Ayush Kanyal

002692009

Computer Vision Assignment 1

PART A: Fundamentals

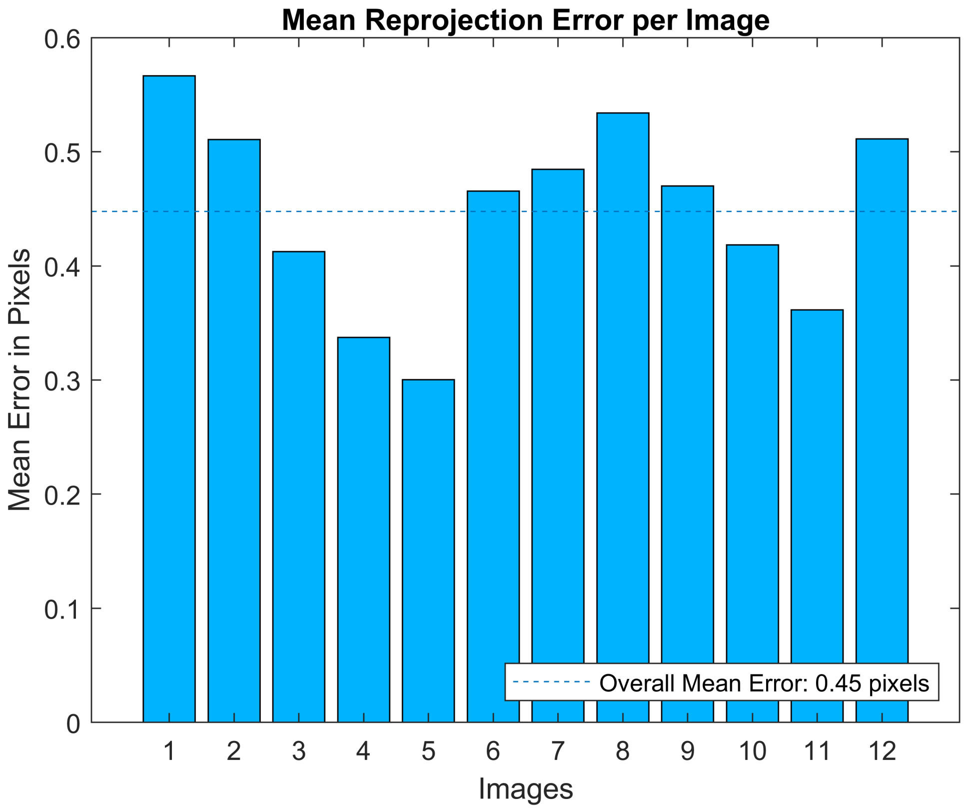
1) MATLAB: 1. Go over the camera calibration toolbox demonstration and calibrate the OAK-D camera https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html.

Ans)

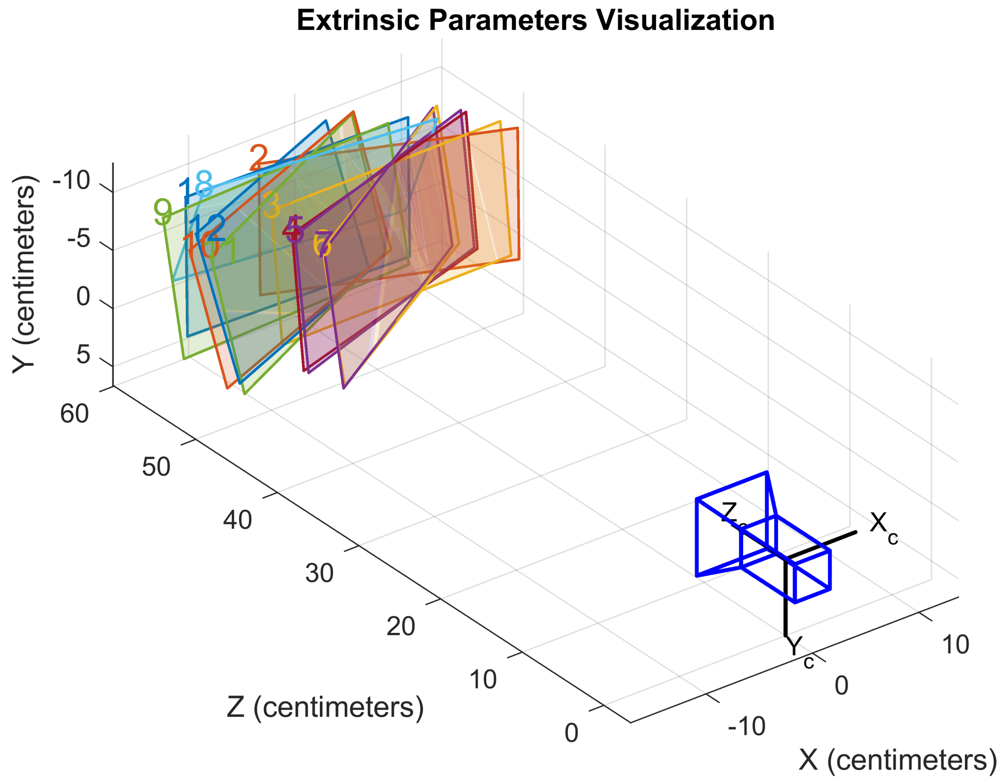
Video solution in Part A folder.

PDF appended below.

```
% Define images to process
imageFileNames = {'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475017779.jpeg',
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475035781.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475038720.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475041398.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475041758.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475045188.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475051774.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475108141.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475161625.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475165991.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475175983.jpeg',...
    'C:\Users\kanya\OneDrive\Desktop\CV\Chessboard capture\16641475189935.jpeg',...
    };
% Detect calibration pattern in images
detector = vision.calibration.monocular.CheckerboardDetector();
[imagePoints, imagesUsed] = detectPatternPoints(detector, imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Read the first image to obtain image size
originalImage = imread(imageFileNames{1});
[mrows, ncols, ~] = size(originalImage);

% Generate world coordinates for the planar pattern keypoints
squareSize = 2.470000e+00;  % in units of 'centimeters'
worldPoints = generateWorldPoints(detector, 'SquareSize', squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] = estimateCameraParameters(imagePoints, worldPoint
    'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
    'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'centimeters', ...
    'InitialIntrinsicMatrix', [], 'InitialRadialDistortion', [], ...
    'ImageSize', [mrows, ncols]);

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams);
```




```
% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');
```




```
% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);
```

```
            Standard Errors of Estimated Camera Parameters
            ----------------------------------------------

Intrinsics
----------
Focal length (pixels):   [ 1523.3867 +/- 4.2075     1528.6228 +/- 4.0811  ]
Principal point (pixels):[ 1000.0736 +/- 1.2439      537.3685 +/- 1.5367  ]
Radial distortion:       [    0.0843 +/- 0.0072      -0.0790 +/- 0.0414  ]

Extrinsics
----------
Rotation vectors:
                         [    0.0329 +/- 0.0025       0.0812 +/- 0.0021       -0.0364 +/- 0.0003  ]
                         [    0.1167 +/- 0.0018       0.4419 +/- 0.0015       -0.0636 +/- 0.0003  ]
                         [    0.0454 +/- 0.0028       0.1339 +/- 0.0021       -0.1093 +/- 0.0003  ]
                         [    0.0050 +/- 0.0019      -0.1962 +/- 0.0017       -0.1001 +/- 0.0003  ]
                         [   -0.0133 +/- 0.0018      -0.2347 +/- 0.0016       -0.1014 +/- 0.0003  ]
                         [   -0.0711 +/- 0.0013      -0.4616 +/- 0.0012       -0.1271 +/- 0.0004  ]
                         [   -0.0582 +/- 0.0013      -0.4861 +/- 0.0012       -0.1160 +/- 0.0004  ]
                         [    0.4803 +/- 0.0018       0.1510 +/- 0.0014       -0.0740 +/- 0.0003  ]
                         [   -0.0600 +/- 0.0018       0.1125 +/- 0.0017       -0.1117 +/- 0.0003  ]
                         [   -0.1454 +/- 0.0019      -0.2246 +/- 0.0016       -0.1797 +/- 0.0003  ]
                         [   -0.1296 +/- 0.0017      -0.3355 +/- 0.0015       -0.1655 +/- 0.0003  ]
                         [   -0.3103 +/- 0.0014      -0.4022 +/- 0.0013       -0.0540 +/- 0.0004  ]

Translation vectors (centimeters):
                         [  -13.6380 +/- 0.0446     -10.4004 +/- 0.0559       55.7622 +/- 0.1601  ]
                         [   -5.1725 +/- 0.0467      -9.3498 +/- 0.0576       57.8223 +/- 0.1476  ]
```

```
[   -7.4679 +/- 0.0425       -8.1966 +/- 0.0534       53.2881 +/- 0.1486  ]
[   -8.7785 +/- 0.0409       -8.6025 +/- 0.0495       49.0246 +/- 0.1370  ]
[   -8.7981 +/- 0.0405       -8.6671 +/- 0.0490       48.6547 +/- 0.1362  ]
[   -8.4420 +/- 0.0379       -8.7050 +/- 0.0458       45.7378 +/- 0.1313  ]
[   -8.6273 +/- 0.0375       -8.8364 +/- 0.0454       45.2955 +/- 0.1308  ]
[  -12.8375 +/- 0.0442      -11.3348 +/- 0.0544       54.5398 +/- 0.1610  ]
[  -16.4937 +/- 0.0434      -10.0643 +/- 0.0548       54.8031 +/- 0.1557  ]
[  -17.0995 +/- 0.0426       -9.3254 +/- 0.0508       50.7641 +/- 0.1502  ]
[  -16.5994 +/- 0.0410       -9.5143 +/- 0.0491       48.8366 +/- 0.1441  ]
[  -17.0707 +/- 0.0416      -11.0773 +/- 0.0502       50.0071 +/- 0.1485  ]
```

```
% For example, you can use the calibration data to remove effects of lens distortion.
undistortedImage = undistortImage(originalImage, cameraParams);

% See additional examples of how to use the calibration data.  At the prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('StructureFromMotionExample')
```

PART B: MATLAB/Python Prototyping 2.

2) Write a MATLAB/Python script to find the real-world dimensions (e.g. diameter of a ball, side length of a cube) of an object using perspective projection equations. Validate using an experiment where you image an object using your camera from a specific distance (choose any distance but ensure you are able to measure it accurately) between the object and camera.

Ans)

Video solution in part B.

PDF Appended below.

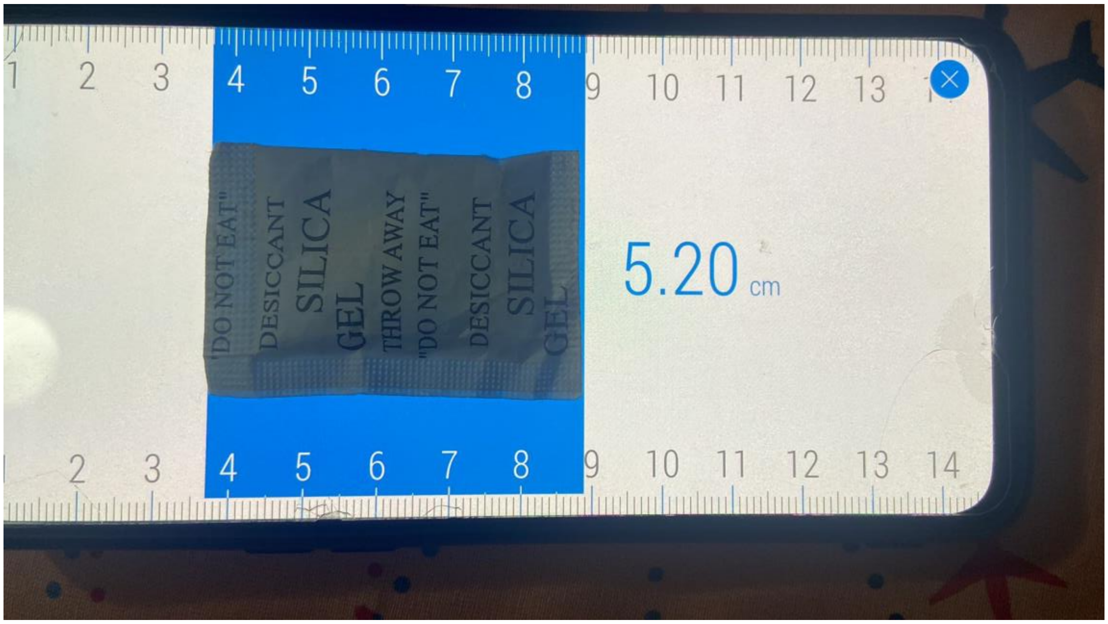
```
I = imread('16641553972049.jpeg'); % Read the image
imshow(I); % Display the image
[x y] = ginput(2); % reads two points. x is a 2x1 column vector with x
coordinates and y is a 2x1 column vector with y coordinates.
```

```
%Focal length from part A
fx = 1523.38;
fy = 1528.62;

%distance between camera and object
z0 = 29.22;

%point1
x1 = z0*(x(1)/fx);
y1 = z0*(y(1)/fy);

%point2
x2 = z0*(x(2)/fx);
y2 = z0*(y(2)/fy);


% using Euclidean distance to find the distance between point1 and point2
dist = sqrt((x2-x1)^2 + (y2-y1)^2);
disp("the distance between the two points is");
```

```
the distance between the two points is
```

```
disp(dist);
```

```
5.2384
```



"DO NOT EAT"
DESICCANT
SILICA
GEL
THROW AWAY
"DO NOT EAT"
DESICCANT
SILICA
GEL
5.20 cm

PART C: Application development Familiarize with the Depth AI SDK.

3) Setup your application to show a RGB stream from the mono camera and a depth map stream from the stereo camera simultaneously. Is it feasible? What is the maximum frame rate and resolution achievable?

Ans)

Yes, its achievable. On a 10 Gb/s type c cable I was able to get 19-20 fps with RGB camera running at 4K resolution and stereo at 400p.

Video solution on part C folder.

4) Run the camera calibration tutorial. Compare the output with answers from Part A calibration results

Ans) Camera matrix obtained from MATLAB tool was:

[[1523.3867 +/- 4.2075 0. 1000.0736 +/- 1.2439]

[0. 1528.6228 +/- 4.0811 537.3685 +/- 1.5367]

[0. 0. 1.]]

And from the DepthAi camera calibration was:

[[1540.043 0. 997.75275]

[0. 1536.3896 523.34106]

[0. 0. 1.]]

Video solution in part C