Image Processing Digital Assignment 2

Name: Om Ashish Mishra

Registration Number: 16BCE0789

Slot: G2

The Question:

Explain about Discrete Fourier Transform-Discrete Cosine Transform-Haar Transform and Hough Transform

The Answer:

Discrete Fourier Transform

The Fourier transform is a mathematical function that takes a time-based pattern as input and determines the overall cycle offset, rotation speed and strength for every possible cycle in the given pattern. The Fourier transform is applied to waveforms which are basically a function of time, space or some other variable. The Fourier transform decomposes a waveform into a sinusoid and thus provides another way to represent a waveform.

Here 2D DFT expression is given below:

$$f(m,n) \xrightarrow{\text{2D-DFT}} F(k,l)$$

where $F(k, l)$ is defined as

$$F(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n) e^{-j\frac{2\pi}{M}mk} e^{-j\frac{2\pi}{N}nl}$$

$$F(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n) e^{-j\frac{2\pi}{M}nl} e^{-j\frac{2\pi}{N}mk}$$

The discrete Fourier transform is an important image processing tool which is used to decompose an image into its sine and cosine components. It converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input

sequence. An inverse DFT is a Fourier series, using the DTFT samples as coefficients of complex sinusoids at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The DFT is therefore said to be a frequency domain representation of the original input sequence. If the original sequence spans all the non-zero values of a function, its DTFT is continuous (and periodic), and the DFT provides discrete samples of one cycle. If the original sequence is one cycle of a periodic function, the DFT provides all the non-zero values of one DTFT cycle.

In image processing, the samples can be the values of pixels along a row or column of a raster image. The Discrete Fourier Transform does not contain all frequencies forming an image but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domain are of the same size.

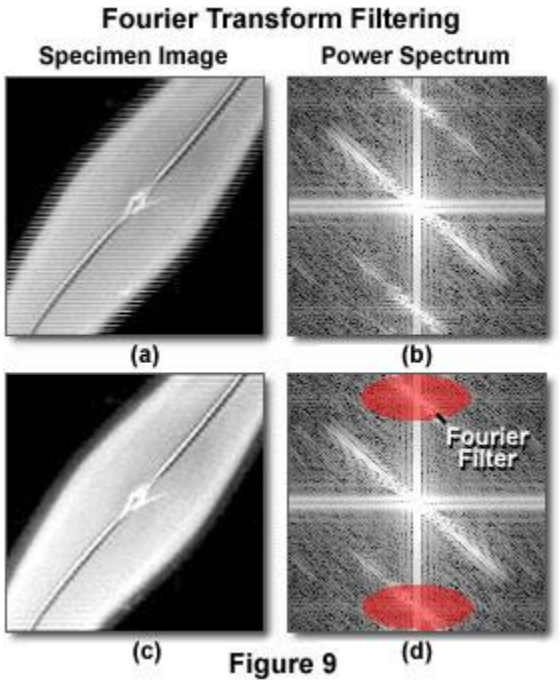
For a square image of NxN, the two dimensional DFT is given by:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi (\frac{ki}{N} + \frac{lj}{N})}$$

where *f(a,b)* is the image in the spatial domain and the exponential term is the basis function corresponding to each point *F(k,l)* in the Fourier space. The equation can be interpreted as: the value of each point *F(k,l)* is obtained by multiplying the spatial image with the corresponding base function and summing the result.

The basis functions are sine and cosine waves with increasing frequencies, *i.e. F(0,0)* represents the DC-component of the image which corresponds to the average brightness and *F(N-1,N-1)*represents the highest frequency.

The Fourier Transform is used if we want to access the geometric characteristics of a spatial domain image. Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain.



