CSc 8830: Computer vision Assignment-1 Report

SAI TEJASWI CHAKRAVARAM

002762775

documentation to execute the script.

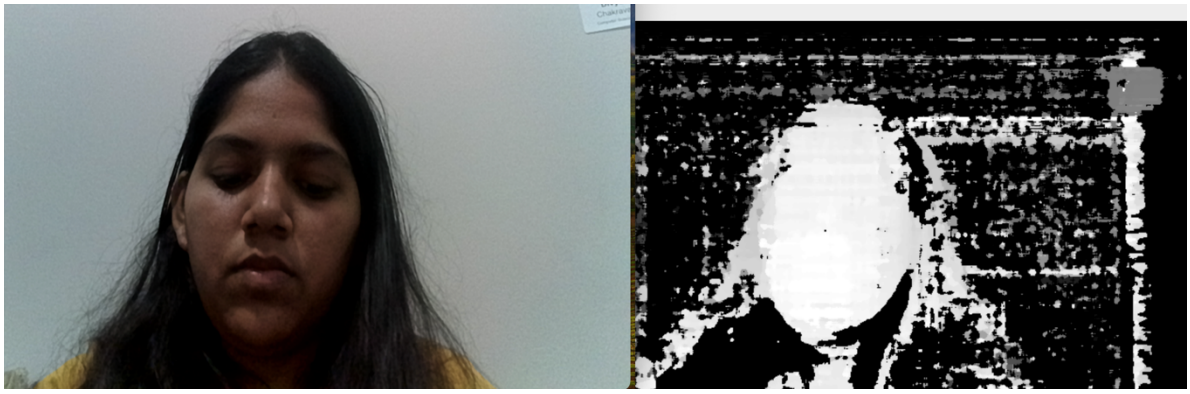
Preparatory exercises

(4). With the OAK-D camera, set up your application to show a RGB stream from the mono camera and a depth map stream from the stereo camera simultaneously. Make a note of what is the maximum frame rate and resolution achievable?

Method: 1. Import Required Libraries: OpenCV, NumPy, and depthai are among the libraries that must be imported.
2. Set Parameters: Establish parameters like RGB and depth camera resolutions, RGB frame resizing dimensions, and stereo depth configuration options.
3. Build Pipeline: Get the depthai pipeline started.
4. RGB Camera Configuration: Establish a colour camera node and set its parameters for resolution.
Get the RGB stream's output link created.
5. Set Up Mono Cameras: Establish left and right mono camera nodes, then adjust their board sockets and resolutions.
6. Set Up Stereo Depth: Establish a stereo depth node and set up features like left-right check, extended disparity, and subpixel disparity. Connect the stereo depth node to the left and right mono cameras.

7. Establish Output Queues: Set up RGB and disparity output queues.
8. Initialise FPS Variables: This step sets the parameters used to calculate frames per second (FPS).
9. Run Pipeline: The pipeline can be run by utilising the depthai device context manager.
10. Retrieve Frames: Get frames for RGB and depth streams continuously from the output queues.
11. Show Frames: Use OpenCV to show the RGB frame and depth map. To improve visualisation, resize the RGB frame. To improve visibility, normalise the depth map.
12. Calculate and Display FPS: Use the console to calculate and print the frame rate.

Note: The provided stereo-vision camera is used to display the depth map and in RGB. 90 frames per second is the maximum frame rate that was attained. The observed resolution for RGB and depth map was 1080.



Assignment :

1. **Report the calibration matrix for the camera chosen and verify (using anexample) the same.**

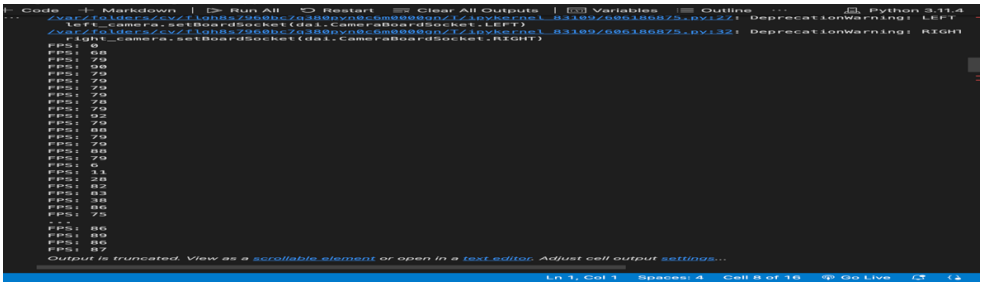
Method 1:

Obtain Images for Calibration: a. Utilise the capture_color_images() and capture_monochrome_images() functions to obtain images. The monochrome (left and right) and colour (RGB) cameras' images are captured by these functions, in turn.

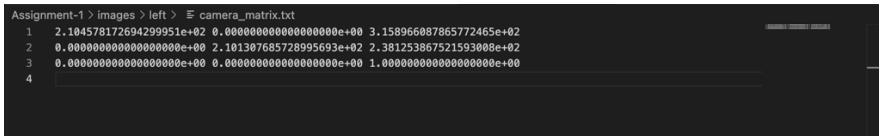
2. Calibrate Cameras: a. Set up each camera individually by using the caliberate_camera() function. Using the camera matrix and distortion coefficients as calibration parameters, this function locates the chessboard corners in the photos that are taken and stores the calibration settings.
3. Load Camera Parameters: a. Use the load_camera_parameters() function to import the previously saved files containing the camera matrix and distortion coefficients.

4. Adjust distortion: a. Using the loaded camera parameters, apply the correct_distortion() function to adjust distortion in an input image.

5. Determine Calibration Error: a. Use the calculate_error() function to determine the calibration error by contrasting an undistorted image's chessboard square size with a known size. This stage confirms that the calibration was accurate. 6. Use an Example to Confirm Calibration:
a. Open a sample picture.
b. Input the camera's settings.
c. Use the loaded parameters to fix the image's distortion.
d. Determine the example image's calibration error.



Calibration of camera Matrix:



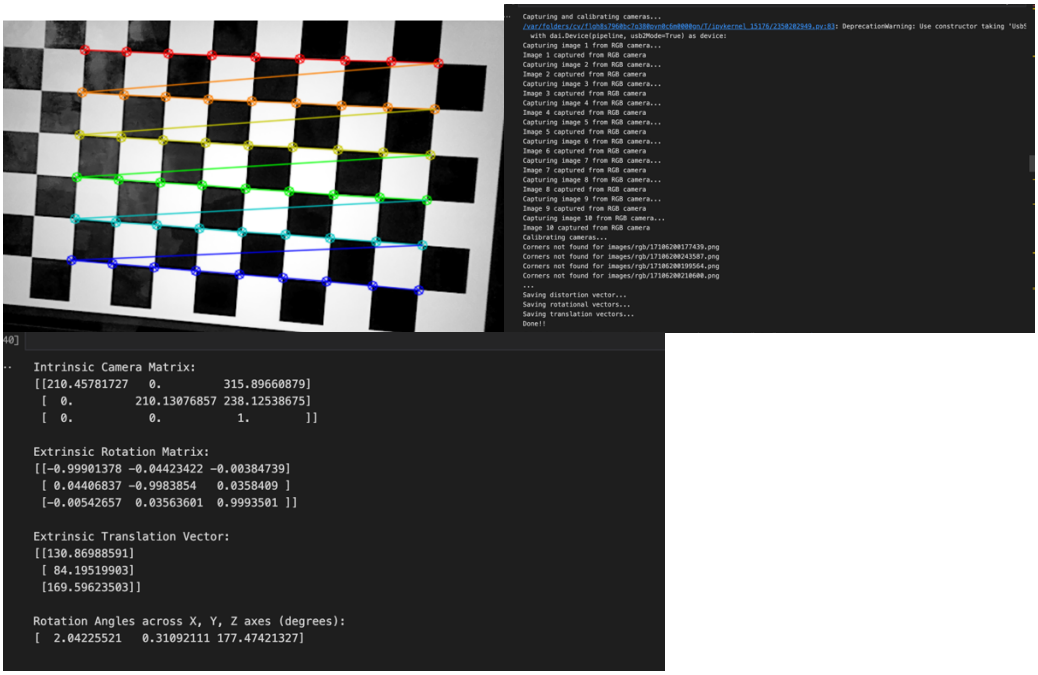
Finding a square's side on the calibration checkerboard and reporting the calibration error allows you to verify the above matrix.

