EE 569 HOMEWORK 3

By

Morayo Ogunsina

USC ID : 7371213793

ogunsina@usc.edu

Issued 2/14/2021

Due 3/12/2021

(PARTIALLY COMPLETED HOMEWORK)

Note to Graders :

I found this homework more challenging than the previous and have applied myself to completing it to the best of my ability, given the submission deadline.

I plan to complete them later.

Table of Contents

Problem 1 :

Geometric Image Modification

I. Abstract and Motivation

In digital image processing, geometric transformations such as translations, rotations and scaling are used to manipulate the appearance of images. They also find applications in the removal of (geometric) distortions and can be used to construct new images that give new perspectives as seen in panoramas and mosaics and various special effects.

Basically, geometric transformations involve applying some mathematical procedure which modifies the pixel locations in different coordinates, and this involves mapping from one coordinate to another. For this problem, I applied the principle of mapping pixels from image coordinates to the cartesian coordinates to create a special effect called the disc shaped warping where an image is warped around a disc.

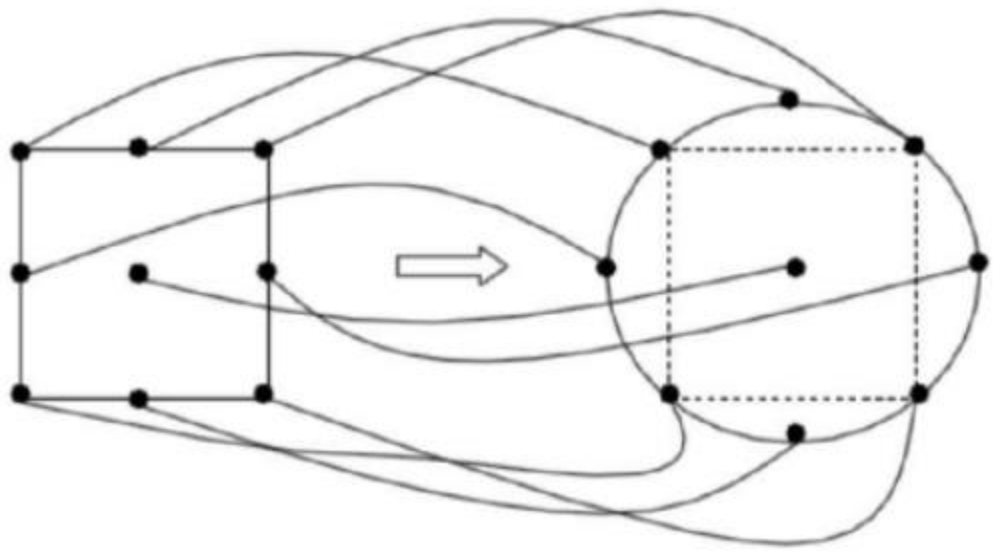


Figure 1 : The points on the squares have corresponding points mapped to the disc.

II. ***Approach and Procedures***

For the image, it must satisfy these requirements :

1. Boundary points on the square should be mapped to Boundary points on the circle/disc
2. The centers or midpoints should also match, so the points on the horizontal and vertical locations are kept the same while others are changed.
3. The mapping should be reversible, so it must be linear.

The following Coordinate System are used:

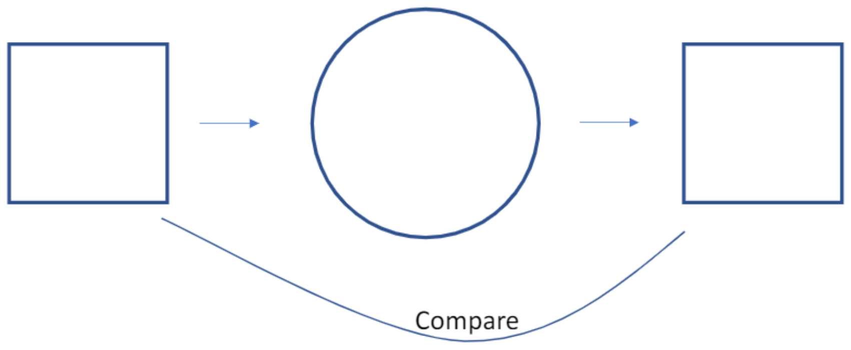
Original Image Image Pixel Coordinates – (***i,j***)

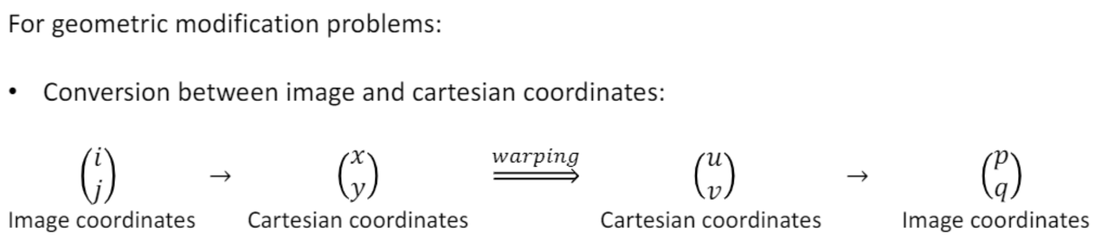
Original Image Cartesian Coordinates – (***x,y***)

Disc Image Cartesian Coordinates – (***u,v***)

Disc Image Image/Polar Coordinates – (***p,q***)

There are two main coordinates involved in achieving a disc-warped image and those are the Image coordinates which is represented by (i,j) and the cartesian coordinates, (x,y). In the conversion, the new coordinates translate to (u,v) and (p,q) as shown in the image below:





For geometric modification problems:

- Conversion between image and cartesian coordinates:

$$\underset{\text{Image coordinates}}{\begin{pmatrix} i \\ j \end{pmatrix}} \rightarrow \underset{\text{Cartesian coordinates}}{\begin{pmatrix} x \\ y \end{pmatrix}} \xRightarrow{warping} \underset{\text{Cartesian coordinates}}{\begin{pmatrix} u \\ v \end{pmatrix}} \rightarrow \underset{\text{Image coordinates}}{\begin{pmatrix} p \\ q \end{pmatrix}}$$

Figure 2 : Coordinate conversions for warping (from discussion notes)

Here, the pixel coordinates (***i,j***) are mapped to the cartesian coordinates (***x,y***) first, then mapped to the (***u,v***) coordinates corresponding to the warping process and finally mapped to the polar coordinates for the images, (***p,q***), giving a 4-step process.

To warp the square image into a disc-shaped image, I chose an ***Elliptical Grid Mapping*** method discussed, and implemented the given formula from the paper by **X et all** . The key formulas are given below:

Square to disc mapping:

$$u = x\sqrt{1-\frac{y^2}{2}} \qquad v = y\sqrt{1-\frac{x^2}{2}}$$

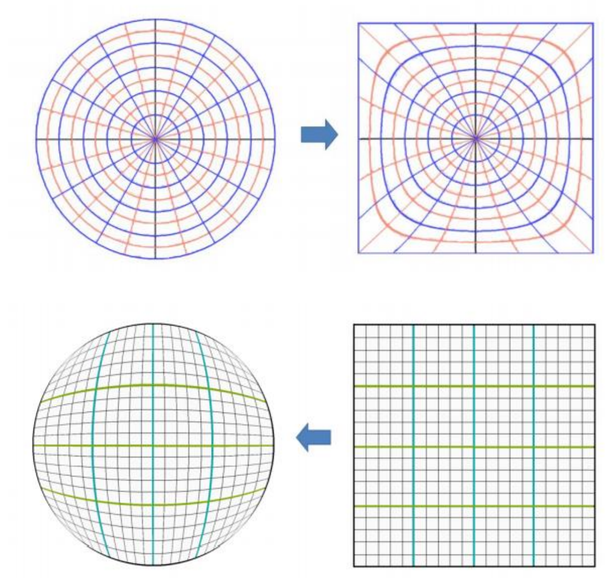
Figure 3: (u,v) coordinates for Square to Disc Mapping

And for reconstructing the original image from the warped image, the formula below was applied:

Disc to square mapping:

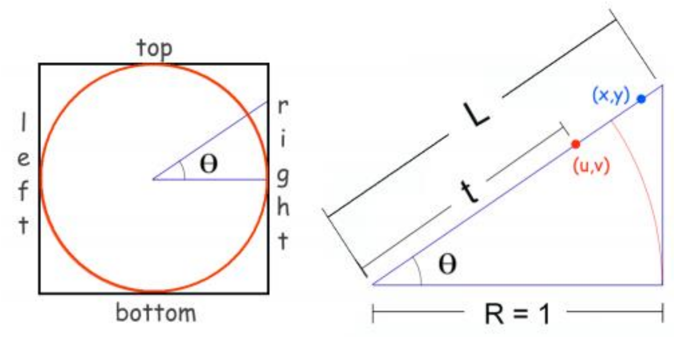
$$x = \frac{1}{2}\sqrt{2+u^2-v^2+2\sqrt{2}\,u} - \frac{1}{2}\sqrt{2+u^2-v^2-2\sqrt{2}\,u}$$
$$y = \frac{1}{2}\sqrt{2-u^2+v^2+2\sqrt{2}\,v} - \frac{1}{2}\sqrt{2-u^2+v^2-2\sqrt{2}\,v}$$

