Subject: EE569

Homework 3

Name: Shreyansh Dnyanesh Daga

USC ID: 6375-3348-33

Email: sdaga@usc.edu

Date: 11/03/2013

Problem: 1 Spatial Warping

Motivation:

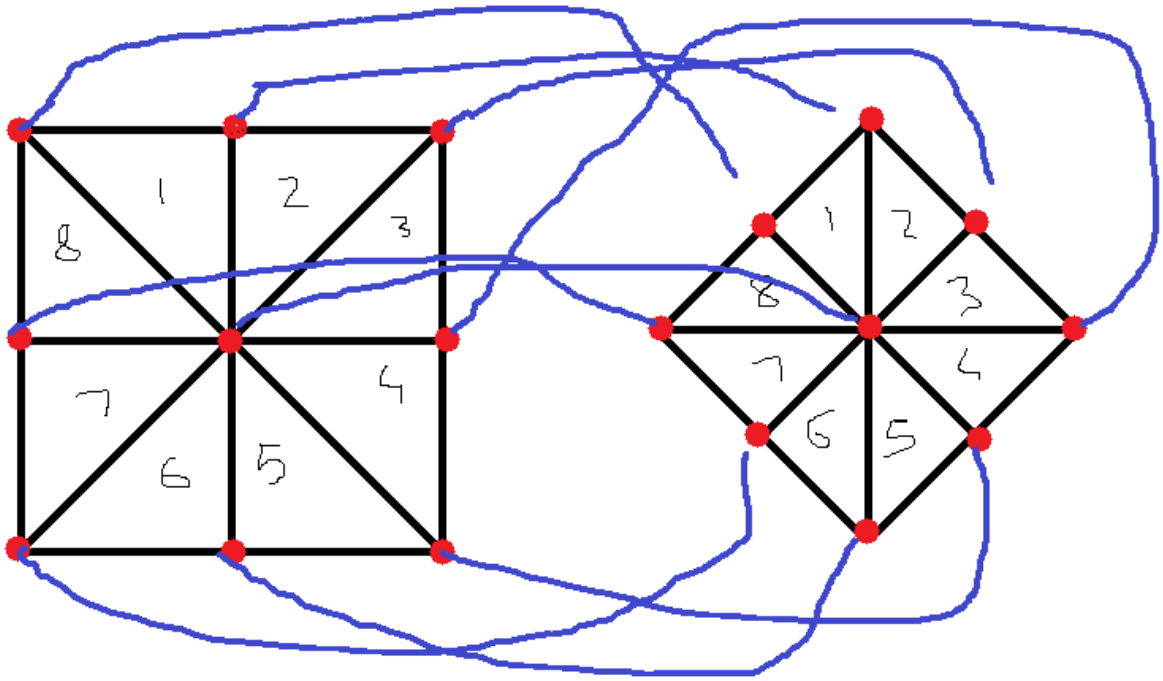
Spatial warping concerns with the modification of the shape and structure of the Image to the desired shape and structure of the output image. Spatial Warping is an important concept and can be applied to many application where there is a demand for the change in the shape and structure of the perceived image. For example in games and many simulation applications, suppose there is a glass or any shape changing object, how would the image through it look when viewed by the user is important and we can modify the perceived shape of the image which the user is viewing. Like this there are plenty examples where spatial warping is needed. Another example is for exceptionally different art images or posters where the desired image needs to be warped into a shape of square or a circle etc.

Approaches and Procedure and results:

P1.A Warping to Diamond shape

In this part of the problem we are presented with a image which is by default in square shape(500x 500). We need to warp this image into a shape of diamond as shown by the example. One simple approach is to break down the image into small triangles and then warp these triangles into their respective shape using a linear warping polynomial. For these we need to have a set of control points which will define how the image points will be warped, because they will follow the motion of control points.

So one possible way to warp square image into diamond image is shown as follows.



Warping of square to diamond based on control points

Thus we can then warp a triangle into another triangle as we know the three co-ordinate points of each triangle. This can be done as follows.

U = {Set of output Triangle x - co-ordinate} X = {Set of input Triangle x – co-ordinate}

V = {Set of output Triangle y - co-ordinate} Y = {Set of input Triangle y – co-ordinate}

We can then form the below relationship between these co-ordinate points

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} a0 & a1 & a2 \\ b0 & b1 & b2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

Where matrix elements [a] and[b] can be found as follows using the control points which define that region.

Let Matrix A be as follows

$$\begin{bmatrix} 1 & X1 & Y1 \\ 1 & X2 & Y2 \\ 1 & X3 & Y3 \end{bmatrix}$$

Matrix A

Where these are the control points(X1,Y1 etc)

So to get matrix [a] and [b] we have

$a = A^{-1}U$ and $b = A^{-1}V$ Where U and V are the output image control pts

However, this relationship will give us mapping from input point to outpoint using the Matrix(a,b). To get mapping from output to input we need to reverse the control points that is substitute X for U and U for X and Y for V and V for Y. This will give us the mapping from output to input.

THAT IS WE KNOW THE OUTPUT IMAGE POINTS, WE JUST NEED ITS RESPECTIVE INPUT IMAGE POINT TO GET THE PIXEL INFORMATION.

In the transformation image shown at the top, lets see how the first triangle is warped. This is done in matlab.

