Project : 3D Scanner

Individual Progress Report : 8

Group Number : 2

Submitted to : Yohan Fougerolle

Dated : 27/11/2016

Centre Universitaire Condorcet - Master Computer Vision

1. OBJECTIVE

The goal of this report is to show individual or sub-group progress report and to update the teacher that no one is lacking behind and the teacher should be aware if someone is lacking behind.

2. TASKS ASSIGNED LAST WEEK

- Mohit and Zain - $Filteringbydepthandsavingdata$.
- Sepideh - $PlanarRemoval$.
- Yamid - $StreamingthroughGUIQtwithadvancement$.
- Julia and Marc - $OptimizingandCombiningCode$.
- Leo and Wajahat - $ApplyingMeshesandFinishinICP$.
- Utpal, Thomas, Carmen and Omair -$Solving3Dreconstructionissues$.

Github link : https://github.com/WajahatAkhtar/Project-S.E.

2.1. Mohit and Zain

The task assigned to me on Thursday was to store the filtered data from Kinect in a vector.
The group decided to store the data in a vector so that it can be accessed by our main data class within the program and be available for post processing such as ICP and 3D reconstruction. This way of storing in the vector, we assume, will increase processing speed compared to if we store it on disk.

Since I do not know how to use vector class of STL I took 2 days studying it and now able to use vector to store, and manipulate data of various types using the functions push_back (), size (), empty () etc. I can also use iterators to traverse my vector object.

To use it in our project I tried declaring a vector object $std :: vector < PointXYZRGBA > filtered\_data$;

Now I need to store the filtered point cloud data in my vector object but at the moment I am unable to decipher, in our code for filtering , which variable or pointer is pointing to filtered data.I tried using $filtered\_data.pushback$ ($cloud\_filtered$);

to push values to my vector but without any success. Leo has agreed to help me with this task.

We were given the task of filtering by depth and saving the data, it worked in live stream but we faced some delay problems. So, we were trying not to save images directly to the disk but to save the images in a vector of arrays. The first image will b saved in the first block and the second in the second block and so on. And now as we are left with only some small part of filtering, major part was done so zain will alone work on it and from now onwards me and Yamid will work on GUI.

2.2. Sepideh

Planar removal algorithm detail: The aim of the report is to detail all the activity made on planar removal. Apparently, this process is known as Claude filtering in some literature, which means selection of an object or region of interest (ROC) in a 3D model and suppression of other part. For instance for a given 3D points as shown in Fig.1 the ROC is define by a bonding box, however it is not obligatory to be cubical box rather it can be any shape.

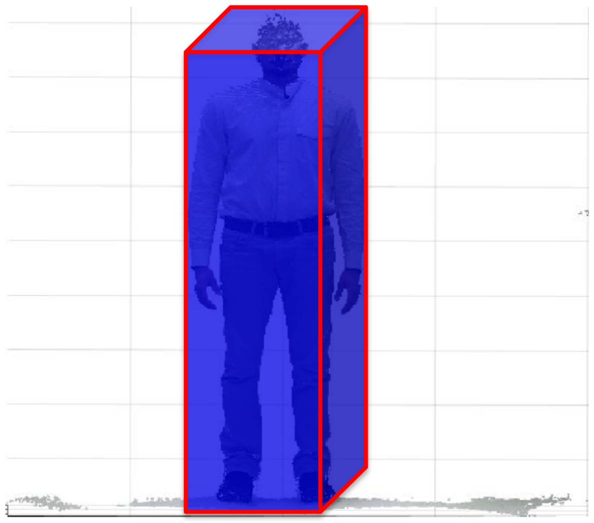

Figura 1: bonding box definition

Bonding box has 6 planes, in 3D scanner application two planes are more important the bottom and the top planes as shown in Fig.1. In fact, by having this bonding box and adjusting top/bottom planes we have a ROC region inside and non-ROC outside the box. As result the faction is defined with 2 inputs and 2 outputs. The first input of the function is a 3D Claude in PLY format and second input is a set of parameters, which defines ROC. The output of the function is 2 3D PLY models of the select object and the other one is background model (part lays out of region of interest).

$$[Background\_ply, Target\_object] = planar\_removal(3DpointsPLY, parameters).$$

Parameters includes:

- **Bottom plane:** plane position, plane orthogonal vector, and plane orientation
- **Top plane:** plane position, plane orthogonal vector, plane orientation, and distance from bottom plane

The algorithm will be located in PCL visualizaer.

2.3. Yamid

According to the suggestions from the group, the GUI has been totally redesigned into a more Photoshop-like interface. All the elements have been put into separate docks and toolbars, which can be attached inside the main window at every border, or left independent to float around the screen. The color of the interface has also been changed to gray, making it easier to work with the 3D models. In the Fig. 2 it can be noted how the elements are grouped in the different docks attached around the visualization widget.

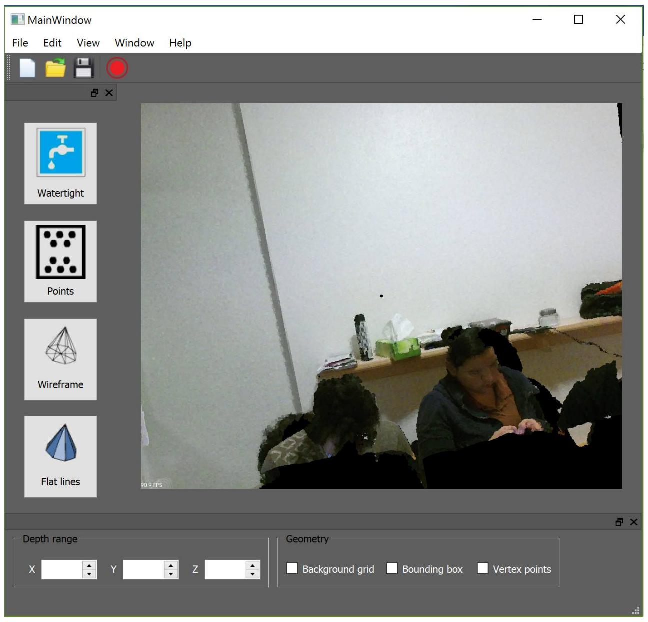

Figura 2: New interface with elements in docks attached to the main window

As mentioned before, these docks and toolbars can also be dettached, leaving the visualization window alone to cover the whole space of the screen. In Fig. 3 it's possible to see how the application looks with those docks in an independent manner. It's also worth to note that these docks are resizeable and also be definitely closed ; a menubar was also included in the main window. In order to implement the new interface, a separate Project was created and the source code from the previous one migrated to the new ; this means that the buttons and controls have the same functionalities as in the previous interface.

The new project replaces the one that was previously uploaded at Github, in order to avoid any type of redundancies.

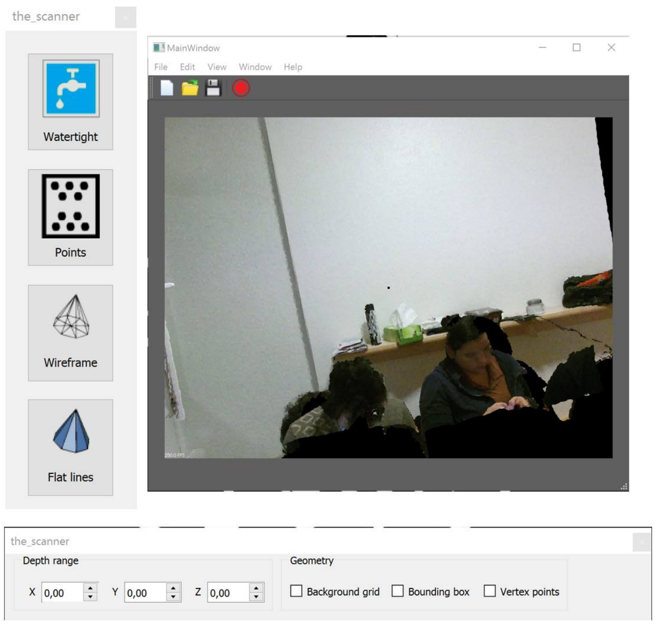

Figura 3: Interface with deattached docks

2.4. Julia and Marc

During this week I finished all 6 labs in c++, so now I understand how to create classes, how to work with pointers in 1d, 2d arrays. Was organized a meeting were we share our knowledge about basics of oop. Also from Thursday I joined group of ICP. Till now I read about point clouds, how to registered them. I have problems with running examples, but I think I will fix it in few days, because errors not about code itself, but about pcl library.

2.5. Wajahat and Leo

At first, we tried ICP on the same point cloud with a manual given initial transformation. It worked perfectly. More research, understanding is done on the ICP method of PCL. Now we are able to stitch at least two point clouds and get a normal result.

I have noticed a problem of illumination, that occurs when there is one strong light source in the room, where the person is being scanned, which causes illumination differences on the point clouds, which after stitching point clouds can be clearly seen. However there is not much that can solve a problem, except for better lighting in the room for scanning.

For the data we have, there is still a problem of stitching more than two point clouds, but it may be caused by the simple fact, that the ones we have used are not quite overlapping enough and some of them are too different from each other. What we are now trying to do is to divide all data into groups and to align point clouds within these small groups first, than align the group point clouds between each other.

On the first picture below you can see the result of stitching point cloud from picture 2 with another point cloud. You can see the lighting problem on the back of Omar.

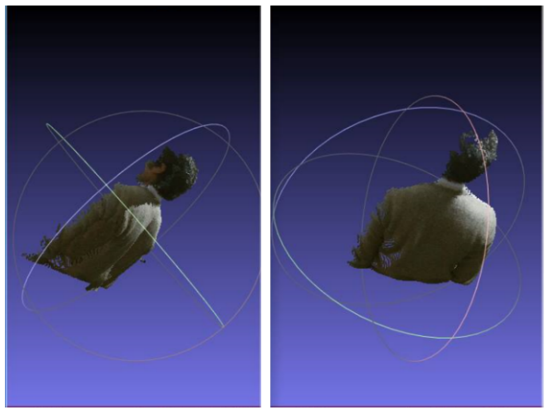

Figura 4: Result of stiching point cloud

As for stitching more than two point clouds we decided to test out different approaches. The first one we will try is filtering by convergence result. It will probably save us from cases when one of the point clouds is really bad (person moved too much or something like that). So if we realize that two point clouds do not converge enough, we will not take to consideration one of them. We also will try smoothing for the stitched groups of point clouds, so we can possibly get less outlier.

As mentioned above that when we align point cloud with itself through icp without any feature or normal estimation it aligns perfectly well, as clearly seen below we have an original cloud in white and a transformed one in green with some translation and rotation.

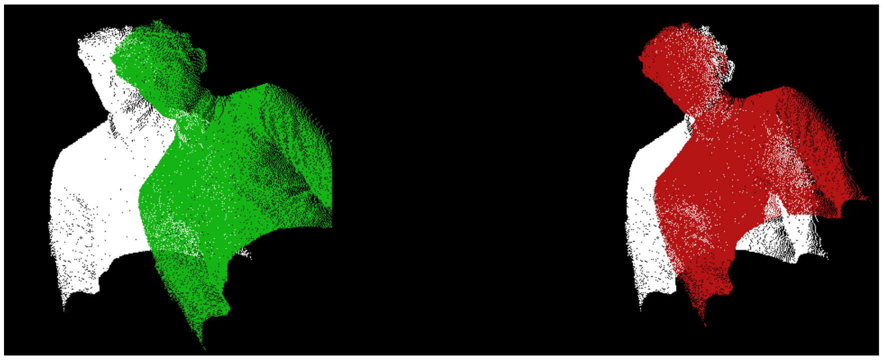

Figura 5: Left before ICP , right ICP with 1 iteration

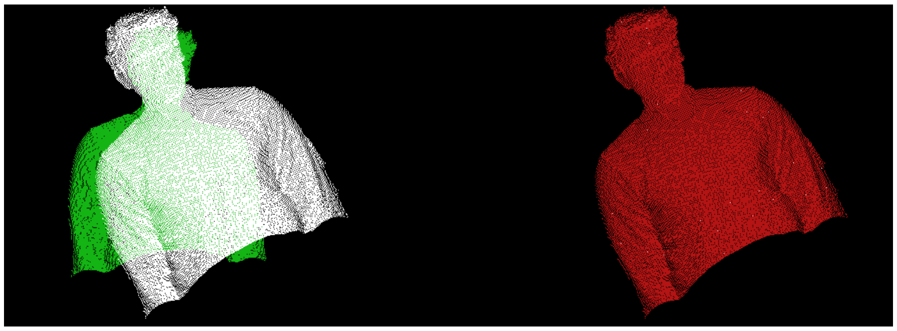

Figura 6: Left before ICP , right ICP with 50 iteration

so when we applied icp, within few iterations (i.e 50), the cloud aligns with the transformed one and within every iteration it checks the threshold we set on the basis of the overlapping, and maximum correspondence distance works fine but if we move towards shoulders or sides of the point cloud data that can be seen below we see no alignment as a result of which we start getting bad output, as we were using incremental icp so if such result is added to next point cloud it totally change the results and that's the reason we were having bad results and our cloud was disappearing. From now onward we will try to find out how to remove this error as it is not coming from the parameters we have set in our code as above the same parameters work on the point clouds, so we have to figure out whether it requires smoothing before adding to other cloud or the dataset of point cloud we have is not too good.

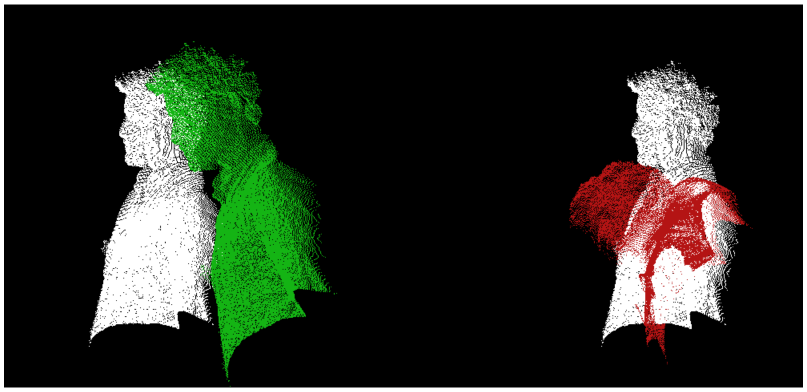
Figura 7: Left before ICP , right ICP with 200 iteration

As we are able to find where we were having problem in alignment so our next task is to apply the points we have mentioned above through smoothing and increasing number of frames on the side to have more overlap.

2.6. Utpal, Thomas, Carmen and Omair

After the last meeting we had, we focus on the theory of the papers
of poisson surface reconstruccion and tried to get the clear understanding of the algorithm. What we understand is that, for a given point cloud first step in poisson reconstruction is to find an indecator function M(x,y,z) = a F(x,y,z) + (1-a)B(x,y,z), whe M is the model of boundary, F is point insite the model and B is points out side the model. to estimate the indecator function,we pass the point cloud model through gaussian filter to get the noisy model then we subtract the noisy model from original point cloud to get the model of the boundary.ie $S - S * G = M$ and then we optimise the model M using Poisson Equation $\bigtriangledown \cdot \bigtriangledown \cdot$ M = $\bigtriangledown \cdot \bigtriangledown \cdot$ S

And analysed that the result by poisson may not be good in these two cases:

- 1. If size of gaussian is not appropriate because if it is large then the indecator function M will be so strong that optimisation using poission equation will not result in any improvement on the other hand if size of gaussian kernel is small then then the indecator function will not be appropriate to get the actual surface,
- 2. If the point cloude data is missing some where the approximation of indecator function will not be appropriate.

However as we could not get desired results from this algorithm, we contacted Cansen and Yohan for guidance.The approach of the meeting was that we clear some clues about the algorithms we were using inside the watertight, which some of them complicate the algorithm itself. Moreover we present results of Greedy Projection Triangulation (GPT) algorithm which got a better approximation of mesh surface, although with holes, than Poisson method.

Moreover, Cansen also advised us to explore this technique more and apply some hole-filling algorithm for the hole. But we decide to continue as well with Poisson method with the aim of simplify the code.

Results:

- **Greedy Projection Triangulation.** This method does not have the same particularity of the Poisson algorithm,create directly a watertight mesh without holes, so we have worked on a Holes filling algorithm.

  We did some research about the existing algorithms (Volfill, Triangulation, ...). The main problem in our situation is that it exists no holes filling algorithm implemented in the PCL library other than the Poisson algorithm... so we add the choice between adding a new library to the project or coding our own algorithm. We decided to go with the second option because it will force us to code more and use tools as the STL library. For the choice of the method, we found a very good paper [Link] (from Varnuska, Parus and Kolingerova) about a very basic triangulation : They scan all the reconstructed mesh in order to find the the triangles without neighbours and get the corresponding edges. After getting an edge, they check all the connected edges that are in the same case in order to build the the shape of the hole. When we know all the holes, we will do a triangulation of the hole to fill it with triangles.

  We will start the implementation of the algorithm, when we are able to read the datas from the PointCloud datatype (PCL) outside a PCL functions (this work is in progress and it will be finished very soon).

- **Poisson.** After the meeting with Cansen and Yohan we decide to start again from crash with this algorithm and keep it simple, starting from the basic and move and adding new approach step by step, you can find the code [here]. By doing this we could had the following results:

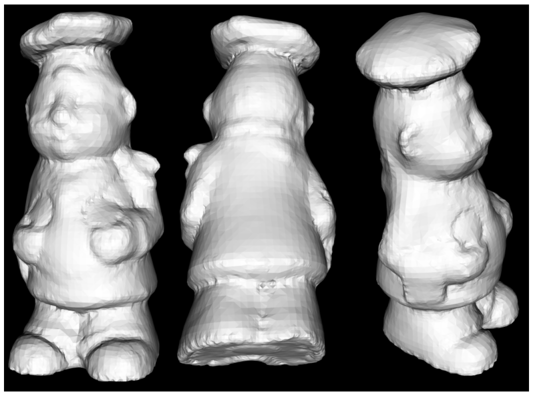

Figura 8: PCD model - Chef

The figure 8 is the reconstruction of a PCD simple model, as it can seen, we have a good result, without holes or overmeshing and we can apreciate details of the object. by simplifying the code and not overloading the data with algorithms that we fairly understand we finally obtain a better result. However, when we test the code with objets with more details and complicated form the problem of overmeshing is present again, (e.g Fig.9 and Fig. 10), this is because we do not have a completely çlear"point data (on Fig. 10 it can be seen the points out the region of interest), and like Poisson method is sensity to the noisy.

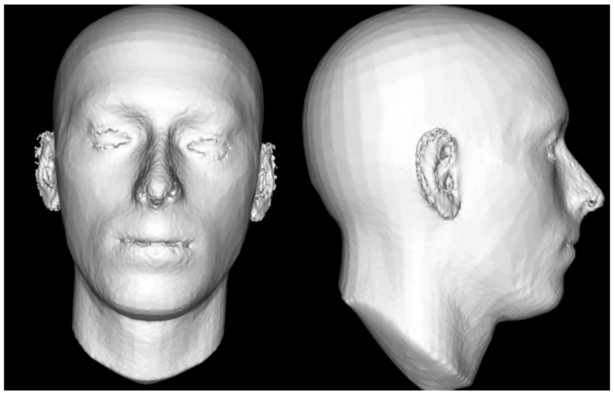

Figura 9: PLY model - Head

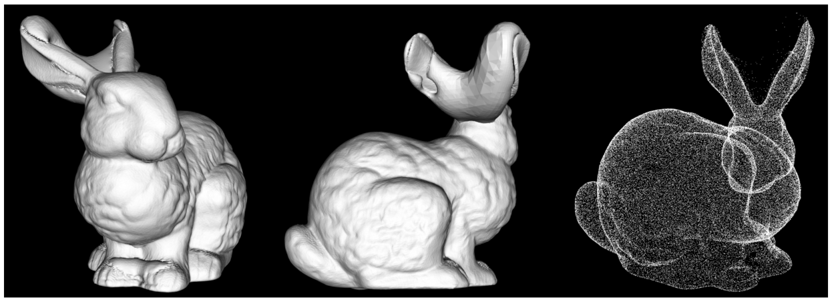

Figura 10: PLY model - Bunny

Conclusion: We had a progress with Poisson algorithm, we could solve the problem of meshing the axis that was presented before. However we keep have difficulties when running a complicate shape model. we need to explore a new strategy and start addying methods to our code to try to have an accuare reconstruction to any kind of object because the aim is to reconstruct a whole body, which one is not a simple object.

By another had, the Greddy Triangulation has also their difficulties because it is not easy to filling the holes. So we are not sure if we should choose Poisson and explore more the greedy triangulation, we would like to extern this concern to Cansen and Yohan in order to take the best option.

3. Future Work

- Zain - *Filteringbydepthandsavingdata*.
- Sepideh and Utpal - *PlanarRemoval*.
- Yamid and Mohit - *StreamingthroughGUIQtwithadvancement*.
- Marc - *OptimizingandCombiningCode*.
- Leo, Wajahat and Julia - *ApplyingMeshesandFinishinICP*.
- Thomas, Carmen and Omair -*Solving3Dreconstructionissues*.

Github link : https://github.com/WajahatAkhtar/Project-S.E.