_x(k)$, then N point DCT of $x(n)$

$$\underset{x(n)}{N-\text{point}} \leftrightarrow \underset{y(n)}{2N-\text{point}} \xleftrightarrow{DFT} \underset{Y(k)}{2N-\text{point}} \leftrightarrow \underset{C_x(k)}{N-\text{point}} \quad \text{--------} \quad 2.1$$

The sequence $x(n)$ is related to $y(n)$ by

$$y(n) = x(n) + x(2N-1-n) \qquad \text{------2.2}$$

$$= \begin{cases} x(n), 0 \le n \le N-1 \\ x(2N-1-n), N \le n \le 2N-1 \end{cases}$$

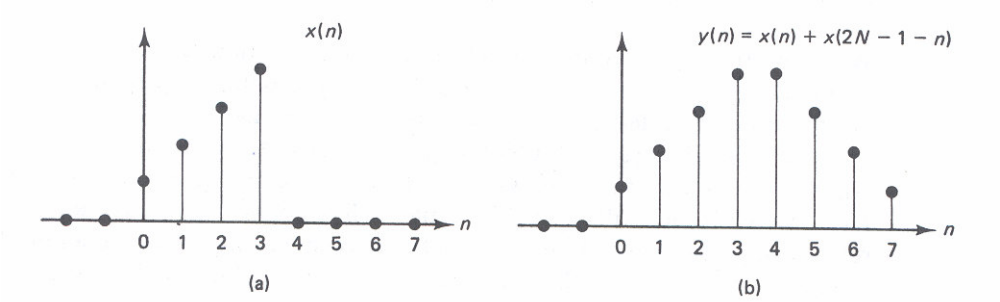


Fig 2.6

An example of $x(n)$ and $y(n)$ when $N = 4$ is shown in Fig. 2.6. The sequence $y(n)$ is symmetric with respect to the half-sample point at $n = N - \frac{1}{2}$. A periodic sequence $\tilde{x}(n)$ is formed by repeating $x(n)$ every N points, $\tilde{x}(n)$ has artificial discontinuities, since the beginning and end part of $x(n)$ are joined in the repetition process. A periodic sequence $\tilde{y}(n)$ is formed by repeating $y(n)$ every $2N$ point, however, $\tilde{y}(n)$ no longer contains the artificial discontinuities. This is shown in Fig. 2.7 for $x(n)$ and $y(n)$ and for $X(n)$ and $Y(n)$ shown in Fig. 2.6.

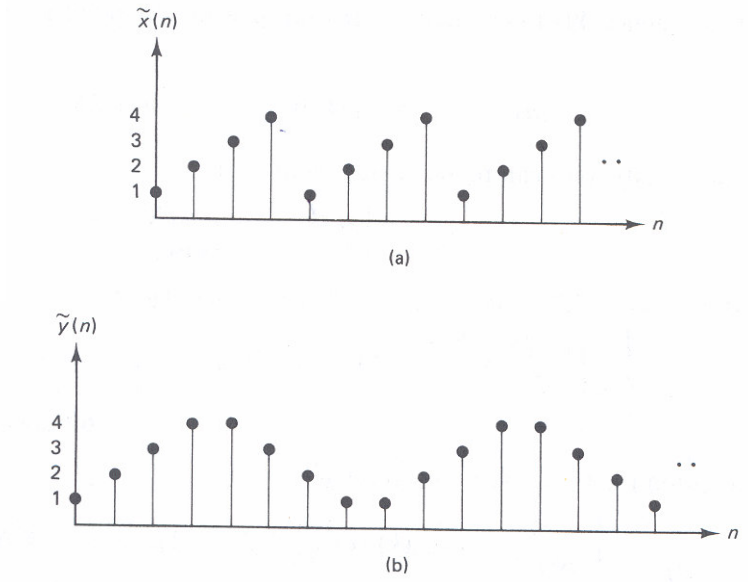


Fig. 2.7

The $2N$ point DFT $Y(k)$ is related to $y(n)$ by

$$Y(k) = \sum_{n=0}^{2N-1} y(n) W_{2N}^{kn}, 0 \le k \le 2N-1 \qquad 2.3a$$

Where

$$W_{2N} = e^{-j(2\pi/2N)} \qquad 2.3b$$

From (2.3a) & (2.3b)

$$Y(k) = \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{kn}, 0 \le k \le 2N-1 \qquad 2.4$$

With a change of variables and after some algebra, 2.4 can be expressed as

$$Y(k) = W_{2N}^{-\frac{k}{2}} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\Pi}{2N} k(2N+1), 0 \le k \le 2N-1 \qquad 2.5$$

The N point DCT of $x(n)$, $C_x(k)$, is obtained from $Y(k)$ by

$$C_x(k) = \begin{cases} W_{2N}^{\frac{k}{2}} Y(k), & 0 \le k \le 2N-1 \\ 0, & otherwise \end{cases} \qquad 2.6$$

From (2.5) and (2.6)

$$c_x(k) = \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2n+1), \quad 0 \le k \le 2N-1 \qquad 2.7$$
$$0, \qquad otherwise$$

Equation (2.7) is the definition of the DCT of $x(n)$. From (2.7), $C_x(k)$, is a N point sequence, and therefore N values of $x(n)$ are represented by N values of $C_x(k)$. If $x(n)$ is real, $C_x(k)$ is real. If $x(n)$ is complex, so is $C_x(k)$. To derive the inverse DCT relation, we relate $C_x(k)$ to $Y(k)$, $Y(k)$ Y(k) to $y(n)$, y(n), then $y(n)$ to $x(n)$. First consider determining $Y(k)$ from $C_x(k)$. Although $Y(k)$ is a $2N$ point sequence and $C_x(k)$ is an N point sequence, redundancy in