The range of pixels that is the width and height is normalized (500,500) to (1,1) Thus we will then see that the point (0.25, 0.25) in the output image will correspond to point (0,0) in the input image. And thus we have our reverse mapping for triangle 1 which can be applied to all the triangles.

EE569_HW3_P1.m

```
1   %Triangle 1
2   U = [0.25 0.00 0.50];
3   V = [0.25 0.50 0.50];

4

5   X = [0.00 0.00 0.50];
6   Y = [0.00 0.50 0.50];

7

8   A = [1 U(1) V(1);1 U(2) V(2);1 U(3) V(3)];

9

10  A_inv = inv(A);

11

12  c = A_inv*X';
13  d = A_inv*Y';

14

15  a = A*U';
16  b = A*V';
17
18  tmat1 = [c';d'];
19  tmat2 = [a';b'];
20  disp('Triangle_1');
21  disp(tmat1);
22  disp(tmat2);
```

Output Image Control Points

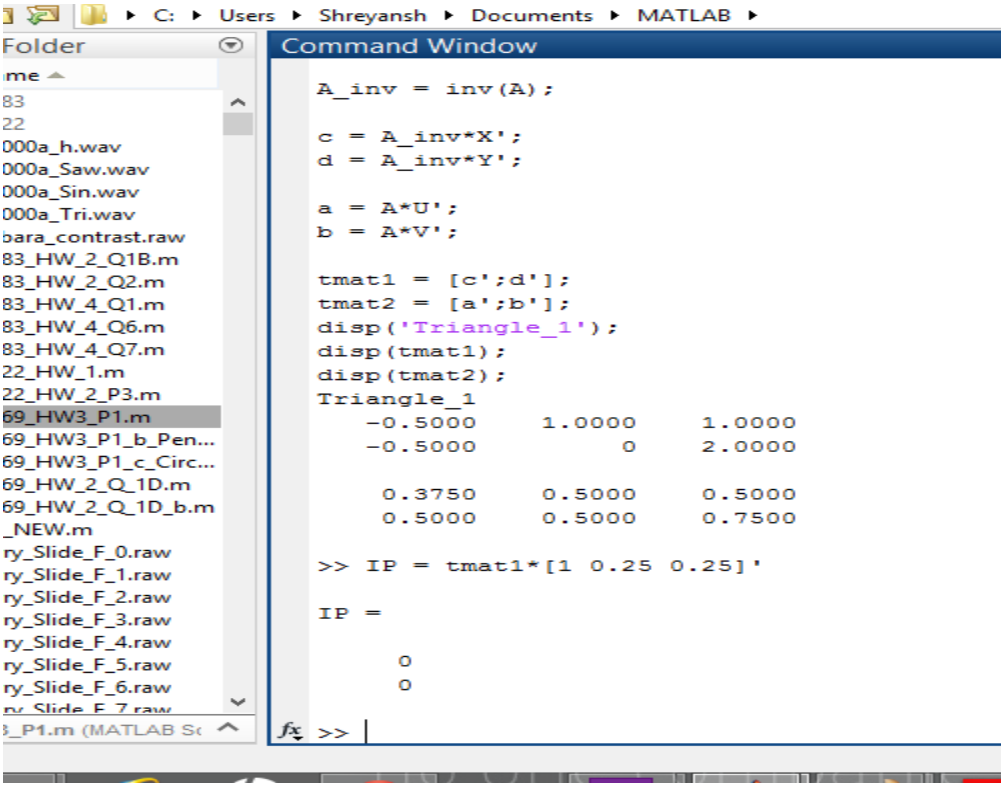
Input Image Control Points

A transformation matrix

This will give mapping from output to input

tmat1 will give reverse mapping

Generating Transformation Matrix for triangle 1

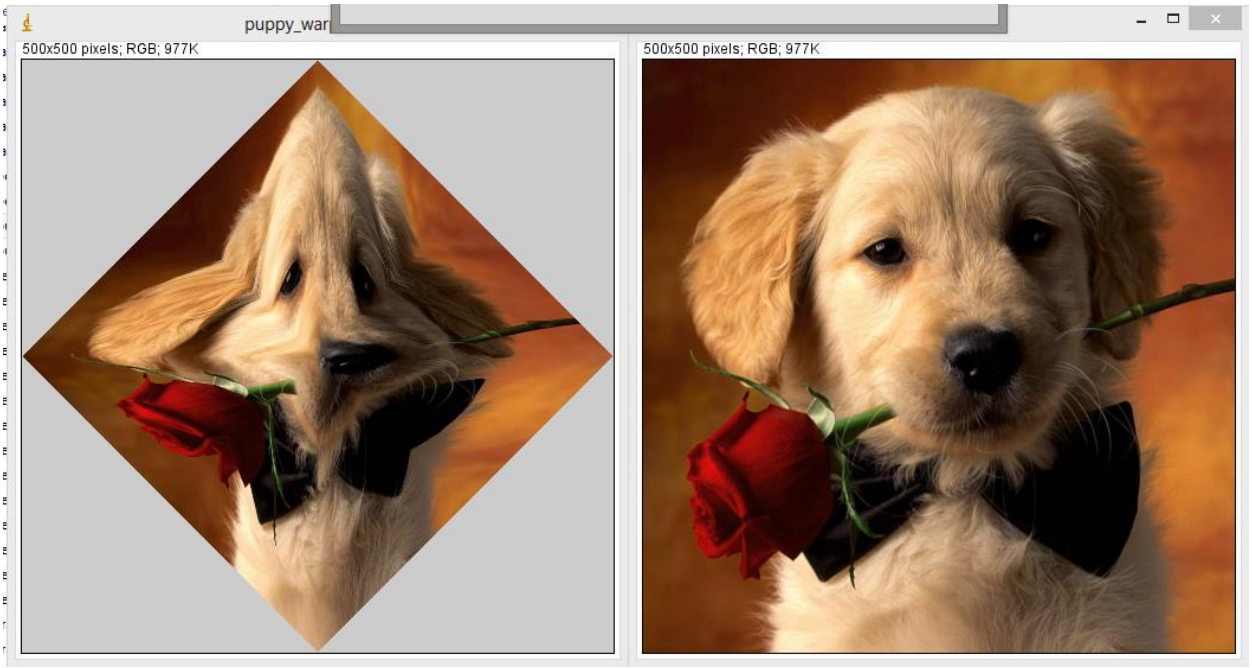


Thus we see that if we give input as the point (0.25, 0.25) of the output image, we get (0,0) point of the input image

Thus the algorithm will be as follows

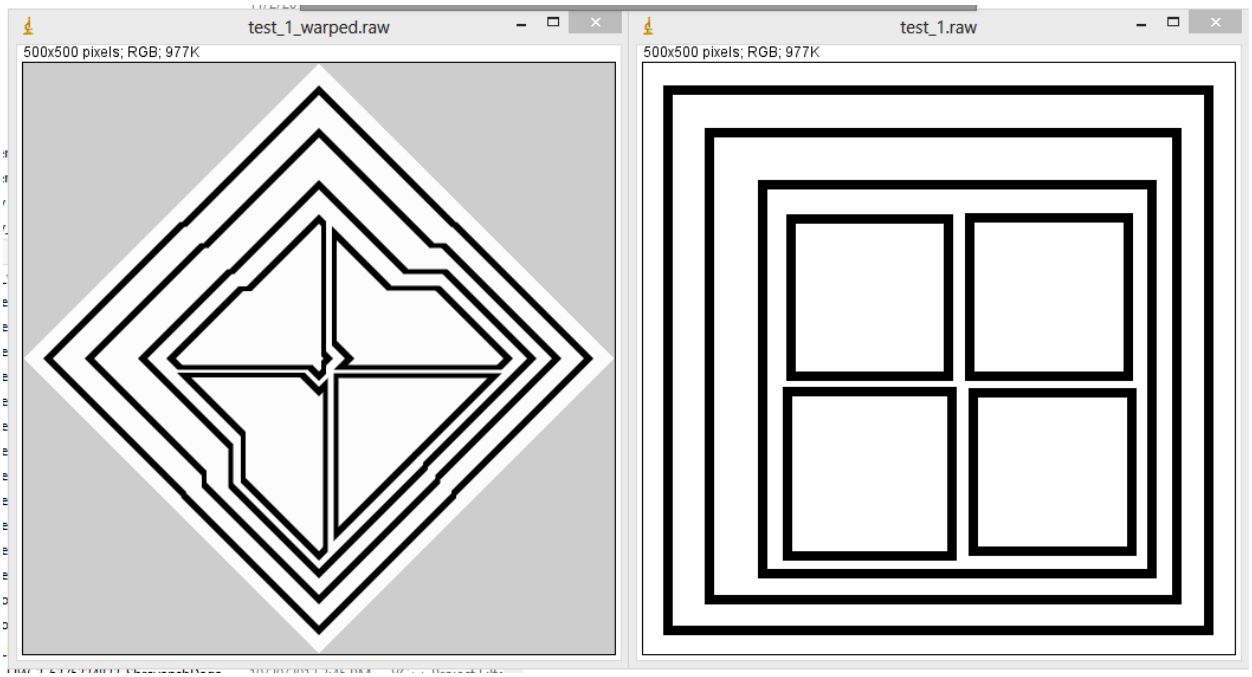
1) Select the control points U, V and X, Y
2) Generate A matrix from U, V for reverse mapping and from X, Y for forward mapping.
3) Calculate A^{-1} using matlab
4) Find transform matrix [a] and [b] using A^{-1} and X, Y
5) Using transform matrix map the ouput points to the input points
    a. For this First find in the output image whether the Point under consideration is within the triangle or not.
    b. This is done by using the condition of whether the point lies within a triangle defined by the three control points.
    c. If the point lies within and on the boundary of the triangle, proceed further, else ignore
6) Get the intensity/Pixel value from this obtained input point and give it to the ouput point under consideration.
7) Repeat this for all the triangles and all the points in the output image.
8) If the point does not lie within any of the 8 triangles, set its value to black.

Using this algorithm we get the following results for Problem 1A



Left Warped Image Right Original Image

I also tested this algorithm on another test image as shown below.



Left Warped test image Right Original Test Image.

From this we see that there is some distortion on the edges where the triangle are joined with each other, more prominently on the +-45^{o} and +- 135^{o} axis lines.

But on the whole, this method fairly considers all the required points mentioned.

