UNIVERSITY OF HAIFA

جامعة حيفا אוניברסיטת חיפה

RT Systems Lab

Weekly Report number (4)

Contact Us

Julian Ewaied 215543497 Namir Ballan 326165156

8/14/2023

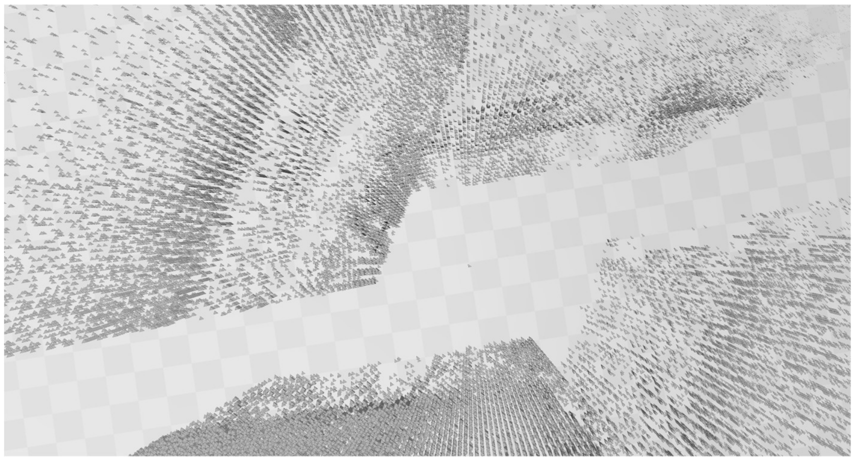
Contents

This Week's Tasks

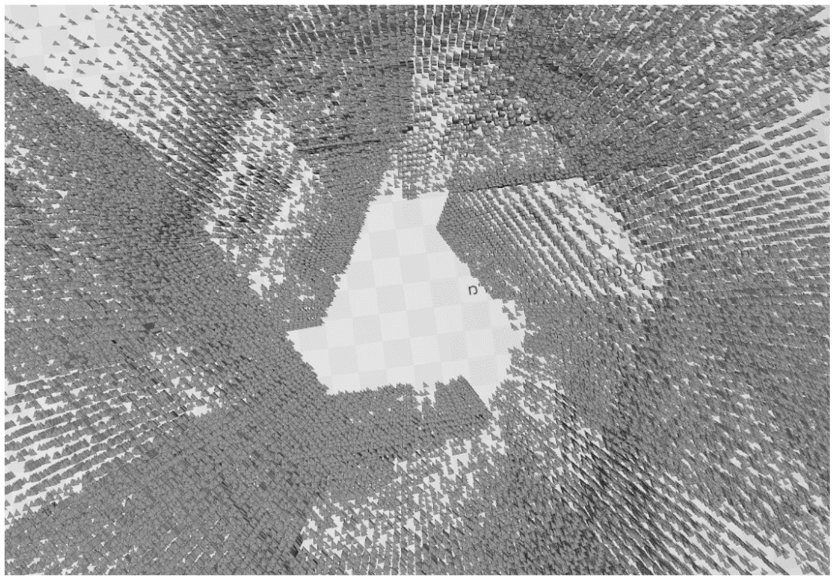
- ☞ Checking the rotation is done well
- ☞ Matching between two representations of the same point in two different frames
- ☞ Making the code more efficient
- ☞ Automating the filtering policy

Checking Rotations

I started by looking at the 3D representation as it is more flexible, and noticed something odd in the output when we zoomed in:



There were two "holes" in the room outline, meaning the rotations were done wrong, and when I checked, I found out I rotated the points from the file after I added it to all the points from pervious files. I fixed the bug and got the following results.



At first sight, this is extremely wrong, but by far I have recognized a pattern in the data – way too many outliers. Therefore, I used SAD as a filtering argument in the interval $[a, b]$. I did a binary-search to find the best $[a, b]$, and found the "best" SAD range of $[100{,}125]$. Bar claimed this is some sort of overfitting, therefore we need to run on other rooms in the future. Either ways, we got the results:

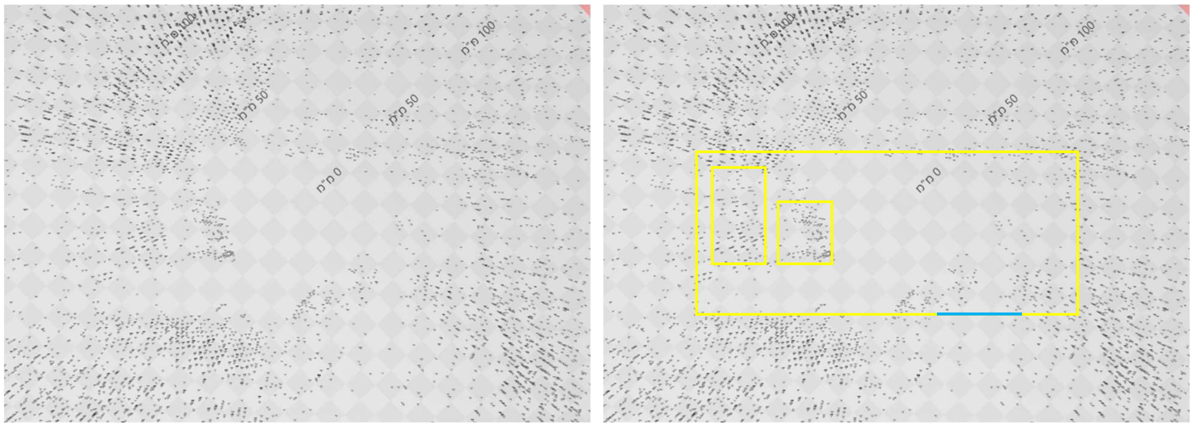
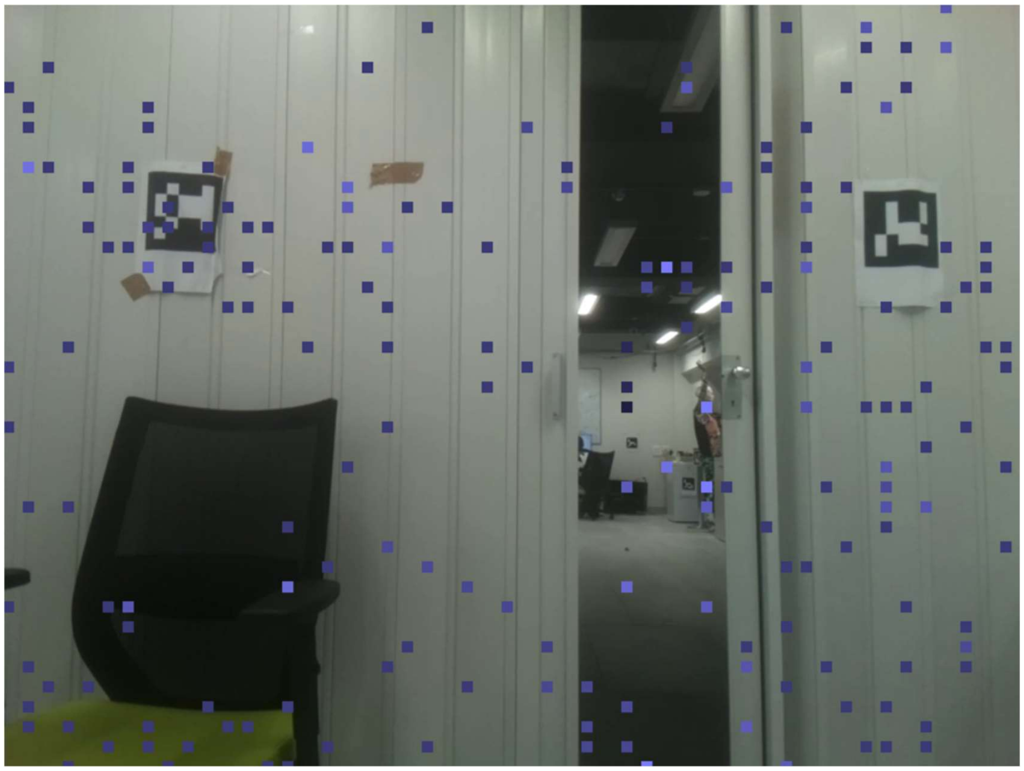


Figure1 : the results (left image), and how they should be interpreted. The yellow rectangles are a chair and a table that appeared in one of the videos. We can still see effective noise next to the entrance, which, Bar claims, is the result of the floor since it's of an almost uniform color. We will be looking for solutions soon.

At the end, we attach a picture showing how the floor is problematic. We can see the floor is quite bright, meaning its interoperated as close points.



Automating Filtering Policy

We considered multiple options, of which we mention the following:

(1) Manually deciding on a bounded interval of SAD values, which we found hard.

(2) Deep NN on all the parameters of each point: dx, dy, SAD and z, where will take another room's data as a training data, find the optimum supervised by a human-made selection.