Applying Discrete Fourier Transform to an Image

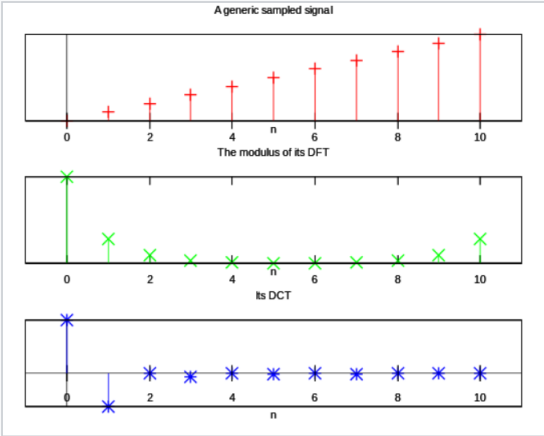
Discrete Cosine Transform

The transformation the Joint Photographic Experts Group chose for the task was the Discrete Cosine Transformation (DCT). Recall that the preprocessing portion of algorithm partitions the image into 8 x 8 blocks, so the DCT is an 8 x 8 matrix. The values of the matrix are given below.

$$U = \frac{1}{2}\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{11\pi}{16} & \cos\frac{13\pi}{16} & \cos\frac{15\pi}{16} \\ \cos\frac{2\pi}{16} & \cos\frac{6\pi}{16} & \cos\frac{10\pi}{16} & \cos\frac{14\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{22\pi}{16} & \cos\frac{26\pi}{16} & \cos\frac{30\pi}{16} \\ \cos\frac{3\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{27\pi}{16} & \cos\frac{33\pi}{16} & \cos\frac{39\pi}{16} & \cos\frac{45\pi}{16} \\ \cos\frac{4\pi}{16} & \cos\frac{12\pi}{16} & \cos\frac{20\pi}{16} & \cos\frac{28\pi}{16} & \cos\frac{36\pi}{16} & \cos\frac{44\pi}{16} & \cos\frac{52\pi}{16} & \cos\frac{60\pi}{16} \\ \cos\frac{5\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{25\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{45\pi}{16} & \cos\frac{55\pi}{16} & \cos\frac{65\pi}{16} & \cos\frac{75\pi}{16} \\ \cos\frac{6\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{30\pi}{16} & \cos\frac{42\pi}{16} & \cos\frac{54\pi}{16} & \cos\frac{66\pi}{16} & \cos\frac{78\pi}{16} & \cos\frac{90\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{49\pi}{16} & \cos\frac{63\pi}{16} & \cos\frac{77\pi}{16} & \cos\frac{91\pi}{16} & \cos\frac{105\pi}{16} \end{bmatrix}$$

Elements of the 8 x 8 DCT matrix.

A **discrete cosine transform** (**DCT**) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical for compression, since it turns out (as described below) that fewer cosine functions are needed to approximate a typical signal, whereas for differential equations the cosines express a particular choice of boundary conditions.



DCT-II (bottom) compared to the DFT (middle) of an input signal (top).

The Discrete Cosine Transform works by separating images into parts of differing frequencies. During a step called quantization, a part of compression actually occurs the less important

frequencies are discarded, and is hence, ‘lossy’. Then, only the most important frequencies that remain are used to retrieve the image in the decompression process. As a result, reconstructed images contain some distortion.

The Discrete Cosine Transform Equation

The DCT equation computes the i,jth entry of the DCT of an image

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N} \right] \cos\left[\frac{(2y+1)j\pi}{2N} \right]$$

$$C(u) = \left\{ \begin{array}{ll} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{array} \right\}$$

Where,

p(x,y) is the x,yth element of the image represented by the matrix p.