Reverse Mapping

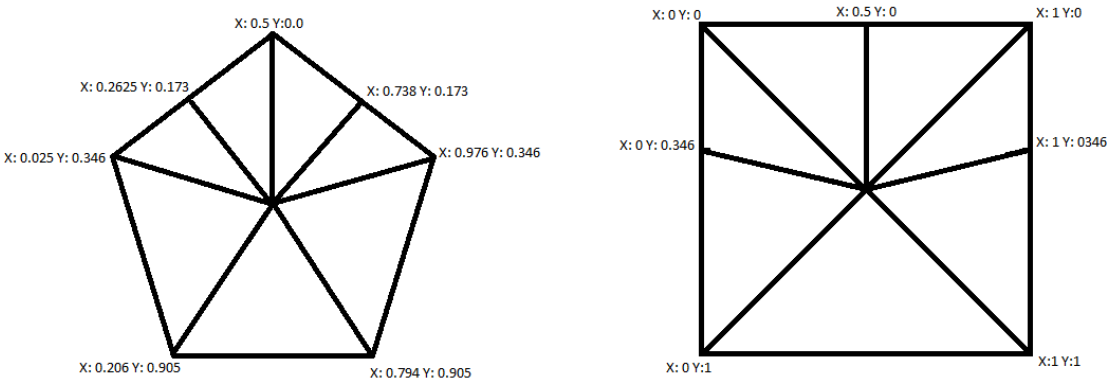
Using same procedure but with interchanged values of U,V with X, Y we get the reverse mapping from output image to input image as follows.

P1. B Warping to Pentagon shape:

In this part of the problem we are given with another image, and we need to warp it to a pentagon shaped image.

For this again we divide and break down the input image into small triangles, and warp these triangles to their respective triangles in the output image.

Since this is not a strict symmetric shape, the selection of control points is very important. We select the control points and the triangles as follows



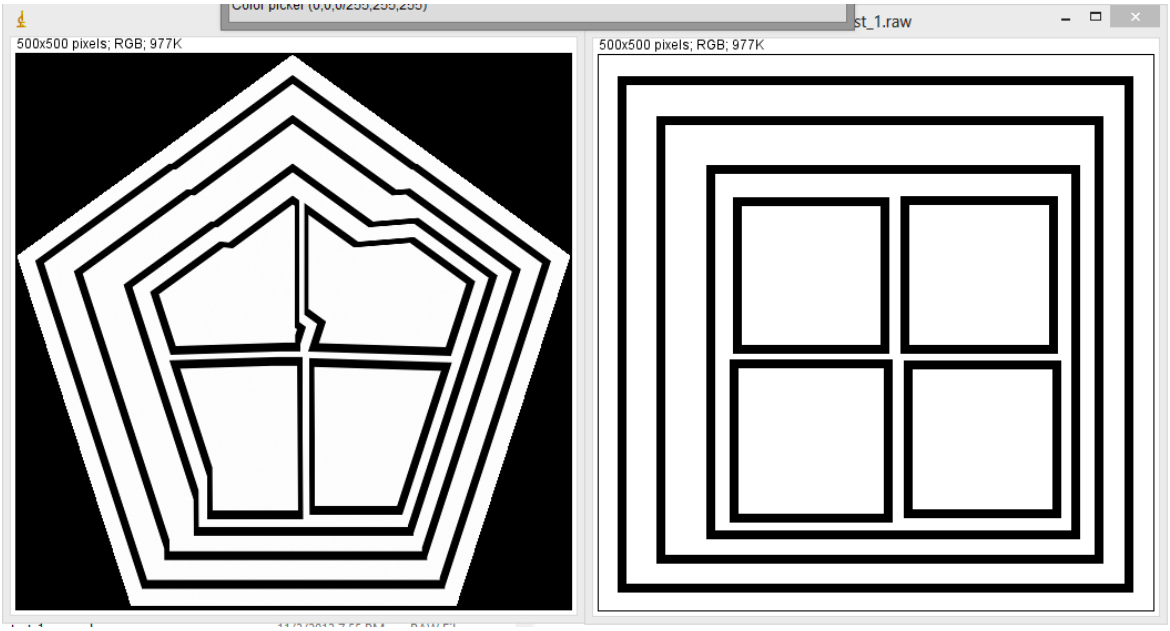
Left output image control points, right input image control points.

We begin, by considering the angles of the regular pentagon, in this case the angles of the pentagon come out to be 108^{o} , using this and some simple geometry we get the control point co-ordinates for the pentagon as shown, now since this is a regular pentagon, it will be higher than the bottom sidea bit.

One more thing to be noted is that the, upper two squares were further divided into 4 triangles and the lower three remain same as shown and there is no need for further division. Thus again using the same method of triangle mapping as used in problem 1.A we proceed further to get the following image warping for the cowboy image.

