DTM assignment 4

Qu Wang, 4700686, Pablo Ruben, 4273818

22 Jan. 2019

1 Questions

1. Does it matter how you pick your seed points, ie. is there a better way than to randomly pick the seeds?

   Randomly picking seeds was implemented in the code submitted as it was found to be rather sufficient. Of course, there might be some problems - for instance if some of the seed points have already been classified (therefore, a check is performed):

```
if SeedInd[reg_index] in region_dict :
    reg_index += 1
    continue
```

   Another more problematic aspect is if one of the planes that should be identified does not have any seed points. Therefore, once the first set of seeds has been visited, the non-classified points are used to create an additional set of seeds:

```
if reg_index == len(SeedInd)-1:
    print("looking for new seeds")
    unclassified = []
    for index in range(0, len(pts)):
        #print("looking for orphan points")
        if (index in region_dict) == False:
            #print("adding", index)
            unclassified.append(index)
    SeedInd+=list(range(0,len(unclassified),math.ceil(jparams['minimal-segment-count']/eject_trigger)))
    eject_trigger+=1
```

   Another approach for selecting the seed points might have been to compute some additional attributes such as curvature first. This approach was not chosen as it seemed a rather big computational effort to fit a plane for each point and its neighborhood.

2. During region growing you can determine the neighbours of a point using 1) a fixed radius search and 2) its k nearest neighbours. What works best?
   In short, both methods were found to work accurately. Of course, tweaking needs to be performed separately for both separately but comparable results can be reached in both cases.
   For the two optimised result, they both work fine for roofs, and knn is better on trees and edge of roofs (see figure 1 for details, figure 5 and figure 7 for overall result.)
   One difference that was observed is that if a too small k is used for the knn, segmentation is still performed, although leading to oversegmentation (see figure 2, 7 for different k, and figure 3, 4 for different r).

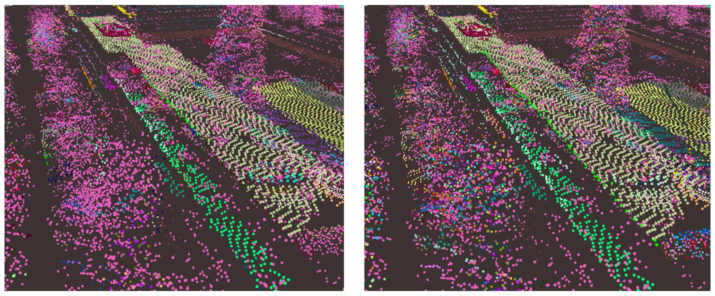

Figure 1: Left: optimised knn result; right: optimised radius search result

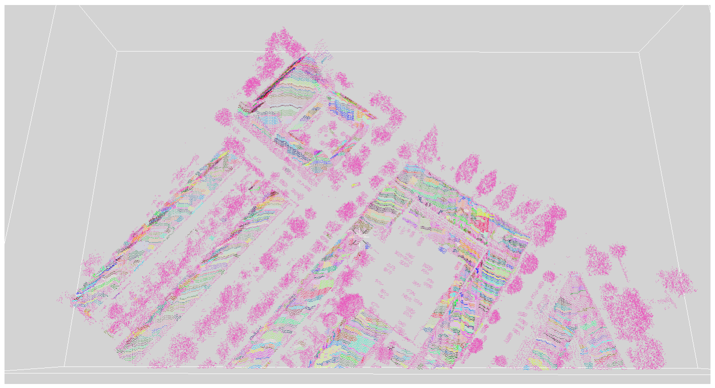

Figure 2: Example of knn segmentation achieved with a k setting of 3 (epsilon:0.1, min count:15, thinning:2)

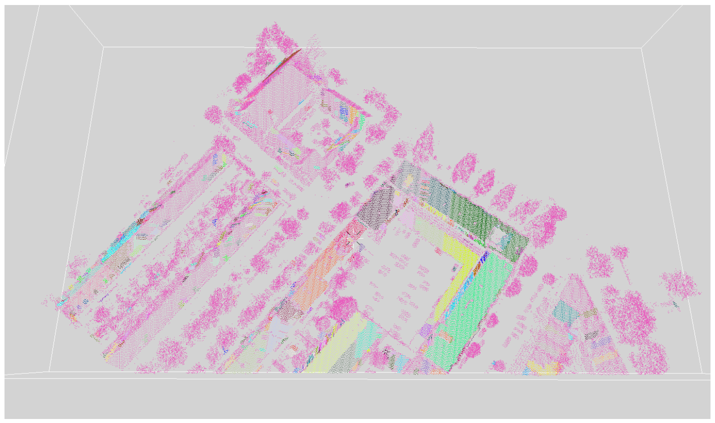

Figure 3: Example of radius segmentation achieved with a radius setting of 0.5 (epsilon:0.1, min count:15, thinning:2)

3. How does the thinning factor affect the segmentation result?
   Yes, the thinning factor does definitely affect the result of the segmentation. This can mainly be observed for planes that were identified with few points, which are already on the edge of the minimum count when performing the segmentation. A good example can be found on the bottom right of figure 4 and 5.