(3) Filtering out all points of low heights, which didn't really work since we lost too much data.

(4) Replacing every "cube" of points with their median by dividing the space into equal cubes.

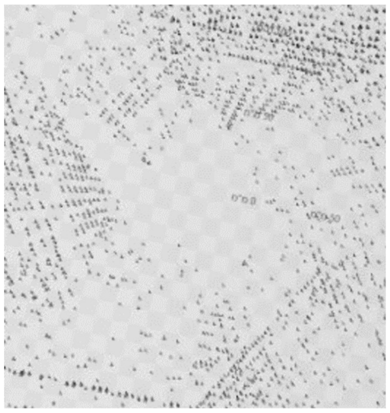


Figure2 : the top-down map after filtering low-height points.

(5) Filtering the floor and white walls data using the block-average algorithm described below.

Block Average Algorithm

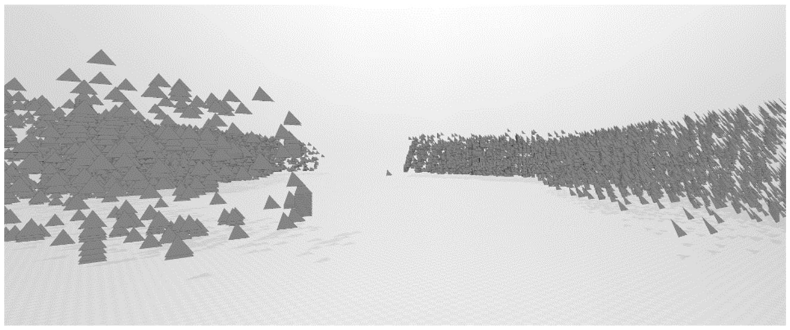
For each macro-block y in the image:

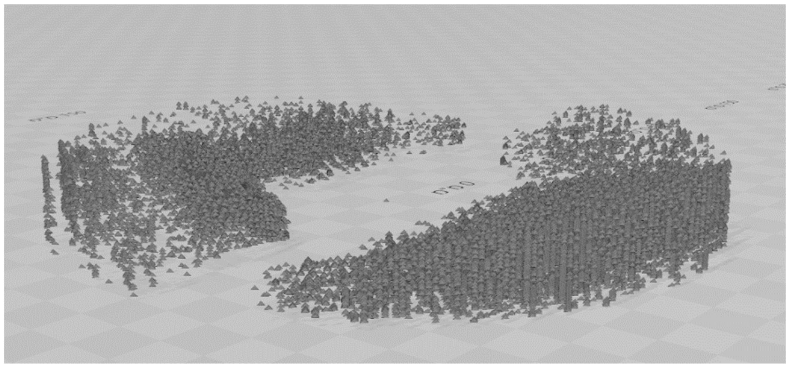
1. sample 16 arbitrary (and not random) points, for instance, the diagonal of the macro-block.
2. Average their colors values: if $c_1, \ldots, c_{16} \in [255]^3$, calculate $\bar{c}_y = \frac{1}{16}\Sigma_{i=1}^{16} c_i$.
3. If $\frac{1}{8}\Sigma_{x\in\Gamma(y)} \left|\left|\bar{c}_x - \bar{c}_y\right|\right|_2 \leq f$, where $\Gamma(y)$ is all blocks adjacent (horizontally, diagonally, and vertically) to y, and f is some global constant, get rid of the point resulted by the center of y.

This is a useful filtering algorithm, since if two blocks have almost the same color average, meaning they're useless, the difference between their color-averages is very small, and therefore we get rid of them. Obviously, this algorithm is not very robust to blocks where all their neighbors are of some color average, and only one is different.

Matching Points

We can notice that points that we get are points with a shifted height (y coordinates). This is problematic in 3D, so we had to add the points height to y-coordinate of them. In addition, we filtered the points into a circle of radius $10\sqrt{3}m$; here are the results (we can see the walls of some height).





Note: these results are from before we fixed the bug in the rotations.

Making the code more efficient

We decided that the code is sufficiently efficient in terms of running time (average running time is 0.2 seconds in my computer, and 10 times slower in raspberry-pi, so approximately 2 seconds), and is the most efficient in terms of kernel time (we read each file only once, so we do the minimal number of necessary queries). Therefore, I will be taking this task off the list with no further improvements, for the current time.

Next Week's Tasks

☞ Implementing Average Color Filtering Algorithm

☞ Implementing Randomized Color Filtering Algorithm

☞ Implementing the Median Compressing Algorithm

☞ Getting Data from another room

☞ Searching up DNN libraries for C++.