$Y(k)$ due to the symmetry of $y(t)$ allows us to determine $Y(k)$ from $C_x(k)$. Specifically, from (2.5)

$$Y(k)=\begin{cases} W_{2N}^{-k}Y(2N-k), & 0\le k\le 2N-1 \\ 0, & k=N \end{cases} \qquad 2.8$$

From (2.6) and (2.8)

$$Y(k)=\begin{cases} W_{2N}^{-k}C_x(k), & 0\le k\le N-1 \\ 0, & k=N \\ -W_{2N}^{-\frac{k}{2}}C_x(2N-k), & N+1\le k\le 2N-1 \end{cases} \qquad 2.9$$

The sequence $Y(k)$ is related to $y(n)$ through the $2N$ point inverse DFT relation given by

$$y(n)=\frac{1}{2N}\sum_{k=0}^{2N-1}Y(k)W_{2N}^{-kn},0\le n\le 2N-1 \qquad 2.10$$

From (2.2), $x(n)$ can be recovered from $y(n)$ by

$$x(n)=\begin{cases} y(n), & 0\le n\le N-1 \\ 0, & otherwise \end{cases} \qquad 2.11$$

From (2.9), (2.10), and (2.11), and after some algebra, we get

$$x(n)=\begin{cases} \frac{1}{N}\left[\frac{C_x(0)}{2}+\sum_{k=1}^{N-1}C_x(k)\cos\frac{\pi}{2N}k(2n+1)\right], & 0\le n\le N-1 \\ 0, & otherwise \end{cases} \qquad 2.12$$

Equation (2.12) can also be expressed as

$$x(n)=\begin{cases} \frac{1}{N}\sum_{k=1}^{N-1}w(k)C_x(k)\cos\frac{\pi}{2N}k(2n+1), & 0\le n\le N-1 \\ 0, & otherwise \end{cases} \qquad 2.13a$$

Where

$$w(k)=\begin{cases} \frac{1}{2}, & k=0 \\ 1, & 1\le k\le N-1 \end{cases} \qquad 2.13b$$

Equation (2.13) is the inverse DCT relation. From (2.7) and (2.13), we get FDCT

Discrete Cosine Transform Pair

$$C_x(k) = \begin{cases} \sum_{n=0}^{N-1} 2x(n)\cos\frac{\pi}{2N}k(2n+1), & 0 \le k \le N-1 \\ 0, & otherwise \end{cases} \quad \textbf{FDCT} \qquad 2.14a$$

$$x(n) = \begin{cases} \frac{1}{N}\sum_{k=1}^{N-1} w(k)C_x(k)\cos\frac{\pi}{2N}k(2n+1), & 0 \le n \le N-1 \\ 0, & otherwise \end{cases} \quad \textbf{IDCT} \qquad 2.14b$$

From the derivation of the DCT pair, the DCT and inverse DCT can be computed by Computation of Discrete Cosine Transform

Step 1. $y(n) = x(n) + x(2N-1-n)$

Step 2. $Y(k) = DFT[y(n)]$ ($2N$ Point DFT computation)

Step 3. $C_x(k) = \begin{cases} W_{2N}^{\frac{k}{2}} Y(k), & 0 \le k \le N-1 \\ 0, & otherwise \end{cases}$

Computation of Inverse Discrete Cosine Transform

Step 1. $Y(k) = \begin{cases} W_{2N}^{-k} C_x(k), & 0 \le k \le N-1 \\ 0, & k = N \\ -W_{2N}^{-\frac{k}{2}} C_x(2N-k), & N+1 \le k \le 2N-1 \end{cases}$

Step 2. $y(n) = IDFT[Y(k)]$ ($2N$ Point inverse DFT computation)

Step 3. $x(n) = \begin{cases} y(n), & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$

In computing the DCT and inverse DCT, Steps 1 and 3 are computationally quite simple. Most of the computations are in Step 2, where a $2N$ point DFT is computed for the DCT and a $2N$ point inverse DFT is computed for the inverse DCT. The DFT and inverse DFT can be computed by using Fast Fourier transform (FFT) algorithms. In addition, because $y(n)$ has symmetry, the $2N$ point DFT and inverse DFT can be computed by computing the N point DFT

and the N point inverse DFT of an N point sequence. Therefore, the computation involved in using the DCT is essentially the same as that involved in using the DFT.

In the derivation of the DCT pair, we have used an intermediate sequence $y(n)$ that has symmetry and whose length is even. The DCT derived is thus called an even symmetrical DCT. It is also possible to derive the odd symmetrical DCT pair in the same manner. In the odd symmetrical DCT, the intermediate sequence $y(n)$ used has symmetry, but its length is odd. For the sequence $x(n)$ shown in Fig. 2.8(a), the sequence $y(n)$ used is shown in Fig. 2.8(b). The length of $y(n)$ is $2N-1$, and $\tilde{y}(n)$, obtained by repeating $y(n)$ every $2N-1$ points, has no artificial discontinuities. The even symmetrical DCT is more commonly used, since the odd symmetrical DCT involves computing an odd-length DFT, which is not very convenient when one is using FFT algorithms.

