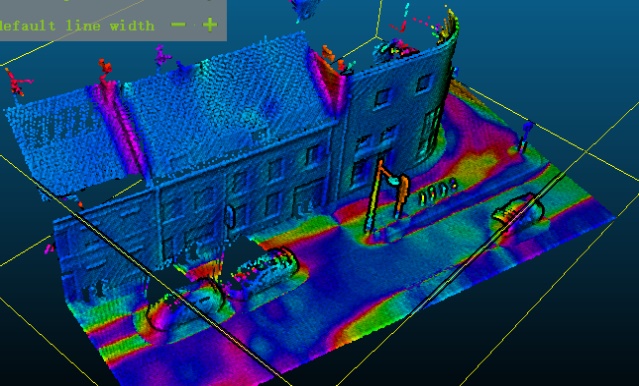
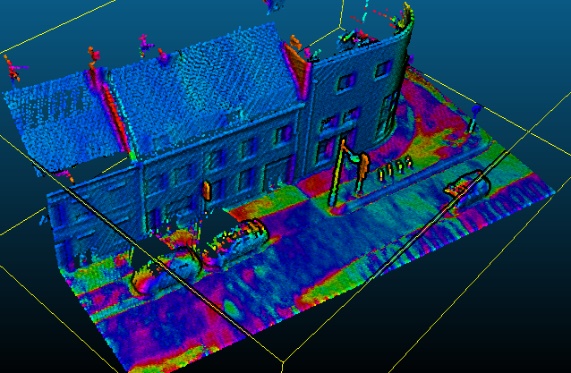
TP 4 : Neighborhood descriptors

Leman FENG

[flm8620@gmail.com](mailto:flm8620@gmail.com)

lemanfeng.com

**Question 1 (2 points) : If you use a too small / too big radius, what is the effect on the normal estimation? Use screenshots to support your claims.**

****

Left: radius=0.5, right: radius=1.0

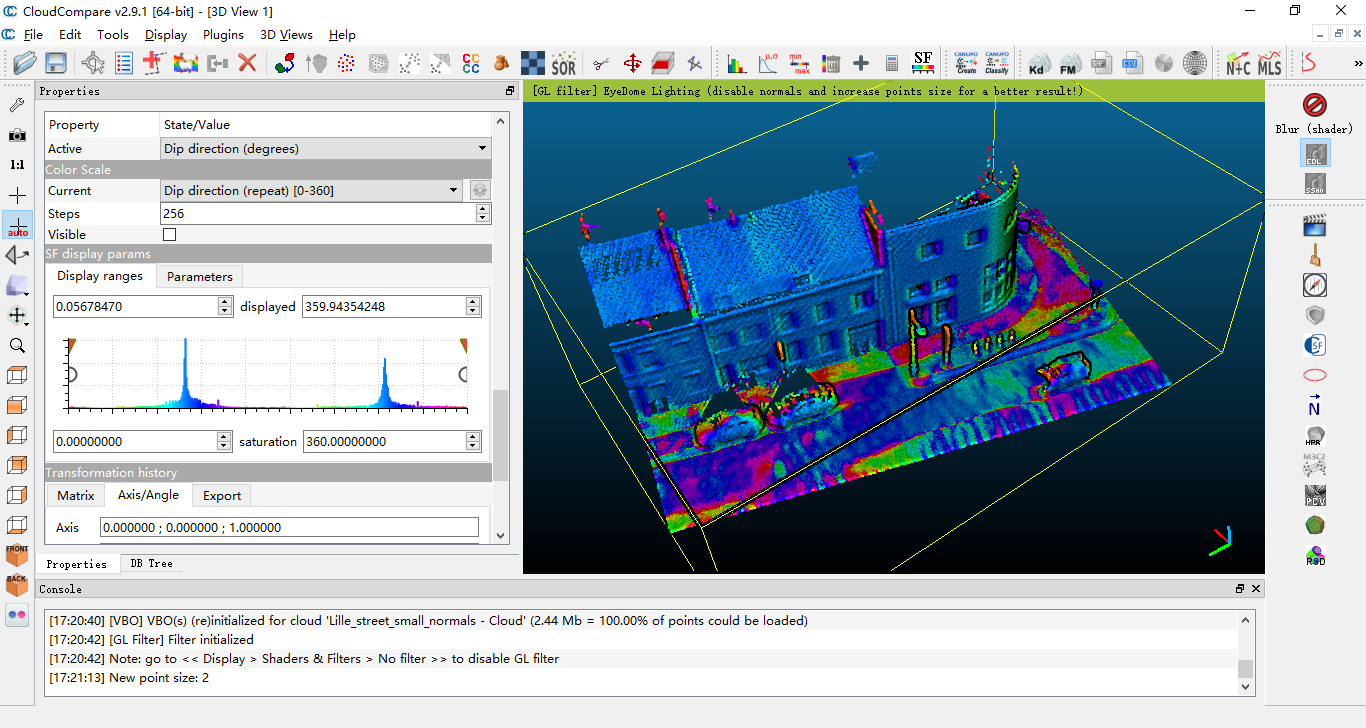
Large radius well smooth out the normal estimation.

**Question 2 (1 point) : How would you choose the neighborhood scale for a good normal estimation?**

A simple approach may be trying multiple times and set a fix value for radius if you know your point cloud has real length unit and you know you will only work for the same scenario (like 0.1-0.5 meter for outdoor scene).

If you need a real adaptive radius, maybe we can use the scale-space, meaning you use a “Mexican hat” filter together with PCA, just like SIFT detector in 2D images.

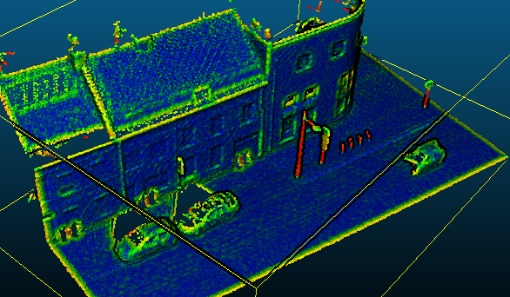
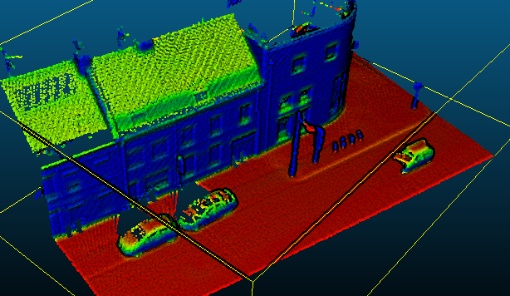
**Question 3 (2 point) : Show a screenshot of your normals converted as “Dip” scalar field in CloudCompare.**

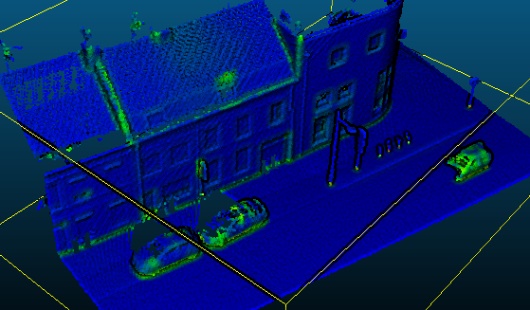
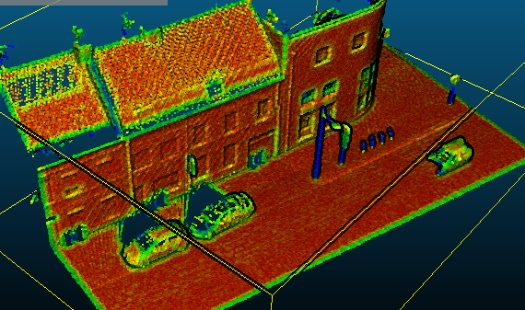


**Question 4 (1 point) : How could you measure the quality of your normal estimation with the eigenvalues?**

The normal direction should be very smooth on flat ground (radius can’t be too small), but it should also capture the small detail of cars (radius can’t be too large).

**Question 5 (4 points) : Show screenshots of the 4 features as scalar fields of the cloud. Can you explain briefly the names of the 3 last features considering their definition with eigenvalues?**





From left to right, top to down: vert, line, plan, sphere

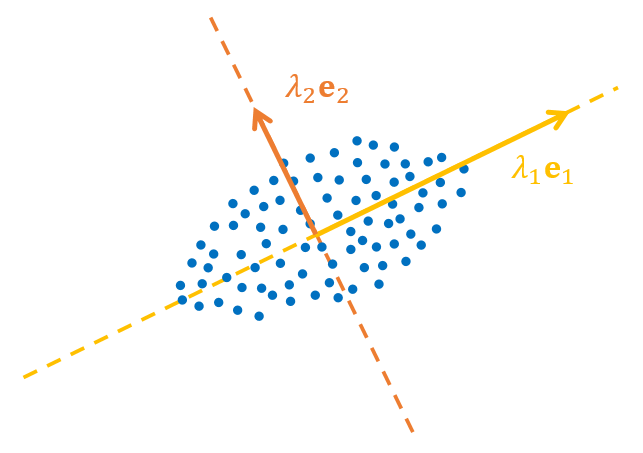
Explain:

Linearity: largest eigenvalue should be much larger the other two

Planarity: second eigenvalue should be close to the largest one, but the third(normal direction) should be very small

Sphericity: all three eigenvalues should be nearly the same.

# C. (BONUS) Mini-Challenge : point classification



I use four features (vert, line, plan, sphere) evaluated for neighbors with in radius R. Then I added 6 more sphere along 3 directions of eigenvector for evaluating another six group of features, totally 4\*7=28 features per point.

I trained a MLP network with 4 layers, which have 200, 60, 30, 6 neurons. I choose 16384 points for each class, and separate 10% of them for validation.

I used Keras library to train this network. After 150 epochs, the validation accuracy is 91%.

Here are three views of predicted labels for “Lille test” file:

