Teraki 3D Assignment

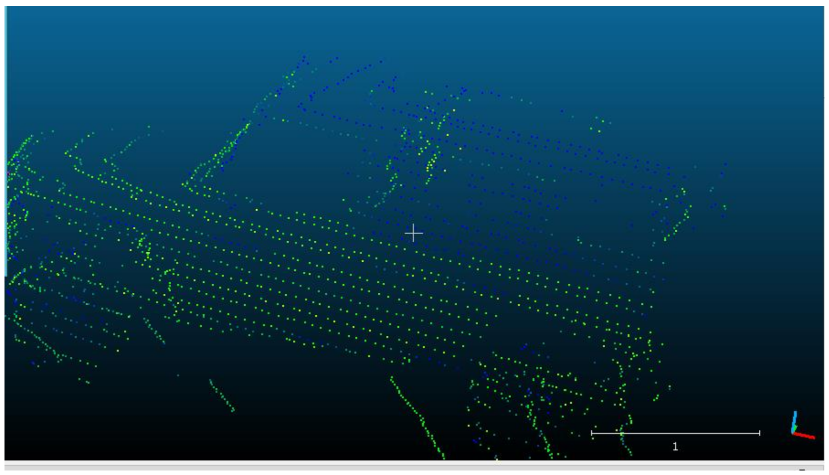
Michelle Tang

May, 2019

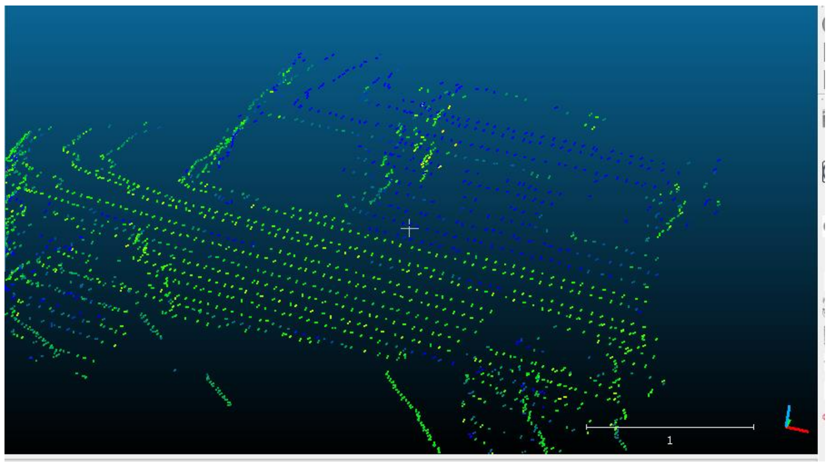
1. Visualization at Cloudcompare

Careful examination of the 3D points at Cloudcompare shows that the decompression B has the best accuracy. Decompression A follows and C has the worst accuracy.

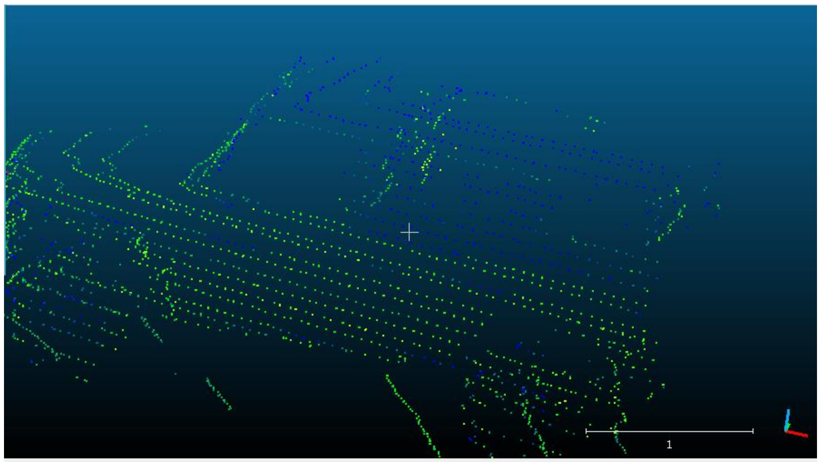
Below are representative images to illustrate how I got the conclusion. I tried to zoom in the image and examine an area that the points are not too dense, for example, the line at the bottom of the car.



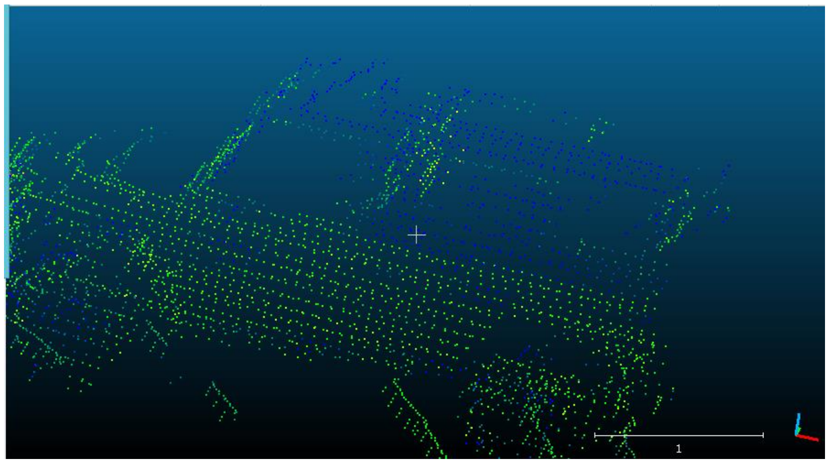
Uncompressed



Uncompressed + A



Uncompressed +B



Uncompressed + C

2. Calculation of distance in C++

To quantitatively compare the decompression results, one needs to calculate the distance that each point shifted. For this purpose, I wrote a class called PointCloud in C++. Here are the main functions in it:

readPLY: Read .ply file and output the location data into a 2D array called "points" .

pairing: For each point in uncompressed dataset, I search through the decompressed dataset using a simple nested for loop and find the point with smallest distance to it. I write that point in the same row as this uncompressed point, and put their distance as the last column of the row.

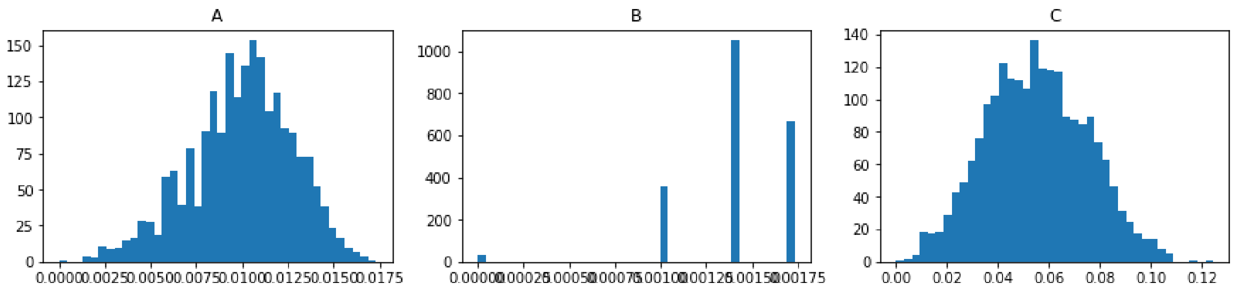
Choice of Algorithm: In terms of algorithm efficiency, I just used a simple brute force algorithm to pair the points. It has a complexity of O(n^2). If the data size is much bigger and we want a more efficient algorithm, we could build a tree structure (partition the space by repeatedly choosing a point that split the current partition in half). For each element in the other data set, we search the tree for the closest point by repeatedly looking for the partition point that is closer to it and ignore the space that the other partition point belongs to. . This way we only need to search part of the tree and it should give a search time of O(n x log(n)). Generally speaking, most efficient algorithms for pairing involve limiting the search space by applying some data structure.

(Note: one potential problem of the algorithm I am using is that more than one uncompressed point can correspond to one decompressed point. However this won't hinder us from drawing the conclusion. For details, please see part 3.)

In the main(), I read each of the decompression files, and call pairing function 3 times to calculate the distance between uncompressed dataset and decompressed A or B or C. In the end I have 3 output files. Each has 2109 rows and 7 columns. The columns are x_uncompressed, y_uncompressed, z_uncompressed, x_decompressed, y_decompressed, z_decompressed, distance from left to right.

3. Data analysis and plotting in python

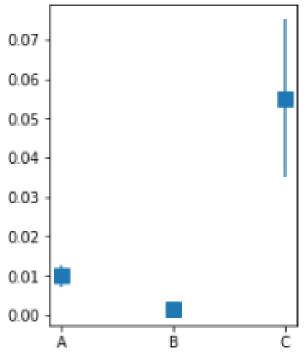
I first read in the three .txt file into 2D array. Then plot the histogram of the last column (distance) of each array.



As shown in the figure above, histogram from decompression A and C have a distribution similar to normal distribution, with A centered around 0.01 and C centered around 0.06. Decompression B has only three major peaks, around 0.001, 0.001414, 0.001731. Without further analysis, we can already see that the accuracy is: B > A > C, which is consistent with the Cloudcompare result.

Here is some further analysis:

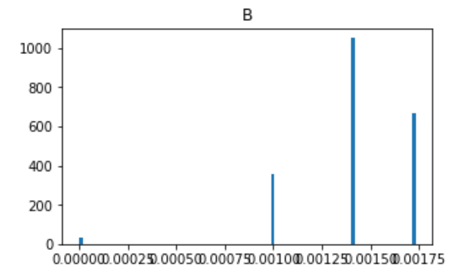
(1) Mean and standard deviation: A : 0.009990 ± 0.002788
B : 0.001421 ± 0.000306
C : 0.055162 ± 0.019976



One could also come up other metrics in addition to mean and standard deviation, for example: Quantile or Maximum distance points etc. But in this case, the conclusion is already pretty clear from mean and standard deviation. So I didn't apply other metrics.

(2) As mentioned previously, C++ pairing algorithm may result in more than one uncompressed points correspond to the same decompressed point. In another word, the algorithm fails to find the correct corresponding point in the decompressed data set. To check if this happens, I wrote a function in python that check if each decompressed points (their coordinates are in column 3,4,5) in output files have duplicates in other rows or not.
From the python notebook, we can see that such duplication does not happen for A and B, but happens for C. One explanation for this is that **the location of the decompressed points are shifted so much that the nearest point is no longer the correct corresponding point.** So even if we had not known the mean and distance values, we could have already suspected that C is the worst decompression, just from the duplication function.

(3) Histogram of B shows a very interesting pattern. Using bin number 40, it shows only 3 major peaks and 1 minor peak. If I use much smaller bin, for example, bin number 100, we still only see 3 major peaks and 1 minor peak. This is because most distances are around 0.001, 0.001414, 0.001731.



Here is my explanation of why there are only three major peaks:

If one looks at the input data carefully, one can find that the location data (X,Y, Z) has 7 digits after decimal point, but the last 4 are usually 0001 to 0005, or 9999, which is not the case for intensity data in the 4th column. This indicates the location measurement is only accurate 3 digits after decimal point. When a decompressed data set such as B is very close to the original data set, each point could be shifted ±0.001 in each of the X, Y Z dimension. If all 3 dimensions shifted 0.001, this corresponds to the distance peak of 0.00173 in the histogram (square root of (3 X ((0.001)^2)) = 0.00173). If 2 dimensions shifted 0.001, this corresponds to the distance peak of 0.001414 in the histogram (square root of (2 X ((0.001)^2)) = 0.001414). If only one dimension shifts, then this point should be in the peak of 0.001. Because data set B is very close to the original points, most points only shifted 0.001 in one or two or three of the dimensions in X Y Z. So it will only end up 3 major peaks when we calculate distance histogram.

In summary, decompression B gives a very accurate representation of the original location. If I were not already told these are the decompressed data from the same location, I would have guessed it is the 3D location of another time point while the car is moving (because most points shifted the same amount of distance). Dataset B could also be computer simulated data because the value in each dimension always centers the original value ± 0.001.

Since we are already told B is a decompression from the same location, this histogram pattern could indicate a small systematic error/bias during (de)compression, and it may be able to correct easily.