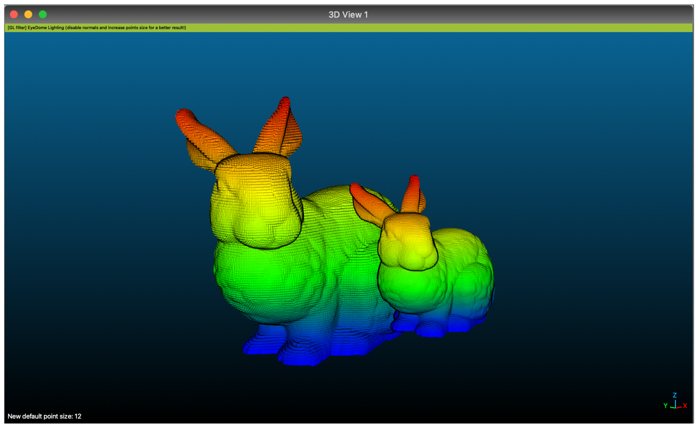
NPM3D - TP1

Aubin TCHOÏ

January 2023

Question 1

The figure below shows a screenshot of the original bunny and the transformed bunny together with EDL and perspective projection.



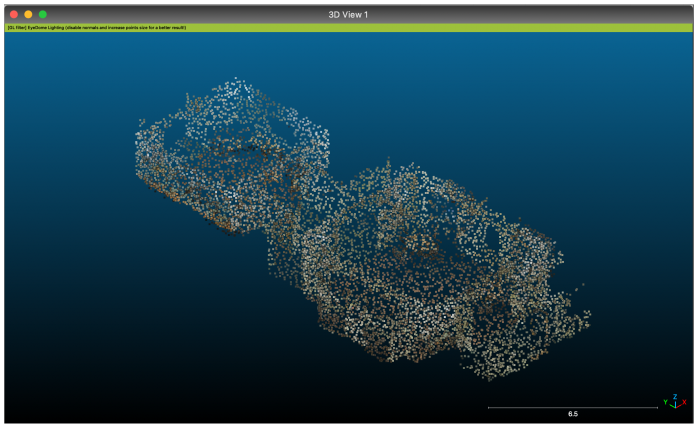
[GL filter] EyeDome Lighting (disable normals and increase points size for a better result!)

New default point size: 12

FIGURE 1 – Original and scaled-down bunnies

Question 2

The figure below shows a screenshot of the decimated "indoor_scan" point cloud with a factor of 300. The image is shown with an orthographic projection and a point size of 6. You can see that the density of the decimated point cloud is uneven.



[GL filter] EyeDome Lighting (disable normals and increase points size for a better result!)

FIGURE 2 – Decimated indoor scan point cloud

Question 3

The table below lists the computation time recorded for brute force spherical neighborhood search with a 20 cm radius and brute force KNN neighborhood search with $k = 100$.

n_queries	10	whole point cloud
spherical	0.207 s	18 h
knn	0.483 s	41 h

TABLE 1 – Time recorded for neighborhood searches

As expected, the brute force implementations are not a option when it comes to applying it on the whole point cloud. Additionally, the KNN search is slower than the spherical search, which is understandable because although they both require computing all the distances to the current point, the KNN search requires an additional sorting step to find the k closest points to the current one.

In the current implementation the whole array is not sorted, the introselect algorithm is used to only partially sort the array for better performances.

Question 4a

The figure below shows the inference time for 1000 queries for various leaf sizes on a spherical neighborhood search with a radius of 20 cm.

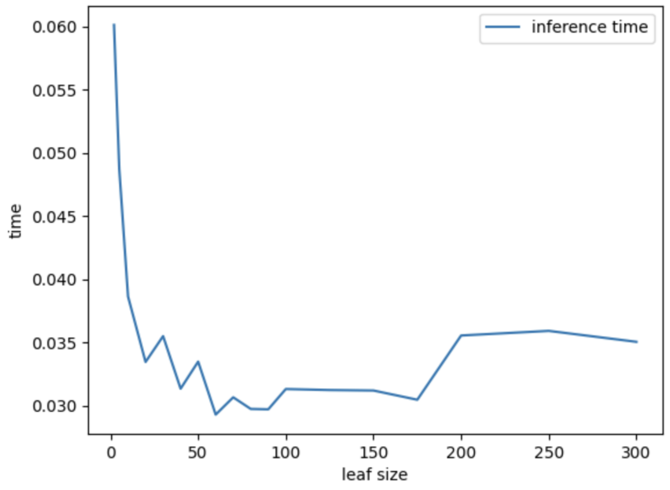


FIGURE 3 – Inference time for 1000 queries

The optimal leaf size is found to be equal to 61. There is actually a trade-off in the choice of the leaf size when it comes to the inference time. Indeed, the tree obtained for a very small leaf size would contain many nodes, for a leaf size of 1 it would have as many leaves as there are points in the cloud, but most importantly when looking for the neighbors we need to traverse the tree downwards to the leave containing the point and then upwards to check on every node if there could be neighbors on the other side of the hyperplane. Since there are many nodes for a small leaf size this search can become costly. For very large leaf sizes, the query also slows down because there are too many pairwise distance computations.

The small oscillations observed on the figure around a value of leaf size of 40 or so could be due to the alternation between dimensions in each level of depth in the tree.

If we take a look at the build time, it only decreases when the leaf size increases, since it means you can stop the building process earlier. The figure below compares the build and inference times for 10000 queries (increased from before for comparisons on the same scale).

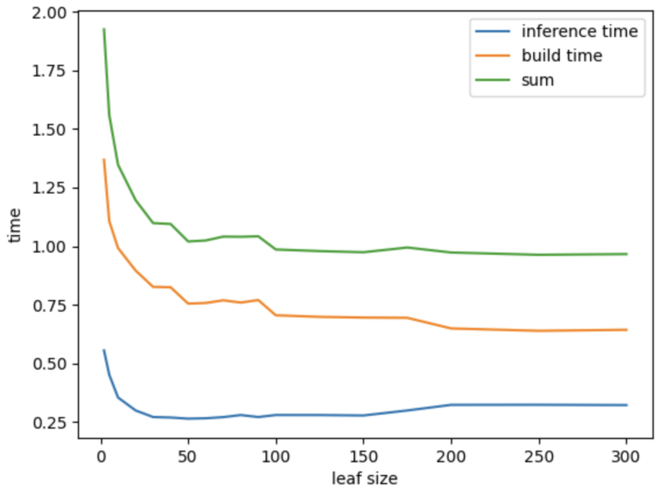


FIGURE 4 – Build and inference time for 10000 queries

Question 4b

The figure below presents the timings obtained with a KDTree with a leaf size of 61 as a function of the radius of the spherical search.

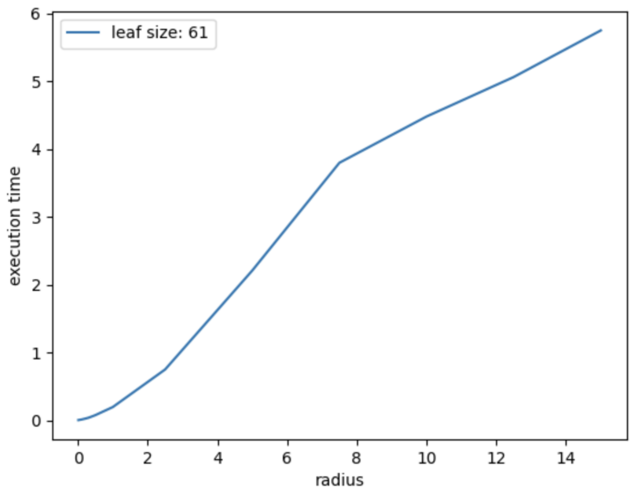


FIGURE 5 – Inference time for 1000 queries

The inference time increases with the radius, which is expected because need to look for more neighbors and thus explore more nodes in the tree. In the current setting it would take 107 seconds to compute 20 cm neighborhoods for all the points in the cloud, which is 600 times less than the brute force method.

Bonus question

The figure below shows a screenshot of the sub-sampled "indoor_scan" point cloud. The size of the voxels is equal to 0.18, which is the value for which this subsampled point cloud contains roughly the same number of points as the decimated one. Three different methods were used to represent the position, color and label of the voxel ; the point chosen to represent the voxel is the barycenter of the points in it, its color is the color of the point closest to the barycenter and its label is the most frequent one among the points in the voxel. The image is shown with an orthographic projection and a point size of 6 similarily as the decimated point cloud.

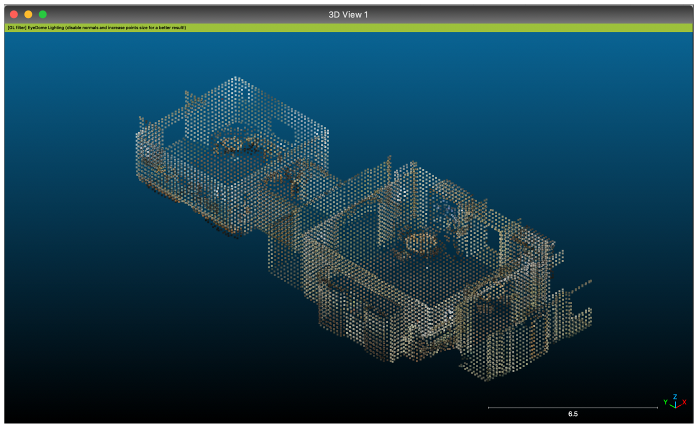

3D View 1
[GL filter] EyeDome Lighting (disable normals and increase points size for a better result)
6.5
Z
Y
X

FIGURE 6 – Grid-subsampled indoor scan point cloud

On a first glance it is much easier to identify the scene alongside the various items in it and the density of the point cloud is quite even. The grid subsampling method also seems to smooth the point cloud by reducing the noise and removing outliers from the point cloud, whereas decimation seemed to increase it as it seems to create holes in the point cloud. One additional remark is that grid subsampling is not sensible to the ordering of the points in the array that is used to store them contrarily to decimation, which makes sense since there is a priori no particular ordering.