# Assignment – 1

# Part-A

## Camera Calibration:

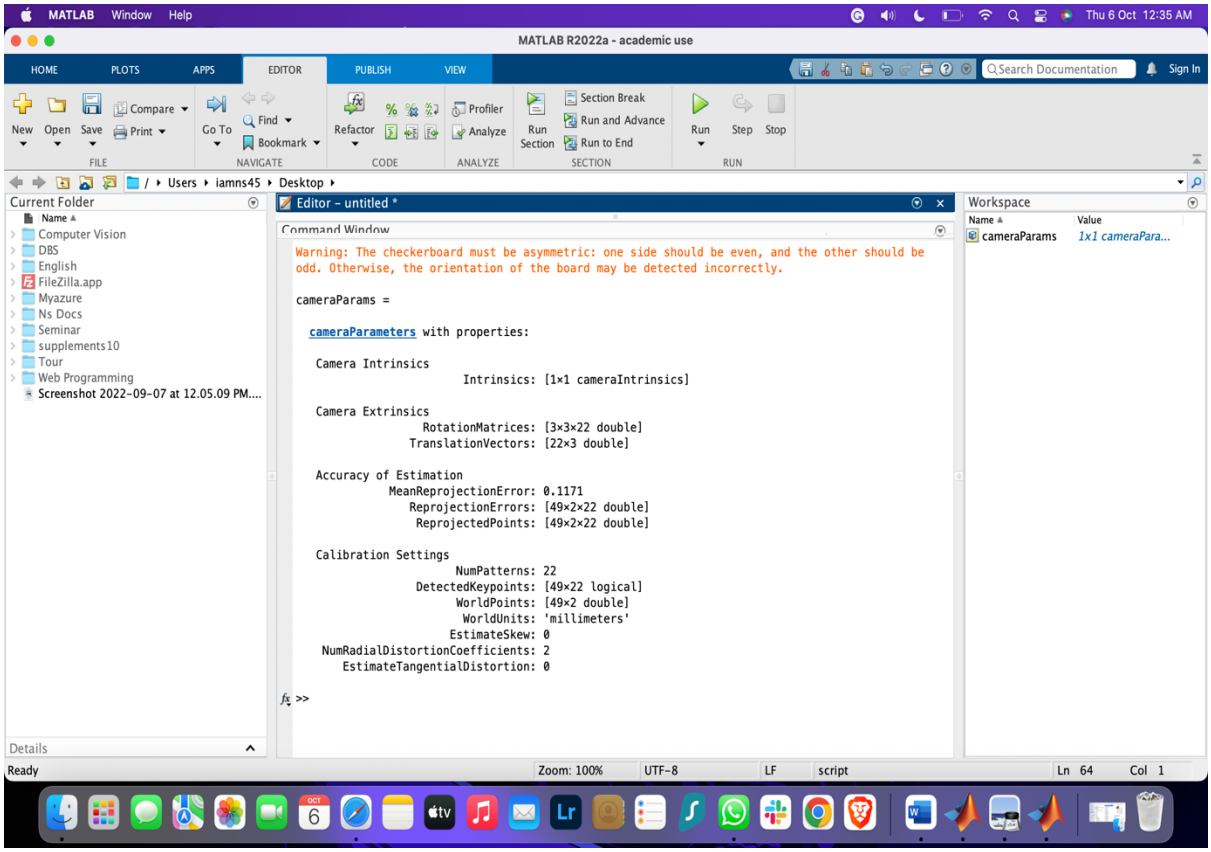
For Camera Calibration, we took the images of the chessboard by using the 0AK-D camera from a distance of 42.4 inches.

