

RT Systems Lab

**Weekly Report number (0)**

Julian Ewaied 215543497 Namir Ballan 326165156

Contents

No table of contents entries found.

# This Week's Tasks

* Understanding the problem
* Building minimal coding environment
* Understanding the data formats and choosing one
* Running sample code

# Tasks Done

## Understanding the project

We analyzed the used methods for 3D mapping using a camera and a raspberry pi microprocessor. We understood the computational and physical limitations we have, and chose our project.

Our project focuses on building a 3D model of a rooms using only pictures in which we know where the camera vertically is, and therefore using two or more locations in different frames of the same point using motion vectors to match (which is fast since it is done in the hardware), we can calculate the coordinates of a point, and hence build a depth map of the room as a points cloud as we will be discovering more details later on. Notice that this project has been implemented previously, but didn’t use MV's effectively as we will do to approximate the numbers. More details to be explained in later reports.

## Running a sample code

We are planning on using C++ to code, but since most modules which support raspberry pi are in python, we decided to first try to use the already existing python interface thanks to Bar, Arthur, and David in the lab. They offered us actual data from the lab using a drone, and a python code to try see how things work in the project with simple interfaces. We had a 3D coordinates table of all the feature points, and had to make a top-view 2D map of the room. With simple coding we managed to do that- here's the code we wrote:

def topdown\_view(depth: np.ndarray):

    depth[:, 2] = np.clip(depth[:, 2], 0, 700)

    depth[:, :2] = np.squeeze(cv2.undistortPoints(depth[None, :, :2], cam\_mat, dist\_coeff))\*depth[:, 2:]

    # empty picture

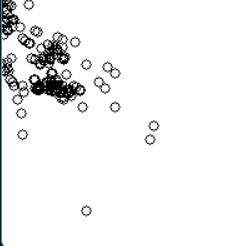
    pic = np.full((350,350,3), 255,dtype=np.uint8)

    for d in depth:

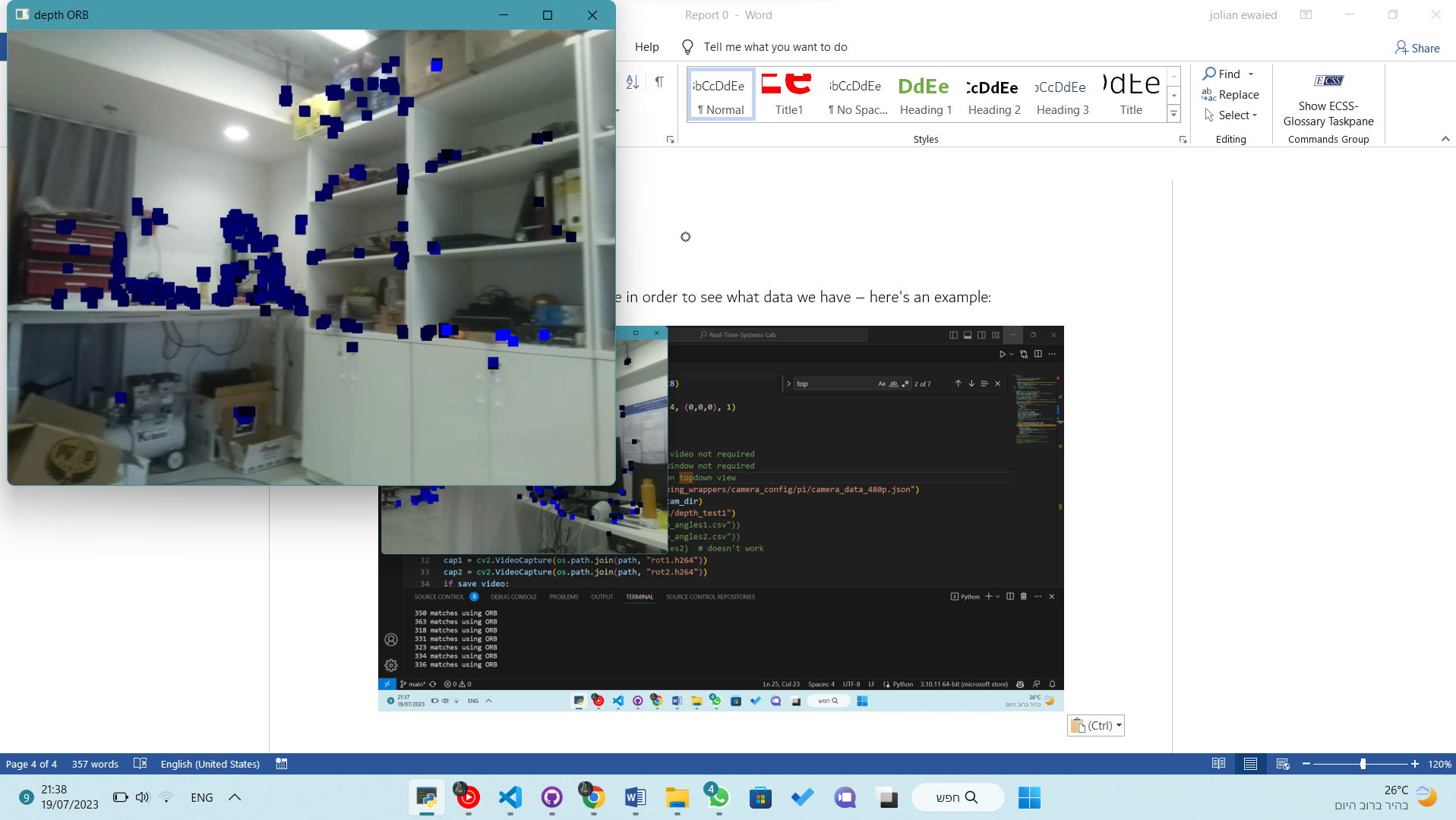
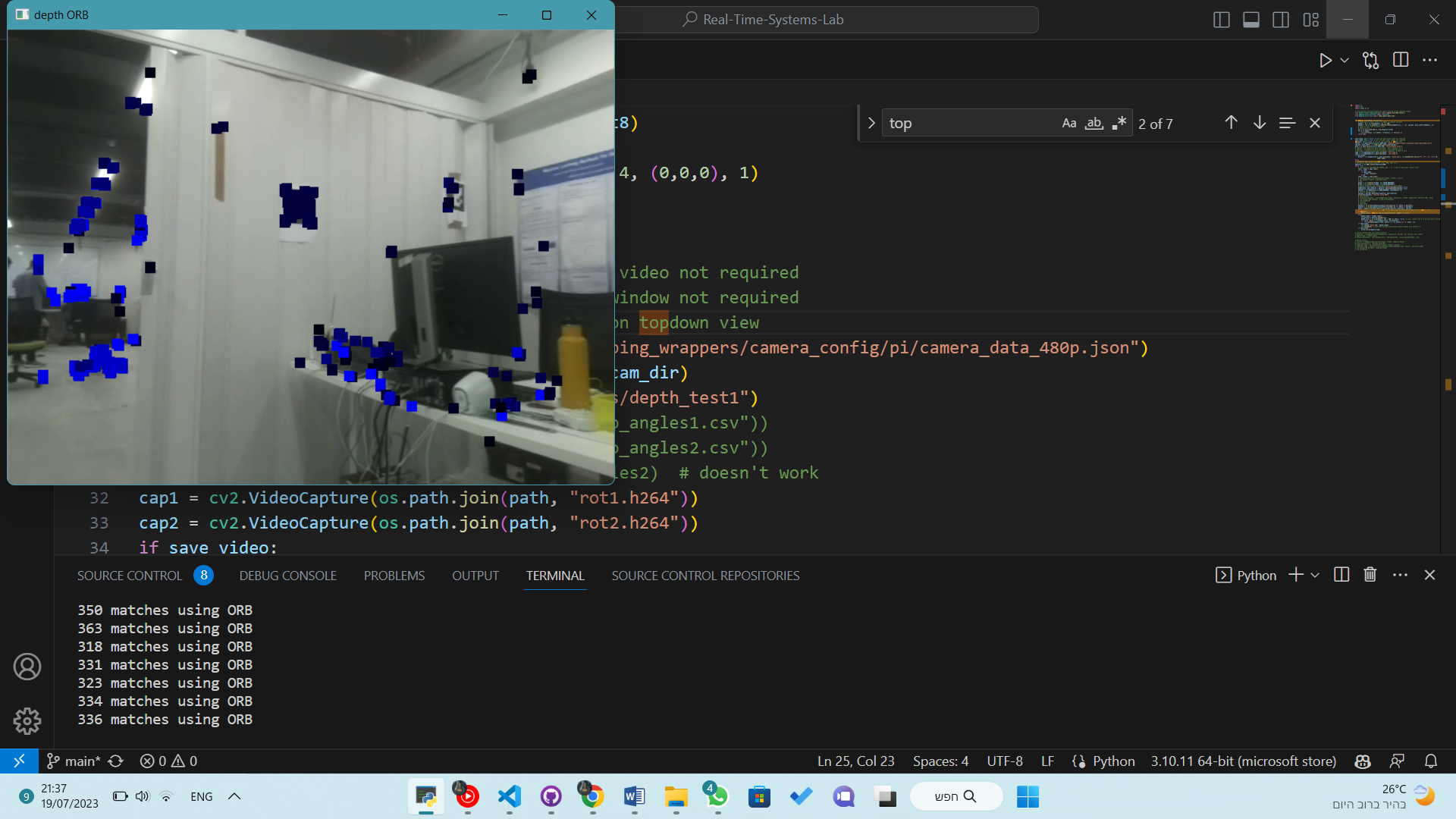
        cv2.circle(pic, (int(d[0]), int(d[1])), 4, (0,0,0), 1)

    return pic

the results are not very promising:



We also ran an already-existing code in order to see what data we have – here's an example:



## Understanding code formats