Left warped image, right original image

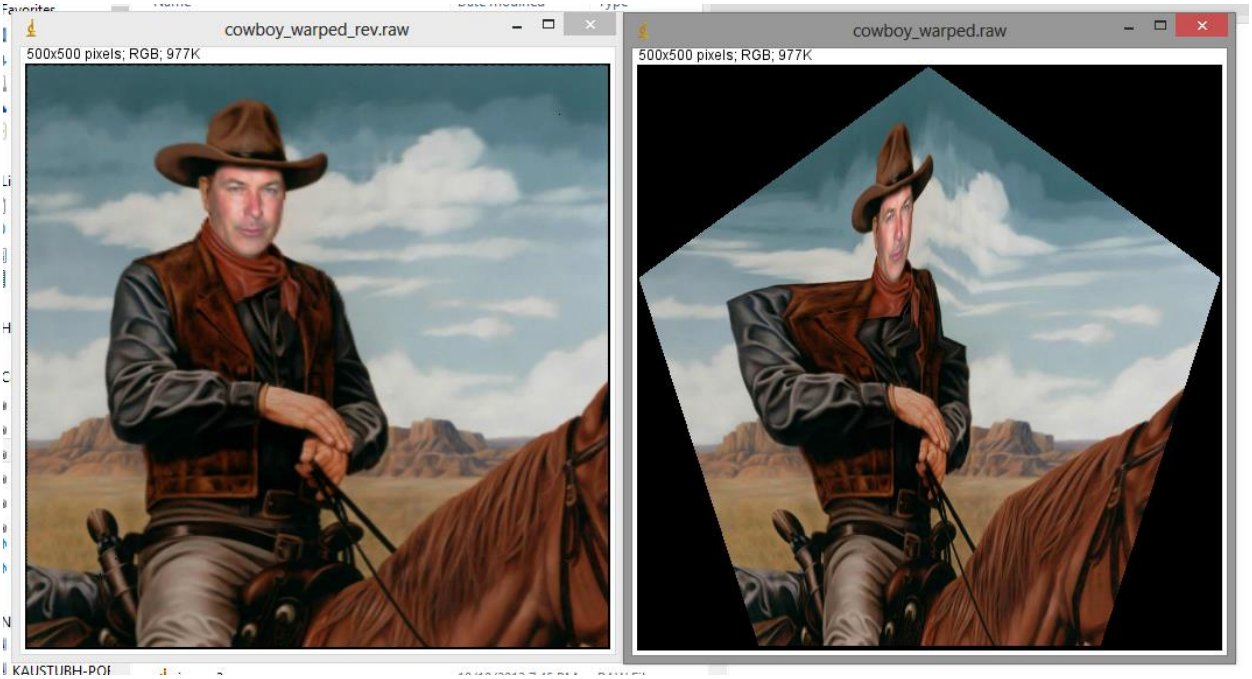
The same is applied to the test image and we get the following results to see how the algorithm works



Left warped image, right original image

Reverse Mapping

When we implement reverse mapping, that is we try to obtain original image from warped image, we get the following result.

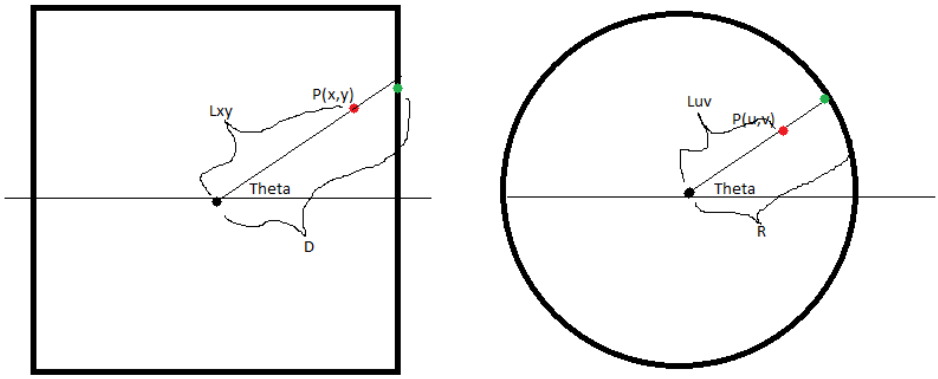


P1. C Warping to Circle shape:

This part of the problem requires us to warp the square image into a circle image as shown in the example. Now the circle shape is not linear like pentagon and diamond shapes. This will require us to either use a second order polynomial equation or some other method to warp the image.

One simple method approach is shown below, while scanning the input image from pixel to pixel, we calculate its distance from the center and find the angle it makes with the axis points as shown, next we extend its vector to see where it meets the square boundary. This will give us the angle where the ouput pixel should be put and also the scaling factor with which we have to scale all the points in that vector to warp it to the circle.

This can be illustrated using the following diagram.