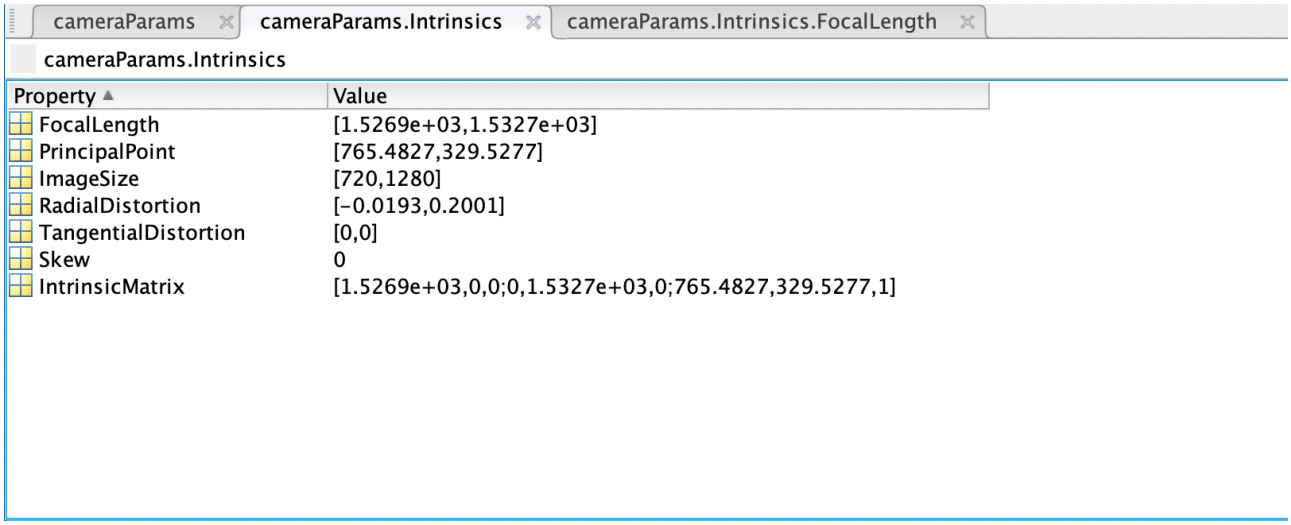
Calibrating the OAK-D camera using MATLAB:

```
 % Auto-generated by cameraCalibrator app on 06-Oct-2022
%-------------------------------------------------------


% Define images to process
imageFileNames = {'/Users/iamns45/cv/image1.jpg',...
    '/Users/iamns45/cv/image2.jpg',...
    '/Users/iamns45/cv/image3.jpg',...
    '/Users/iamns45/cv/image4.jpg',...
    '/Users/iamns45/cv/image5.jpg',...
    '/Users/iamns45/cv/image6.jpg',...
    '/Users/iamns45/cv/image7.jpg',...
    '/Users/iamns45/cv/image8.jpg',...
    '/Users/iamns45/cv/image9.jpg',...
    '/Users/iamns45/cv/image10.jpg',...
    '/Users/iamns45/cv/image11.jpg',...
    '/Users/iamns45/cv/image12.jpg',...
    '/Users/iamns45/cv/image13.jpg',...
    '/Users/iamns45/cv/image14.jpg',...
    '/Users/iamns45/cv/image15.jpg',...
    '/Users/iamns45/cv/image16.jpg',...
    '/Users/iamns45/cv/image17.jpg',...
    '/Users/iamns45/cv/image18.jpg',...
    '/Users/iamns45/cv/image19.jpg',...
    '/Users/iamns45/cv/image20.jpg',...
    '/Users/iamns45/cv/image21.jpg',...
    '/Users/iamns45/cv/image22.jpg',...
    };
% 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 = 1.950000e+01;  % in units of 'millimeters'
worldPoints = generateWorldPoints(detector, 'SquareSize', squareSize);
```

```
% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] =
estimateCameraParameters(imagePoints, worldPoints, ...
    'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
    'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'millimeters', ...
    '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);

% 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')
```

