Project software engineering

3D Scanner

Group Members : *Wajahat Akhtar, Omair khalid , Thomas Drevet, Lev Kolezhuk, Maria del carmen Moreno Genis,Yuliia kamkova, Yamid Espinel , Zain Bashir, Mohit Ahuja,Utpal Kant,Marc Blanchon, Sephideh haddadi*

Submitted to : *Yohan FOUGEROLLE and Cansen JIANG*

Dated : *03/01/2017*

Centre Universitaire Condorcet - Master Computer Vision

Contents

List of Figures

1 Objective

To create the software product for performing the 3D scanning and creating a 3D model by using QT environment on Windows using C++. A 3D model is a digital representation of a physical object. If you already have an object, and you want it in a digital form, that's what we do. We use advanced 3D scanning equipment to capture and transform them into 3D digital models. Tasks:

2 Key Steps

The following steps are the building block of the Project :-

- Choose the model of the 3D camera (sensors) for the project
- Choice of the libraries
- Installation of PCL on Windows.(Interfacing with Qt)
- Getting data from Kinect V2.
- Filtering Coordinates.
- Point Cloud Registration (ICP) .
- Implementation of Meshing/ 3D Reconstruction
- Graphical User Interface (GUI)
- Project Management
- Results.
- User Manual

3 Choice of Sensor

3.1 Sensors Available

There are hundreds of sensors available in the market for 3D reconstructions but looking at the quality and the amount of budget 2 sensors were choosen The Kinect V2 Sensor and R200 both having their own qualities and drawbacks but among those two the best one we choosed is qiven below.

3.2 Chosen Sensor

We used Kinect V2 Sensor in our project for 3D scanning due to the following reasons.

- Time of Flight sensor.
- RGB camera of 1280x960 pixel resolution.
- Viewing angle = 43 degree (horizontal) by 57 degree (vertical).
- Frame rate = 30 FPS .

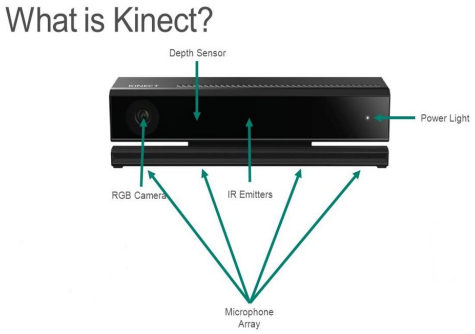

Figure 1: Kinect V2 Sensor

R200 sensor was not selected although it has some extra features which kinect v2 doesn't have like portability , light wait to carry etc. but looking at the task given to us kinect v2 overcomes R200 depending on resolution and the kind of data we acquired performance and depth we wanted.

Figure 2: R200

4 Choice of libraries

At the beginnig of the project we focused in the search of sources available to work with point clouds. We find diferent approachs but we selected the one more accurate to our needs and level of knowledge.

The three more highlights options that we find are following described and also is explained why we do not use or use it.

4.1 Available Libraries

Below here is the the list of libraries which are available for 3D reconstruction and there features.

4.1.1 Reconstruct Me Library

This is a software interface created to manipulated 3D objects. It is only needed a kinect, or other 3D depth camera), in order to obtain the 3D information of the object and then the software will be able to apply the different applications that it provides. This software is easy to use and understand and also is provides the facility of work with the 3D reconstruction in real-time. Unfortunately, this software is not open source, and, as that is one essential requirement, we discard it as an option.

4.1.2 CGAL (Computational Geometry Algorithms Library)

This is an open source project with easy access by a class library dedicated to the processing of point clouds and polygonal and volumetric meshes. This library makes extensive use of advanced c++ features such as templates and trait. Like most of their computations are backed up by interval arithmetic and multi-precision libraries, users with high precision requirements tend to use it. Although this library contains a large documentation and accessibility to example sets, its main focus is in computational geometry and computer graphics so, the manipulation of the library is quite complex and demand a solid knowledge of programming and also one of the big problems with this library is the complexity of interfacing it with QT. For these reason we decide to not implemented.

4.2 Chosen Library

As there were many libraries available but we choose PCL due to the following reason below.

- **PCL (Point Cloud Library).** PCL is a class library dedicated to the acquisition, processing and visualization of point cloud datasets. It provides components very useful for a variety of operations on point clouds, for example: Point cloud cleaning and filtering, normal estimation, spatial search, registration, surface reconstruction, visualization, etc. Moreover PCL contains a valuable resource with large set of recent research-grade algorithms. Also it makes extensive use of non trivial c++ features, which means the facility of add modification to the functions, not been limited of the "basic" definition of them. At the end we decide to use this library as it provides a wide documentations with tutorials and quite explanation, it is not to complicate to manipulate it and it is focused at the 3D perception and robotics applications.

5 PCL Installation

5.1 Steps required for installing PCL Library

There are few basic steps required for the installation of Point cloud library Its easy to install if right version of PCL is choosen else it will take you a lot of time even months to figure out the problems and interface it correctly.In the start when we starting interfacing qt with PCL we choosed version 1.6 as it is the open source available official version online we tried installing it by building libraries individually such as OpenNI2,Boost,Eigen,FLANN,Qhull,VTK which can also be done all together but there were many problems in installing them individually as they have many independences problem.But as i mention choosing the right version is also a big problem if chosen a correct version it can save your lot of time, So after a lot of research and Problems we were able to find the best version of PCL that is PCL 1.8 which can be installed on windows.

