UNIVERSITY OF HAIFA

אוניברסיטת חיפה جامعة حيفا

RT Systems Lab

Weekly Report number (0)

Contact Us

Julian Ewaied 215543497 Namir Ballan 326165156

7/21/2023

Contents

This Week's Tasks

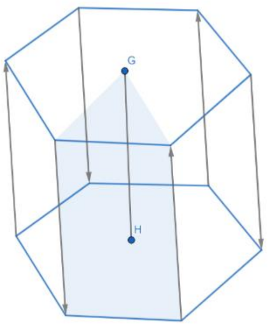
- Understanding the problem
- Building minimal coding environment
- Understanding the data formats and choosing one
- Running sample code
- CSV files reader class implementation

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 video frames in which we know vertical camera position, and therefore using multiple disparity matrices (using motion vectors which is fast since it is done in the hardware), we can calculate the (x, y, z) coordinates of a point, and hence build a depth map of the room as a points cloud. Notice that this way we get more instances of the same point, and therefore more information, with less resolution (a tradeoff). The path of the drone is the key change – it will scan a particular angle vertically, then rotate, and scan another angle, and so on. Look at the diagram for better explanation.



The drone moves along axis GH. Up and down, where after every vertical movement, it rotates by a certain angular displacement $\Delta\theta = \frac{360°}{n}$, where n describes the resolution we need. Notice that each vertical axis gives us a points cloud, and we need to connect them all together, and that we will leave for other teams to do.

In case this new path works, we might also test spiral path which is a bit more complex yet more powerful, depending on how much time we have left.

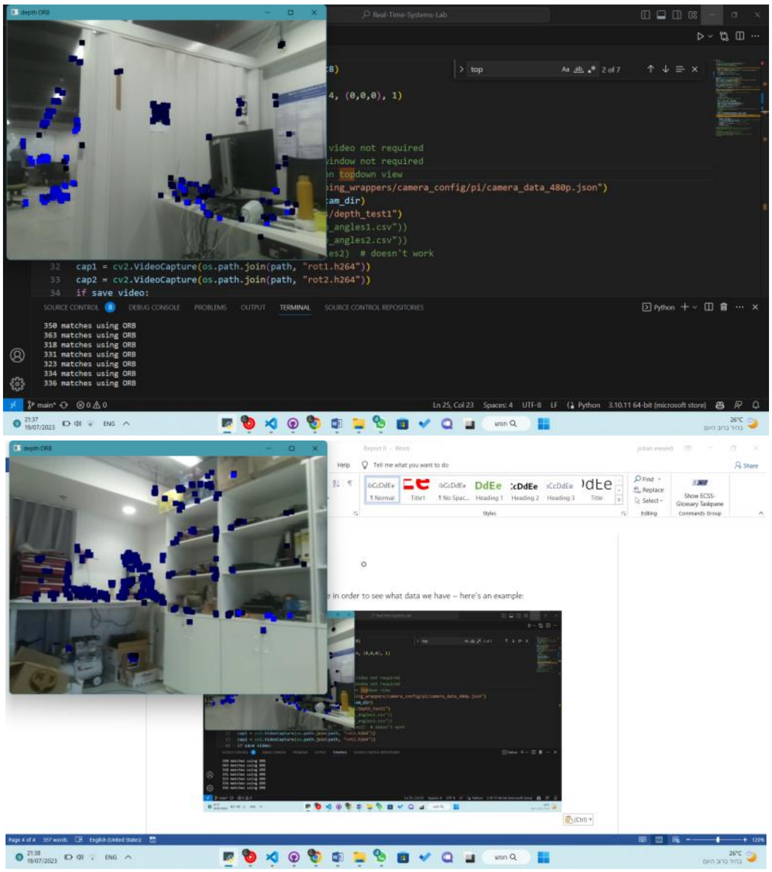
Running a sample code

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. We had 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[2])), 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 files formats

At first, we tried to work with HDF5 files. We found it very complex for our purpose, so we decided to use a library called cnpy with npy files which required a header file called zalib.h, which was not found on our machine. After a long fight with the code we decided to convert the data into csv files. We wrote a python code that takes a path to an .npy file, and using python API writes the relevant data into the csv file in the line format: d_x, d_y, SAD.