## OUTPUT:



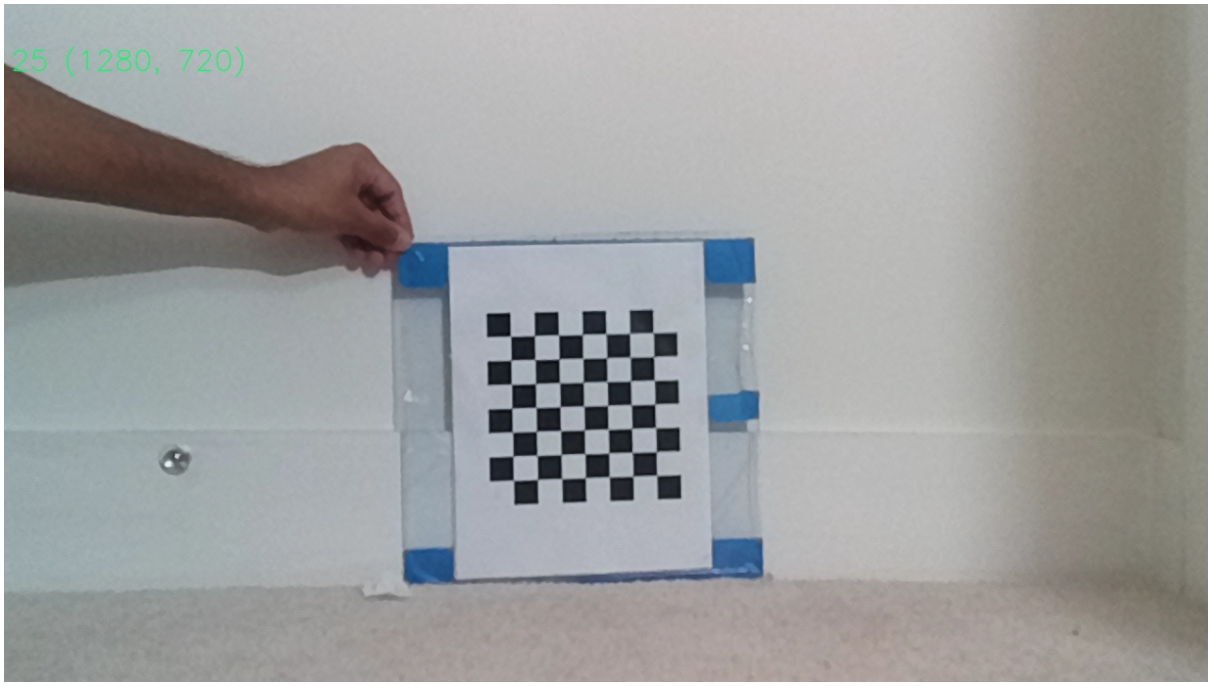



# PART B

MATLAB Script to find the real-world dimension

## Code:

```
I=imread("/Users/iamns45/cv/image6.jpg ")
imshow(I)
[x,y]=ginput(2)
z_dist=939.8
fx=1329.3020
fy=1330.4565
x1=z*(x(1)/fx)
x2=z*(x(2)/fx)
y1=z*(y(1)/fy)
y2=z*(y(2)/fy)
dist=sqrt((y2-y1)^2+(x2-x1)^2)
fprintf("The Estimated distance Between 2 points",dist)
```




**Output:**
**The Estimated distance between 2 points is 21.2mm**
**The original distance is 20.5mm**

## PART-C

It is feasible to show to show an RGB stream from the mono camera and a depth map stream from the stereo camera simultaneously.