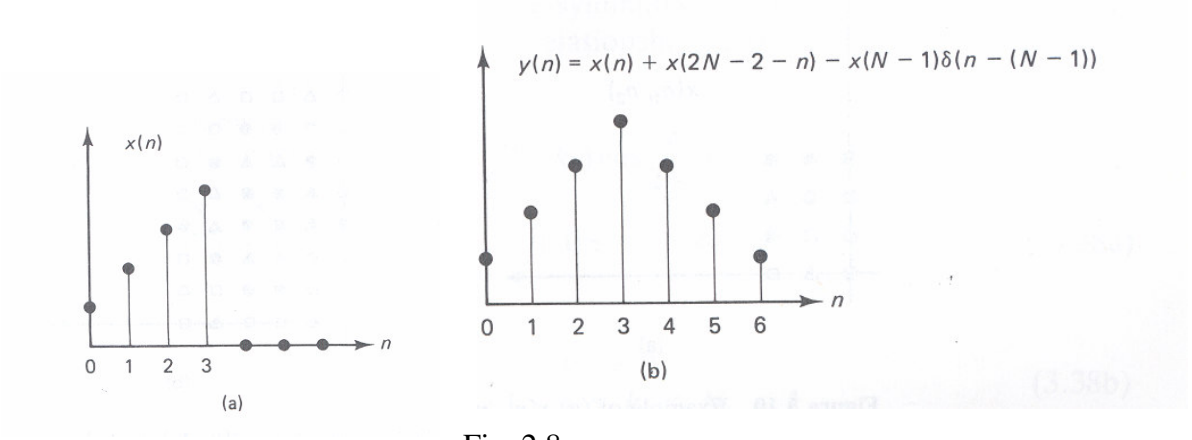


Fig. 2.8

2.3.2 The Two-Dimensional Discrete Cosine Transform

The 1-D DCT can be extended straightforwardly to two dimensions. Let $x(n_1, n_2)$ denote a 2-D sequence of $N_1 x N_2$ points that is zero outside $0 \le n_1 \le N_1 - 1,\ 0 \le n_2 \le N_2 - 1$. Derive the 2-D DCT pair by relating $x(n_1, n_2)$ to a new $2N_1 x 2N_2$ point sequence $y(n_1, n_2)$, which is then related to its $2N_1 x 2N_2$ point DFT $Y(k_1, k_2)$. Then relate $Y(k_1, k_2)$ to $C_x(k_1, k_2)$, the $N_1 x N_2$ point DCT. Specifically,

$$\begin{array}{ccccccc} N_1 x N_2 - point & & 2N_1 x 2N_2 - point & & 2N_1 x 2N_2 - point & & N_1 x N_2 - point \\ x(n_1, n_2) & \leftrightarrow & y(n_1, n_2) & \leftrightarrow & Y(k_1, k_2) & \leftrightarrow & C_x(k_1, k_2) \end{array}$$

The sequence $x(n_1,n_2)$ is related to $y(n_1,n_2)$ by

$$y(n_1,n_2) = x(n_1,n_2) + x(2N_1 - 1 - n_1, n_2) + x(n_1, 2N_2 - 1 - n_2) + x(2N_1 - 1 - n_1, 2N_2 - 1 - n_2) \quad (2.16)$$

An example of $x(n_1,n_2)$ and $y(n_1,n_2)$ when $N_1 = 3$, $N_2 = 4$ is shown in Fig. 2.9(a) and (b), respectively.

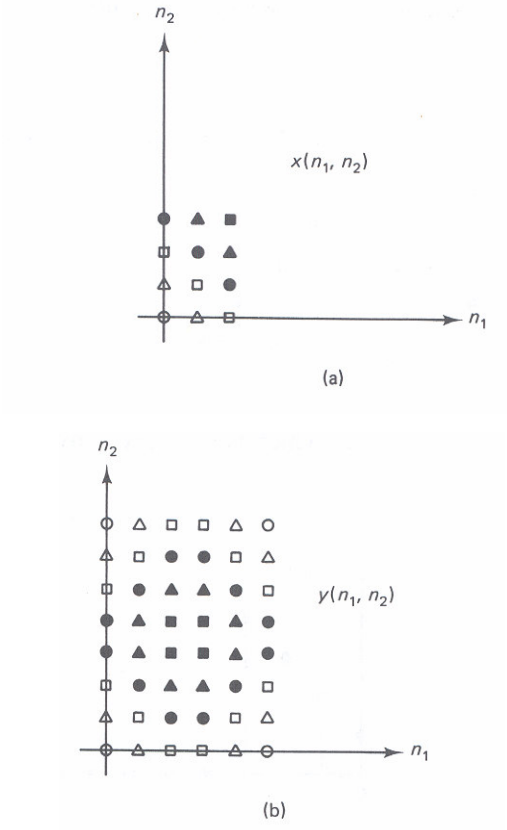


Fig. 2.9

A periodic sequence $\tilde{x}(n1, n2)$ with a period of $N_1 x N_2$ obtained by repeating $x(n_1,n_2)$ is shown in Fig. 2.10(a). A periodic sequence $y(n_1,n_2)$ with a period of $2N_1 x 2N_2$ obtained by repeating $y(n_1,n_2)$ is shown in Fig. 2.10(b). The artificial discontinuities present in $\tilde{x}(n1, n2)$ are not present in $\tilde{y}(n1, n2)$. The sequence $y(n_1,n_2)$ is related to $Y(k_1,k_2)$ by

$$Y(k_1,k_2) = DFT\left[y(n_1,n_2)\right]. \quad 2.17$$

The $N_1 x N_2$ point DCT of $x(n_1,n_2)$, $C_x(k_1,k_2)$, is obtained from $Y(k_1,k_2)$ by

$$C_x(k_1,k_2)=\begin{cases} W_{2N_1}^{\frac{k_1}{2}} W_{2N_2}^{\frac{k_2}{2}} Y(k_1,k_2), & 0\le k_1\le N_1-1, \quad 0\le k_2\le N_2-1 \\ 0, & otherwise \end{cases} \qquad 2.18$$