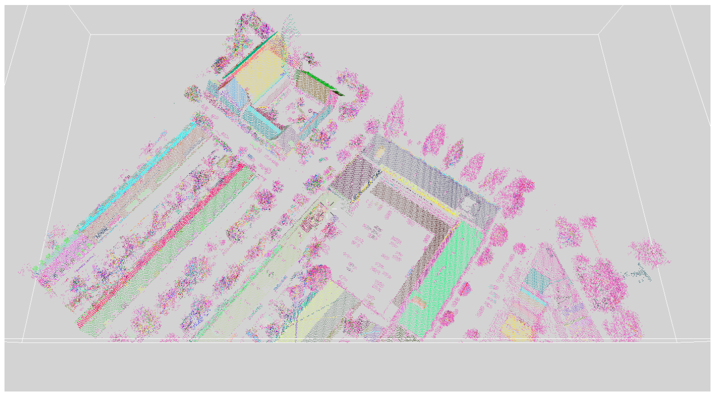
Figure 4: Example of same segmentation with a thinning of 2 (radius: 2, epsilon:0.1, min count:15,)

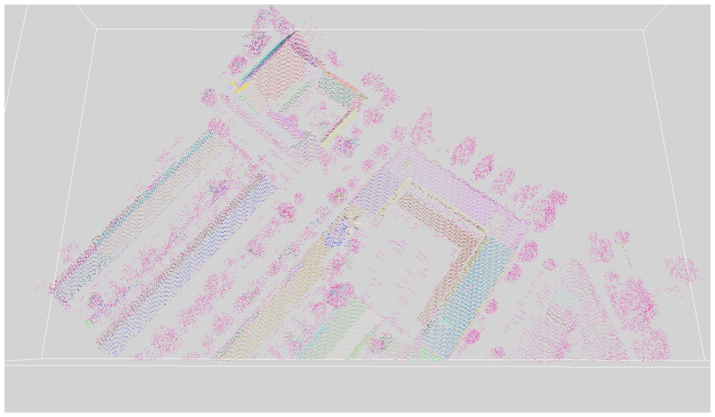
Figure 5: Example of same segmentation with a thinning of 5 (radius: 2, epsilon:0.1, min count:15,)

4. Do you get better results if you re-estimate the plane to all the points in the region as it is growing?
   The results are indeed improved if the plane is recalculated. However, as the plane estimation is computing intensive, it was decided to perform it only when 20% had been added to the region being grown. Furthermore, also for computational efficiency reasons - it was decided not to check the existing points of the region for their distance with the new plane and remove them if required.

```
if len(curr_region)>last_plane_calc*1.2:
    #print("recalculating plane")
    seed_neighbs = []
    for n_ind in curr_region:
        seed_neighbs.append(pts[n_ind])
        normal_v, plane_pt = fit_plane(seed_neighbs)
        last_plane_calc = len(curr_region)
```

   An example of the segmentation obtained in figure 4 without plane re-estimation can be found in figure 6. One can clearly observe that recalculating the planes does prevent over-segmentation.

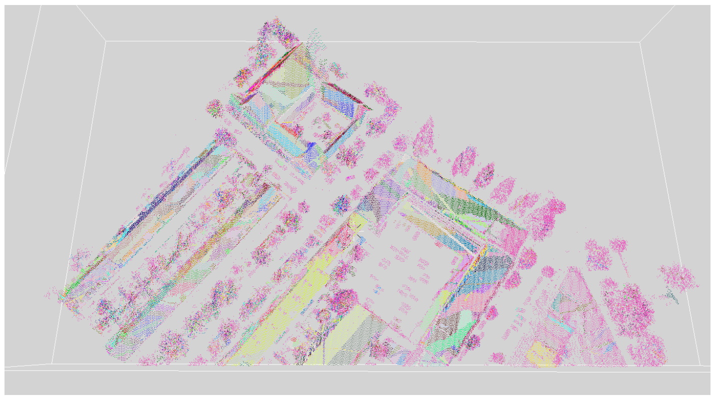

Figure 6: Example result without plane re-estimation (radius: 2, epsilon:0.1, min count:15, thinning:2)

5. Do you think you can improve your results if different/additional criteria are used for the valid_candidate() function?
   Ideas for improving the valid_candidate function might be:
   - to calculate the curvature of the point to check if the plane indeed belongs to a plane. - however, the first option and principal components analysis is well known to have only limited noise resistance and giving wrong values on edges.