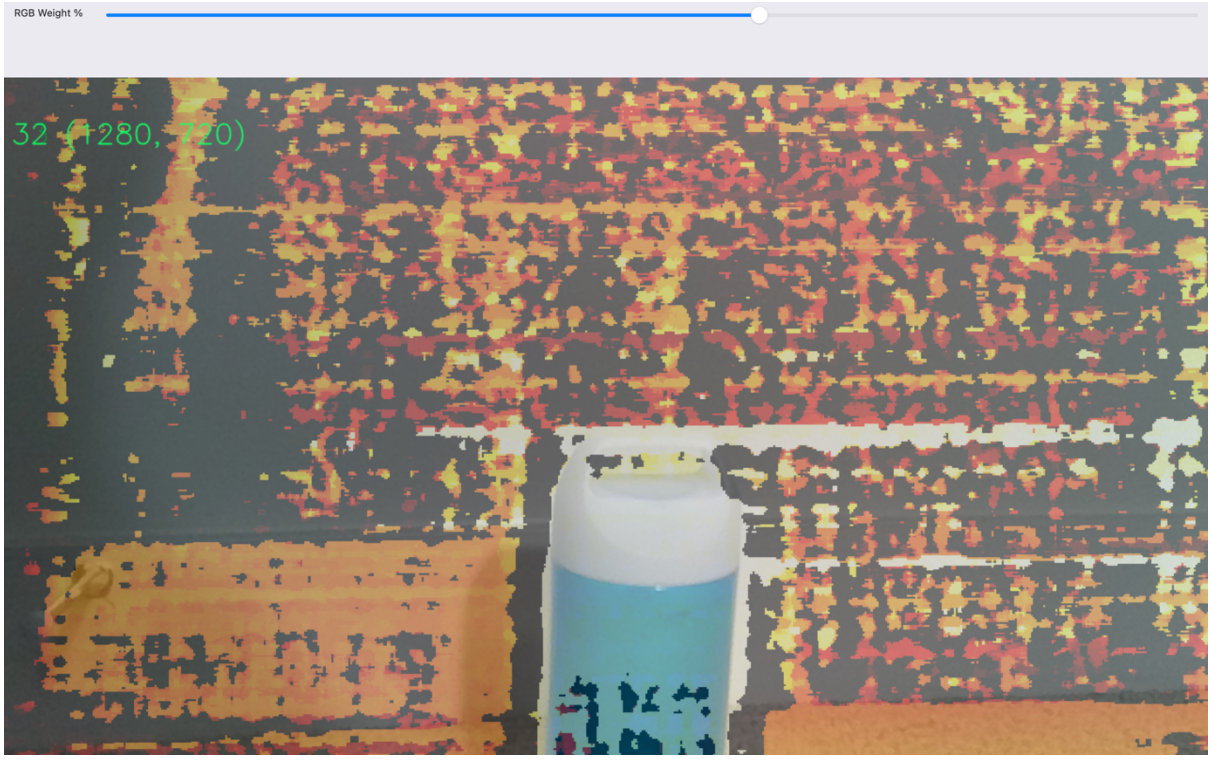



Fig. Combination of RGB and depth

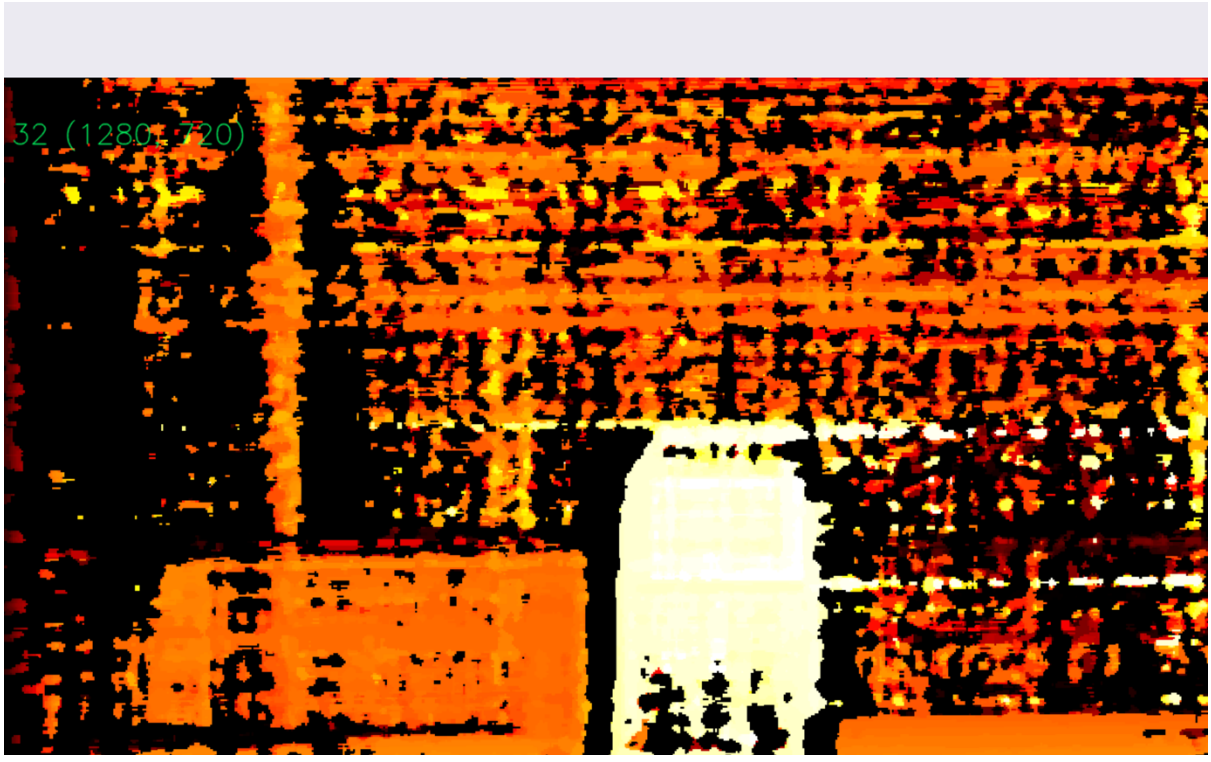



Fig. Depth map


Fig. RGB

Maximum 32fps
1280 720 resolution(720p)

**Video Link:**

https://drive.google.com/file/d/1AJ_OfxIJZOrexerqGdtAFZs5G8MWNX4B/view?usp=sharing

**GitHub link:**

**https://github.com/Iamns45/Computer-Vision.git**