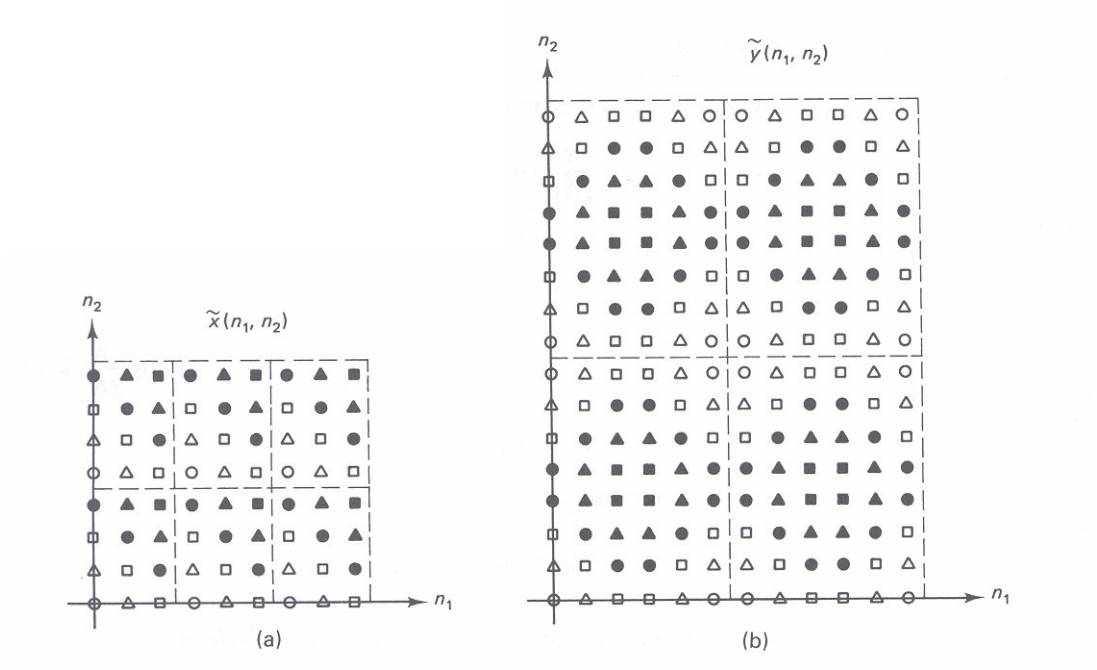


Fig. 2.10

From (2.16), (2.17), and (2.18), and after some algebra

$$C_x(k_1,k_2)=\begin{cases} \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} 4x(n_1,n_2)\cos\frac{\pi}{2N_1}k_1(2n_1+1)\cos\frac{\pi}{2N_2}k_2(2n_2+1), \\ for 0\le k_1\le N_1-1, 0\le k_2\le N_2-1 \\ 0, otherwise \end{cases} \qquad 2.19$$

Equation (2.19) is the definition of the 2-D DCT. The inverse DCT can be derived by relating $C_x(k_1,k_2)$ to $Y(k_1,k_2)$, exploiting the redundancy in $Y(k_1,k_2)$ due to the symmetry of

$y(n_1, n_2)$, relating $Y(k_1, k_2)$ to $y(n_1, n_2)$ through the inverse DFT relationship, and then relating $y(n_1, n_2)$ to $x(n_1, n_2)$. The result is

$$x(n_1, n_2) = \begin{cases} \dfrac{1}{N_1 N_2} \displaystyle\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w_1(k_1) w_2(k_2) C_x(k_1, k_2) \cos \frac{\pi}{2N_1} k_1 (2n_1 + 1) \cos \frac{\pi}{2N_2} k_2 (2n_2 + 1), \\ for 0 \le n_1 \le N_1 - 1, 0 \le n_2 \le N_2 - 1 \\ 0, otherwise \end{cases}$$

2.20a

Where

$$w_1(k_1) = \begin{cases} \dfrac{1}{2}, & k_1 = 0 \\ 1, & 1 \le k_1 \le N_1 - 1 \end{cases} \qquad 2.20b$$

$$w_2(k_2) = \begin{cases} \dfrac{1}{2}, & k_2 = 0 \\ 1, & 1 \le k_2 \le N_2 - 1 \end{cases} \qquad 2.20c$$

From (2.19) and (2.20), FDCT is written as

Two-Dimensional Discrete Cosine Transform Pair

$$C_x(k_1, k_2) = \begin{cases} \displaystyle\sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} 4x(n_1, n_2) \cos \frac{\pi}{2N_1} k_1 (2n_1 + 1) \cos \frac{\pi}{2N_2} k_2 (2n_2 + 1), \\ for 0 \le k_1 \le N_1 - 1, 0 \le k_2 \le N_2 - 1 \\ 0, otherwise \end{cases} \quad \textbf{FDCT} \qquad 2.21a$$

$$x(n_1, n_2) = \begin{cases} \dfrac{1}{N_1 N_2} \displaystyle\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} w_1(k_1) w_2(k_2) C_x(k_1, k_2) \cos \frac{\pi}{2N_1} k_1 (2n_1 + 1) \cos \frac{\pi}{2N_2} k_2 (2n_2 + 1), \\ for 0 \le n_1 \le N_1 - 1, 0 \le n_2 \le N_2 - 1 \\ 0, otherwise \end{cases}$$

IDCT 2.21b

Chapter 3

DISCRETE FRACTIONAL COSINE TRANSFORM

Introduction

There is a close relationship between the conventional Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT). This chapter introduce another transform, the Discrete Fractional Cosine Transform (DFRCT), which has a similar relationship with the Discrete Fractional Fourier Transform (DFRFT). The DFRCT share many useful properties of the regular DCT, and it has a free parameter, its fraction. When the fraction is zero, we get the cosine modulated version of the input signal. When it is unity, we get the conventional DCT. As the fraction changes from 0 to 1 we get different forms of the signal which interpolate between the cosine modulated form of the signal and its DCT representation. Thus, DFRCT is a general form of DCT which has an additional free parameter, and with this free parameter it may find its place in many applications where DCT is found to be useful.

3.1 FRACTIONAL OPERATIONS

The fourth power of 7 may be defined as 7^4= 7x7x7x7, but it is not obvious from this definition how one might define $7^{3.5}$. It must have taken sometime before the common definition $7^{3.5} = 7^{7/2} = \sqrt{7^7}$ emerged. The first and second derivatives of the function f (x) are commonly denoted by:

$\frac{df(x)}{dx}$ and $\frac{d^2 f(x)}{dx} = \frac{d}{dx}\left[\frac{df(x)}{dx}\right] = \frac{d[df(x)/dx]}{dx} = \left(\frac{d}{dx}\right)^2 f(x)$ respectively.

Higher order derivatives are defined similarly. Now what the 1.5th derivative of a function might mean is not clear from the above definition. Let F (μ) denote the Fourier transform of $f(x)$. The FT of the nth derivative of $f(x)$ $\left[i.e.\frac{d^n f(x)}{dx^n}\right]$ is known to be given by $(i2\pi\mu)^n$ F (μ), for any positive integer n. Now let us generalize this property by replacing n with the real order '**a**' and take it as the ath derivative of $f(x)$. Thus to find $\frac{d^a f(x)}{dx^a}$, the ath derivative of $f(x)$, find the inverse Fourier transform of $(i2\pi\mu)^a$ F (μ). In both of these examples we are dealing with the fractions of an operation performed on an entity, rather than fractions of the entity itself. $2^{0.5}$ is the square root of the integer 2. The function $[f(x)]^{0.5}$ is the square root of the function $f(x)$. But $\frac{d^{0.5} f(x)}{dx^{0.5}}$ is the 0.5th derivative of $f(x)$ with $\left(\frac{df(x)}{dx}\right)^{0.5}$ being the square root of the derivative operator $\frac{d}{dx}$. The process of going from the whole of an entity to fractions of it underlies several of the more important conceptual developments. e.g. fuzzy logic, where the binary 1 & 0 are replaced by continuous values representing our certainty or uncertainty of a proposition.

3.2 FRACTIONAL COSINE TRANSFORM

The DFRCT, which is the generalization of cosine transform. The real part of FRFT kernel was chosen as the kernel for Fractional Cosine Transform as in case of (CT) where real part of FT is chosen as (CT) kernel.

The FRFT belongs to the class of time frequency representation that has been extensively used by the signal processing community. In all the time frequency representation, one normally uses a plane with two orthogonal axis corresponding to time & frequency. If we consider a signal $x(t)$ to be represented along the time axis and its ordinary FT $X(F)$ to be represented along the frequency axis, then the FT operator (denoted by F) can be visualized as change in representation of signal corresponding to counter clock wise rotation of the axis by an angle $\pi/2$ [9,10]. This is consistent with some of absorb properties of FT. For example, two successive rotations of a signal through $\pi/2$ will result in an inversion of time axis. Moreover, four successive rotations will leave the signal unaltered since a rotation through 2π of the signal should leave the signal unaltered. The FRFT is a linear operator that corresponds to rotation of signal through an angle which is not multiple of $\pi/2$, that is it is the representation of the signal along the axis u making an angle α with time axis.

The FRFT $F^{\alpha}(u)$ of a function $f(x)$ is defined as

$$F^{\alpha}(u) = R_F^{\alpha}[f(x)](u) = \frac{1}{\sqrt{2\pi}} \int_{-\alpha}^{\alpha} k_{\alpha}(x,u) f(x) \exp(-jux/\sin\alpha) dx, \qquad (3.1)$$

Where

$$k_\alpha(x,u)=\frac{\exp\left[j\frac{1}{2}\alpha\right]}{\sqrt{j\sin\alpha}}\exp\left[\frac{1}{2}j\left(x^2+u^2\right)\cot\alpha\right] \qquad \textbf{(3.2)}$$

For $\alpha=\frac{1}{2}\pi$, for which $k_{\pi/2}(x,u)=1$, we have the normal FT, while for $\alpha\to 0$ we have the identity transformation: $F^\circ(x)=f(x)$. The FRFT of an even function is even, while the FRFT of an odd function is odd. Consider one sided function $f(x)$, with $f(x)=0$ for $x\prec 0$, and define the FRCT as

$$F^\alpha(u)=R_c^\alpha[f(x)](u)=R_F^\alpha[f(x)\pm f(-x)](u)=F^\alpha(u)\pm F^\alpha(-u) \qquad (3.3)$$

$$=\sqrt{\frac{2}{\pi}}\int_0^\alpha k_\alpha(x,u)f(x)\cos(ux/\sin\alpha)dx, \qquad (u\geq 0)$$

which reduces to the normal Cosine Transform (CT) for $\alpha=\frac{1}{2}\pi$. To express the relationship between the FRFT of one sided function Cosine Transform (CT) and Sine transform (ST) of this function in a different way, it can be written as

$$2F^\alpha(\pm u)=F_c^\alpha(u)\pm\exp(-j\alpha)F_s^\alpha(u). \qquad (3.4)$$

Considering the kernels of the fractional transforms R^α,

$$R_F^\alpha:(1/\sqrt{2\pi})k_\alpha(x,u)\exp(-jux/\sin\alpha) \qquad (3.5)$$

$$R_c^\alpha:(2/\sqrt{2\pi})k_\alpha(x,u)\cos(ux/\sin\alpha) \qquad (3.6)$$

R_c^α is related to the even part of R_F^α. In general we conclude that, to determine the FRCT of a one sided function f(x), we can as well determine the FRFT of the evenly extended two-sided function $f(x)+f(-x)$.

3.3 CALCULATION OF DFRCT KERNEL

With the advent of computers and enhanced computational capabilities the Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) came into existence in evaluation of FT for real time processing. Further these capabilities are enhanced by the introduction of DSP processors and Fast Fourier Transform (FFT) algorithms. On similar lines, so there arises a need for discretization of DFRCT [11]. This section introduces a discrete version of continuous fractional cosine transform using discrete Hermite eigenvector.

There are four types of DCT's and their kernel matrices are given as [12]:

1. DCT1

$$C_{N+1}^{I} = \sqrt{\frac{2}{N}}\left[k_m k_n \cos\left(\frac{mn\pi}{N}\right)\right] \tag{3.7}$$

For $m, n = 0,1,2,.....N.$

2. DCT-II

$$C_{N}^{II} = \sqrt{\frac{2}{N}}\left[k_m \cos\left(\frac{m\left(n+\frac{1}{2}\right)\pi}{N}\right)\right] \tag{3.8}$$

For $m, n = 1,2,.....N-1.$

3. DCT-III

$$C_{N}^{III} = \sqrt{\frac{2}{N}}\left[k_m \cos\left(\frac{\left(m+\frac{1}{2}\right)n\pi}{N}\right)\right] \tag{3.9}$$

For $m, n = 1,2,.....N-1.$

4. DCT-IV

$$C_N^{IV} = \sqrt{\frac{2}{N}}\left[k_m \cos\left(\frac{\left(m+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\pi}{N} \right)\right] \quad (3.10)$$

For $m, n = 1,2,.....N-1.$

k_m and k_n in the above four definition are defined as

$$k_m = \begin{cases} \frac{1}{\sqrt{2}}, & m = 0 \ and \ m = n \\ 1, & others \end{cases}$$

DCT-I kenel have symmetric structure and periodic with period 2. The periodicity means that repeated applications of DCT-I would give the original sequence. DCT-IV is same as DCT-I for symmetry and periodicity, but DCT-II and DCT-III operators are forward and inverse transverse pair of each other and are non-periodic. Here DCT-I will be chosen and will be used in developing DFRCT.

3.3.1 Eigenvectors and Eigenvalues of kernel matrices:

The DFT kernel matrix has four distinct eigen values (1, -j, -1, j) and its multiplicities are summarized in Table 3.1.

Table 3.1 Eigen value multiplicities of DFT-I kernel matrices

N	Multiplicity of 1	Multiplicity of -j	Multiplicity of -1	Multiplicity of j
4m	*m+1*	*m*	*m*	*m-1*
4m+1	*m+1*	*m*	*m*	*m*
4m+2	*m+1*	*m*	*m+1*	*m*
4m+3	*m+1*	*m*	*m+1*	*m*

Because the DFT has only four distinct eigenvalues, the DFT eigenvectors will constitute four eigenspaces. It is trivial to find that any vector spanned by the DFT eigenvectors corresponding to the same eigenvalue is still a DFT eigenvector. Therefore there exist infinite eigenvectors for the DFT kernel matrix. The multiplicities of DFT eigenvalues are just the dimensions of eigenspaces.

All the DFT eigenvectors are even or odd. The even eigenvectors with the eigenvalues 1, –1 in addition, the odd eigenvectors correspond to the eigenvalues j or -j.

The DCT-I eigenvectors can be attained from the DFT eigenvectors [13, 14].

If $v = [v_0, v_1, ..., v_{N-2}, v_{N-1}, v_{N-2}, ..., v_1]^T$ is an even eigenvector of the $(2N-2)$ point DFT kernel matrix. $F_{2N-2}v = \lambda v(\lambda = 1, -1)$. Then

$$v = [v_0, \sqrt{2}v_1, ..., \sqrt{2}v_{N-2}, v_{N-1}]^T \tag{3.11}$$

will be an eigenvector of the N point DCT-I kernel matrix, where λ is the corresponding eigenvalue.

$$C_N^1 \hat{v} = \lambda \hat{v} \tag{3.12}$$

Where

$$C_N^1 = \sqrt{\frac{2}{N-1}} \begin{bmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \cos\frac{\pi}{N-1} & \cdots & \cos\frac{(N-2)^2\pi}{N-1} & \frac{1}{\sqrt{2}}\cos\frac{(N-1)\pi}{N-1} \\ M & M & \ddots & M & N \\ \frac{1}{\sqrt{2}} & \cos\frac{(N-2)\pi}{N-1} & \cdots & \cos\frac{(N-2)^2\pi}{N-1} & \frac{1}{\sqrt{2}}\cos\frac{(N-2)(N-1)\pi}{N-1} \\ \frac{1}{2} & \frac{1}{\sqrt{2}}\cos\frac{(N-2)\pi}{N-1} & \cdots & \frac{1}{\sqrt{2}}\cos\frac{(N-2)(N-1)\pi}{N-1} & \frac{1}{2}\cos\frac{(N-1)^2\pi}{N-1} \end{bmatrix}$$

The eigenvalues of DCT-I kernel matrices are only 1 and –1. Their multiplicities are shown in Table 3.2.

Table 3.2 Eigen value multiplicities of DCT-I kernel matrices

N	**Multiplicity of 1**	**Multiplicity of -1**
Odd	$\frac{N+1}{2}$	$\frac{N-1}{2}$
Even	$\frac{N}{2}$	$\frac{N}{2}$

Now the N point DFRFT kernel is given by [15]:

$$F_{N,\alpha} = V_N D_N^{\frac{2\alpha}{\pi}} V_N^T = V_N \begin{bmatrix} 1 & & & 0 \\ & e^{-j\alpha} & & \\ & & 0 & \\ 0 & & & e^{-j(N-1)\alpha} \end{bmatrix} V_N^T \tag{3.13}$$

Where $V_N = [v_0 | v_1 | ..... | v_{N-1}]$, v_k is the kth order DFT hermite eigenvector, and α indicates the rotation angle of transform in time-frequency plane. When $\alpha=0$, $F_{N,\alpha}$ is an identity operator. If $\alpha=\pi/2$, the DFRFT becomes the conventional DFT. Similar to DFRFT, the *N* point DFRCT kernel can be identified as:

$$C_{N,\alpha} = V_N D_N^{\frac{2\alpha}{\pi}} V_N^T = V_N \begin{bmatrix} 1 & & & 0 \\ & e^{-j\alpha} & & \\ & & 0 & \\ 0 & & & e^{-j(N-1)\alpha} \end{bmatrix} V_N^T \tag{3.14}$$

Where $V_N = [v_0 | v_1 | ..... | v_{N-1}]$, v_k is the DCT-I order eigenvector given by equation 3.11, and α indicates the rotation angle of transform in the time-frequency plane. When $\alpha=0$, $C_{N,\alpha}$ is an

identity operator. If $\alpha=\pi/2$, the DFRCT becomes the conventional DCT. The steps were constructing the *N* point DFRCT kernel with angular parameter α are summarized as follow :

Step 1: Compute the M_c point DFT hermite even eigenvectors.