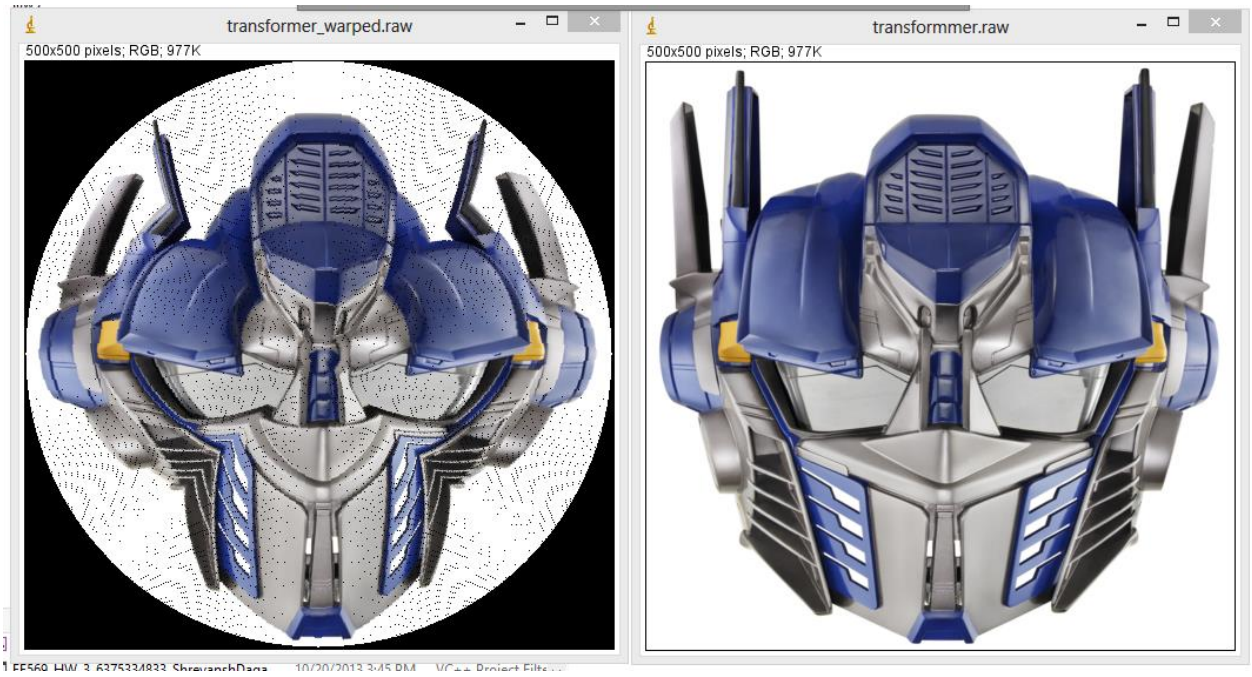


Relationship between points from square to circle

Now as shown we have Lxy, D, Theta and R = 0.5

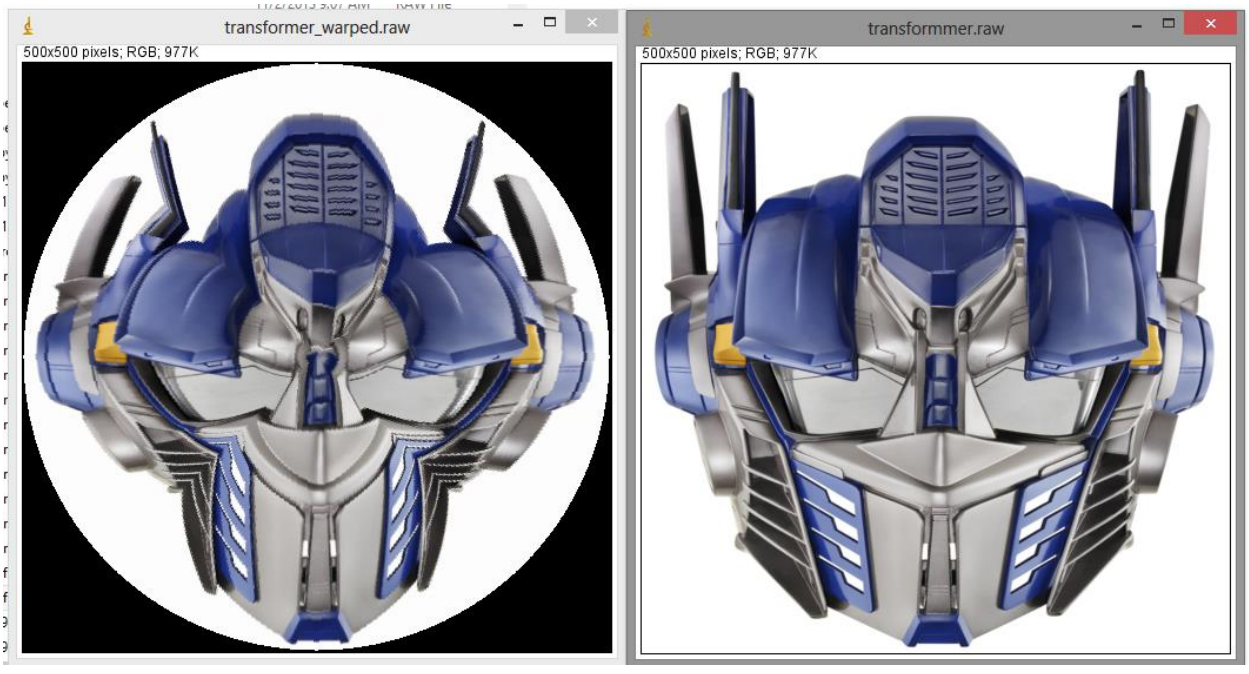
Using these we have to find Luv to find out where to put the input pixel to the output pixel. D/R will be the scaling factor which will decide Luv from Lxy as shown, now R is fixed and D will vary since the image is a square image Hence we need to compute D for every pixel.

We get the following warping when applying this method.



Left warped image, right original image

However, if use the reverse mapping from output to input using the same method but instead we scan the output image using polar co-ordinates, we get a cleaner and nice image as shown below



Left warped image, right original image

Thus we see that, using reverse mapping we get a cleaner image than forward mapping, This is due to the fact that we are using interpolation to get the pixel information for those co-ordinates which are fractional and not discrete

Discussion

After the implementation of all three shape warpings, we find that the most clean warping is obtained by the reverse mapping, since we have fixed the output points(which are discrete) and we obtain its respective input point which can be fractional but we obtain the pixel information using interpolation.

Also from mapping back from warped image to original image, as wel have already lost pixel information in the warping, we loose further pixel information and we do not get the expected original image but a bit blured image which is due to the fact that interpolation is applied again to reconstruct the original image from the warped image.