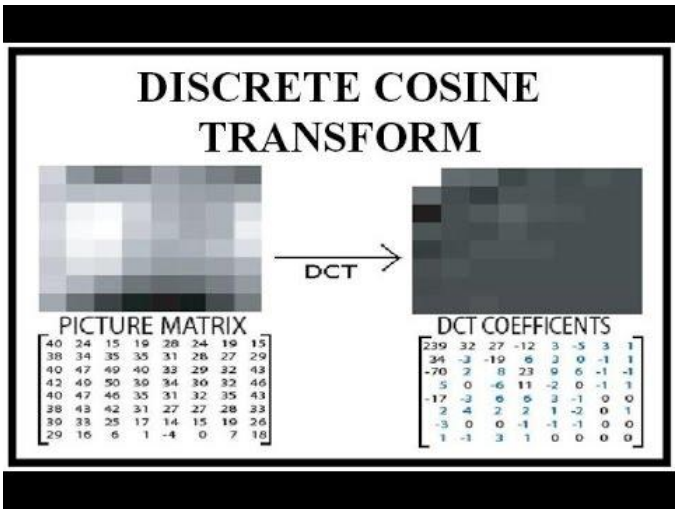
N is the size of the block that the DCT is done on.

The equation calculates one entry (i,jth) of the transformed image from the pixel values of the original image matrix. For the standard 8x8 block that the jpeg compression uses, N equals 8 and x and y range from 0 to 7. Therefore D(*i*,*j*) would be as in equation

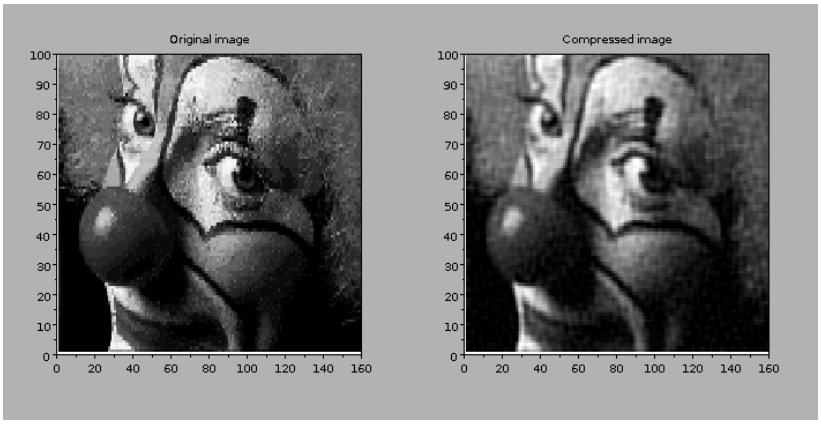
$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} p(x,y) \cos\left[\frac{(2x+1)i\pi}{16} \right] \cos\left[\frac{(2y+1)j\pi}{16} \right]$$

The Matrix form of DCT is defined as

$$T_{i,j} = \left\{ \begin{array}{ll} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N} \right] & \text{if } i > 0 \end{array} \right\}$$



Matrix Form of DCT



Applying Compression using DCT

Haar Transform

The starting point for the definition of the Haar transform is the Haar functions $h_k(z)$, which are defined in the closed interval $[0,1]$.The order k of the function is uniquely decomposed into two integers p,q

$$k = 2^p = q - 1,\ k = 0,1.....,L-1,\ and\ L = 2^n \quad (4.6.1)$$

where

$0 \le p \le n-1, 0 \le q \le 2^p$ for $p \neq 0$ and $q = 0$ or 1 for $p = 0$

Table (4.5.1) summarizes the respective values for $L = 8$. The Haar functions are

$$h_0(z) \equiv h_{00}(z) = \frac{1}{\sqrt{L}},$$

$$h_k(z) \equiv h_{pq}(z) = \frac{1}{\sqrt{L}} \begin{cases} 2^{\frac{p}{2}} & \frac{q-1}{2^p} \le z < \frac{q-1}{2^p} \\ -2^{\frac{p}{2}} & \frac{q-1}{2^p} \le z < \frac{q}{2^p} \\ 0 & otherwise\ in\ [0,1] \end{cases} \quad (4.6.2)$$

