Optimisation Report

Blanchon Marc - MsCV

Face Recognition

Point Cloud Stitching

Subject: Creating a software to perform point cloud stitching.

Point Cloud Stitching

Abstract

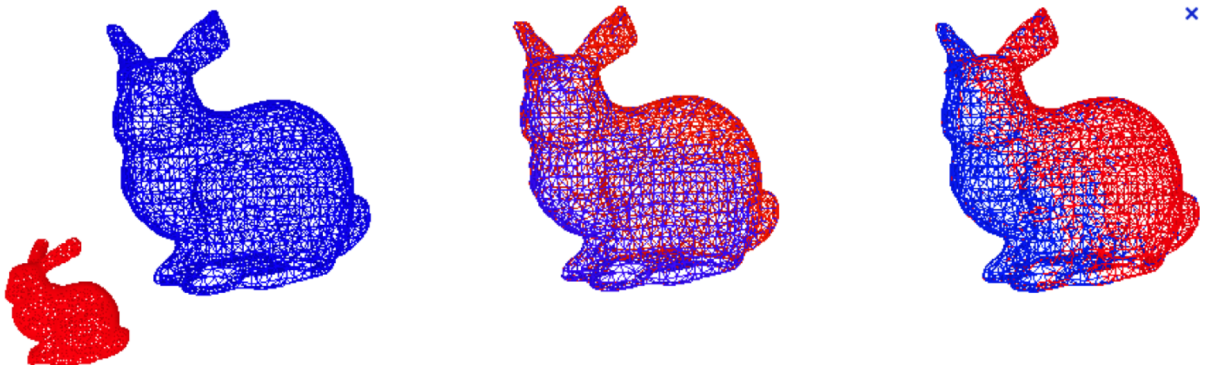
In this project, we have to create a Matlab code able to perform point cloud registration and to stitch different point cloud. In fact this is the principle of the ICP.
As humans, we can see this problem as a puzzle solving but for a machine the difficulty is much more important. We will discuss on this work using three main parts, the first one will be the principle of the point cloud, the registration, the merging.
The second part will talk about the code and the last about improvement to finish with a conclusion.

The Principle

A point cloud is a set of 3D points that form a scene. These points are uniques and can be separated from the point cloud (we can use this property for registration). This form of data (pointcloud) is mainly used to determine an object by it's surface and the usual way to get a point cloud is to scan an object.

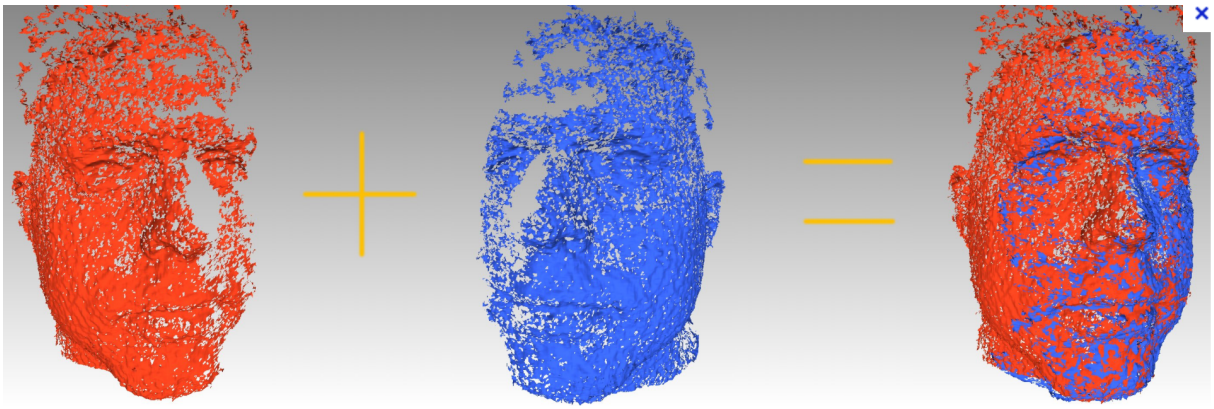
After this formal definition, we can enter more deeply in the subject.
How to Merge a point cloud in a manner to obtain, at the end only, one surface composed of different point clouds?

So assuming that we have two surfaces that are compatible, which means that they have one face partially or totally compatible. Then we are able to perform a proper registration. This action consist of register each points that belongs to both of the point clouds and to determine from this a transformation matrix to stick the two surfaces together.



We can see in this example that we have two different point clouds, and after a scan of each points and the determination of common points, a transformation matrix has been deduced and we can move a point cloud according to the other to "fit" the points.

This was the general case, now we can express more about our subject. In the previous example we can see that we have two exact or almost exact same shapes so almost same point clouds. It consist to unify the two points clouds and remove the duplicates.



Now we have an example of point registration which is called a reconstruction, we have two incomplete point clouds, and the first added to the second gives us our final shape. In fact we refer only on partial matching and we deduce in the same way a transformation matrix and add the point clouds according to this transformation.

This is our case…

We have three different point clouds, all three independents or just dependent on a low amount of point and we have to find a technique to match and to stitch them.

In order to perform such computation, we will use Matlab and we will try to use Matlab built-in functions to perform this ICP (Iterative Closest Point).

To be consistent and to reflect our level of Matlab coding. We will not use external libraries or codes, and we will stick to the build-in libraries and functions of Matlab.
Our results may be better with external functionality but we will not be able to understand each point of these codes.

The Code

In our code we will use the “ICP” library of Matlab.
Before starting, all lines of the code are commented and here we will discuss more about the theory than the code directly.
Every steps used in the code to obtain the final result are still in the code and commented till it was working and not completely useless so we can see a kind of evolution in the script.

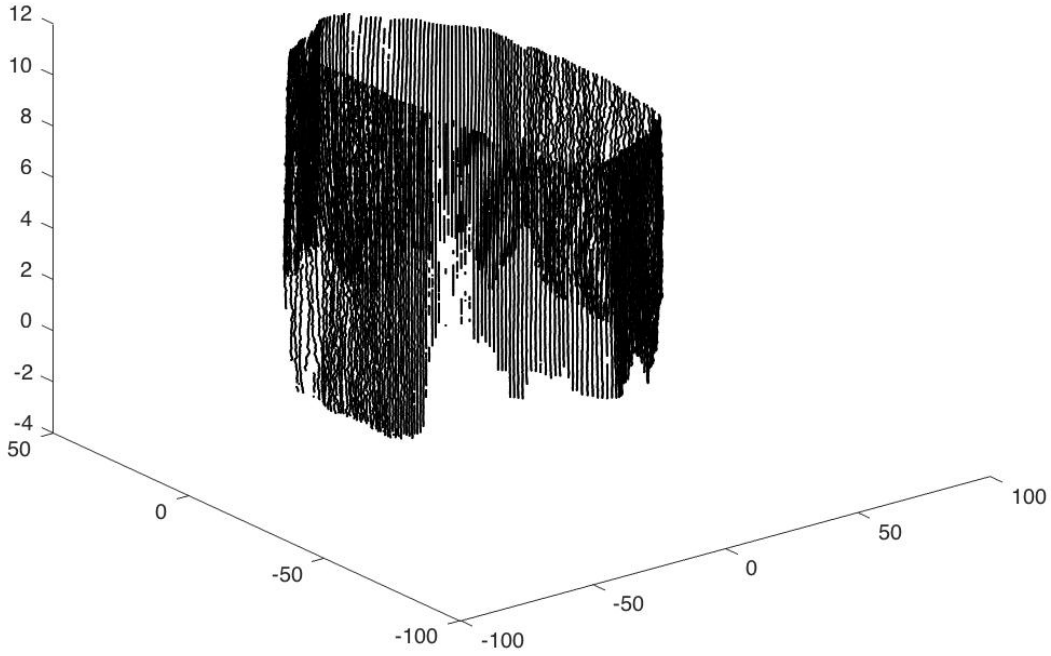
So in our case, three point clouds has been provided to accomplish this task. After loading for the first time these “.mat” data, we was able to check the types of coordinates we had.

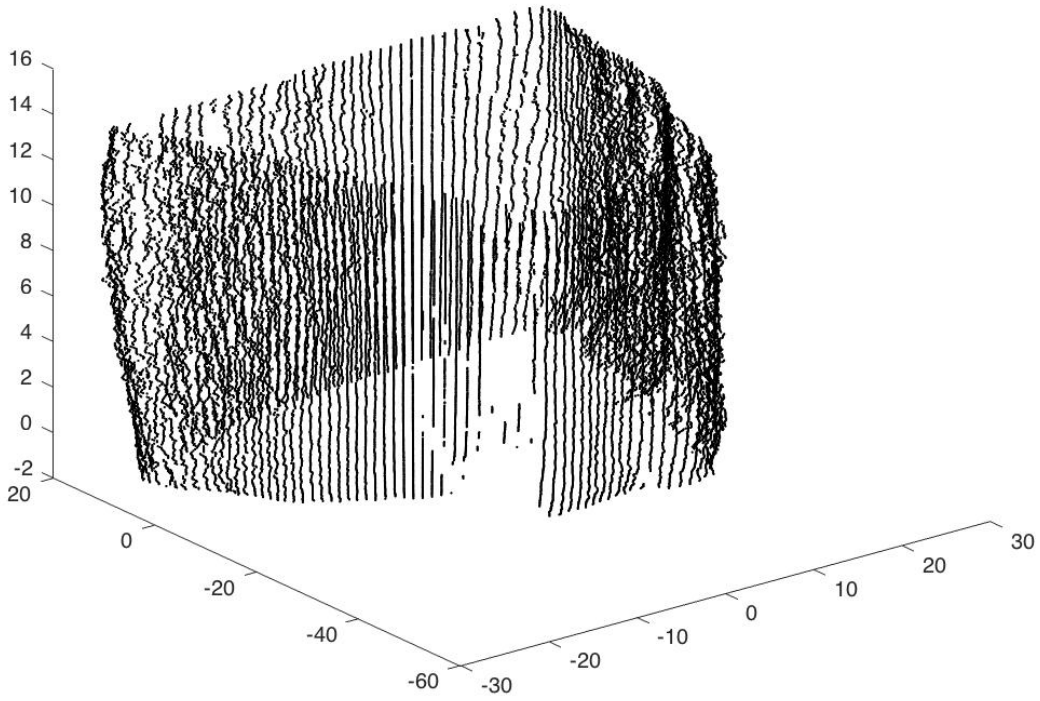
In fact each point clouds are defined only by three vectors containing each X , Y or Z.
Our first guess was to convert these vectors into one matrix 3 x Nb_of_Points and then to convert into homogeneous coordinates.
After checking how to convert coordinates to point cloud, we choose to change our homogeneous coordinates to X, Y, Z coordinated to avoid a concatenation of matrix.

In our first reflexion, and we will stick to this logic. Comparing three point clouds

```
load('HK-233_a_09-Sep-2015_ver1_XYZ+_newZ_nb_clean.mat') %load first pointcloud
figure;
plot3(Xclean, Yclean, Zclean, 'k.');%plot the pointcloud

cfirst=[Xclean ; Yclean ; Zclean]; %concatenate the coordinate to one vector
cfirst=cfirst';%transpose to have the format we want to convert in homogeneous coordinate

%hfirst = cart2hom(cfirst); % convert to homogeneous


ptCloudRef = pointCloud(cfirst); % point Cloud convertion
```

directly or comparing two point cloud to form one, then compare the third with the assembled one, are the same processes.
So we repeat exactly the same process, only renaming our variables. Then with our plot3 we are able to plot our two firsts points clouds.


12
10
8
6
4
2
0
-2
-4
50
0
-50
-100
-100
-50
0
50
100


16
14
12
10
8
6
4
2
0
-2
20
0
-20
-40
-60
-30
-20
-10
0
10
20
30

We can see that we have proper point clouds in 3D by definition.

The next step is directly point registration, and for this, our main reference will be https://fr.mathworks.com/help/vision/ref/pcregrigid.html .

The first approach was to directly perform a transformation check between the two clouds using pcregrigid function but this approach was very slow and gave bad results, or even more, errors.

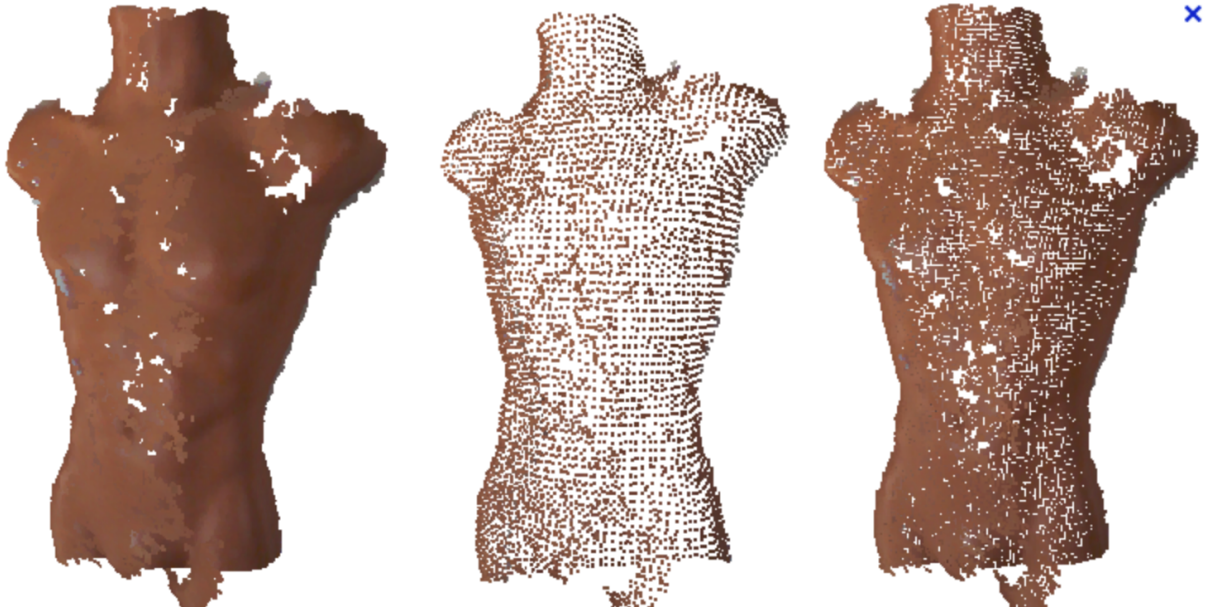
So after some research on internet, we found that the down sampling on a point cloud will increase the speed of our program and also the accuracy.
In fact it is understandable about the speed of the program, but there was a doubt about accuracy.

To explain briefly, if we down sample, we reduce the amount of points according to an algorithm. In lot of point clouds, there are many points and a lot are near to each others. The main problem interact here, if our tolerance of acceptation, or our criterion of matching is too big, then the point can fit some near other points that are not especially the real one. By down sampling, we reduce this probability to stick to a point that is not the real one because of the tolerance of the registration algorithm.

```
gridSize = 0.05; %Set a grid size
% down sample the point cloud by the grid size, it speed up the process and
% increase the accuracy
fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize);
moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);


tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true); %set the transform


ptCloudAligned = pctransform(ptCloudCurrent,tform); %transform the second point cloud accoring to the t
mergeSize = 0.001; %set a merging size
ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize); %merge the two pointcloud
```

So to focus on the code, we see that we define a grid size to down sample and apply the pcdownsample algorithm from Matlab on both of our point clouds.
We can't see a big difference for one reason, we use a low value of grid because if we increase too much, the results become totally wrong.

In order to see the effect of the down sampling, we can see this on the next page figure.

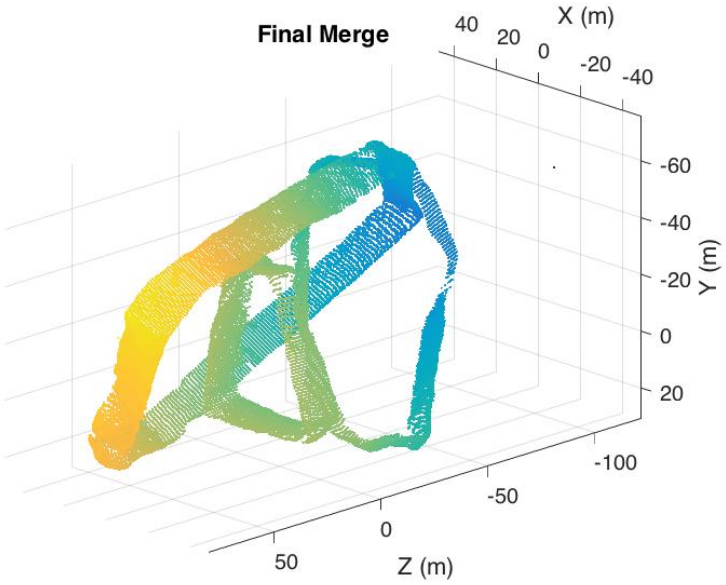


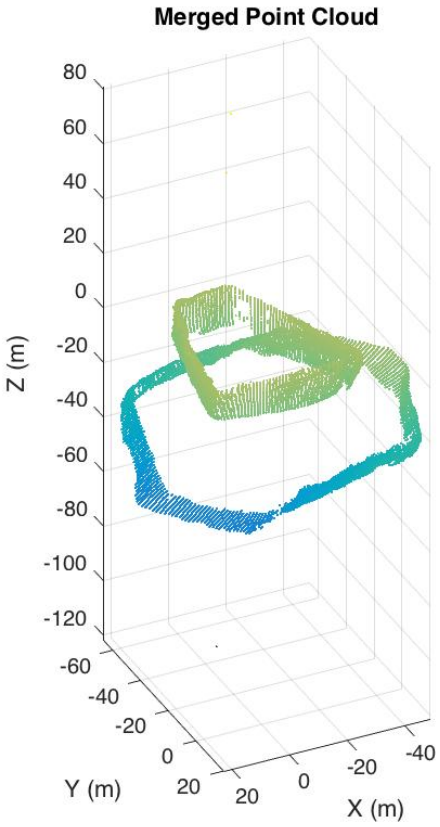
After this step of down sampling, we are calculating our transformation matrix with the function pcregrigid (point cloud registration rigid).
To be totally honest, we tried every possible parameters, but we sticked to the matlab website example because this is the only one that is working or that is the one that is creating a real transformation matrix.

After this we are just creating a new point cloud by applying the transformation mask to our target. And we merge the two points clouds using the merge function of matlab.

The last step is to display our first merging of two point clouds.
And then repeat the exact process as an iterative process to consider this new point cloud as our fixed point cloud and the third point cloud as our target.

Now we can display the results.


Final Merge
X (m)
40
20
0
-20
-40
-60
-40
-20
0
20
Y (m)
-100
-50
0
50
Z (m)


Merged Point Cloud
80
60
40
20
0
-20
-40
-60
-80
-100
-120
Z (m)
-60
-40
-20
0
20
Y (m)
20
0
-20
-40
X (m)

In the first figure we can see the merging of our two point clouds, and in the second figure, the merge of our three point clouds.

We can discuss first about results, they are not that good. In fact it can come from a lot of variables, precision, too small grid for down sample…

In fact it exist better algorithm for ICP implemented by users, the main goal of a project is not to succeed but to understand errors, and to understand the concept.

We can also discuss about the problems encountered during the project, in fact a lot of problems happened :

- Parameters badly set
- Values given for grid too arbitrary
- Inability to perform transformation on the point clouds
- Inability to convert coordinates vectors into a matrix
- Merging algorithm giving errors
- Transformation applied but no changes on the plot
- …

After all these issues, in fact it was maybe more easy to swap to a video on internet or to a code directly given from internet but after all, even if the results are not that good we have a merging process of point clouds. This is not precise and there is a reason, probably some parameters for Matlab functions are missing to reduce the tolerance of the merging process.

Improvements

To this project we can apply lot of improvements. I think in our case, the best improvement possible would be to change the programming language to be more consistent in the coding and in the accessibility.
We all have preferences and python or C++ would be a good idea to change from Matlab and to expand the code. And also these programming languages have already some libraries that allows us to perform image processing operations in couple lines of code.

As we had to stick to Matlab, we can cite some improvements of the code:

- Using external library
- Looking deeply in function to find where tolerance is changeable
- More Matlab coding practice to avoid small errors (to earn time for heavier problems)

In order to improve a bit more the program, we could be able to develop our own function to provide us a transform function, using distance matching, least square or other point comparison.

The last improvement that has been found is maybe to add an initial transform matrix as guess, like this the user have to operate a bit on the program but is can reduce the system charge and also improve the results

Conclusion

To conclude, we can see that we developed a program able to stitch point clouds. The accuracy is not good, depending on parameters it can be worse or maybe better.
In this report, the whole process of development has been developed, from research to programming.

In the zip file will be provided the code that can be run from anywhere through Matlab.