Where $M_c = 2(N-1)$

Step 2: Use Step 1 to compute the DCT-I eigenvectors from the DFT hermite even eigenvectors.

Step 3: Determine the DRFCT transform kernel by the following equation.

$$C_{N,\alpha} = V_N D_N^{2\frac{\alpha}{\pi}} V_N^T$$

Where $V_N = [v_0 | v_1 | ..... | v_{N-1}]$, v_k is the DCT-I eigenvector obtained from the kth order Hermite eigenvector given by equation (3.11).

Chapter 4

IMAGE COMPRESSION USING DFRCT

Introduction

Image compression using transform coding yields extremely good compression with controllable degradation of image quality. By adjusting the cut-off of transform coefficient, a compromise can be made between image quality and compression factor. The human visual system response is very dependent on spatial frequency. If we could somehow decompose the image into a set of waveforms, each with a particular spatial frequency, we might be able to separate the image structure the eye can see from the structure that is imperceptible. The DCT can provide a good approximation to this decomposition. In this chapter, DRFCT is used as tool for image compression.

4.1 IMAGE COMPRESSION ALGORITHM

Following steps are used in image compression using Discrete Fractional Cosine Transform are:-

1. An image is first partition into non-overlapped *NxN* sub images.
2. A 2-D DRFCT is applied to each block at particular value of 'a'. For simplicity, the 'a' order along x and y direction are taken to be same. This is done to convert the gray scale levels of pixels in spatial domain into coefficients in frequency domain.
3. The coefficients are normalized by different scale according to cut-off selected. To attain higher compression the larger value of cut-off is selected. The quantized coefficients are

rearranged in a zigzag scan order i.e. from lower frequency components to higher frequency components to be further compressed by efficient lossless coding strategies.

4. At decoder end simply inverse process of encoding by using inverse 2-D DFRCT is performed. Inverse DFRCT is obtained by negative value of 'a' that used in forward DFRCT.

4.2 CHARACTERISTIC TO JUDGE COMPRESSION ALGORITHM

Image quality describes the fidelity with which an image compression scheme recreates the source image data. There are four main characteristics to judge image-compression algorithms:

1. Compression Ratio
2. Compression Speed
3. Mean Square Error
4. Peak Signal to Noise Ratio

These characteristics are used to determine the suitability of a given compression algorithm for any application.

4.2.1 Compression Ratio

The compression ratio is equal to the size of the original image divided by the size of the compressed image. This ratio gives how much compression is achieved for a particular image.

The compression ratio achieved usually indicates the picture quality. Generally, the higher the compression ratio, the poorer the quality of the resulting image. The trade-off between compression ratio and picture quality is an important one to consider when compressing images.

Some compression schemes produce compression ratios that are highly dependent on the image content. This aspect of compression is called data dependency. Using an algorithm with a high degree of data dependency, an image of a crowd at a football game (which contains a lot of detail) may produce a very small compression ratio, whereas an image of a blue sky (which consists mostly of constant colors and intensities) may produce a very high compression ratio.

4.2.2 Compression Speed

Compression time and decompression time are defined as the amount of time required to compress and decompress a picture, respectively. Their value depends on the following considerations:

- the complexity of the compression algorithm
- the efficiency of the software or hardware implementation of the algorithm
- the speed of the utilized processor or auxiliary hardware

Generally, the faster that both operations can be performed, the better. Fast compression time increases the speed with which material can be created. Fast decompression time increases the speed with which the user can display and interact with images.

4.2.3 Mean Square Error

Mean Square Error measures the cumulative square error between the original and the compressed image.

The formula for MSE is giving as

$$\mathrm{MSE} = \left(\frac{1}{\mathrm{NxN}} \right) \sum_{x=1}^{N} \sum_{y=1}^{M} [I(x,y) - I'(x,y)]^2$$

Where NxN is the size of image

$I'(x,y)$ And $I(x,y)$ are the matrix elements of the decrypted and original elements at *(x,y)* pixel.

4.2.4 Peak Signal to Noise Ratio

Peak signal-to-reconstructed image measure known as PSNR.

The formula for PSNR is given as

$$PSNR = 20 * log\,10 \left(\frac{255}{sqrt(\,MSE\,)} \right)$$

Here signal is original image and noise is error in reconstructed image.

In general, a good reconstructed image is one with low MSE and high PSNR. That means that the image has low error and high image fidelity.

.

Chapter 5

IMAGE ENCRYPTION USING DRFCT

Introduction

In this chapter, a novel technique for image encryption that uses cascaded multistages of DFRCT with random phase mask at each stage is presented. This technique provides more secure image encryption and zero mean square error, with the addition of extra degree of freedom provided by DRFCT.

5.1 KEYS IN DRFCT

It has been recently noticed that Fractional Transforms can be used in encryption of images [15]. In this encryption utilizes cascaded multistage DFRCT with random face filters in between. The n-stage of DFRCT can provide n-dimensional extra keys indicated by the fractional orders. In case of two-dimensional DFRCT transform, there are two different fractional orders along x-axis and y-axis respectively. Such a system can have n-1 random phase filters, so that the total encryption keys can be increased to as many as 3n-1. Thus the security strength of the encryption keys may be greatly enhanced.