Table (4.5.1): Parameters for the Haar functions

K	0	1	2	3	4	5	6	7
P	0	0	1	2	2	2	2	2
q	0	1	1	1	1	1	3	4

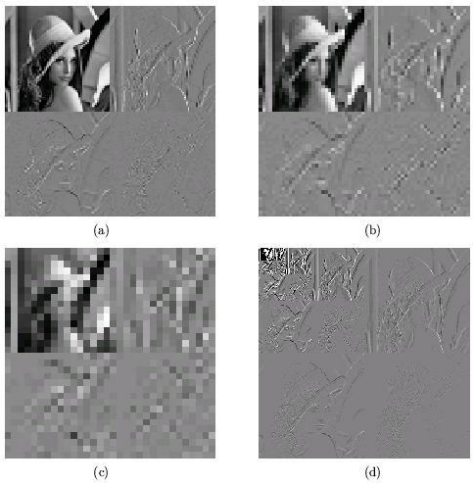
The Haar transform matrix of order L consists of rows resulting from the preceding functions computed at the points $z = m/L,\ m = 0,1,2,...,L-1.$ For example, the 8 x 8 transform matrix is

$$H = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2}_1 \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix} \quad (4.6.3)$$

It is not difficult to see that $H^{-1} = H^T$ that is H is orthogonal.

In mathematics, the Haar transform is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis. Wavelet analysis is similar to Fourier analysis in that it allows a target function over an interval to be represented in terms of an orthonormal basis. The Haar sequence is now recognised as the first known wavelet basis and extensively used as a teaching example.

The Harr transform is the imaging-related operation which ties to multiresolution analysis. Its basis functions are the oldest and simplest known orthonormal wavelets.



Applying Harr Transform

Hough Transform

The Hough transform is a way of finding the most likely values which represent a line (or a circle, or many other things).

You give the Hough transform a picture of a line as input. This picture will contain two types of pixels: ones which are part of the line, and ones which are part of the background.

For each pixel that is part of the line, all possible combinations of parameters are calculated. For example, if the pixel at co-ordinate (1, 100) is part of the line, then that could be part of a line where the gradient (m) = 0 and y-intercept (c) = 100. It could also be part of m = 1, c = 99; or m = 2, c = 98; or m = 3, c = 97; and so on. You can solve the line equation y = mx + c to find all possible combinations.

Each pixel gives one vote to each of the parameters (m and c) that could explain it. So you can imagine, if your line has 1000 pixels in it, then the correct combination of m and c will have 1000 votes.

The combination of m and c which has the most votes is what is returned as the parameters for the line.

The **Hough transform** is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform"[1] after the related 1962 patent of Paul Hough. The transform was popularized in the computer vision community by Dana H. Ballardthrough a 1981 journal article titled "Generalizing the Hough transform to detect arbitrary shapes"

In automated analysis of digital images, a subproblem often arises of detecting simple shapes, such as straight lines, circles or ellipses. In many cases an edge detector can be used as a pre-processing stage to obtain image points or image pixels that are on the desired curve in the image space. Due to imperfections in either the image data or the edge detector, however, there may be missing points or pixels on the desired curves as well as spatial deviations between the ideal line/circle/ellipse and the noisy edge points as they are obtained from the edge detector. For these reasons, it is often non-trivial to group the extracted edge features to an appropriate set of lines, circles or ellipses. The purpose of the Hough transform is to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects.

The simplest case of Hough transform is detecting straight lines. In general, the straight-line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, vertical lines pose a problem. They would give rise to unbounded values of the slope parameter m. Thus, for computational reasons, Duda and Hart proposed the use of the Hesse normal form

$$r = x \cos\theta + y \sin\theta$$

Where r distance from the origin to the closest point on the straight line, and *theta* is the angle between the axis and the line connecting the origin with that closest point.