Building minimal environment

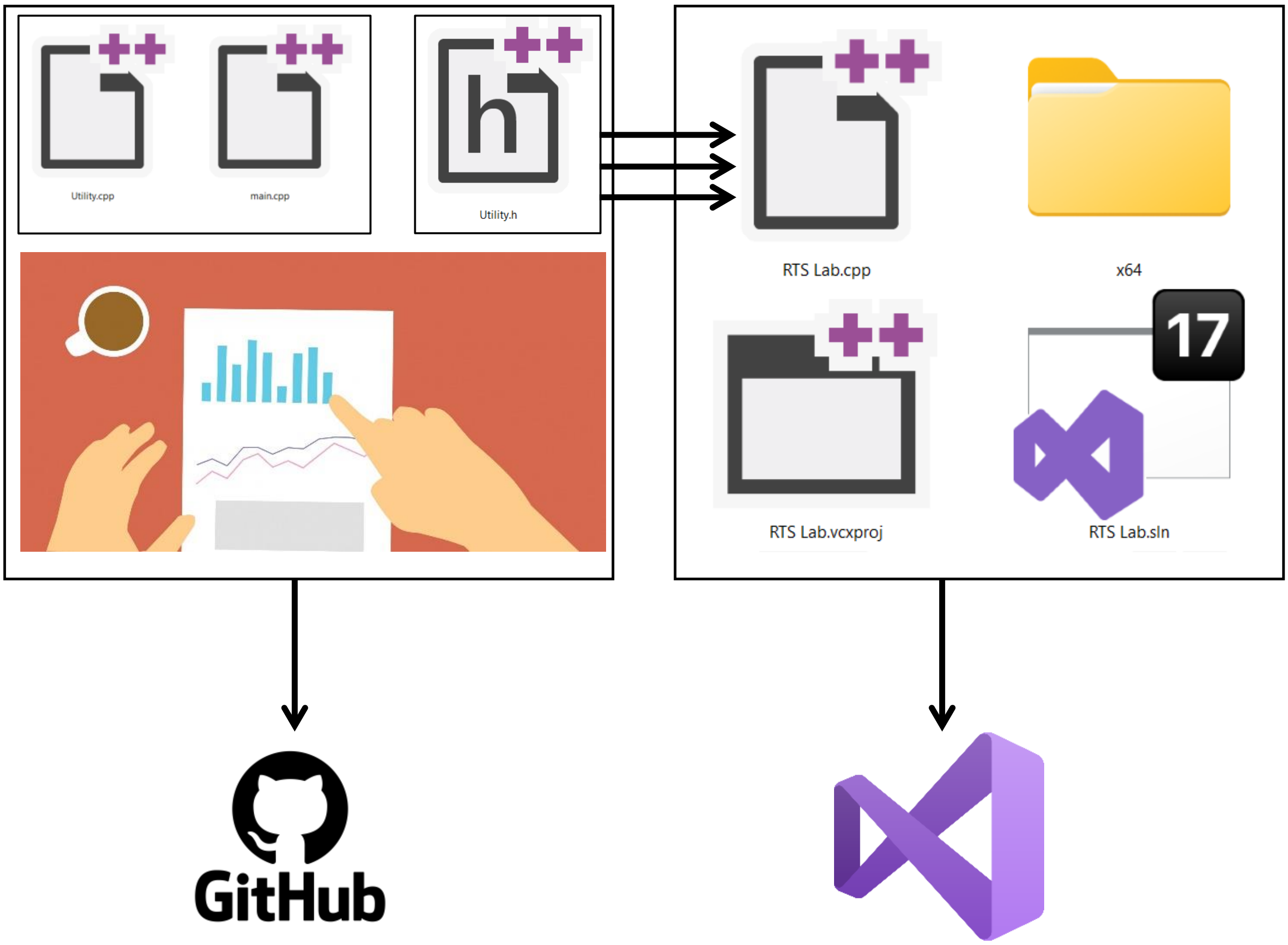
We tried to run the C++ code, but it didn't work in visual studio environment, so we had two options:

a) To use a complex shell code and .json file to use some C++ compiler we don't have yet.

b) To build a .sln file in the repository, which makes a lot of irrelevant files

Eventually we did a clever solution to combine both. Here's a diagram that explains the concepts.

This way, only the relevant code and the reports are transmitted to the Github repository, while the



building files are put in a different directory, which has its own main file (RTS lab.cpp in this case), which calls for the main function in the repository, and this way we can run the code easily with visual studio, and not get the build files. Further information about how to install the repository can be found in the readme file in the repository. Here is the repository link.

Github Repository GitHub

Implementing CSV reader class

We implemented a simple C++ object-oriented interface in which we have a file object and a few operations to do on it:

- Opening the file
- Closing a file
- Changing path
- Reading all the file
- Reading a specific range of frames from a file

Notice that every contiguous amount of lines represents a matrix of motion vectors for all the macro blocks between two frames, so we return a vector of matrices of pairs (d_x, d_y, SAD). Notice that this data structure might change in the future, but for the current time it is enough.

Next Week's Tasks

- Understanding the camera matrix.
- Understanding the trigonometry to calculate the points.
- Understanding video files format and how to extract the frames.
- Installing Eigen library for matrix calculations.
- Installing opencv library for C++.
- Implementing video files interface.