Figure 4: (***x,y***) coordinates for Disc to Square Mapping



Source : [1509.06344] Analytical Methods for Squaring the Disc (arxiv.org)

The idea is to map each point inside the disc to a corresponding point in the square-shaped image and we can use the radial property of our desired image to achieve this. This means that it starts from a point source and spreads radially outwards, mapping the angular coordinates (***u,v***) to the cartesian points, (***x,y***). This is illustrated below:



If θ is the angle between the point (u,v) and the x-axis, these equations must hold:

$$\cos\theta = \frac{u}{\sqrt{u^2+v^2}} = \frac{x}{\sqrt{x^2+y^2}}$$

$$\sin\theta = \frac{v}{\sqrt{u^2+v^2}} = \frac{y}{\sqrt{x^2+y^2}}$$

$$\tan\theta = \frac{v}{u} = \frac{y}{x}$$

Source : [1509.06344] Analytical Methods for Squaring the Disc (arxiv.org)

My approach to this implementation are explained below:

The input image coordinates are mapped to the cartesian coordinates first by subtracting the (image_height/2 - 1) and (image_width/2 - 1) from the image coordinates (***i,j***) respectively to get the x and y coordinates in the cartesian plane. These new values must be normalized by dividing by (image_height/2 - 1) and (image_width/2 – 1) respectively, to avoid negative coordinates.

```
long double x = (i - ((ImgHeight / 2) - 1)) / (long double)((ImgHeight / 2) - 1);
long double y = (j - ((ImgWidth / 2) - 1)) / (long double)((ImgWidth / 2) - 1);
```

Next, we make use of the formula in Figure 3 to get the (***u,v***) coordinates for our desired warped image, after which they are multiplied by (image_height/2 - 1) and (image_width/2 – 1) respectively and we add (image_height/2 - 1) and (image_width/2 – 1) respectively again. Like so :

```
unsigned int p = (u * ((ImgHeight / 2) - 1)) + ((ImgHeight / 2) - 1);
unsigned int q = (v * ((ImgWidth / 2) - 1)) + ((ImgWidth / 2) - 1);
```

These (***p,q***) coordinates are our polar coordinates in which we use to map the original (***i,j***) pixel coordinates to our desired disc-image.

For the reconstructed image, Here, we do conversion from the (***u,v***) image coordinate to the square image coordinates by using the equation in Figure 4. Next, these results xnew and ynew are converted to the final image pixel coordinates as shown below,

```
unsigned int l = round((xnew * (((ImgHeight / 2) - 1))) + ((ImgHeight / 2) - 1));
unsigned int m = round((ynew * (((ImgWidth / 2) - 1))) + ((ImgWidth / 2) - 1));
```

The previous (p,q) values of the warped image are then mapped to these new coordinates.

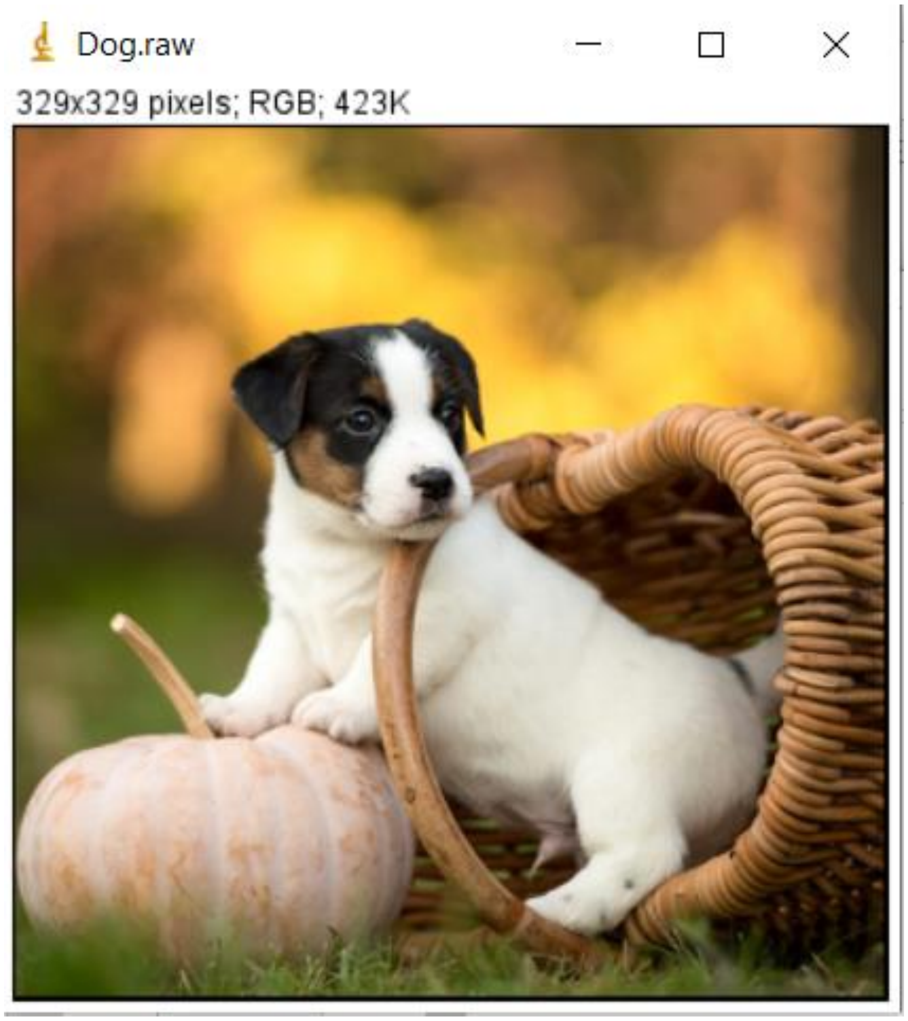
It is important to note that for the inversion, the inverse of Nowell's Elliptical Grid Mapping was derived [1].

The steps for **Square-to-Disc Warping** are summarized to :

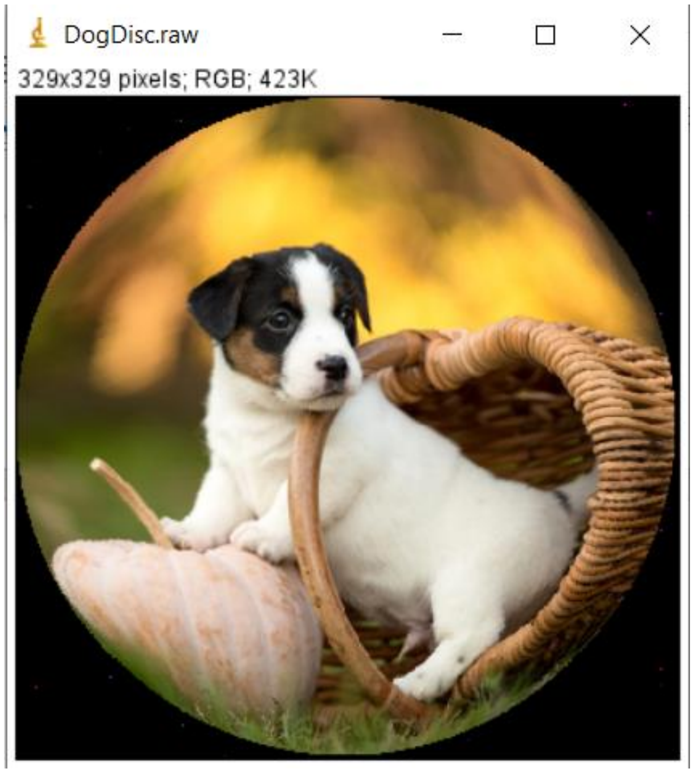
- STEP 1 : Run program (check README for instructions).
- STEP 2: Read image file into a 3D array, for easy manipulation
- STEP 3: Construct 2, 3D heap output for the Disc and Reconstructed images
- STEP 4: Loop through the image pixels and calculate the values for the coordinates (**x,y**), (**u,v**).
- STEP 5: Calculate the polar coordinates from these values as illustrated in Figure 2.
- STEP 5: Map the original image data to the disc heap array constructed using the (**p,q**) coordinate conversions.
- STEP 7: Do the same for the reconstructed image using the inversion coordinates denoted by (***l,m***), computed from the xnew and ynew disc coordinates. This is the inverse address mapping.
- STEP 8: Write outputs to file to view on image viewing software.

I. ***Experimental Results***

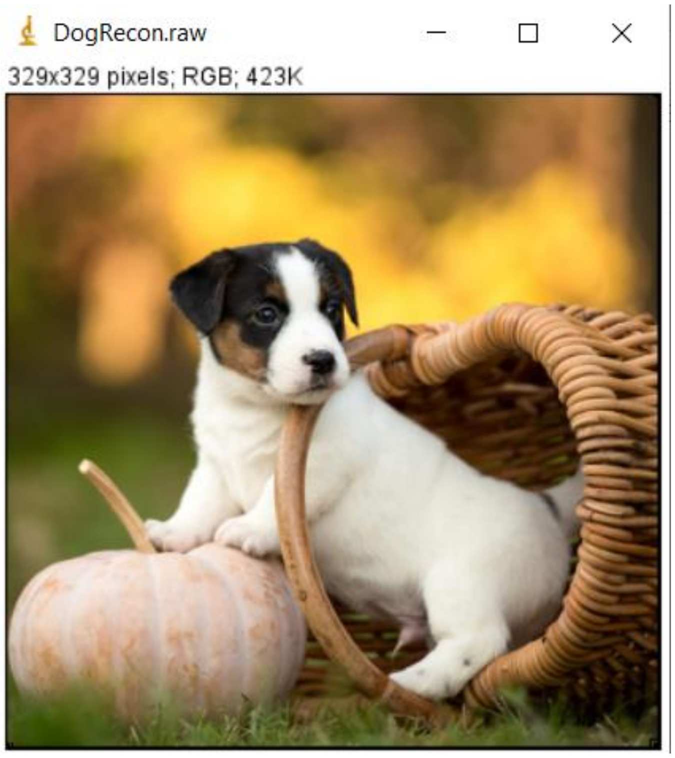
For Dog Image,



Original Dog Image

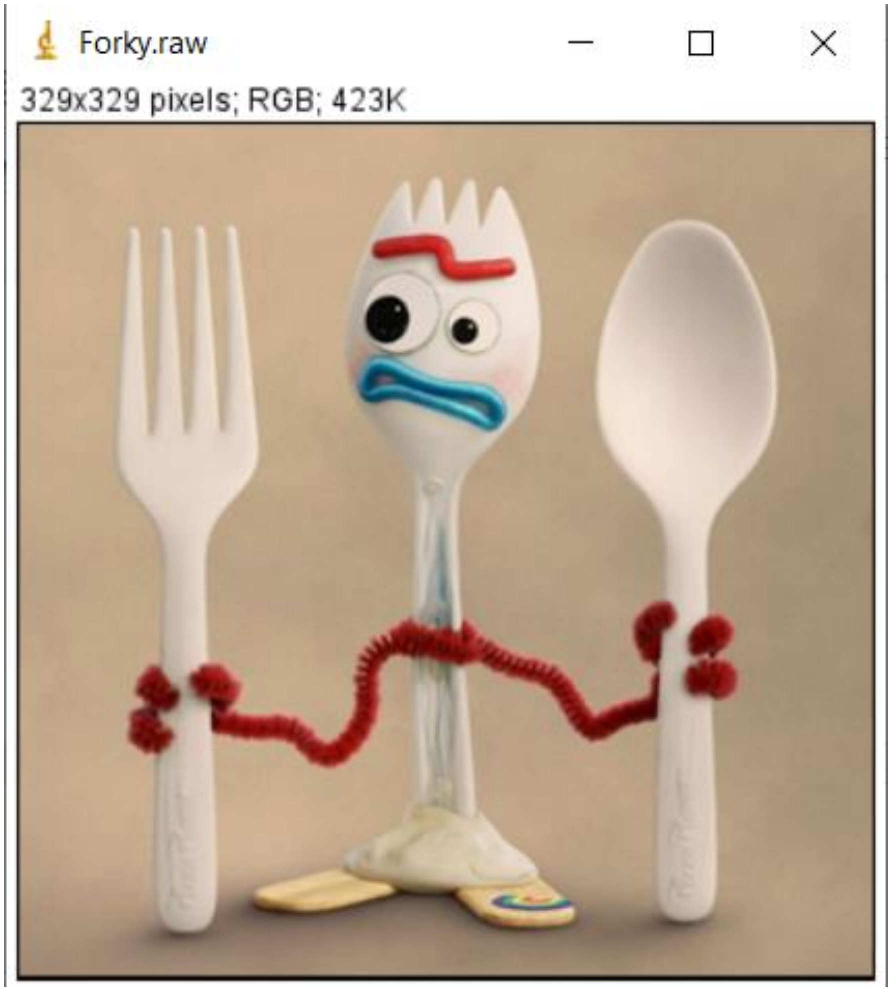


Disc-Warped Dog Image

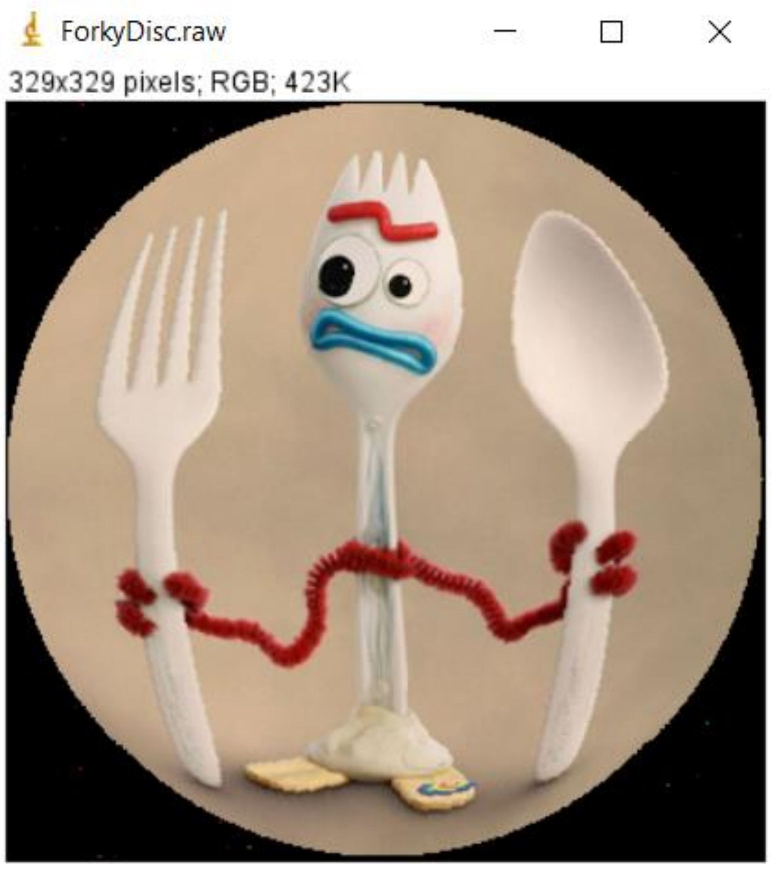


Reconstructed Dog Image

<u>For Forky Image,</u>

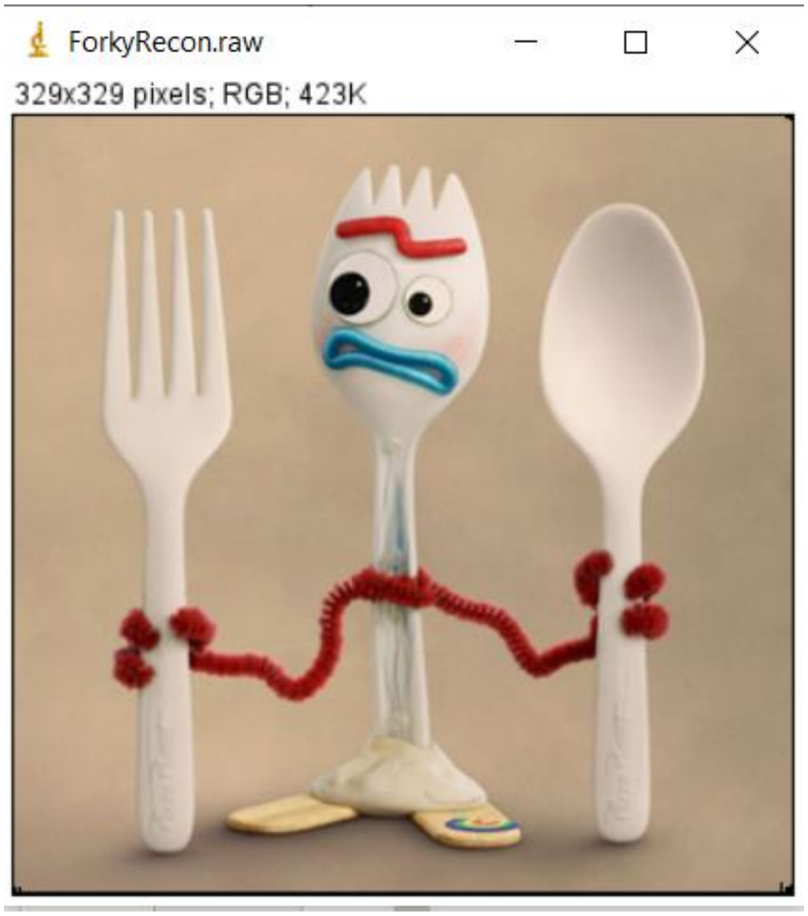


Original Forky Image


ForkyDisc.raw

329x329 pixels; RGB; 423K

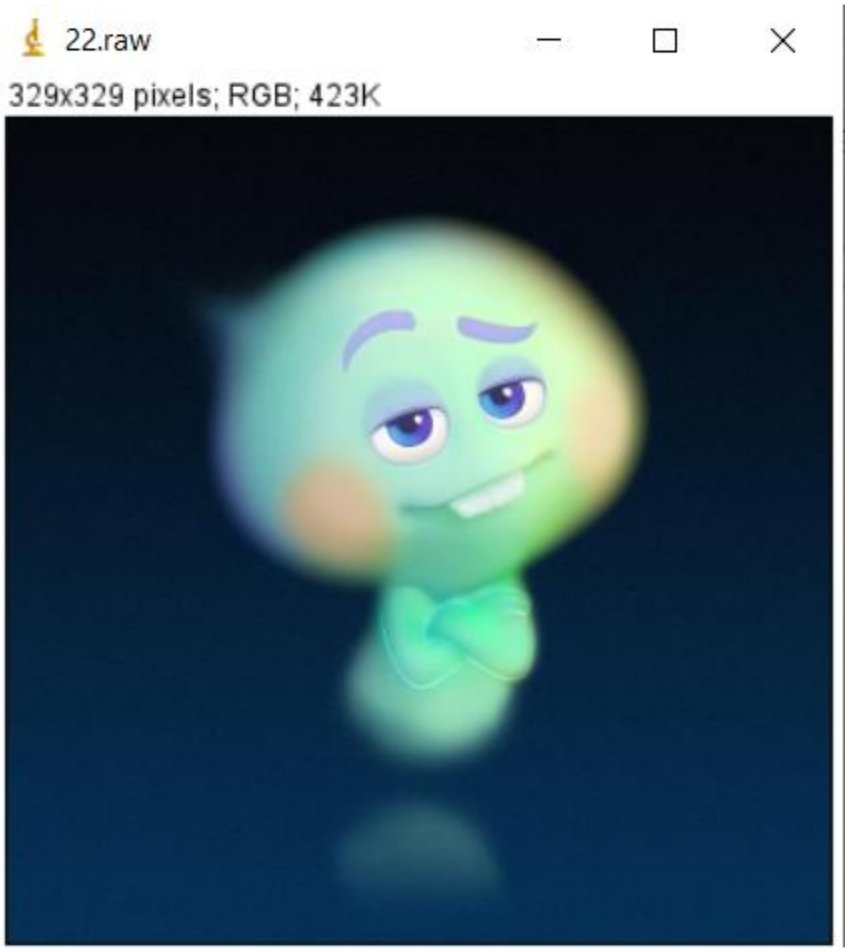
Disc-Warped Forky Image


ForkyRecon.raw

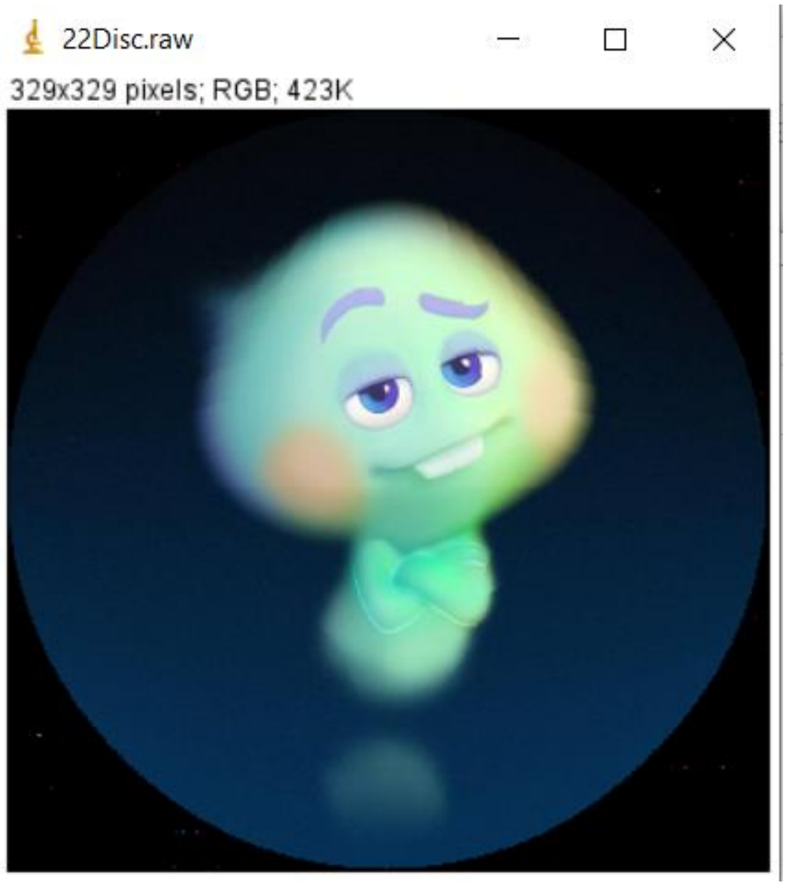
329x329 pixels; RGB; 423K

Reconstructed Forky Image

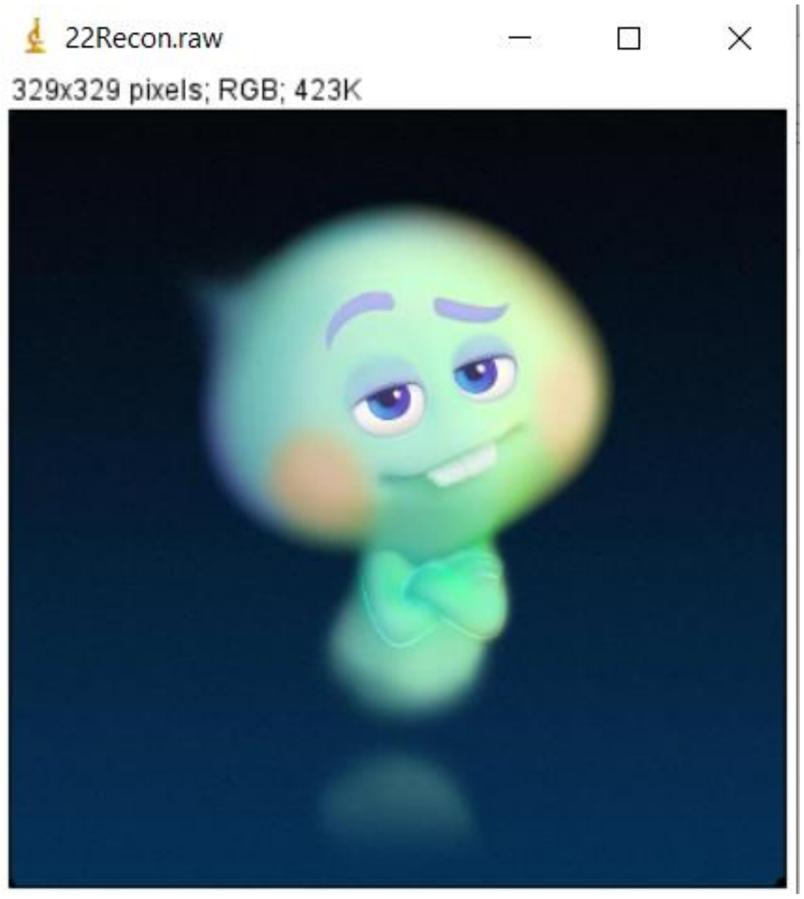
For 22 Image,



Original 22 Image



Disc-Warped 22 Image



Reconstructed 22 Image

II. <u>Discussion</u>

For this homework, I had some trouble figuring out a lot of the concepts and could give the full expected results. The only part I was able to get was warping the square image into a circle. For the distortion on the top left (as seen in the owl image), and the scaling of the original image to a bigger one, I think I need more time to figure that out.

From the experimental results, I observed that for the warped images, i.e disc-shaped images, the disc is completely warped into a circular shape and embedded within the original image size.

Also, there are some colored pixel spots on the black background images which become artifacts to the image. I think this might be because I did not use bi-linear interpolation, or it might be due to the absence of a control structure I did not include in my code which is to ensure if the coordinates are part of the desired warped image then it must be that (x2 + y2 should be greater than 1)? If I had more time, I could investigate this fix.

For the reconstructed image, it looks perfectly alright with no blurring or distortion and no observable artifact. So, I can conclude that the mapping is a linear one.