For simplicity, the 'a' order along x and y direction are taken to be same i.e. $\alpha_x = \alpha_y = \alpha$ and three stages of DFRCT are cascaded together. Thus in the intermediate planes two randomly encoded phase masks are used.

5.2 ALGORITHM FOR IMAGE ENCRYPTION

Algorithm consists of two parts Encryption to encrypt image and Decryption to retrieve image.

5.21 Encryption

The image encryption can be described as follows [16]. Let $f(x_0, y_0)$, a real valued two-dimensional data, denote the image that is to be encrypted. To this data DFRCT is applied continuously three times with the fractional orders α_1, α_2 and α_3, respectively. In the intermediate stages two random phase masks

$H_1(x_1, y_1) = [-i2\pi\phi_1(x_1, y_1)]$ and $H_2(x_2, y_2) = [-i2\pi\phi_2(x_2, y_2)]$ respectively are used. Where $\phi_2(x_2, y_2)$ are randomly generated homogeneously distributed functions.

Thus the resultant transformed function $\psi(x, y)$ can be written as

$$\psi(x, y) = F_c^{\alpha 3}\{\psi_2(x_2, y_2)H_2(x_2, y_2)\}(x, y) \qquad 6.1$$

with

$$\psi_2(x_2, y_2) = F_c^{\alpha 2}\{\psi_1(x_1, y_1)H_1(x_1, y_1)\}(x_2, y_2) \qquad 6.2$$

and

$$\psi_1(x_1, y_1) = F_c^{\alpha 1}\{f_2(x_0, y_0)\}(x_1, y_1). \qquad 6.3$$

The resultant transformed function $\psi(x, y)$ can be regarded as the encrypted image.

5.2.2Decryption

The decryption process is the reverse operation with respect to the encryption. Firstly take DFRCT of order -α_3 on the encrypted image $\psi(x, y)$ and multiplying the random phase

mask $H_2^*(x_2, y_2)$, and then the mid term function $\psi_2(x_2, y_2)$ is obtained. Then perform DFRCT of order -α_2 on the function $\psi_2(x_2, y_2)$ and multiplying the random phase mask $H_1^*(x_1, y_1)$, thus function $\psi_1(x_1, y_1)$ will be recovered. After another DFRCT of order -α_1 on the function $\psi_1(x_1, y_1)$, the original image $f(x_0, y_0)$ is obtained. Here the random phase masks $H_1^*(x_1, y_1)$ and $H_2^*(x_2, y_2)$ are complex conjugates $H_1(x_1, y_1)$ and $H_2(x_2, y_2)$, respectively.

Chapter 5

IMAGE ENCRYPTION USING DRFCT

Introduction

In this chapter, a novel technique for image encryption that uses cascaded multistages of DFRCT with random phase mask at each stage is presented. This technique provides more secure image encryption and zero mean square error, with the addition of extra degree of freedom provided by DRFCT.

5.1 KEYS IN DRFCT

It has been recently noticed that Fractional Transforms can be used in encryption of images [15]. In this encryption utilizes cascaded multistage DFRCT with random face filters in between. The n-stage of DFRCT can provide n-dimensional extra keys indicated by the fractional orders. In case of two-dimensional DFRCT transform, there are two different fractional orders along x-axis and y-axis respectively. Such a system can have n-1 random phase filters, so that the total encryption keys can be increased to as many as 3n-1. Thus the security strength of the encryption keys may be greatly enhanced.

For simplicity, the 'a' order along x and y direction are taken to be same i.e. $\alpha_x = \alpha_y = \alpha$ and three stages of DFRCT are cascaded together. Thus in the intermediate planes two randomly encoded phase masks are used.

5.2 ALGORITHM FOR IMAGE ENCRYPTION

Algorithm consists of two parts Encryption to encrypt image and Decryption to retrieve image.

5.21 Encryption

The image encryption can be described as follows [16]. Let $f(x_0, y_0)$, a real valued two-dimensional data, denote the image that is to be encrypted. To this data DFRCT is applied continuously three times with the fractional orders α_1, α_2 and α_3, respectively. In the intermediate stages two random phase masks

$H_1(x_1, y_1) = [-i2\pi\phi_1(x_1, y_1)]$ and $H_2(x_2, y_2) = [-i2\pi\phi_2(x_2, y_2)]$ respectively are used. Where $\phi_2(x_2, y_2)$ are randomly generated homogeneously distributed functions.

Thus the resultant transformed function $\psi(x, y)$ can be written as

$$\psi(x, y) = F_c^{\alpha 3}\{\psi_2(x_2, y_2)H_2(x_2, y_2)\}(x, y) \qquad 6.1$$

with

$$\psi_2(x_2, y_2) = F_c^{\alpha 2}\{\psi_1(x_1, y_1)H_1(x_1, y_1)\}(x_2, y_2) \qquad 6.2$$

and

$$\psi_1(x_1, y_1) = F_c^{\alpha 1}\{f_2(x_0, y_0)\}(x_1, y_1). \qquad 6.3$$

The resultant transformed function $\psi(x, y)$ can be regarded as the encrypted image.

5.2.2Decryption

The decryption process is the reverse operation with respect to the encryption. Firstly take DFRCT of order -α_3 on the encrypted image $\psi(x, y)$ and multiplying the random phase

mask $H_2^*(x_2, y_2)$, and then the mid term function $\psi_2(x_2, y_2)$ is obtained. Then perform DFRCT of order -α_2 on the function $\psi_2(x_2, y_2)$ and multiplying the random phase mask $H_1^*(x_1, y_1)$, thus function $\psi_1(x_1, y_1)$ will be recovered. After another DFRCT of order -α_1 on the function $\psi_1(x_1, y_1)$, the original image $f(x_0, y_0)$ is obtained. Here the random phase masks $H_1^*(x_1, y_1)$ and $H_2^*(x_2, y_2)$ are complex conjugates $H_1(x_1, y_1)$ and $H_2(x_2, y_2)$, respectively.