6. Why are the parameters that you give in params.json the best?
   Basically, it is all a matter of balance. The optimal values used in figure 4 (radius:2) for radius usage and in figure 7 (knn:15) for knn usage were obtained by tweaking the settings and checking the result (, they both have epsilon:0.1, min count:15, thin-ning:2). This process is illustrated by the figures 8, 9, 10,11, 12, and 13 below:

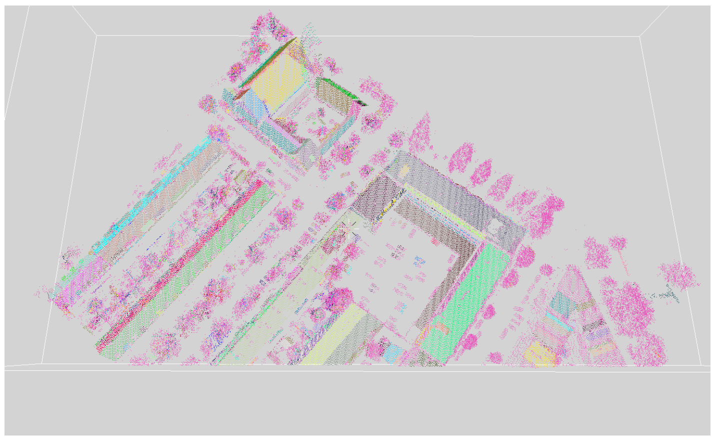

Figure 7: Result of the settings deemed optimal for KNN (knn: 15, epsilon:0.1, min count:15, thinning:2)

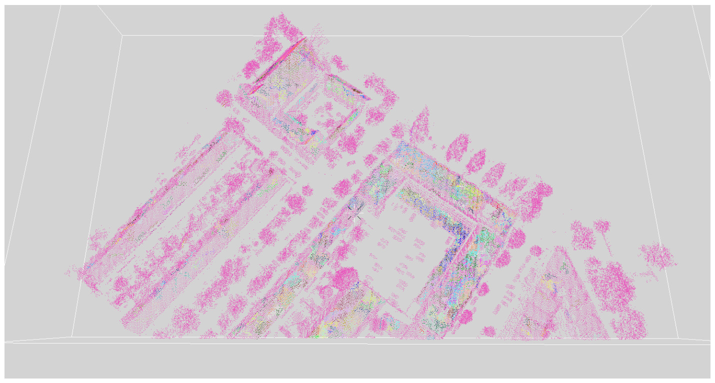

Figure 8: Result obtained for too strict epsilon value (knn: 15, epsilon:0.005, min count:15, thinning:2)

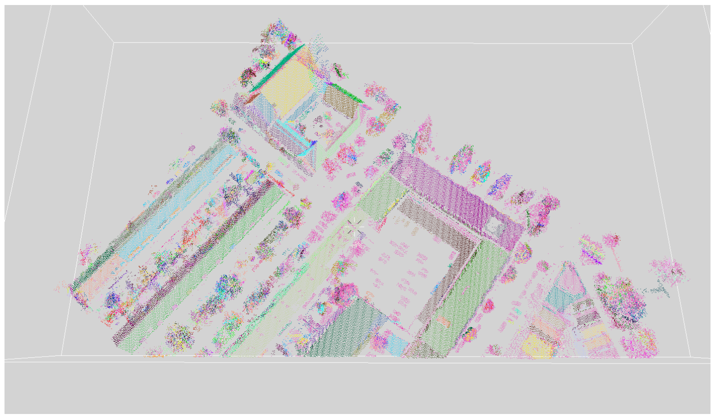

Figure 9: Result obtained for too tolerant epsilon value (knn: 15, epsilon:0.5, min count:15, thinning:2). This leads to a lot of non-existant planes being detected in trees.

