**CV ASSIGNMENT\_01**

**Hema Nikhitha Edupuganti**

**002798014**

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?

Import the required libraries i.e., cv2, time, numpy, and depthai. Now, set the parameters like resolution of RGB camera, mono camera and required dimensions that are used for resizing RGB frame. Using dai.Pipeline(), a pipeline object has to be created. With using respective resolution and board sockets, build RGB camera, left and right mono cameras. Next, build stereo depth node with including parameters such as left-right consistency check, sub-pixel disparity.

Output queues are made for RGB and disparity frames. Enter the main loop where frames are retrieved on continuous loop from output queues created. Now, using OpenCV’s cv2.imshow() function, display RGB frame and depth map frame.

Now, calculate Frames per second (FPS) and print to console. Check for a key press where, once a key is pressed, all the OpenCV windows close and the loop ends.

Output for the same can be seen below:

A screenshot of a computer

Description automatically generated

1) Report the calibration matrix for the camera chosen and verify (using an example) the same.

Use the given captureImages() and captureColorImages() functions to capture images from appropriate camera sources (right, left, or RGB) of OAK-D camera. Use the calibrate() function to find corners, calibrate and store the camera matrix, distortion vector for each camera source.

The camera matrix and distortion vector can be loaded from the saved files by using the load camera function. Utilising the imported distortion vector and camera matrix, use the undistort\_image() function to restore an example image. Using the undistorted image and known data, like the chessboard square's size and the object's distance from the camera, use the calculating calibrate error function to get the calibration error.

Output for the same can be seen below:

A screenshot of a computer

Description automatically generated

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.

Use the given captureImages() and captureColorImages() functions to capture images from appropriate camera sources (right, left, or RGB) of OAK-D camera. Use the calibrate() function to find corners, calibrate and store the camera matrix, distortion vector for each camera source.

The camera matrix and distortion vector can be loaded from the saved files by using the load camera function. Using undistort image function to undistort a sample image using loaded camera and distortion vector. We use OpenCV functions to extract image points for PnP calculation, define 3D real-world points, solve PnP to obtain translation and rotation vectors, and convert the rotation matrix to Euler angles. Output the rotation angles along the X, Y, and Z axes in degrees, also in both intrinsic camera matrix and extrinsic translation vector.

Output for the same can be seen below:

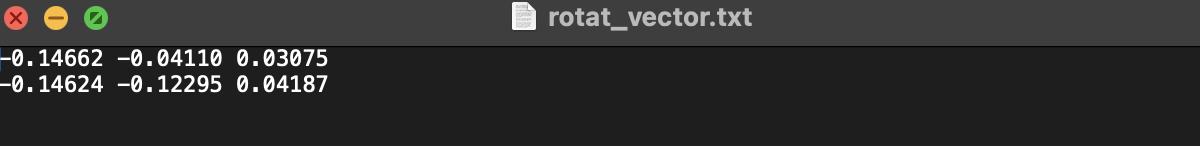
A black and white checkered board with colorful lines

Description automatically generated

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

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.

Use the camera to take a close-up image of the object from a particular distance. Be sure to gauge this distance precisely. We use object detection or manual bounding box annotation o find object’s bounding box coordinates in a picture. The real-world dimensions of the object can be determined using the calculate\_object\_distance() method by taking into account both the object's known distance from the camera and its apparent size in the image.

With the help of the supplied convert\_milli\_to\_inch() function, convert the computed dimensions to the required real-world units. We have the outcome graphically by displaying the computed dimensions on the picture.

Output for the same can be seen below:

A screen shot of a computer

Description automatically generated

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).

Firstly, create a fresh folder for your Flask project. Install Flask in this environment with a simple pip command. The primary Flask application should be created in a Python file such as app.py. Modules like NumPy, OpenCV , and Flask must be imported.

As an illustration, you may have separate routes for the home page, uploading images, and processing them. Using Jinja2 template, we create an HTML template for each route which help to specify page layouts. This involves identifying the object's points of interest by clicking on particular areas or by applying image processing methods to find features. Use the logic to calculate the object's actual dimensions based on the points of interest that were determined in the previous stage. This calls for the use of perspective projection formulae.

Give the calculated measurements and the uploaded file on the webpage. The computed values can be dynamically updated in the DOM using JavaScript. To enhance the look and feel of your web pages and to give users a better experience, use CSS styling. Ascertain the user interface is easy to understand and use.

A screenshot of a computer

Description automatically generatedA screenshot of a calculator

Description automatically generated