2. **Point the camera to a chessboard pattern or any known set of reference points that lie on the same plane. Capture a series of 10 images by changing the orientation of the camera in each iteration. Select any 1 image, and using the image formation pipeline equation, set up the linear equations in matrix form and solve for intrinsic and extrinsic parameters (extrinsic for that particular orientation). You will need to make measurements of the actual 3D world points, and mark pixel coordinates. Once you compute the Rotation matrix, you also need to compute the angles of rotation along each axis. Choose your order of rotation based on your experimentation setup.**

Method:

1. a.Point the camera at a checkerboard pattern or reference points that are located on the same plane to capture images with varying camera orientations.
   b. Take ten pictures in succession, reorienting the camera with each one.
   2. Select an Image: a. From the acquired series, select one image for additional processing.
   3. Determine Relative 3D World Points and Pixel Locations:
   The chessboard pattern's reference points' 3D world coordinates can be measured or determined manually.
   b. Based on this, indicate in the chosen image the pixel coordinates of these locations.
   4. Establish Linear Equations: a. In matrix form, establish linear equations using the image formation pipeline equation.

   5. Determine the Intrinsic and Extrinsic Parameters: a. Compute the intrinsic and extrinsic parameter linear equations using the Linear Least Squares technique.
   b. Take the intrinsic-extrinsic decomposition and extract the rotation matrix R. d. Using the rotation matrix, calculate the angles of rotation along each axis.