<u>Problem 2 : Homographic Image Transformation and Image Stitching</u>

I. *Abstract and Motivation*

Homographic transformation is used to merge or stitch different images that have similar key point features into one image with these similar features overlapping and dissimilar features added to provide a more robust image, as seen in panoramas.

As it is, these different images are taken from different camera viewpoints and using some statistical method or choosing good key-points manually, coinciding pixel locations in the images are used as control points to compute a homography transformation matrix. This matrix, applied through some algorithm, transforms each pixel in terms of scaling, translation, and rotation in such a way that the resulting stitched images form a sensible panorama.

I had the most difficulty with this problem mainly due to the results I arrived at in the computation of the Homography matrix, but I will explain my approach and steps.

II. *Approach and Procedures*

The most important aspect of this problem solution is getting the Homography matrix for the panorama transformation and image stitching.

First, we need to do a perspective projection by mapping points from the image coordinates to the world coordinates and they are related by the formula below:

$$w\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Figure 7: Transform matrix from world to image coordinates source : lecture notes

Images from the different camera viewpoints share some relation by the simple homography represented by the formula below:

$$P2 = HP1$$

H = 3x3 homography transformation matrix

P1, P2 are corresponding image points in the homogeneous coordinates before and after transformation respectively.

To be more specific, the formula can be written in the form below:

$$\begin{bmatrix} x'_2 \\ y'_2 \\ w'_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \ and \ \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x'_2}{w'_2} \\ \frac{y'_2}{w'_2} \end{bmatrix}$$

And transformed into a single-type matrix or H-matrix as part of the system of linear equations as:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -y_3X_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -y_4X_4 \\ 0 & 0 & 0 & x_1 & y_1 & 0 & -x_1Y_1 & -y_1Y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 0 & -x_2Y_2 & -y_2Y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 0 & -x_3Y_3 & -y_3Y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 0 & -x_4Y_4 & -y_4Y_4 \end{bmatrix} \times \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

Source : Lecture notes, homework handout

In this H-matrix, H33 is set as 1. This would ensure our other h values are non-zero.

More detail about the H values are given below:

H11 – fixed scale factor in the x direction, y is unchanged
H12 – scale factor in x direction relative to the y distance from the point of origin
H13 – translation from the x origin
H21 – scale factor in the y direction relative to the x distance from the point of origin
H22 - fixed scale factor in the y direction , x is unchanged
H23 – translation from the y origin
H31 – relative scale factors for x and y as a function of x
H32 – the relative scale factors for x and y as a function of y

The steps for **Image Stitching** are summarized to :

- STEP 1 : Run program (check README for instructions).

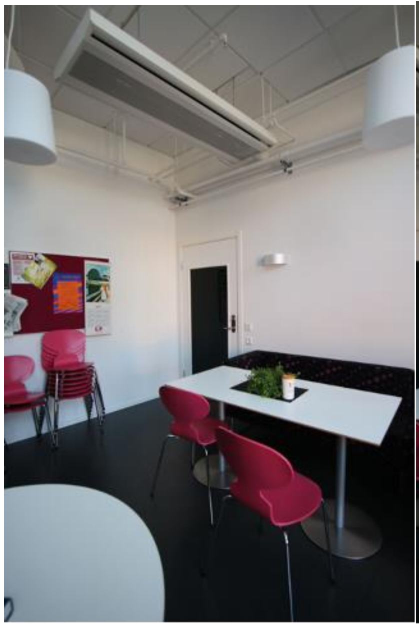
- STEP 2: Read image files i.e ***left.raw***, ***middle.raw*** and ***right.raw*** into a 3D array, for easy manipulation. Also, create a 3D output array for the final stitched image.

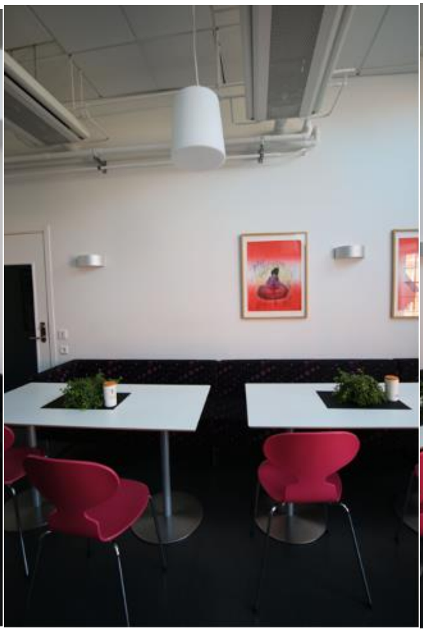
- STEP 3: Compute the H matrix for the left-middle and middle-right stitching.
    - The goal is to replace the Xs and Ys in the equation in Figure 8 with values and get all the corresponding values for H, with H33 being 1 already. (For my final value, I had to manually manipulate the values I got to make a panorama. I will discuss this later).
    - Get the 4 control points using MATLAB's cpselect function and use that to create 8 equations as represented in the equation in Figure 9.
    - The H matrix for both left-middle and middle-right is computed and these are convoluted with the pixel points.

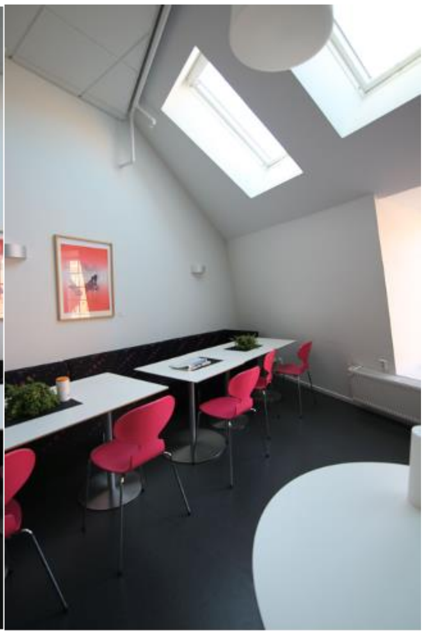
- STEP 4: Now, create a canvas big enough (2000 x 1500), then embed the middle image into it with column (800) and row (300) offsets.
- STEP 5: To stitch the left image transformed image to the middle image in the canvas:
    - first, multiply H Matrix for left and middle with the left image pixels, these are the homogeneous coordinates
    - convert these coordinates to cartesian coordinates as given below: `x = u * 1/W, y = v * 1/ W`
    - Calculate the intensity values for matching left and middle points and do bilinear interpolation
- STEP 5: Do the same for the middle and right images
- STEP 7: Embed the middle image again to create the final panorama image
- STEP 8: Convert the output array from 3D to 1D for writing to the image file.
- STEP 8: Write outputs to file to view on image viewing software.

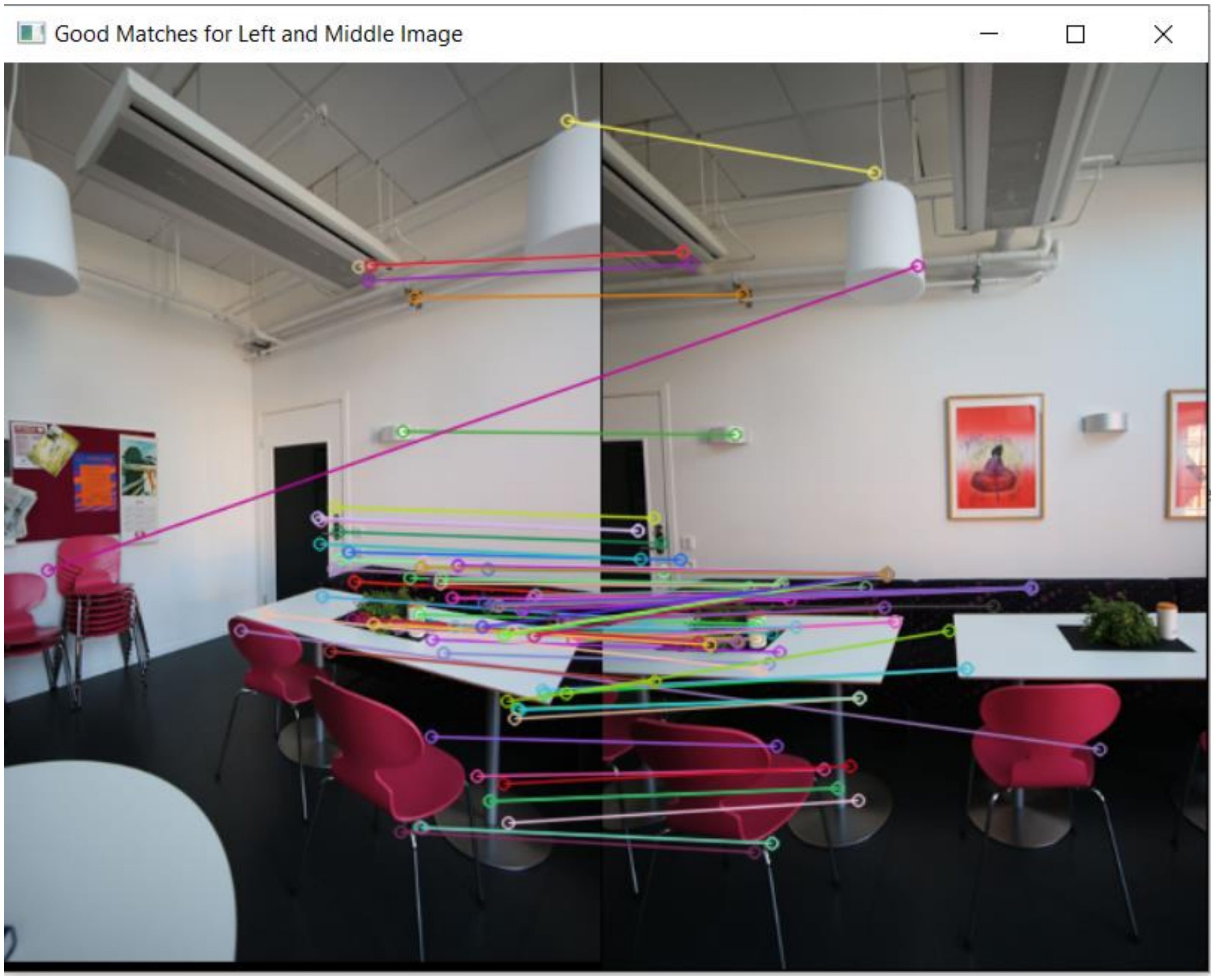
III. Experimental Results

The original images are shown below:

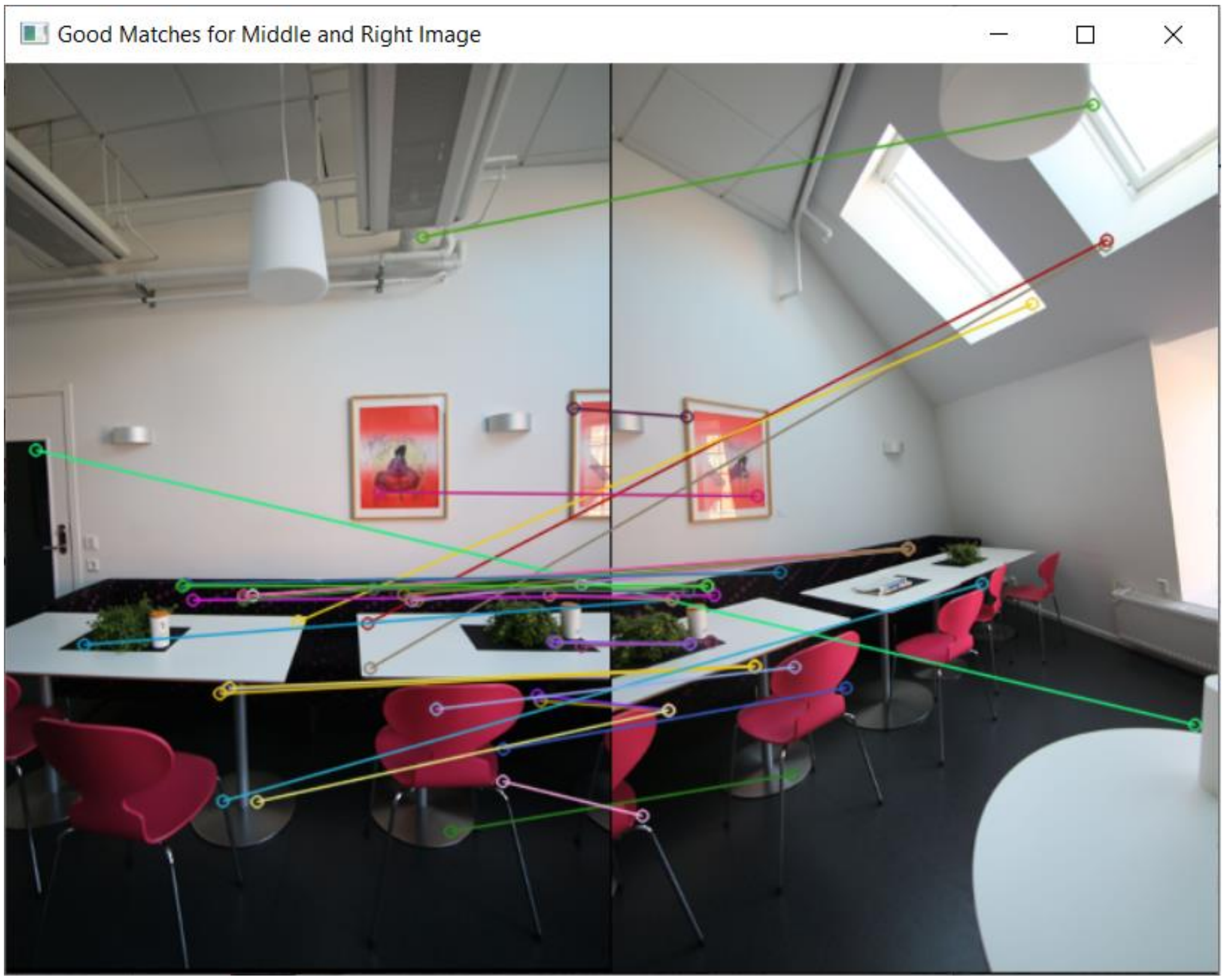

Left Image


Middle Image

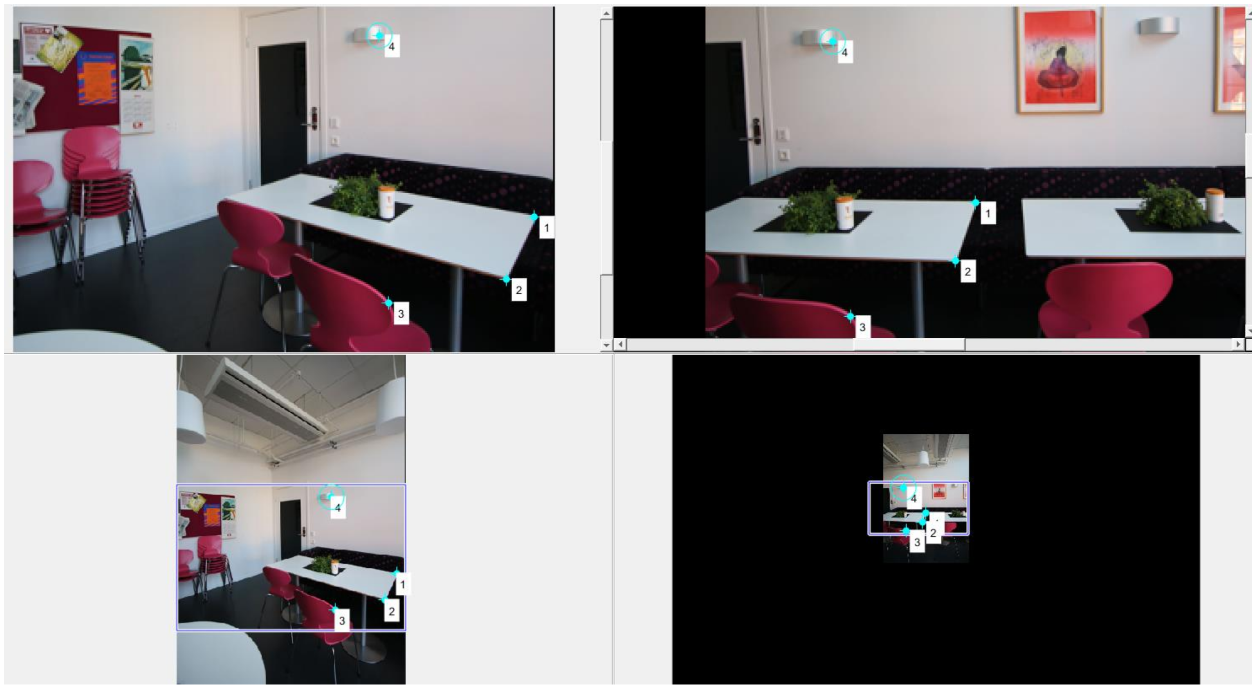

Right Image



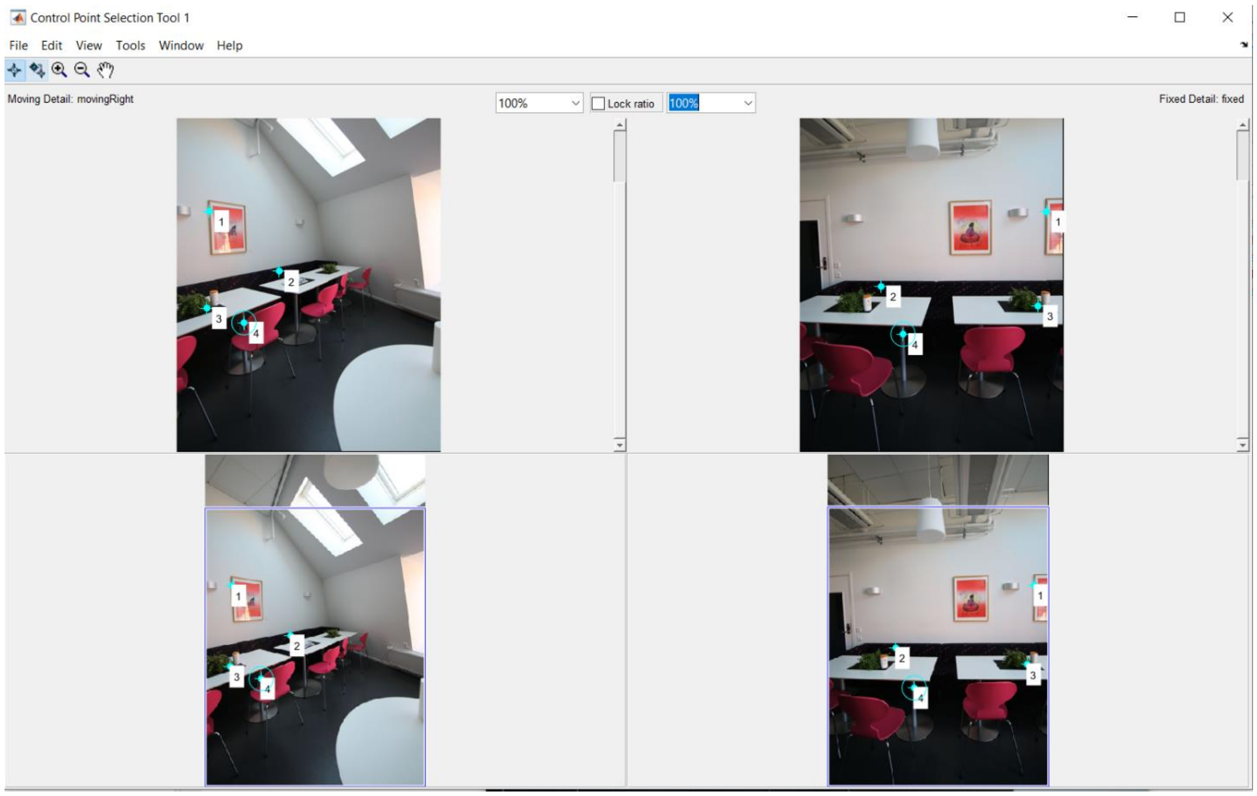
SURF Matched Features Between Left and Middle Images



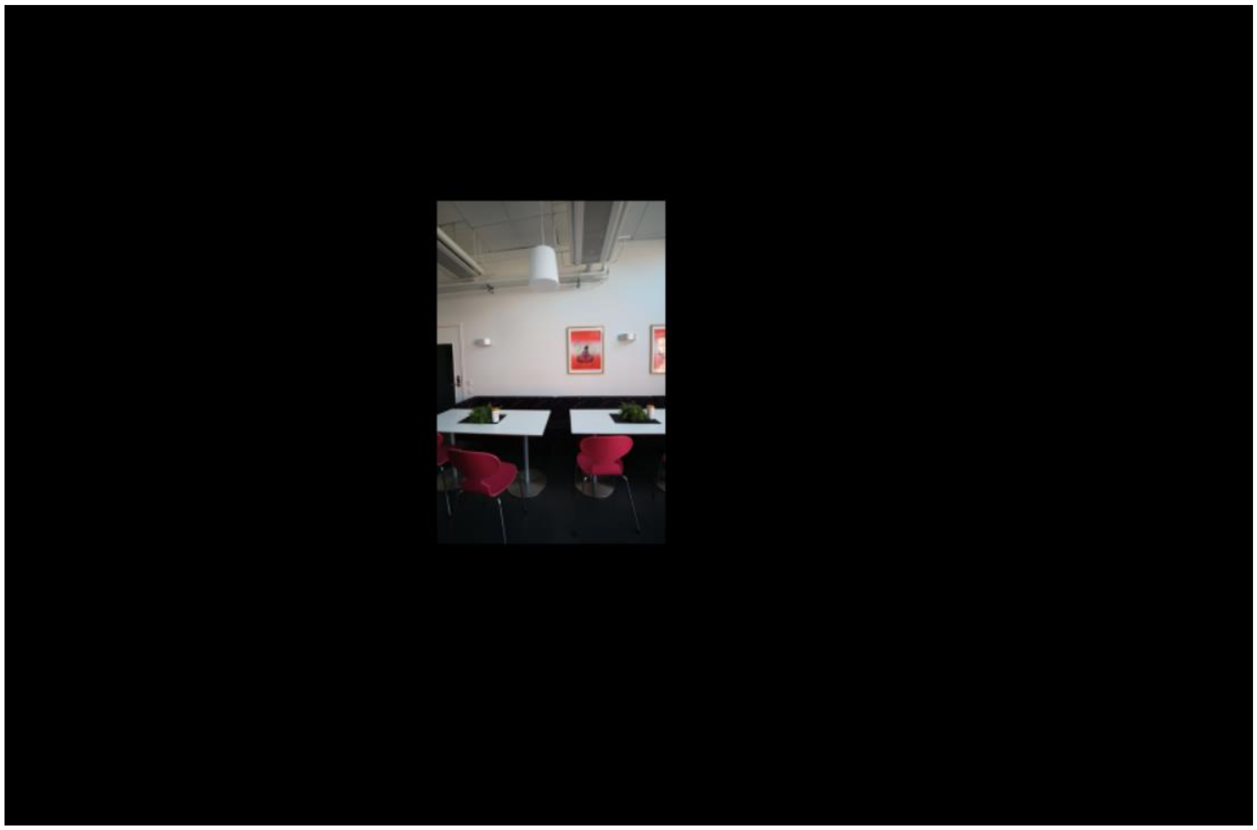
SURF Matched Features Between Middle and Right Images



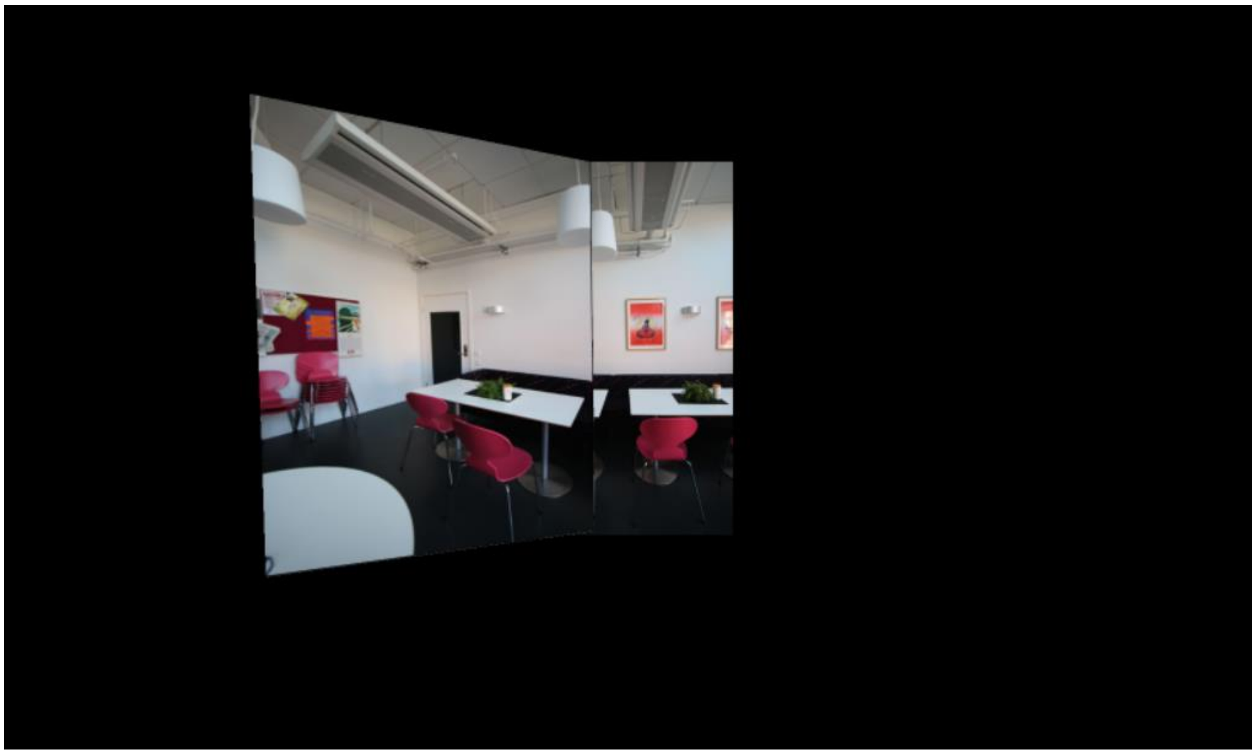
MATLAB cpselect tool to choose control points for Left and Middle


Control Point Selection Tool 1
File Edit View Tools Window Help
Moving Detail: movingRight
100%
Lock ratio
100%
Fixed Detail: fixed
1
2
3
4

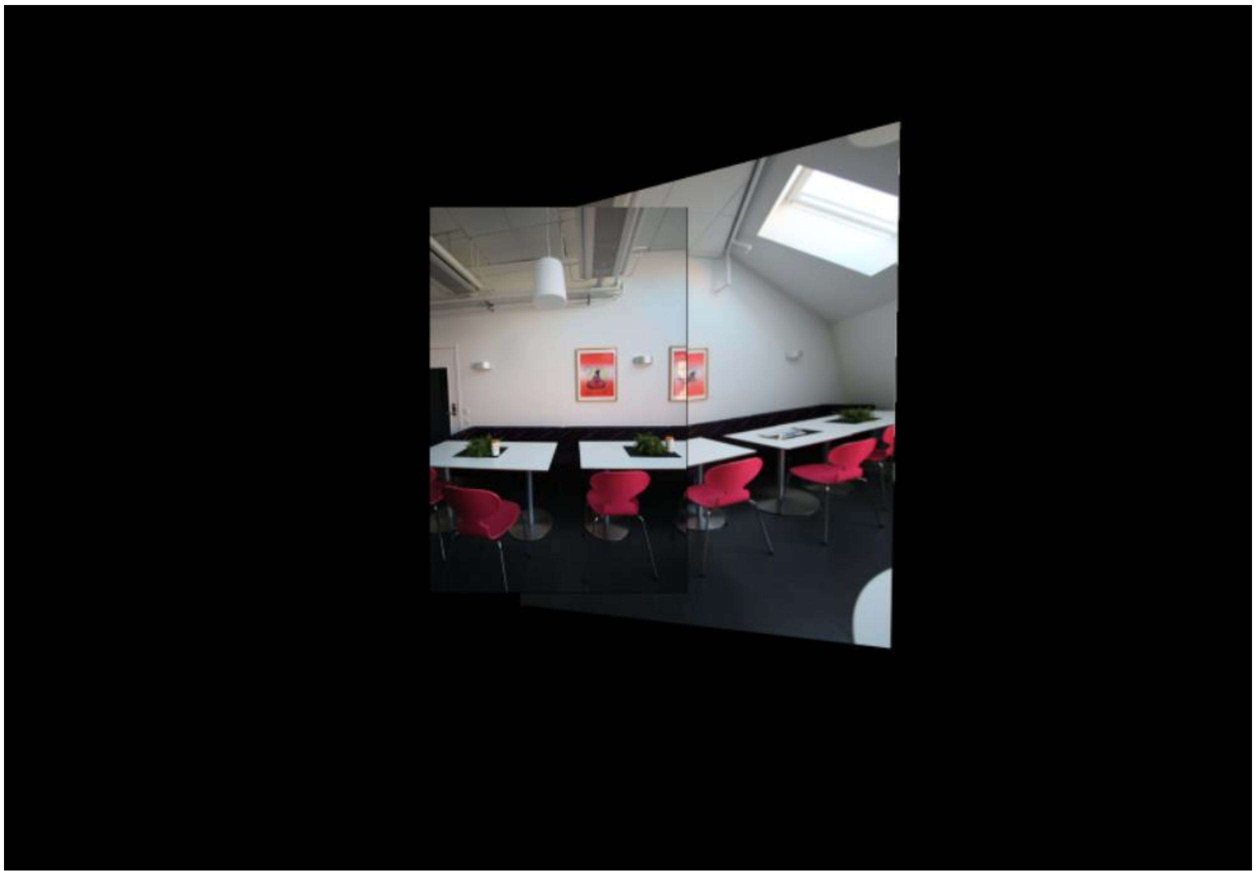
MATLAB cpselect tool to choose control points for Middle and Right images



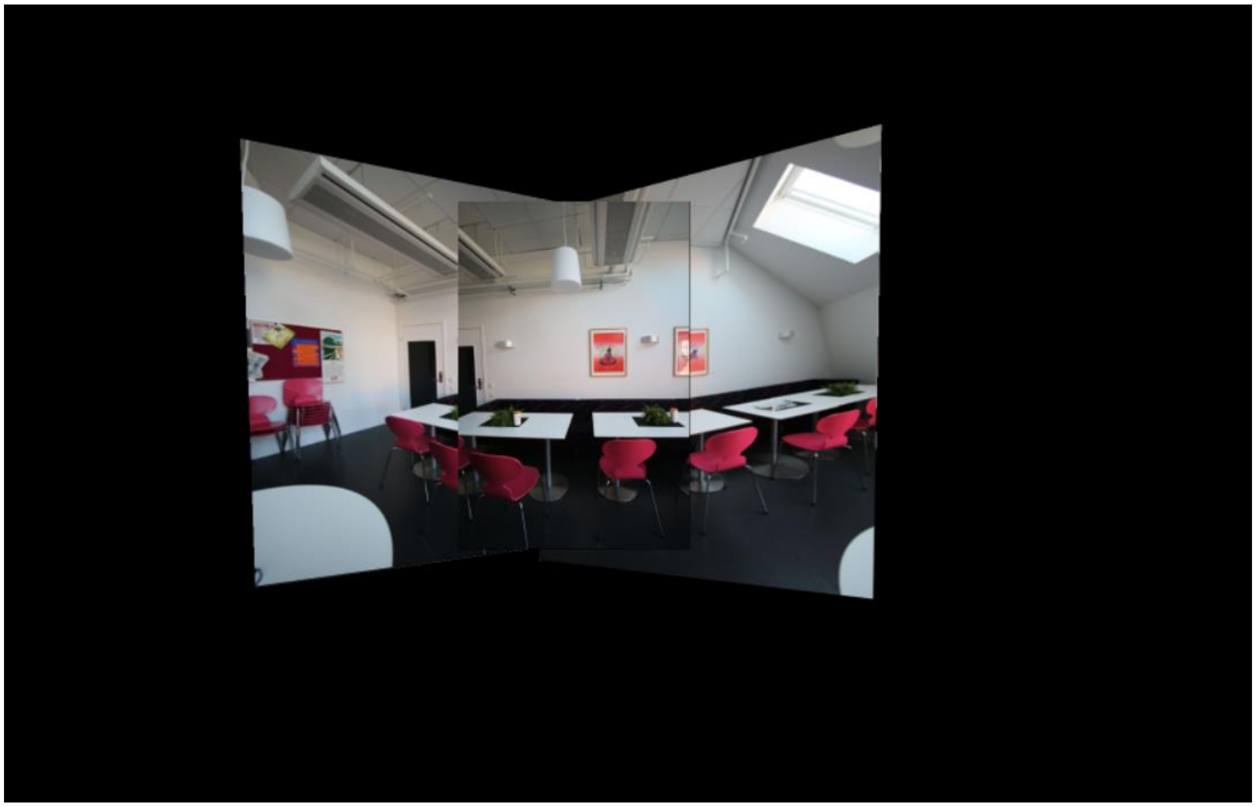
Middle Image placed inside the large canvas for building the panorama



Left Image stitched to the Middle Image on the panorama canvas



Right Image stitched to the Middle Image on the panorama canvas



Result from tweaking H matrix values for Left to Middle and Middle to Right.

IV. ***Discussions***

I did not get a perfectly stitched image for my result mainly due to the difficulty I had with the H matrices. I used OpenCV’s SURF feature to find the corresponding best matches in both image pairs (left-middle, middle-right) and used cpselect to get the coordinates of any suitable 4 points and calculated the H-matrices for left and right transformations in MATLAB. I used these results to in the stitching algorithm but got un-intended results.

As I was unable to figure why the Matrix or my implementation was giving me wrong results, I settled for manually manipulating the H-matrices themselves and finally got some image panorama image as shown in image above.

I think I still need to improve on my implementation and understand how to make my result better. This homework was the most difficult for me to implement so far and I was really discouraged from attempting the last homework. I hope to learn from this and do better in the remaining homework’s.

REFERENCES

[1] D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones", *HP Labs Technical Report*, HPL-96-128R1, 1996.

[2] Lecture Notes, Discussion Notes, Piazza posts and Homework handout.

[3] Matlab panorama example: Feature Based Panoramic Image Stitching - MATLAB & Simulink (mathworks.com)

[4] OPENCV feature matching example: OpenCV: Feature Matching with FLANN

[5] Fong, C.. "Analytical Methods for Squaring the Disc." *arXiv: History and Overview* (2015): n. pag.