Warping is useful for many applications and we can use this method for various applications which demand exceptional special effects on the image. Warping in different shapes is also usefull in various applications which then present a physical poster/ print image of the desired shape as well.

References:

I was unable to configure OpenCV in time hence I have used the following code which will simply give me inverse of matrix in C++

I have used the Matrix Inversion code from the following source

Matrix Inversion Code Taken from
http://chi3x10.wordpress.com/2008/05/28/calculate-matrix-inversion-in-c/
Author: Yu-Tseh Jason Chi
There are 3 functions
MatrixInversion()
GetDeterminant()
GetMinor()

Problem: 3 Texture Analysis and Segmentation

Motivation:

Texture analysis and segmentation deals with identification and analysis of textures of different types. IT is majorly used by different geographical and geological mapping products to identify different textures like, grass, dessert, water, minerals, cities etc. Also Texture analysis and segmentation is used in Computer Vision which employes machine learning so that robots/ computers are able to distinguish between different textures just like humans can. The analysis and processing algorithms have to meet the high requirement of performance for it to work as expected. It is expected that in the next 5 to years we can develop powerfull algorithms which will give robots/computers the power similar to the humans to distinguish between various textures.

Approaches and Procedure and results:

P1.A Texture Image Clustering.

In this part of the problem, we are presented with 12 textures which when seen by human eyes can be easily classified into 4 types namely grass, treebark, sand, water. In this part we use the Laws filter for the texture analysis. Here we use the given kernals for Level, Edge, Spot, Wave, Ripple to generate the filters which will extract the feature set from the image. Since an image is a 2D variable, we also need a 2D filter which can be obtained my tensor product of the given kernals

Thus we have the fllowing 9 law filters which are made using the 3x3 combination of Level, Edge and Spot Kernals.

Laws Filter in 2D are computed as follows

$$L5(*)L5^T$$
$$L5(*)E5^T$$
$$L5(*)S5^T$$

(*) = Tensor Product

3 Sets for L5 filter

Like wise we can also compute 3 sets of laws filters for S5 and E5 as well.

Next we extract features for all the 12 images using the 9 laws filters, this will give us 9 feature images per texture image for all the 12 texture images.

So we have a total of

12x9 = 108 Feature Images

Next we perform feature averaging, that is we represent the feature set of all the pixels in a feature image by an average value which represents the feature value of that feature image.

Averaging can be done using 2 methods

1) Normal Average
2) RMS Average

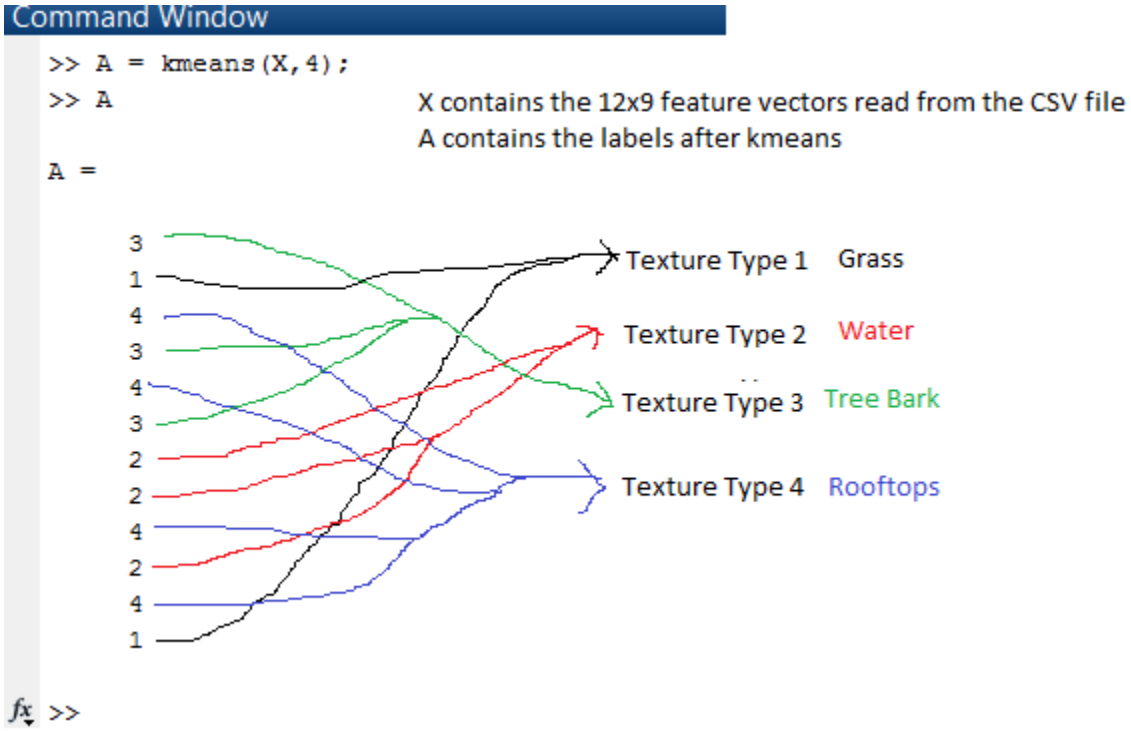
Since there is a chance that feature of a pixel in any feature image can have positive and negative values, we best use the RMS averaging method for the averaging of feature image to represent using a single feature value.