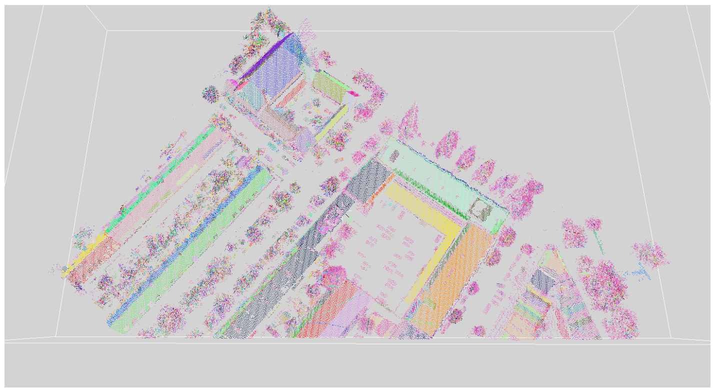

Figure 10: Result obtained for a too low minimum segment count (radius: 2, epsilon:0.1, min count:5, thinning:2). This leads to over-segmentation.

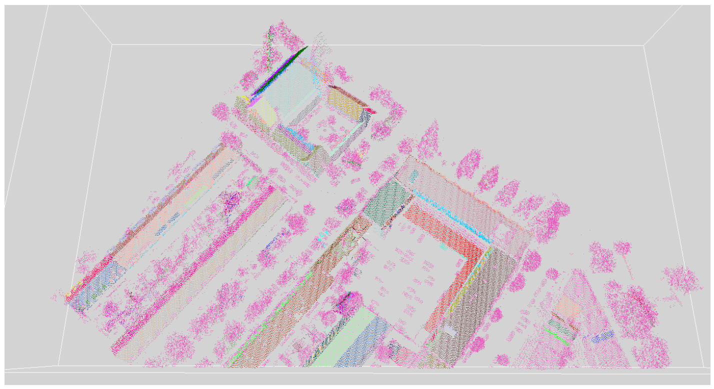
Figure 11: Result obtained for a too high minimum segment count (radius: 2, epsilon:0.1, min count:50, thinning:2). This leads to relatively many planes being rejected due to having less than 50 points.

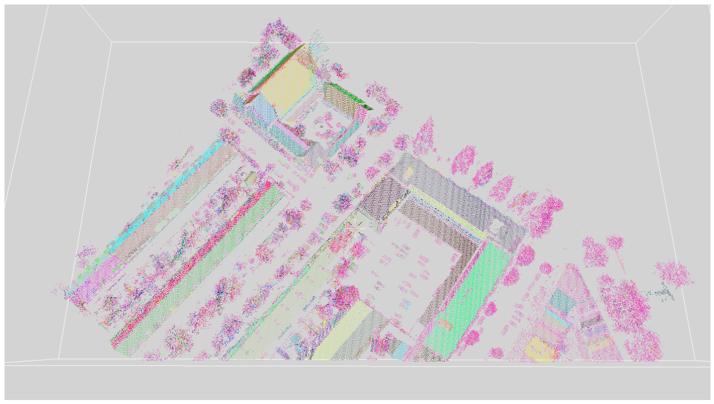
Figure 12: Result obtained for usage of a high k value (knn: 30, epsilon:0.1, min count:15, thinning:2). Also see 2

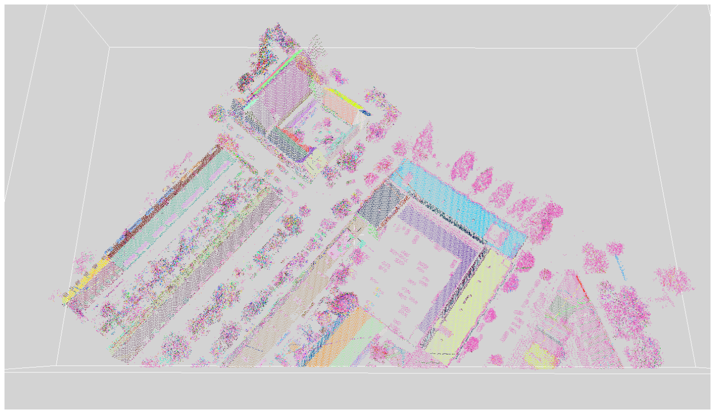
Figure 13: Result obtained for usage of a high radius value (knn: 30, epsilon:0.1, min count:15, thinning:2). Also see 3.

7. Did you make any other significant observations that helped you to improve your segmentation result?
   Usage of dictionaries and their hash table approach can considerably speed up the code during

queries such as knowing whether a point has already been classified.
In order to decrease the chances of a wrong plane being fitted if a point is on an edge, it was decide that to fit the first plane, the point itself, along with only the 3 closest ones would be used. This also means that as soon as the region grows in one direction, the plane is being recalculated rather often (for the 5th, 7th, 9th, etc. point - as the region grows, the frequency of plane recalculations decreases).