P4 Panoramas and Stereo

Negar Nejatishahidin
Computer Science
George Mason University
Virginia, USA
nnejatis@gmu.edu

***Abstract*—This is a report for project three.**

I. P4.1 Spherical Reprojection

The images regarding to this question are shown in figure 1, 2, 3.

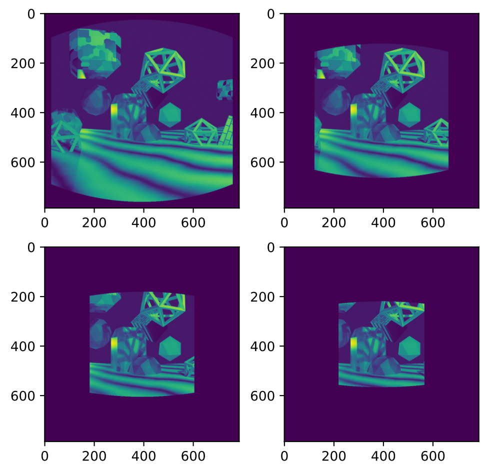


Fig. 1. Shows the four images after the spherical reprojection.

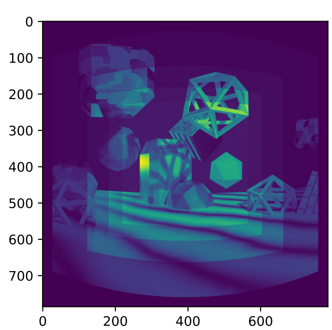


Fig. 2. Shows the images added together, showing that in the center where all images have visibility of the scene, the images properly overlap.

II. P4.2 Panorama Stitching

I have generated 6 images each with 20 degree rotation of camera using blender. The images are shown in figure 10.

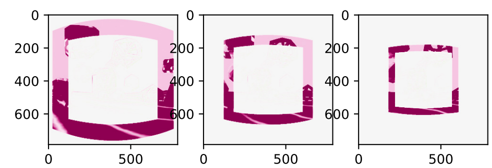


Fig. 3. The "differences" between consecutive Focal Lengths.

After this I reprojected my images to spherical space and I have found the matched between the images and found the homography matrix(here is an affine transformation). **Question**,Figure 5 shows the panorama image result.

The translation matrix for pairs are as follows:

Note that I round the precision to two floating point.

$$pair12 = \begin{bmatrix} 0.99 & -0. & 34.84 \\ 0. & 0.99 & 0.21 \\ 0 & 0 & 1 \end{bmatrix}$$

$$pair34 = \begin{bmatrix} 0.99 & -0. & 34.93 \\ 0. & 0.99 & 0.15 \\ 0 & 0 & 1 \end{bmatrix}$$

$$pair56 = \begin{bmatrix} 1 & -0. & 35.11 \\ 0. & 1 & 0.11 \\ 0 & 0 & 1 \end{bmatrix}$$

Then I added the image with 20mm focal length and the result shown in figure 6. The images stitched together on by one. For example figure 4 shows the results of image 5 and 6 stitching.

III. P4.3 Triangulation

A. *P4.3.1 Projecting Into Image Space*

Question, in general Camera matrix P is as follow :

$$P = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R1 & R2 & R3 & T_x \\ R4 & R5 & R6 & T_y \\ R7 & R8 & R9 & T_z \end{bmatrix}$$

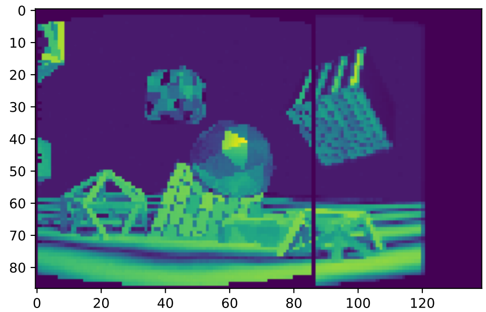


Fig. 4. Image 5 and 6 stitched together

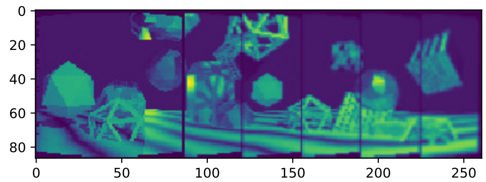


Fig. 5. Panorama image result with 6 images from blender

Therefore P_a and P_b would be :

$$P_a = \begin{bmatrix} f & 0 & 512 \\ 0 & f & 512 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.2 \\ 0 & 0 & 1 & 5 \end{bmatrix}$$

$$P_b = \begin{bmatrix} f & 0 & 512 \\ 0 & f & 512 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 \end{bmatrix}$$

The results are shown in figure 8, 9.

B. P4.3.2 Determining the Size of the Cube

In this section I first used the evaluated 2D points of previous question to make sure if the implementation works correctly. Therefore 2 points are as follow :

$$x_a = [728.4112533725634, 702.6219429125929]$$

$$x_b = [347.604939389232, 753.3961181103705]$$

Triangulation: Linear Solution

Now I do have the 2D points and projection matrices(using the previous question). I have also computed 2 more points as follow(To more accurately compute the cube size):

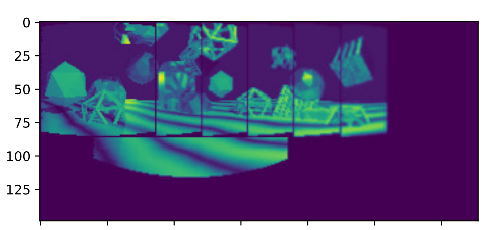


Fig. 6. Panorama image result including the image with f=20

$$x_{a2} = [586, 335]$$

$$x_{b2} = [234, 382]$$

$$x_{a3} = [546, 580]$$

$$x_{b3} = [222, 622]$$

The visualized of the points are shown in figure 10. So, we have all the components to make the matrix A. Using the SVD I have solved the $AX = 0$ equation for each 2 matched points. Therefore, the 3D points have been generated. The function is as follow :

```
def comput3dCoordinate(x_a,x_b,Pa,Pb):

    #matrix A
    A = [x_a[0]*Pa[2]-Pa[0],
         x_a[1]*Pa[2]-Pa[1],
         x_b[0]*Pb[2]-Pb[0],
         x_b[1]*Pb[2]-Pb[1]]

    matrixA = np.matrix(A)
    #svd composition
    u, s, v = np.linalg.svd(matrixA)
    #reshape to 3*3 matrix
    X_temp = np.reshape(v[-1, :4], (4, 1))
    norm = X_temp[3]
    #normalize to get the h
    X = X_temp[0:3]

    return X/norm
```

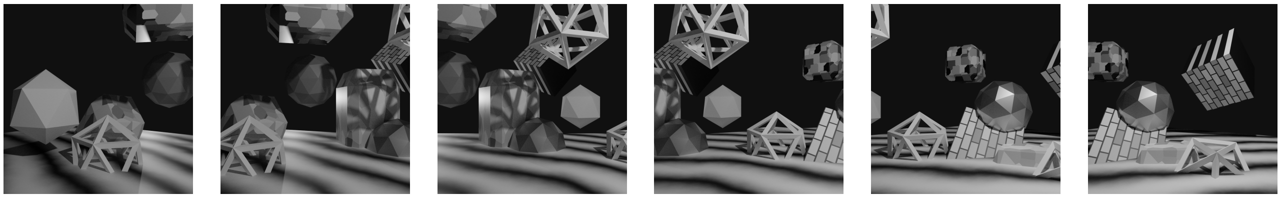
Question, After computing the 3 D points using L2 distance, I have computed 3 distances and average between the results. **The side length of the cube is equal to 1.1519**.

IV. P4.4 Stereo Patch Matching

Question, the epipolar line matched with the (x_a, y_a) is the line that $y = y_a$ and x can be all the value in the range of the length of the x axis of the image. In other world for any value of x, $y = y_a$.

The result are shown in figure 11 and 12.

Question, since the patches along with that area is not available in both images, the function cannot fined a matched patch with the patched of that area, therefore, the result would be more noisy on that area.



(a) 40 degree (b) 20 degree (c) 0 degree (d) -20 degree (e) -40 degree (f) -60 degree

Fig. 7. 6 different images images from blender

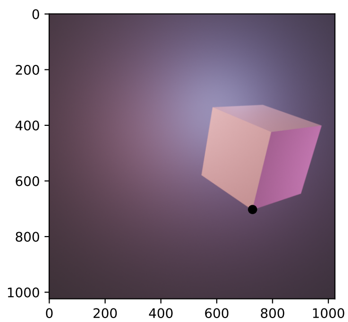


Fig. 8.

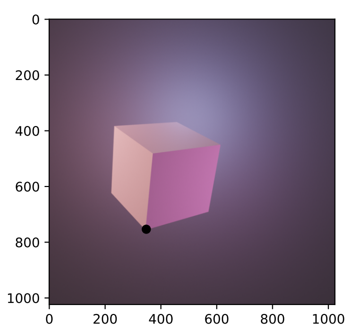
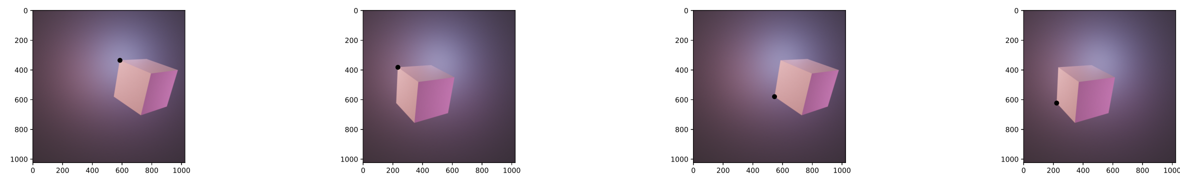


Fig. 9.



(a) x_{a2} (b) x_{b2} (c) x_{a3} (d) x_{b3}

Fig. 10. The x_{a2} and x_{b2} are matched, and the $x_{a3}$and x_{b3} are matched .

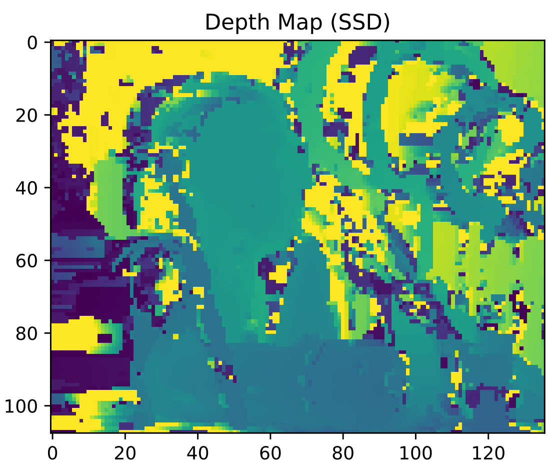


Fig. 11. Generate depth using Sum of Squared Differences.

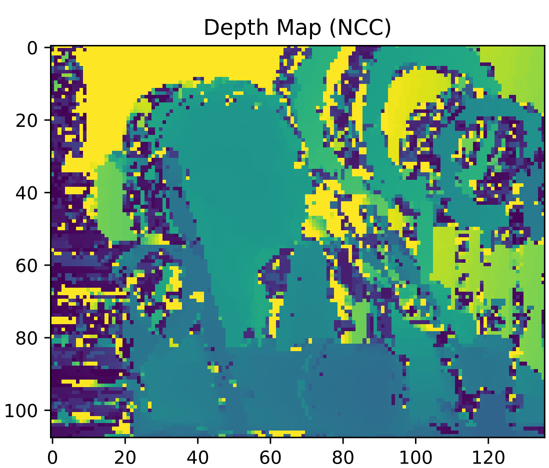


Fig. 12. Generate depth using Normalized Cross Correlation.