|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Digital Image Processing** | **Semester:** | **VII** |
| **Date of Performance:** | **November 26, 2022** | **Batch No:** | **DIP2** |
| **Faculty Name:** | **Prof. Gopal Gupta** | **Roll No:** | **1912052** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** |  |

**Experiment No: 7**

**Title: To study image transforms and morphology**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| To study the process of image transforms and morphing |

|  |
| --- |
| **COs to be achieved:** |
| 1. **Understand fundamental theory and models of image processing** |

|  |
| --- |
| **Theory:** |
| Transformation  Transformation is a function. A function that maps one set to another set after performing some operations. The system would perform some processing on the input image and gives its output as a processed image. It is shown below.  introduction image  Now function applied inside this digital system that processes an image and converts it into output can be called a transformation function.  As it shows transformation or relation, that is how an image1 is converted to image2.  Image transformation.  Consider this equation  G(x,y) = T{ f(x,y) }  In this equation,  F(x,y) = input image on which transformation function has to be applied.  G(x,y) = the output image or processed image.  T is the transformation function.  This relation between the input image and the processed output image can also be represented as.  s = T (r)  where r is actually the pixel value or gray level intensity of f(x,y) at any point. And s is the pixel value or gray level intensity of g(x,y) at any point.  Examples  Consider this transformation function.  transformation1  Let’s take the point r to be 256, and the point p to be 127. Consider this image to be a one-bpp image. That means we have only two levels of intensity which are 0 and 1. So in this case the transformation shown by the graph can be explained as.  All the pixel intensity values that are below 127 (point p) are 0, which means black. And all the pixel intensity values that are greater than 127, are 1, which means white. But at the exact point of 127, there is a sudden change in transmission, so we cannot tell that at that exact point, the value would be 0 or 1.  Mathematically this transformation function can be denoted as:  transformation  Consider another transformation like this  transformation  Now if you will look at this particular graph, you will see a straight transition line between the input image and the output image.  It shows that for each pixel or intensity value of the input image, there is the same intensity value of the output image. That means the output image is an exact replica of the input image.  It can be mathematically represented as:  g(x,y) = f(x,y)  the input and output image would be in this case are shown below.  input image and output image  Transform methods in image processing  An [image transform](https://in.mathworks.com/discovery/image-transform.html) can be applied to an image to convert it from one domain to another. Viewing an image in domains such as frequency or [Hough space](https://in.mathworks.com/help/images/ref/hough.html) enables the identification of features that may not be as easily detected in the spatial domain. Common image transforms include:   * [Hough Transform](https://in.mathworks.com/help/images/object-analysis.html#f11-27827), used to find lines in an image * [Radon Transform](https://in.mathworks.com/help/images/reconstructing-an-image-from-projection-data.html), used to reconstruct images from fan-beam and parallel-beam projection data * [Discrete Cosine Transform](https://in.mathworks.com/help/images/discrete-cosine-transform.html), used in image and video compression * [Discrete Fourier Transform](https://in.mathworks.com/help/images/fourier-transform.html), used in filtering and frequency analysis * [Wavelet Transform](https://in.mathworks.com/help/wavelet/ref/dwt2.html), used to perform discrete wavelet analysis, denoise, and fuse images   Computing the Hough Transform of a Gantrycrane image  Computing the [Hough Transform](https://in.mathworks.com/help/images/ref/hough.html) of a Gantrycrane image.  Detecting straight lines using a radon transform  Detecting straight lines [using a radon transform.](https://in.mathworks.com/help/images/radon-transform.html)  An effective approach to applying image transforms includes using a comprehensive environment for data analysis, visualization, and algorithm development. |

|  |
| --- |
| **Stepwise-Procedure:** |
| 1. Select a photo of your faviourate actor/Actors 2. Click selfie in similar posture & transfer to PC 3. Apply Harr, Walsh, hadamard, cosine, Slant, DWT, DFT, Transfer & plot output 4. Perform edge detection with all types & find best 5. Perform discontinuity with reference to point, line & edge 6. Perform Hough transform 7. Perform Sementation extract Head & Morph image (scale & Rotate) with photo of actor/ Actors 8. Repeat same using Matlab/ Python |