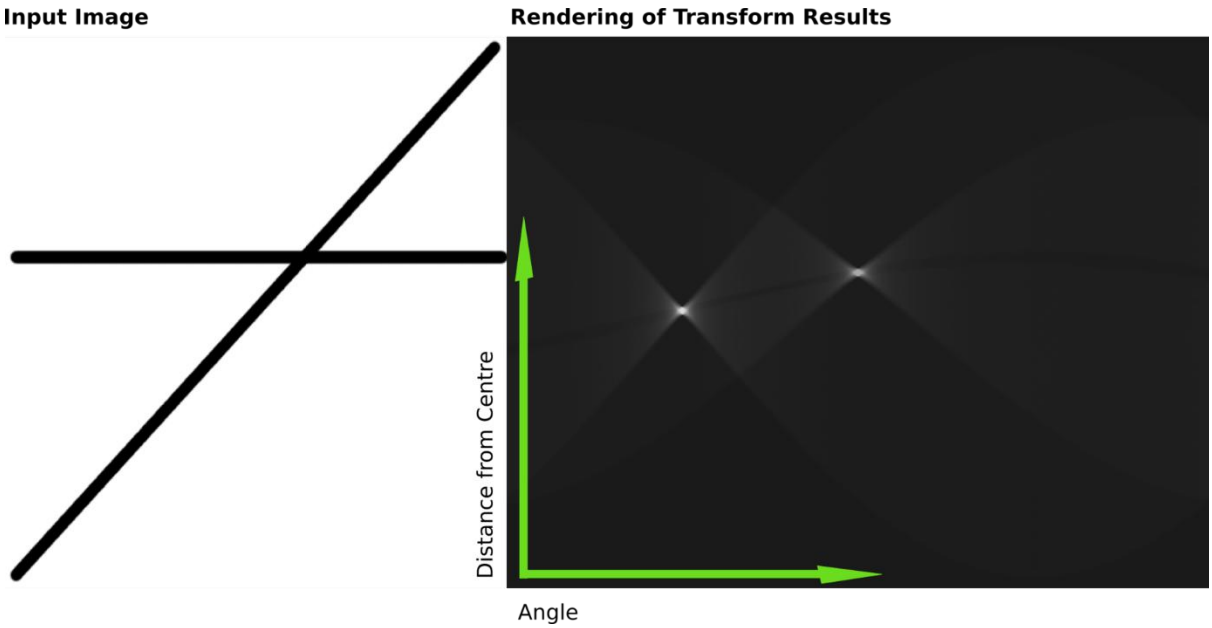
Implementation

The linear Hough transform algorithm uses a two-dimensional array, called an accumulator, to detect the existence of a line. The dimension of the accumulator equals the number of unknown parameters, i.e., two, considering quantized values of r and θ in the pair (r,θ). For each pixel at *(x,y)* and its neighborhood, the Hough transform algorithm determines if there is enough evidence of a straight line at that pixel. If so, it will calculate the parameters (r,θ) of that line, and then look for the accumulator's bin that the parameters fall into, and increment the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off. (Shapiro and Stockman, 304) The simplest way of finding these *peaks* is by applying some form of threshold, but other techniques may yield better results in different circumstances – determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often necessary, in the next step, to find which parts of the image match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines.

The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator—one dimension of this matrix is the quantized angle θ and the other dimension is the quantized distance r. Each element of the matrix has a value equal to the sum of the points or pixels that are positioned on the line represented by quantized parameters (r, θ). So, the element with the highest value indicates the straight line that is most represented in the input image.

Example

The following is a different example showing the results of a Hough transform on a raster image containing two thick lines.



Limitations

The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin.

Also, when the number of parameters is large (that is, when we are using the Hough transform with typically more than three parameters), the average number of votes cast in a single bin is very low, and those bins corresponding to a real figure in the image do not necessarily appear to have a much higher number of votes than their neighbors. The complexity increases at a rate of $\mathbf{O(A^{m-2})}$ with each additional parameter, where A is the size of the image space and m is the number of parameters. (Shapiro and Stockman, 310) Thus, the Hough transform must be used with great care to detect anything other than lines or circles.

Finally, much of the efficiency of the Hough transform is dependent on the quality of the input data: the edges must be detected well for the Hough transform to be efficient. Use of the Hough transform on noisy images is a very delicate matter and generally, a denoising stage must be used before. In the case where the image is corrupted by speckle, as is the case in radar images, the Radon transform is sometimes preferred to detect lines, because it attenuates the noise through summation.