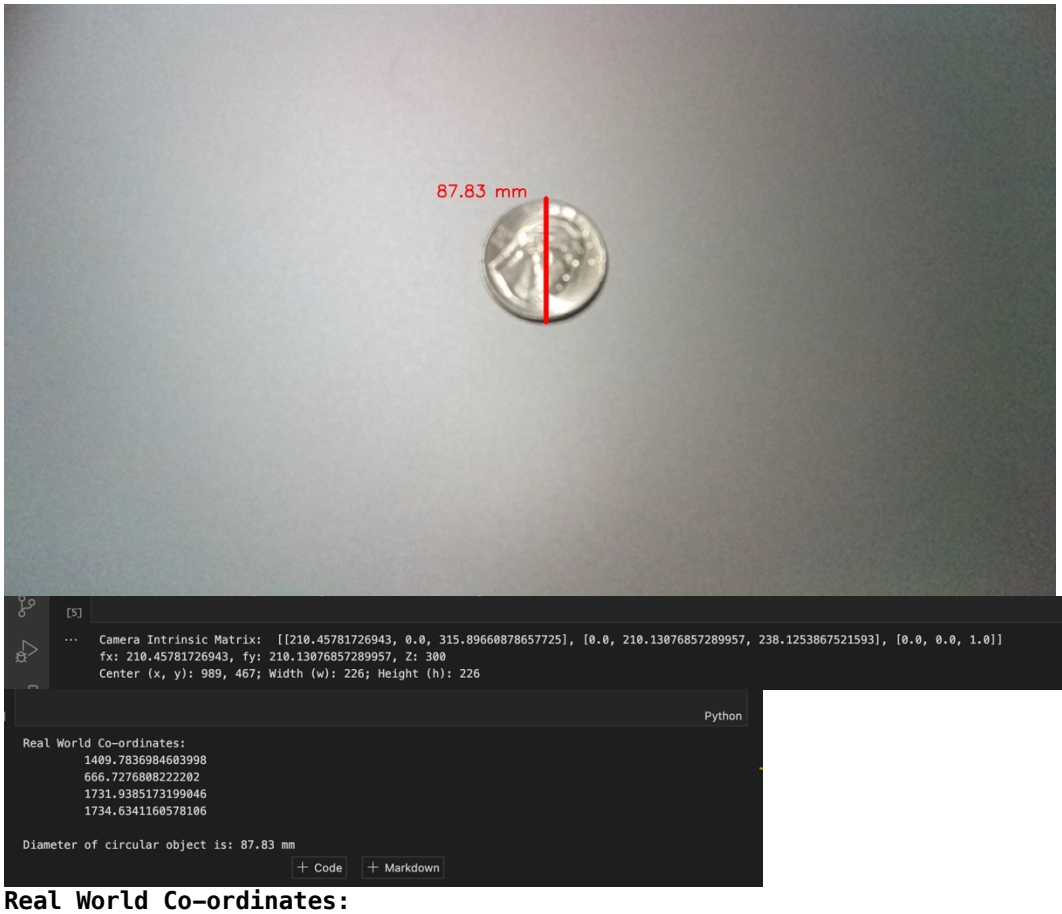


3. Write a script to find the real world dimensions (e.g. diameter of a ball, side length of a cube) of an object using perspective projection equations. Validate using an experiment where you image an object using your camera from a specific distance (choose any distance but ensure you are able to measure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

We choose to use circular objects to find real world dimensions

1. Define the Conversion Function: Put in place a function that converts inches to millimetres. Depending on which units are used in the experiment, this conversion might be required.
   2. Calculate Object Distance: Create a method called calculate_object_distance() that accepts as inputs the taken picture, the object's bounding box coordinates, the focal lengths of the camera (fx, fy), and the distance (Z) between the object and the camera.
   ● Determine the object's corners' real-world coordinates by using the perspective projection formulae.
   ● Determine the actual distance between two corners.
   ● Convert the distance to the appropriate unit (such as an inch or millimetre).
   ● Present the computed measurements on the picture.
   3. Image Processing and Validation: Open the photographed file and provide the appropriate arguments to the calculate_object_distance() function.

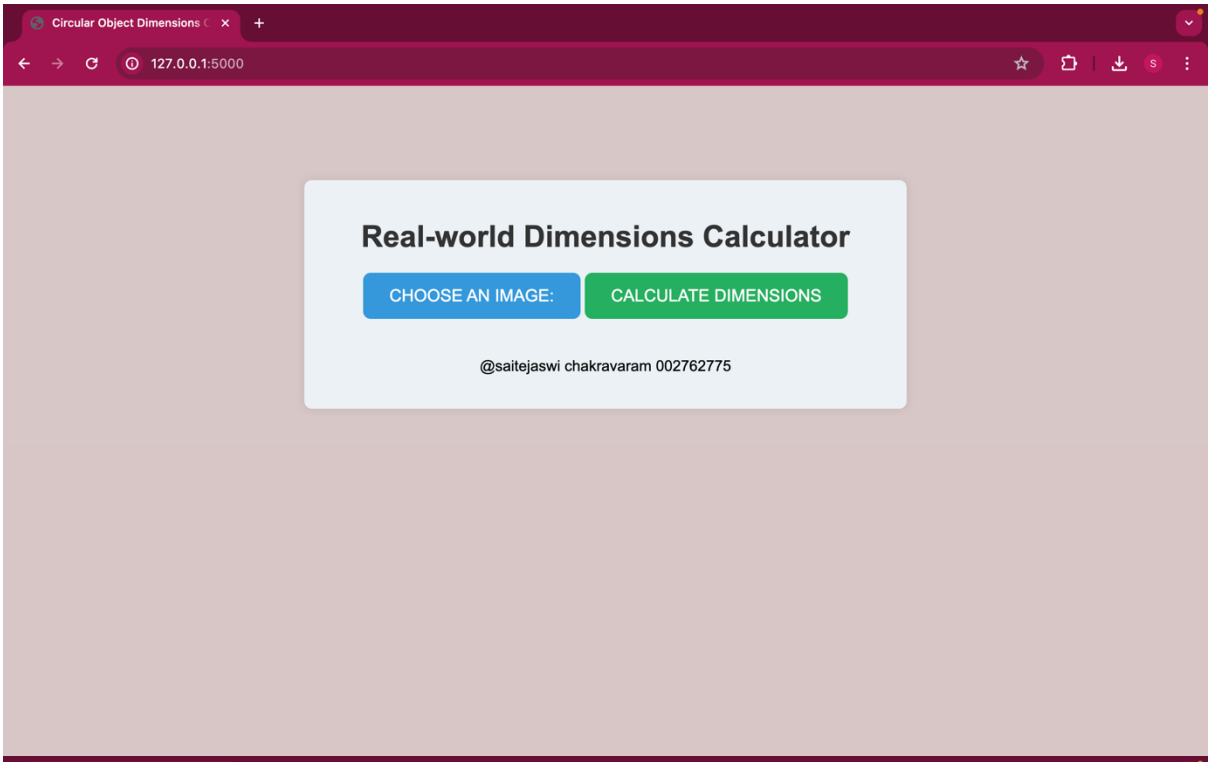
Actual diameter is 87.83 mm.