|  |
| --- |
| **Output** |
| Upload picture screenshots for all approaches with intermediate steps  **HAAR TRANSFORM**  clear all;  close all;  clc;  img = imread(‘Vedant.jpg');  gray=rgb2gray(img);  figure;  imshow(gray);  title('gray image');  % [a2,h2,v2,d2] = haart2(gray,2);  % % imagesc(a2)  % figure;  % imshow(a2);  % title('haar image');  in\_img = im2double(gray);  [in\_colm, in\_row] = size(gray);  %% Performing Transformation  sqrt2 = sqrt(2);  out\_img = zeros(in\_colm,in\_row);  % Computing for Max Level in our case 1024 ie 10 Levels  lvl\_count = 1;  while (lvl\_count\*2 <= in\_colm) % For colm Ops  lvl\_count = lvl\_count\*2;  end  % Transformation Along Colms and downsampling by 2  while (1 < lvl\_count) % 1024, 512, 256,...,1  lvl\_count = floor(lvl\_count/2);  % haar\_matrix = 1/sqrt(2)\*[1 1; 1 -1] implementation  out\_img(1:lvl\_count, :) = (in\_img(1:2:2\*lvl\_count-1, :) + in\_img(2:2:2\*lvl\_count, :))/sqrt2;  out\_img(lvl\_count+1:lvl\_count+lvl\_count, :) = (in\_img(1:2:2\*lvl\_count-1, :) - in\_img(2:2:2\*lvl\_count, :))/sqrt2;  in\_img(1:2\*lvl\_count, :) = out\_img(1:2\*lvl\_count, :); % Updating Colms  end  lvl\_count = 1;  while (lvl\_count\*2 <= in\_row) % For row ops  lvl\_count = lvl\_count\*2;  end  % Transformation Along Rows and downsampling by 2  while (1 < lvl\_count)  lvl\_count = floor(lvl\_count/2); % In this case K will be 1024, 512, 256....1  out\_img(:, 1:lvl\_count) = (in\_img(: ,1:2:2\*lvl\_count-1) + in\_img(: ,2:2:2\*lvl\_count))/sqrt2;  out\_img(:, lvl\_count+1:lvl\_count+lvl\_count) = (in\_img(: ,1:2:2\*lvl\_count-1) - in\_img(: ,2:2:2\*lvl\_count))/sqrt2;  in\_img(:, 1:2\*lvl\_count) = out\_img(:,1:2\*lvl\_count); % Updating Rows  end  figure  imshow(in\_img), title('Haar Transformation on Input Image');  %% Reconstruction of Original Image by Performing IHT  % Inverse transform Along Rows and upsampling by 2  lvl\_count = 1;  while (lvl\_count\*2 <= in\_row)  in\_img(:, 1:2:2\*lvl\_count-1) = (out\_img(: ,1:lvl\_count) + out\_img(: ,1+lvl\_count:lvl\_count+lvl\_count))/sqrt2;  in\_img(:, 2:2:2\*lvl\_count) = (out\_img(: ,1:lvl\_count) - out\_img(: ,1+lvl\_count:lvl\_count+lvl\_count))/sqrt2;  out\_img(:, 1:2\*lvl\_count) = in\_img(:, 1:2\*lvl\_count); % Updating Rows  lvl\_count = lvl\_count\*2;  end  % Inverse transform Along Colms and upsampling by 2  lvl\_count = 1;  while (lvl\_count\*2 <= in\_colm)  in\_img(1:2:2\*lvl\_count-1, :) = (out\_img(1:lvl\_count, :) + out\_img(1+lvl\_count:lvl\_count+lvl\_count, :))/sqrt2;  in\_img(2:2:2\*lvl\_count, :) = (out\_img(1:lvl\_count, :) - out\_img(1+lvl\_count:lvl\_count+lvl\_count, :))/sqrt2;  out\_img(1:2\*lvl\_count, :) = in\_img(1:2\*lvl\_count, :); % Updating Colms  lvl\_count = lvl\_count\*2;  end  figure;  imshow(out\_img,[]), title('Final Restored Image');      **COSINE TRANSFORM**  clear all;  close all;  clc;  img = imread(‘Vedant.jpg');  gray=rgb2gray(img);  figure;  imshow(gray);  title('original gray image');  % discrete cosine transform  J = dct2(gray);  figure;  imshow(log(abs(J)),[])  colormap parula  colorbar  J(abs(J) < 10) = 0;  K = idct2(J);  K = rescale(K);  figure;  imshow(K);  title('Processed Image');      **DISCRETE WAVELET TRANSFORM**  clear all;  close all;  clc;  img = imread(Vedant.jpg');  gray=rgb2gray(img);  figure;  imshow(gray);  title('original gray image');  [cA,cH,cV,cD] = dwt2(gray,'sym4','mode','per');  subplot(2,2,1)  imagesc(cA)  colormap gray  title('Approximation')  subplot(2,2,2)  imagesc(cH)  colormap gray  title('Horizontal')  subplot(2,2,3)  imagesc(cV)  colormap gray  title('Vertical')  subplot(2,2,4)  imagesc(cD)  colormap gray  title('Diagonal') |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| Answer the following questions:   1. What are the different transforms  * Hough Transform, used to find lines in an image * Radon Transform, used to reconstruct images from fan-beam and parallel-beam projection data * Discrete Cosine Transform, used in image and video compression * Discrete Fourier Transform, used in filtering and frequency analysis * Wavelet Transform, used to perform discrete wavelet analysis, denoise, and fuse images |

|  |
| --- |
| **Conclusions:** |
| In this experiment we implemented all major image transforms, found their inverse transform to and displayed the compressed image. |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |