Computer Vision and Pattern Recognition - Coursework 1

Name: Krithika Balaji, Username: KB4317, CID: 01378251
Name: María de la Paz Cardona Sánchez, Username: MDC5017, CID: 01410045

Task 1: Collect Data

Image set, taken with an iPhone 7 camera, can be found in Appendix A.

Task 2: Key point correspondences between images

Task part 1a 59 keypoints were manually found using MATLAB function `ginput()` (Appendix B. Fig. 11).

Task part 1b Keypoints were automatically detected and compared using the Minimum Eigenvalue, Speeded Up Robust Features (SURF) and Harris algorithms with MATLAB built-in functions (Appendix B. Fig. 12, 13).

When the different algorithms were run on the same image pairs, different correspondences were found. From visual inspection of individual correspondences, many outliers were identified (Fig. 1). Since both the chosen object and the grid have repetitive geometric patterns, during correspondence matching, the algorithm would get confused as multiple keypoints have low distances between their descriptors, hence suggesting that one key point in an image has multiple correspondences in another image. This circumstance lead to occasional mislabelling.

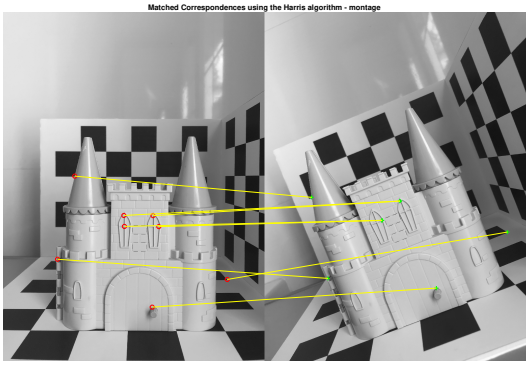

Figure 1. Harris algorithm - outliers can be seen on the window.

To quantitatively describe the quality of the acquired correspondences, the homography matrix from Task 4 was used to project keypoints from image 8019 onto image 8020 (Fig. 3). Using the projected correspondences as the ground truth and the manually and automatically acquired correspondences in image 8020 as the prediction, the Mean Squared Error (MSE) was found (Table 1).

Method	Correspondences	MSE x	MSE y
Manual	59	1.6 x 10^7	1.9 x 10^6
SURF	153	1.12 x 10^{10}	2.18 x 10^9
Harris	8	9.00 x 10^6	6.42 x 10^5
Min Eig	30	7.01 x 10^7	8.17 x 10^6

Table 1. Correspondences found by the manual and automatic methods, along with the MSEs for the x- and y-coordinates.

The Harris algorithm gave the most accurate correspondences, having the lowest MSE for both coordinates. The SURF algorithm found the most correspondences.

Task 3: Camera Calibration

Camera calibration was performed using MATLAB's Camera Calibrator app. Here, 15 images with no object were used. The grid's checkerboard pattern enabled calibration, having black-and-white squares of 41.5 mm side length with easily identifiable corners (Appendix B. Fig 14). Despite using a 3D grid, only one of the rectangular faces was used for the calibration, so the origin was easily identified and parameters and distortions could be obtained.

Task part 1 Intrinsic and extrinsic camera parameters were obtained directly with the calibrator app. Results are displayed on Eq. 1 and Fig. 2. See Appendix B for full extrinsic camera parameters list.

$$\begin{pmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 3409.424 +/- 13.126 & 0 & 0 \\ -1.323 +/- 0.598 & 3410.247 +/- 13.342 & 0 \\ 1983.233 +/- 6.736 & 1489.941 +/- 5.760 & 1 \end{pmatrix} \quad (1)$$

On repeated calibration with a different image set (Appendix B), intrinsic parameters were altered by a small percentage (focal length: 0.89%, principal point: 1.2%). As

the images were taken from different positions, the extrinsic translation and rotation vectors were different.

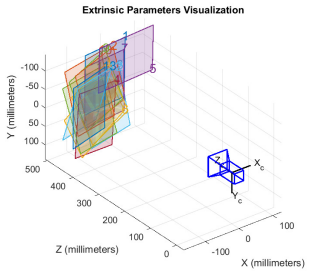

Figure 2. Extrinsic Parameters Visualization.

Task part 2 Radial and tangential distortions were also obtained from running the calibrator app (Table 2). Their small values make them very difficult to be visually appreciated from the images. No other distortions (chromatic or spherical aberration) were visible in any of the images. Thus, the camera's quality is high and it probably counts with embedded post-processing software.

	Radial distortion	Tangential distortion
x-dir	0.1760 +/- 0.0054	-0.5931 +/- 0.0209
y-dir	-0.0014 +/- 0.0005	-0.0004 +/- 0.0007

Table 2. Camera distortions.

Task 4: Transformation estimation

Task part 1a Fig. 17 (Appendix D) shows the image pair chosen for this task. Four pairs of easily identifiable, manually-labelled keypoints (labelled 37-40) were chosen to solve the system of equations. Homography matrix **h** (Eq. 2) was calculated from the relation **Ah**= 0, by applying SVD to **A**.

$$h = \begin{pmatrix} -1.0727 & -1.0097 & -4.2719e-04 \\ -0.7264 & -0.9261 & -2.3227e-04 \\ 3.2606e+03 & 3.4502e+03 & 1 \end{pmatrix} \quad (2)$$

Homography matrix was also found using MATLAB's `estimateGeometricTransform2D()` and selecting `'projective'` as the type of transform. When inputting the same four manually obtained correspondences, the matrix found coincided exactly with Eq. 2. As this function uses the M-estimator SAmple Consensus (MSAC) algorithm, it was tested to see how it can deal with SURF automatically detected keypoints and possible outliers. This method introduced some variation (Appendix D. Eq. 6).

Multiplying keypoints in the left image by **h** gave the corresponding keypoints in the right image (Fig. 3), proving that **h** was correctly calculated.

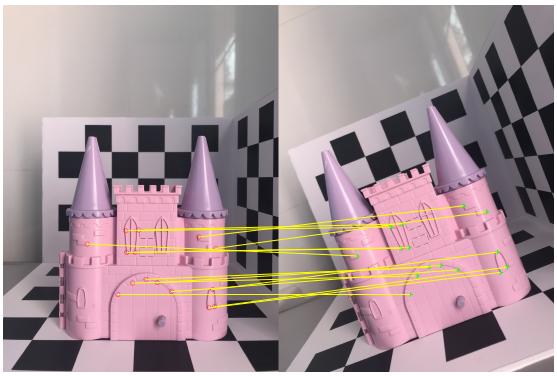

Figure 3. Keypoints and projected correspondences on 15 locations, including the four original ones.

Task part 2a Fig. 18 (Appendix D) shows the pair of images chosen for this task, from which 8 keypoints (37-41, 50-52) were selected. The fundamental matrix **f** was found by solving the equation **Af** $= 0$. The MATLAB function `estimateFundamentalMatrix()` specifying `'Norm8Point'` as the method was also trialled. The obtained matrix was:

$$f = \begin{pmatrix} 2.0595e-07 & 1.7571e-06 & -0.0046 \\ -1.4666e-06 & 1.6114e-08 & 0.0022 \\ 0.0034 & -0.0025 & 1 \end{pmatrix} \quad (3)$$

To check if **f** was correctly calculated, the equation **x′fx**=0 was used, obtaining a value of $-2.7756 * 10^{-17} \approx 0$. SURF-detected keypoints were also used to find the fundamental matrix using the `'RANSAC'` (RANdom SAmple Consensus) method. Again, outliers introduced some variation (Appendix D. Eq. 8).

Epipolar lines (Fig. 4) were found by two different processes: MATLAB's `epipolarLine()` and the equations $l' = Fx$ and $l = F^T x'$ [2].

l' is the epipolar line in the right image corresponding to a point x in the left image; and l is the epipolar line in the left image corresponding to matched point x' in the right image. F is the fundamental matrix. Both methods gave the same epipolar lines.

The two cameras' relative position (forward motion + significant rotation) enabled the epipole to be within the image. Different relative positions, such as converging cameras or parallel motion to the image plane, would give dif-

ferent epipolar lines with an epipole outside the image [1, 2] (Appendix D. Fig. 19).

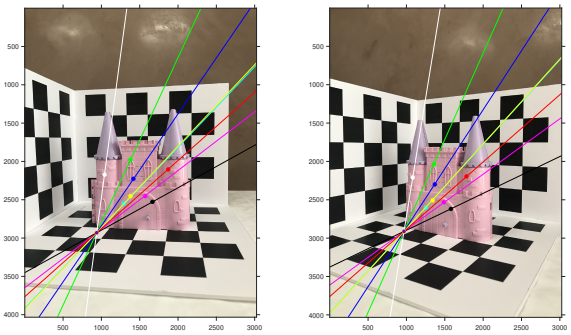

Figure 4. Epipolar lines + Epipole (purple with yellow edge).

Task part 2b From Fig. 4 the epipole can be clearly seen at the intersection of the epipolar lines on both images. Using MATLAB's `isEpipoleInImage()` confirms that the epipole is within the image and that its location corresponds to the intersection in Fig. 4.

Vanishing points and horizon were found using Hough transform and functions `hough()`, `houghpeaks()` and `houghlines()` (Appendix D. Fig. 20). To avoid confusing the algorithm with grids in the different faces, the images were retaken with a flat grid (Fig. 5). The lines do not exactly coincide at a single point, so an intermediate value was estimated, hence the dashed lines.

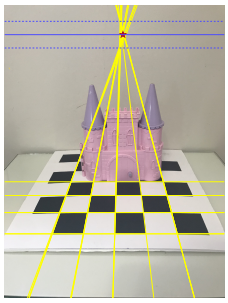

Figure 5. Vanishing points (red star) and horizon (blue line).

See Appendix D for matrix definition **A** for Part 1a and 2a and code for finding **h**, **f** and epipolar lines.

Task 5: 3D geometry

Part 1 Rectified images were found using the function `estimateUncalibratedRectification()`, which returned projective transformations that were input into the function `rectifyStereoImages()`.

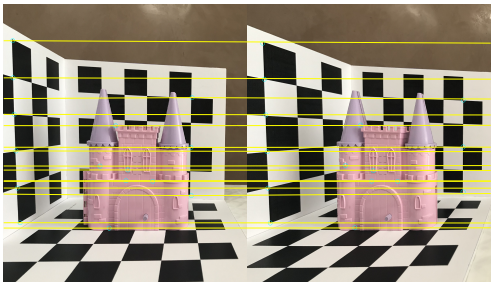

Figure 6. Rectified image pair and corresponding epipolar lines.

The images can be seen to be stereo rectified as the correspondences can be found along horizontal scan lines.

Rectified images were also found following a different approach which involved using the stereo camera calibration app on MATLAB (Appendix E. Fig. 21).

Part 2 The rectified images (Fig. 6) were used as input to the `disparity()` function to get the disparity map (Fig. 7). Because this rectification function did not calibrate the images, the focal length could not be determined. To obtain a depth map, a reliable value of the focal length along with the baseline and the disparity map were needed. Hence, the disparity map was approximated to be the depth map.

From Fig. 7, most of the front side of the castle is visible, further suggesting that rectification was done properly. Certain elements such as the door handle or the windows can be distinguished from the rest, indicating they are at a slightly different depth from the rest of the structure. The disparity map from stereo calibrated rectification (Appendix E. Fig. 24) did not give clear results.

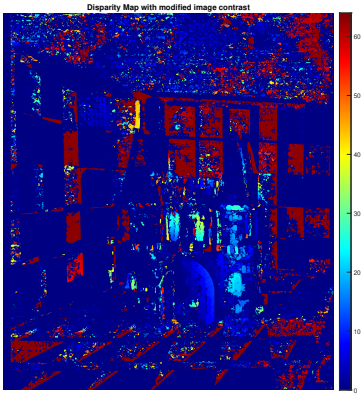

Figure 7. Depth map.

References

[1] D. Hoiem. University of Illinois. Computer Vision CS 543. Epipolar Geometry and Stereo Vision. 2017. https://courses.engr.illinois.edu/cs543/sp2017/lectures/Lecture%2014%20-%20Epipolar%20Geometry%20and%20Stereo%20-%20Vision_Spring2017.pdf. Viewed 21 Feb 2021.

[2] R. Hartley, A. Zisserman. Epipolar Geometry and the Fundamental Matrix. Multiple View Geometry in Computer Vision. 2004. Cambridge University Press, ISBN: 0521540518. https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/HZepipolar.pdf

[3] J. Hays. Georgia Tech. Computer Vision CS 4476. Multiple Views and Motion. https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/

[4] R. Collins. Penn State University. Introduction to Computer Vision CSE Department. Essential and Fundamental Matrices. 2007. http://www.cse.psu.edu/~rtc12/CSE486/lecture19_6pp.pdf. Viewed 8 Feb 2021.

[5] R. Collins. Penn State University. Introduction to Computer Vision CSE Department. The Eight-Point Algorithm. 2007. http://www.cse.psu.edu/~rtc12/CSE486/lecture20_6pp.pdf. Viewed 8 Feb 2021.

Appendices

Appendix A - Task 1

A.1 Grid images - for calibration

Figure 8. Grid image sequence.

A.2 HD Sequence

Figure 9. HD sequence with object.

A.3 FD Sequence

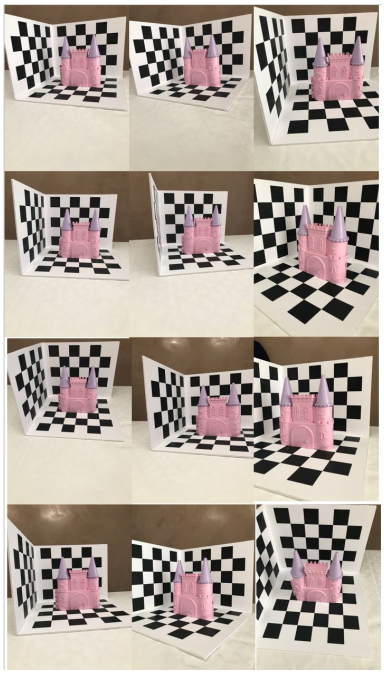

Figure 10. FD Sequence with object.

Appendix B - Task 2

Figures 11, 12 and 13 show the visualization of the matched correspondences found manually and using the Minimum Eigenvalue and SURF algorithms.

Figure 11. Manually acquired correspondences. See number label for correspondences.

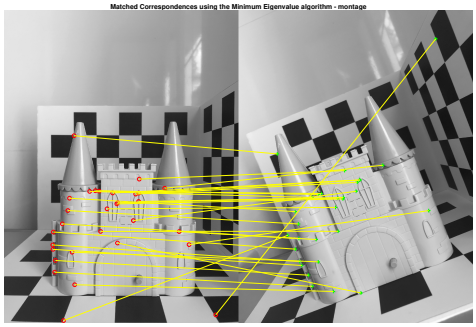

Figure 12. Automatic Correspondences found from Minimum Eigenvalue algorithm. 30 correspondences were found.

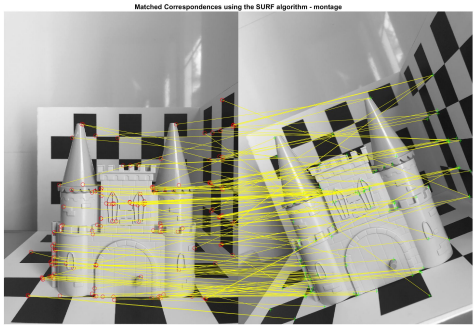

Figure 13. Automatic Correspondences found from SURF algorithm. 153 correspondences were found.

Appendix C - Task 3

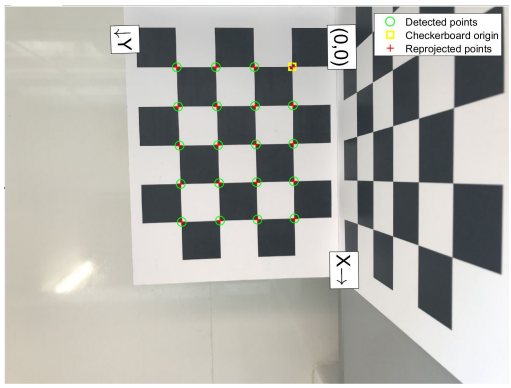

Figure 14. Camera App identification of keypoints for calibration.

C.1 Extrinsic camera parameters Fig. 2.

x-axis	y-axis	z-axis
-0.021 +/- 0.003	-0.089 +/- 0.002	1.546 +/- 0.0003
-0.0316+/- 0.002	-0.099 +/- 0.002	1.299 +/- 0.0003
-0.082 +/- 0.002	-0.070 +/- 0.002	1.170 +/- 0.0002
-0.141 +/- 0.003	-0.084 +/- 0.003	1.673 +/- 0.0003
-0.073 +/- 0.004	-0.069 +/- 0.004	3.126 +/- 0.0004
-0.265 +/- 0.003	0.139 +/- 0.003	2.740 +/- 0.0004
-0.058 +/- 0.002	-0.069 +/- 0.002	1.557 +/- 0.0003
-0.112 +/- 0.002	-0.004 +/- 0.002	1.425 +/- 0.0002
-0.065 +/- 0.002	-0.086 +/- 0.002	1.213 +/- 0.0003
-0.298 +/- 0.002	0.056 +/- 0.002	1.502 +/- 0.0004
-0.368 +/- 0.002	0.135 +/- 0.002	1.489 +/- 0.0004
0.046 +/- 0.002	-0.413 +/- 0.002	1.575 +/- 0.0003
-0.204 +/- 0.002	-0.224 +/- 0.003	1.556 +/- 0.0003

Table 3. Rotation vectors for each of the 13 images used to calibrate the camera.

x-axis	y-axis	z-axis
42.767 +/- 0.927	-136.597 +/- 0.795	468.214 +/- 1.888
4.476 +/- 0.919	-125.450 +/- 0.788	464.154 +/- 1.847
-10.714 +/- 0.924	-85.808 +/- 0.789	468.321 +/- 1.840
19.793 +/- 0.929	-25.105 +/- 0.795	472.693 +/- 1.845
125.358 +/- 0.918	-18.446 +/- 0.802	469.599 +/- 1.894
44.608 +/- 0.931	64.522 +/- 0.797	472.035 +/- 1.854
41.816 +/- 0.927	-101.865 +/- 0.795	471.188 +/- 1.868
33.180 +/- 0.937	-57.515 +/- 0.802	477.801 +/- 1.875
-18.828 +/- 0.920	-126.299 +/- 0.784	464.609 +/- 1.848
-5.875 +/- 0.979	-101.735 +/- 0.826	497.756 +/- 1.894
-3.971 +/- 0.991	-51.426 +/- 0.848	507.526 +/- 1.873
-26.077 +/- 0.855	-96.114 +/- 0.726	430.229 +/- 1.673
-43.482 +/- 0.860	-102.405 +/- 0.727	432.114 +/- 1.702

Table 4. Translation vectors for each of the 13 images used to calibrate the camera.

C.2 Second Trial

Figure 15. Image set for camera calibration trial 2.

Intrinsic Camera Parameters

$$\begin{pmatrix} 3379.383 +/- 30.413 & 0 & 0 \\ -8.121 +/- 2.064 & 3379.989 +/- 30.118 & 0 \\ 2007.702 +/- 13.476 & 1511.258 +/- 10.431 & 1 \end{pmatrix} \quad (4)$$

Extrinsic Camera Parameters

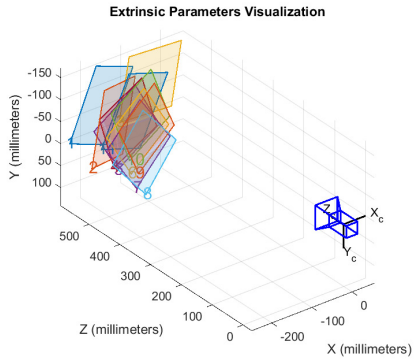

Figure 16. Extrinsic Parameters Visualization.

Appendix D - Task 4

Figure 17. Homography image pair.

D.1 Homography matrix

A definition - homography

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1 * x'_1 & y_1 * x'_1 & x'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_4 & y_4 & 1 & 0 & 0 & 0 & x_4 * x'_4 & y_4 * x'_4 & x'_4 \end{pmatrix} \quad (5)$$

Where (x, y) is a point in the left image and (x', y') is a point in the right image. The subindices indicate the different keypoints.

Homography Matrix MATLAB Code

```
%% Find homography matrix

[U,S,V] = svd(A);

h = V(:,9) ./ V(9,9);
h = reshape(h, [3,3]);
```

Homography matrix using MSAC and SURF detected keypoints

$$\begin{pmatrix} -0.0101e+03 & -0.0215e+03 & -0.0093 \\ 0.0122e+03 & 0.026e+03 & 0.0113 \\ 1.0788e+03 & 2.2982e+03 & 1 \end{pmatrix} \quad (6)$$

Because of the nature of the MSAC algorithm, this matrix changes slightly every time the algorithm is run.

D.2 Fundamental Matrix

A definition - fundamental

$$\begin{pmatrix} x_1 * x'_1 & x_1 * y'_1 & x_1 & y_1 * x'_1 & y_1 * y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_8 * x'_8 & x_8 * y'_8 & x_8 & y_8 * x'_8 & y_8 * y'_8 & y_8 & x'_8 & y'_8 & 1 \end{pmatrix} \quad (7)$$

Where (x, y) is a point in the left image and (x', y') is a point in the right image. The subindices indicate the different keypoints [5].

Fundamental Matrix MATLAB Code

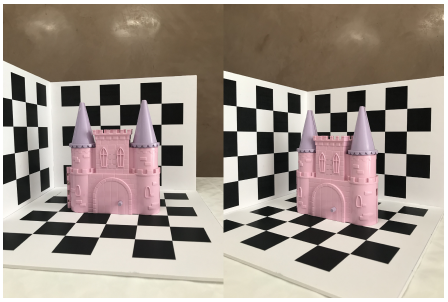

Figure 18. Fundamental image pair.

```
%% Find fundamental matrix

[U,S,V] = svd(A);

f = V(:,9) ./ V(9,9);
f = reshape(f, [3,3]);

% checking f
f_1 = [f(1,1)*xL(1,1)+f(1,2)*xL
    (1,2)+f(1,3);
        f(2,1)*xL(1,1)+f(2,2)*xL
            (1,2)+f(2,3);
        f(3,1)*xL(1,1)+f(3,2)*xL
            (1,2)+f(3,3)];
xprime = [xR(1,1) xR(1,2) 1];

xprime*f_1
```

Fundamental matrix obtained using RANSAC and SURF detected keypoints

$$\begin{pmatrix} 3.0391e-08 & 1.6264e-07 & -3.2115e+04 \\ 3.8535e-08 & 1.0122e-07 & -2.2890e-04 \\ -1.4831e-04 & -4.5967e-04 & 1 \end{pmatrix} \quad (8)$$

D.3 Epipolar lines and epipole

Epipolar lines and epipole MATLAB Code

```
%% Epipolar lines

% plot images
figure;
subplot(1,2,1); imshow(left_image); axis image; hold on;
h = gca;
h.Visible = 'On';
subplot(1,2,2); imshow(right_image); axis image; hold on;
h = gca;
h.Visible = 'On';


colors = ['r', 'g', 'b', 'c', 'y', 'm', 'k', 'w'];
for i=1:8

    % right epiline
    p1 = [ xL(i,1) xL(i,2) 1];
    L_right = [F*p1']';

    % left epiline
    p2 = [xR(i,1) xR(i,2) 1];
    L_left = p2*F;

    color = colors(mod(i, 8)+1);

    % Left image
    subplot(1,2,1);
    % keypoint
    plot1 = plot(xL(i,1), xL(i,2)); hold on;
    set(plot1, 'LineStyle', 'None', 'color', color,...
          'Marker', 'o', 'MarkerFaceColor', color);

    % epipolar lines
    points = lineToBorderPoints(L_left,size(left_image));
    line(points(:,[1,3])',points(:,[2,4])','color', color, 'LineWidth', 1.25); hold on;


    % Right image
    subplot(1,2,2);

    plot2 = plot(xR(i,1), xR(i,2)) ; hold on;
    set(plot2, 'LineStyle', 'None', 'color', color,...
          'Marker', 'o', 'MarkerFaceColor', color);

    points = lineToBorderPoints(L_right,size(right_image)); hold on;
    line(points(:,[1,3])',points(:,[2,4])','color', color, 'LineWidth', 1.25); hold on;

end

%% Find epipole
color = [0.5, 0, 0.7];

% Left Image
[isIn, epipole] = isEpipoleInImage(fNorm8Point, size(left_image))
subplot(1,2,1)
plot(epipole(1,1), epipole(1,2), 'color', 'k', 'Marker', 'o','MarkerFaceColor', color, 'MarkerEdgecolor', 'yellow')

% Right Image
subplot(1,2,2)
[isIn, epipole] = isEpipoleInImage(fNorm8Point', size(left_image))
plot(epipole(1,1), epipole(1,2), 'color', 'k','Marker', 'o','MarkerFaceColor', color,'MarkerEdgecolor', 'yellow')
```

Epipole outside of image

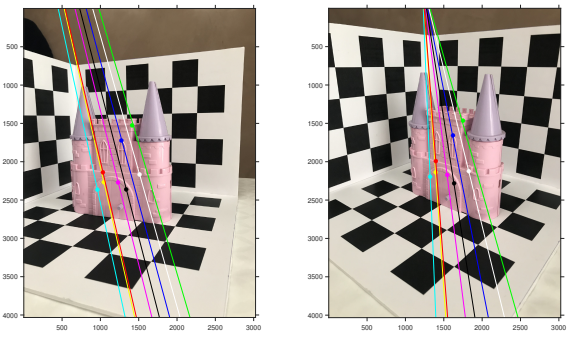

Figure 19. Epipolar lines for a different image pair, where epipole is actually outside the image.

D.4 Vanishing points and Hough Transform

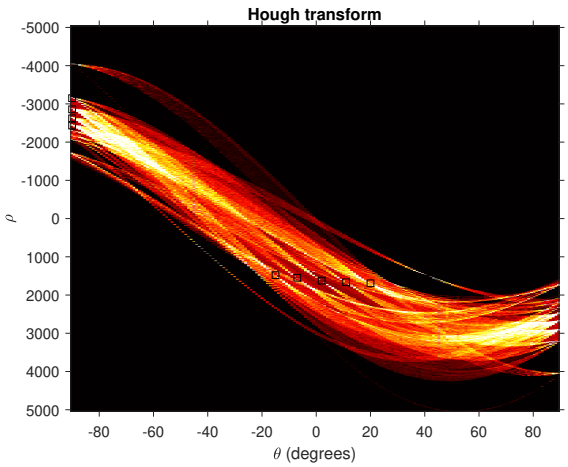

Figure 20. Hough transform.

Appendix E - Task 5

E.1: Images rectified from Stereo Camera Calibrator App

The Stereo Camera Calibrator app was also used to find the stereo rectified images. Below are the rectified images of the 2D calibration grid used to calibrate the camera (Fig. 21), the rectified images of the grid with the castle (Fig. 22), the extrinsic parameter visualization (Fig. 23).

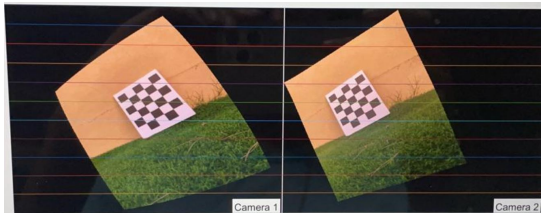

Figure 21. Rectified images of the grid.

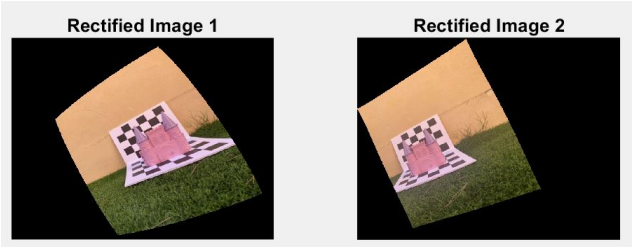

Figure 22. Rectified images of the grid and castle.

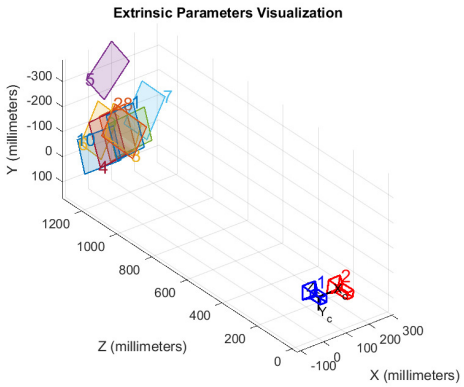

Figure 23. Extrinsic camera parameter visualization.

E.2: Depth map from calibrated rectified images

From the stereo rectified images, the disparity map was found. As seen in Fig. 24, the castle can barely be seen in the disparity map, unlike the disparity map in Fig. 7, where sections of both the grid and the castle can be seen. Hence, despite knowing the focal length of the two cameras, this technique was not used to find the depth map.

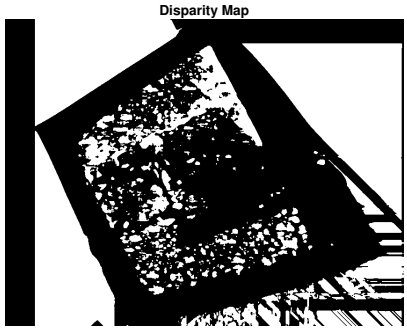

Figure 24. Disparity map calculated based on the images rectified from the Stereo Camera Calibrator App. Has a lot of noise and the grid and the castle can barely be seen.