CSc 8830: CV Assignment-4 Solutions Report

1. Implement an application (must run on web or as an app on mobile device) using the stereo camera where it will recognize, track and estimate dimensions (at least 2D) of any object within 3m distance and inside field-of-view to the camera. You can use barcodes or text recognition tools for identification. However, the entire object must be tracked (not just the barcode or text). Machine/Deep learning tools are NOT allowed.

QR Code Detection:

- The application detects QR codes in the stereo camera's frames using the cv2.QRCodeDetector() class.
- Detected QR codes are outlined with rectangles, and their decoded information is displayed on the frames.

Object Tracking:

- Object detection and tracking are performed using background subtraction.
- Detected objects are represented by bounding boxes, and their movement is tracked between consecutive frames.

Object Dimension Estimation:

- Object dimensions are estimated based on the detected bounding boxes.
- The application calculates the dimensions (width and height) of objects in centimeters using the Euclidean distance between midpoints of the bounding box edges.

Stereo Vision Display:

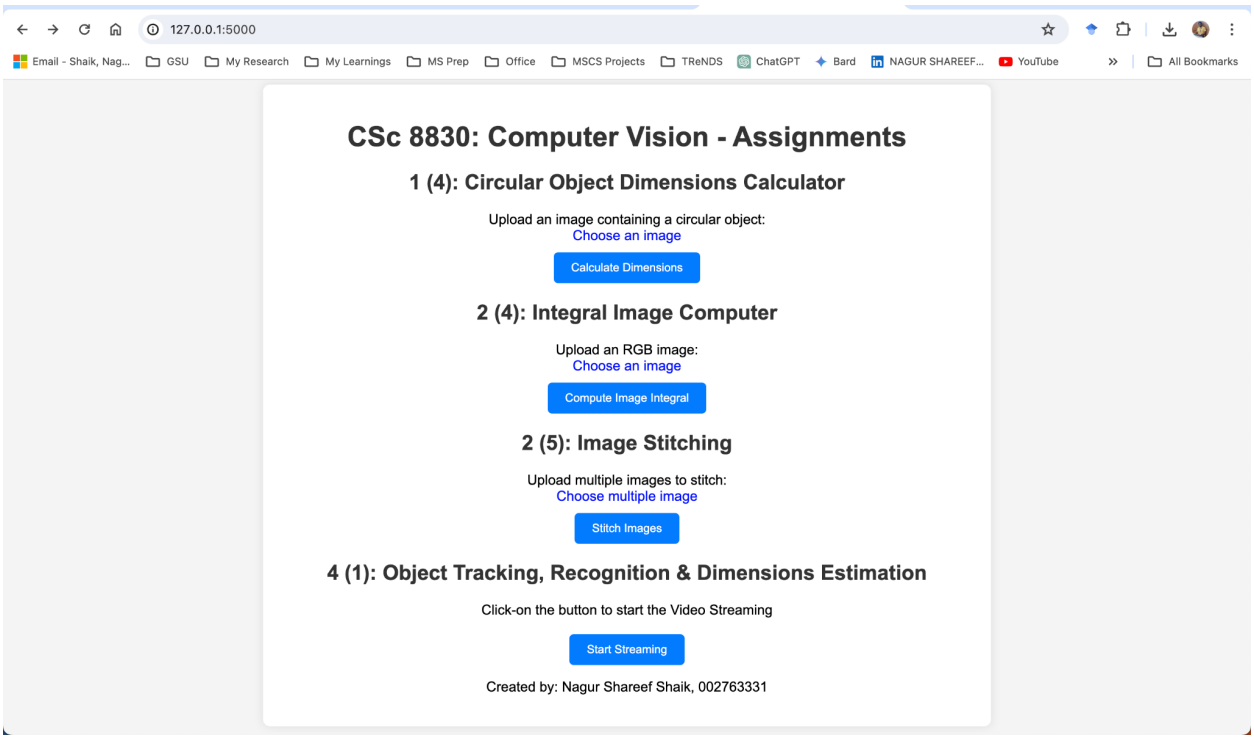
- The application displays the stereo vision (left and right camera frames) with QR code detection and object tracking.
- Additionally, it displays the RGB camera frames with object dimension estimation.

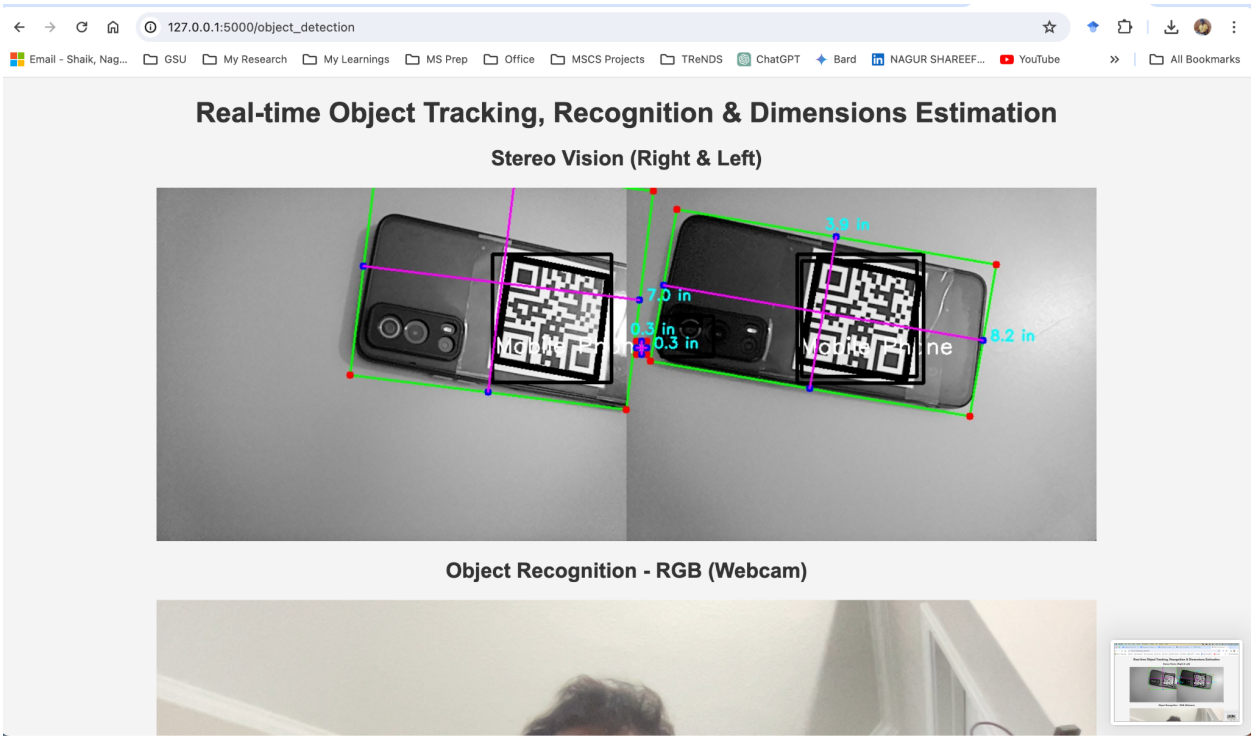
Pipeline Setup:

- The DepthAI pipeline is configured to obtain stereo camera frames and RGB camera frames.
- Separate output queues are defined for depth, left camera, right camera, and RGB camera frames.

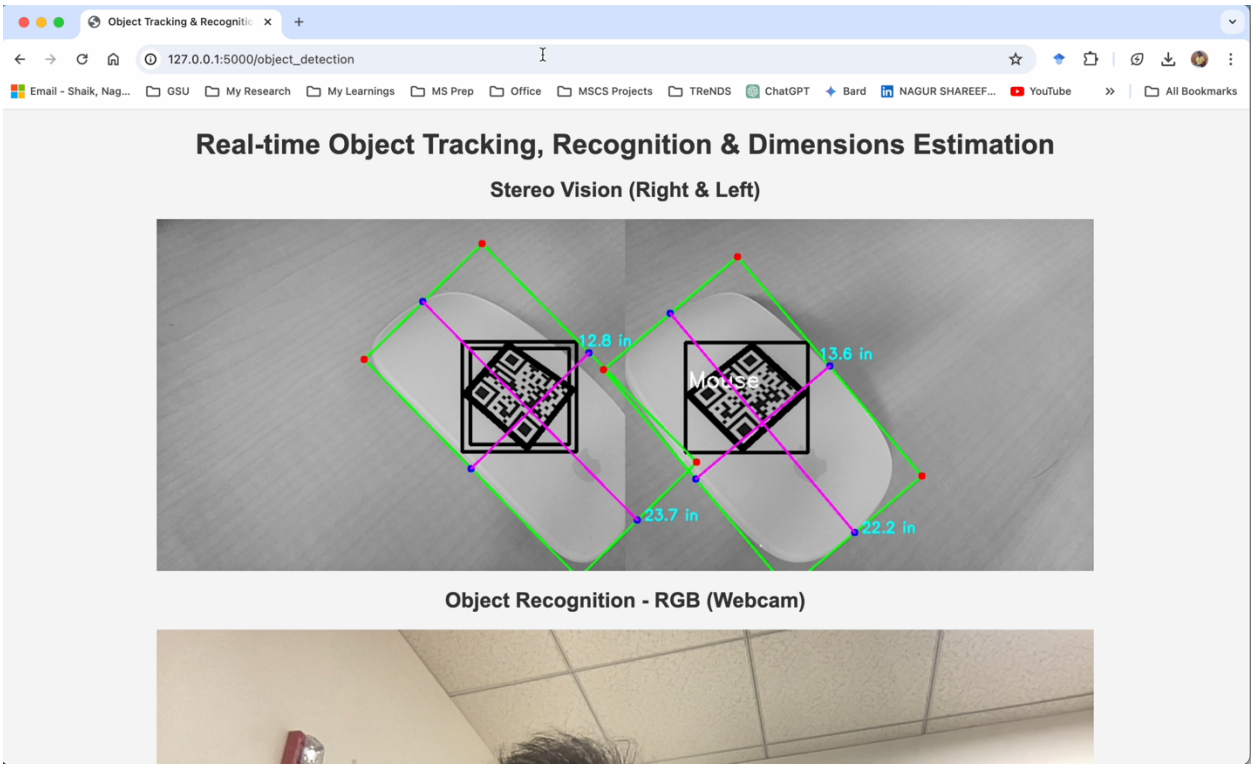
Main Loop:

- In the main loop, frames from the stereo camera and RGB camera are obtained from the output queues.
- QR code detection, object tracking, and object dimension estimation are performed on the frames.
- The processed frames are displayed in separate windows for visualization.





From the above output we can see Mobile phone is recognized, tracked completely & its dimensions were estimated in inches.



From the above output we can see Mouse is recognized, tracked completely & its dimensions were estimated in inches.

2. Use the DepthAI SDK or use ORB3-Visual SLAM (https://github.com/UZ-SLAMLab/ORB_SLAM3) to execute the scripts on your depth camera and run experiments in two different locations. Provide snapshots of your SLAM output and what limitations/corner cases do you observe.

Steps:

- I followed the instructions provided in the DepthAI documentation (https://docs.luxonis.com/en/latest/pages/slam_oak/) to clone and execute the repository for DepthAI-SLAM (https://github.com/bharath5673/depthai-slam).
- This involved setting up the necessary dependencies and running the scripts on my depth camera.

Limitations/Corner Cases:

1. SLAM algorithms often struggle in dynamic environments where objects or people move around. These movements can disrupt the mapping process and cause the system to lose track of its position.
2. Environments lacking distinctive features or textures can be challenging for SLAM algorithms, as they rely on visual features for localization and mapping. If the environment is too uniform, the system may struggle to create a reliable map.

Repo: https://github.com/ShaikNagurShareef/CSc8830-Computer-Vision/tree/main/Assignment-4