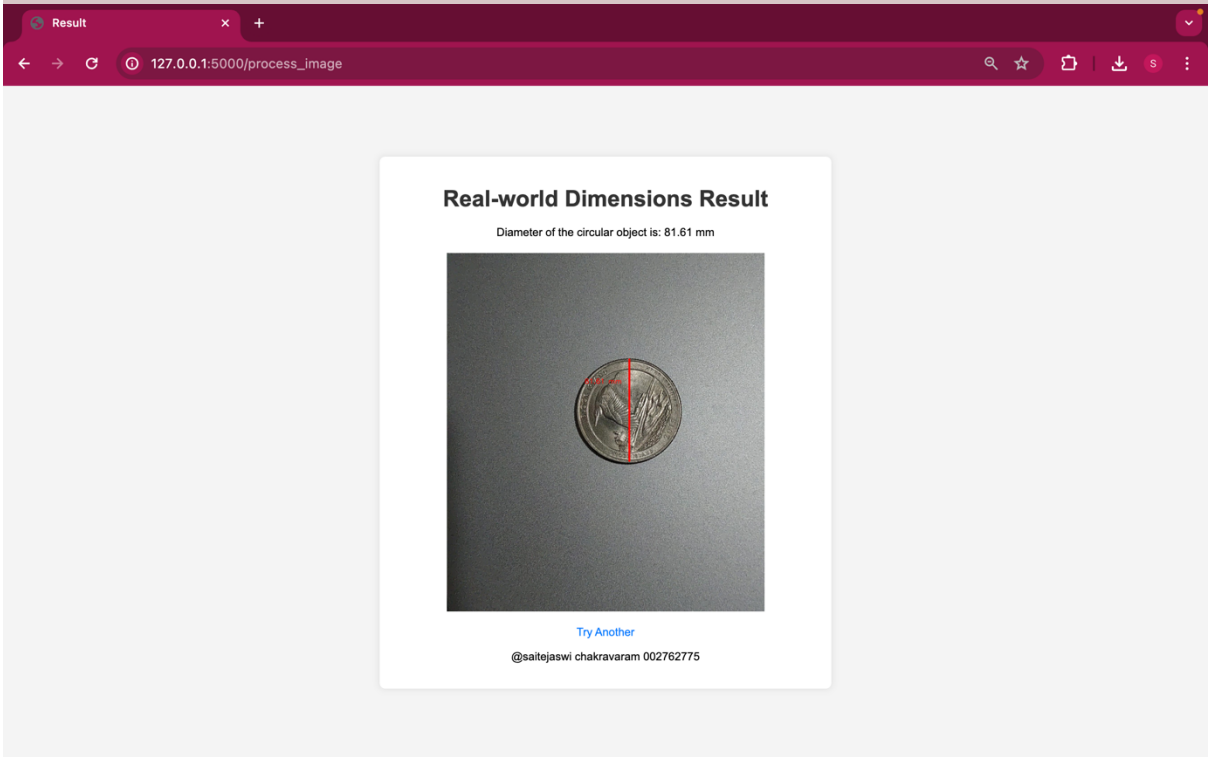


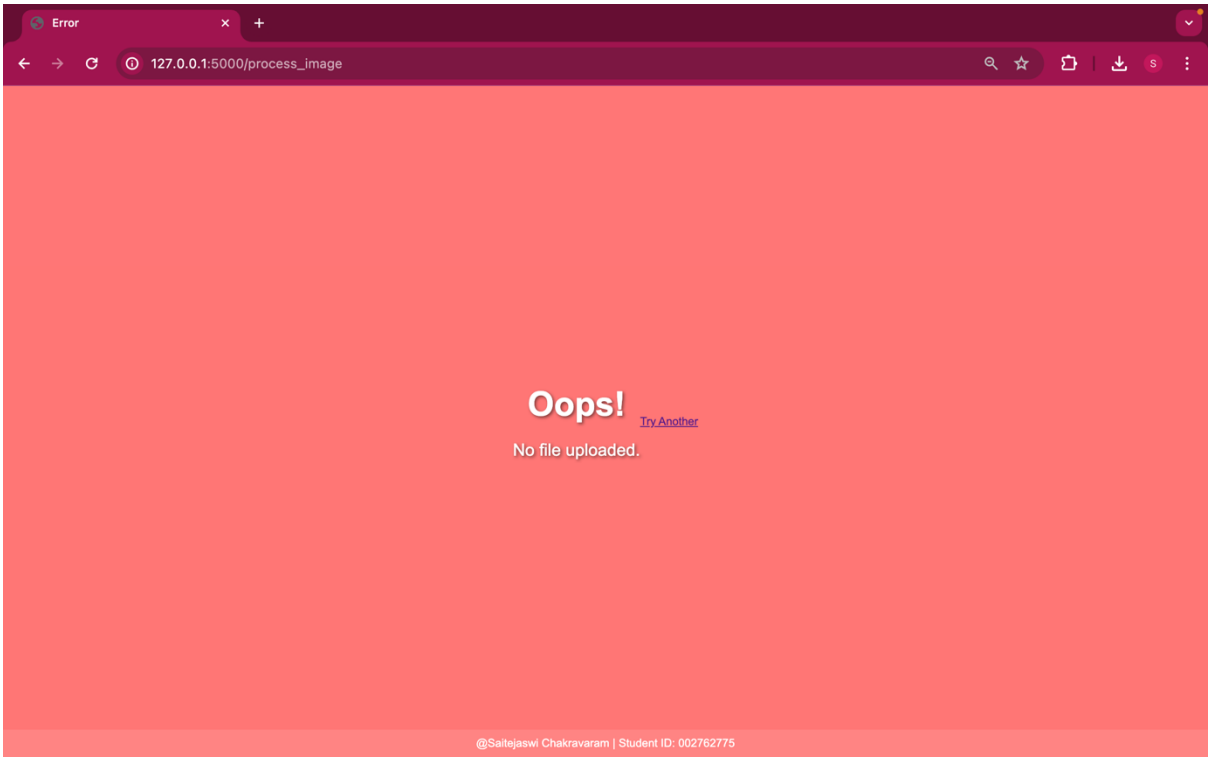

```
Real World Co-ordinates:
        1409.7836984603998
        666.7276808222202
        1731.9385173199046
        1734.6341160578106

Diameter of circular object is: 87.83 mm
```

4.Write an application – must run as a Web application on a browser and be OS agnostic – that implements the solution for problem (3) [An application that can compute real-world dimensions of an object in view]. Make justifiable assumptions (e.g. points of interest on the object can be found by clicking on the view or touching on the screen).


Circular Object Dimensions
127.0.0.1:5000
Real-world Dimensions Calculator
CHOOSE AN IMAGE:
CALCULATE DIMENSIONS
@saitejaswi chakravaram 002762775


Result
127.0.0.1:5000/process_image
Real-world Dimensions Result
Diameter of the circular object is: 81.61 mm
Try Another
@saitejaswi chakravaram 002762775



Github Link:

https://github.com/05saitejaswi/csc8830/tree/main/Assignment-1

vedio demo:
Recording link:
https://gsumeetings.webex.com/gsumeetings/ldr.php?RCID=7584b8559e94eb4a9cd7377a57dab52f