Next we Perform the clustering using k-means method provided, the k-means clustering will cluster the 9 features of the 12 texture images into sets of 4(k = 4) since there are 4 types of textures and each type has 3 different images of same texture.

The k-means method provided does not give the satisfactory clustering of the textures, for both RMS averaging and Normal averaging. Hence I used matlab function kmans().

For this my C++ program writes a csv file which has the 12x9 feature values and then I read this CSV file into matlab in a variable and then perform k means on that variable with k = 4.

The following results were obtained in matlab.



Result obtained after clustering using k means() in matlab

However due to texture9.raw being a blurred grass texture, it was identified as texture type rooftops. Hence in the result we have only 2 textures being identified as Grass while 4 textures are identified as rooftops.

P3.B Texture Segmentation

In this part of the problem, we are presented with an image which by visual inspection has a combination of 6 clusters. We have to perform segmentation and identify the 6 textures as different regions in the output image.

Here we begin with assuming that, each pixel is different texture. This will give us 450x600 textures to cluster. Next using same steps, we create 25 5x5 Laws filters for the feature set extraction of each pixel.

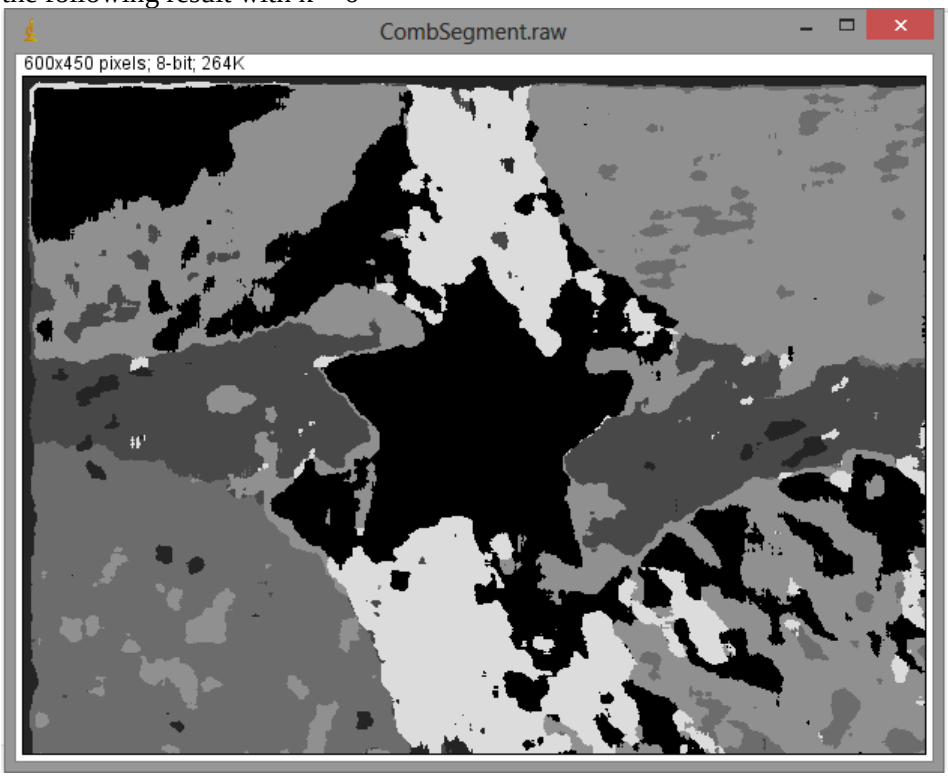
After this step we average the value of pixel using a window of 15, this ensures a proper averaging of the feature value of the pixel. This can be tried using 11x11 and 15x15 window as well.

Next we normalize using the value of the first feature set(the one obtained by $L5(*)L5^T$), and then we proceed to clustering using k means with k = 6 and 7.

With k = 6/7 we obtain the following segmentation with each label value*36 henc we have

1) Label 0 => Pixel Value 0*36 = 0
2) Label 1 => Pixel Value 1*36 = 36
3) Label 2 => Pixel Value 2*36 = 72
4) Label 3 => Pixel Value 3*36 = 108
5) Label 4 => Pixel Value 4*36 = 144
6) Label 5 => Pixel Value 5*36 = 180
7) Label 6 => Pixel Value 6*36 = 216

Thus we get the following result with k = 6



Segmentation with k = 6

With k = 7 we have the following result

Discussion

Thus we see how the texture analysis and segmentation works, texture segmentation primarily performs analysis over the feature set. This practice of extracting feature set from the given data set is widely used in many aspects of statistical processing and descion making.

In problem 1 we were given with different types of textures and we had to cluster them together. For clustering using k means we need to have a maximum variance between the centroids of the given cluster data set (feature set) , then the k means can easily classify the different textures. For this we need to make sure our feature set extraction is sufficient and optimum and also less expensive computationally. As we see in problem 2 where we assumed each pixel to be a unique texture sample We saw how the increase in dataset can affect performance of this method.

Hence to accurately identify the texutres, we need very efficient features.

In problem 3 we can apply either of the methods which will reduce the required feature set in a unique way by not using redudndant information of the feature set. This will greatly optimize the feature set required for k means clustering.

Thus we have seen texture analysis and segmentation.
(Due to Midterm Preparation I was unable to implement Problem 3.C in time I will implement it as soon as possible and submit it as soon as possible)