5.1.1 Installation Instructions:

Prerequisites PC Windows 10 (x64) and a USB 3.0 port for Kinect V2.

Below here is the link to the InstallPCLReadme file which provides all the steps required for installing PCL on Windows.

- Github Link : Install PCL in Windows using QT

5.2 Errors and solutions

- If you get error on QT that there is no compiler or missing compiler , Visual studio 10 or higher version should be installed as mentioned in InstallPCLReadme because QT MSCV doesn't have in built compiler and PCL is been built using visual studio compiler.
- if you run a simple program code in qt after all the installation steps mentioned in order to check basic PCL examples but when you built the program crashes ,It happens because few libraries are not linked properly and are not added to the system environment variables.So open the folder where you are building the code open .exe file as we run in Release mode so .exe file could be found there. when you will run .exe file you will see the error which is causing the program to crash.add those to .dll files path to system environmental variables, Normally you will find QtCore5.dll and OpenNI2.dll file missing.So open My computer property , click Advance System Settings then environmental variables and then search for path in system variables and add path to the variable PATH in system variables by clicking and edit and then new.

5.2.1 Helping Link / Material

- Link : Link 1
- Link : Link2
- Link : Link3

6 Kinetic Grabber/Streaming

6.1 Filtering Coordinates

The idea for this piece of code is to grab point cloud data from Kinect V2, filter the data using a pass-through filter and save the filtered data both on disk and in a vector to be later used for performing ICP and then 3D reconstruction.

Planar removal:

we did the literature survey on the topic and studied one recent paper "Automated Removal of Planar Clutter from 3D Point Clouds for Improving Industrial Object" by T. Czerniawski, M. Nahangi, S.Walbridge, C. Haas published in 33rd International Symposium on Automation and Robotics in Construction (ISARC 2016) [3]. This paper presents following steps and results shown is efficient:

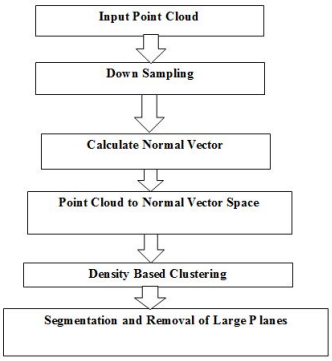

Figure 3: Schematic Planar Removal

But as the algorithm is computationally expensive and our environment is Simple, We proposed a simple analytical process that we take the cluster of points with same y-coordinates and detect the plane having maximum points with same y-coordinates then remove all the points in the point cloud having y-coordinate less than or equal to that plane. And we implemented this with Pass Through filtering process itself and found effective result.

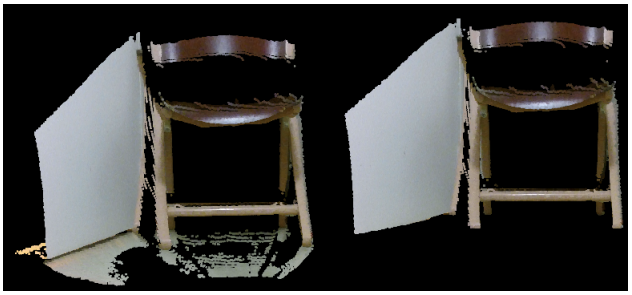

Figure 4: Acquisition with and without Planar Removal

- First of all the typedef pcl::POINTXYZ PointCloudT; is used to replace every instance of pcl::POINXYZTRGBA with the word PointCloudT to help with easy coding. We are using the point type of POINTXYZ from pcl as it gives the xyz information of a single point.
- We then declare two vectors of type POINTXYZ::Ptr named filtered_data and cleaned_data. Filtered_data will contain filtered (filtered along x, y and z axis) point cloud data and cleaned_data will contain only the point clouds that are well defined and have some values stored inside them. Next, a mutex instance named mutex is instantiated from the boost library. This instance is used to protect shared data from being accessed simultaneously by multiple threads/objects.
- A point cloud pointer object of type POINTXYZ is declared called cloud2. An integer named filenum is declared that keeps count of the number of pointcloud files saved. We instantiate an instance of the QElapsedTimer class named timer and use its start function by calling timer.start() to start counting. We need to count time because our approach is to capture a point cloud after a fixed time interval of say 500 milliseconds.
- Pcl viewer is a shared pointer object of the templated class shared_ptr of type PCLvisualizer named viewer and the memory for it is allocated dynamically. The setCameraPosition member function of the PCLvisulaizer class is called on the viewer object to set the x, y, z coordinates camera position and x, y , z coordinates of camera view-up position.
- Ptr to makeShared() functions copies the cloud to the heap and returns a smart pointer to it.
- A grabber instance is declared which is set equal to a shared pointer of type Kinect2Grabber.
- A connection object named connection is declared which is set equal to the grabber calling registerCallBack function.The registerCallback registers a call back function/method to a signal with the corresponding signature and returns a connection object that can be used to disconnect the call back method from the signal again.

- We now set a capturing parameter of type Boolean to true if it is false and vice versa and then go on to invoke the start () function on the grabber object to start grabbing images.
- We then enter the actual data capturing while loop which executes until the capturing state is true. The scoped_try_lock function takes the mutex object as an argument and tries to lock the thread. If the thread is locked and cloud exists.
- Ptr is a typedef for shared_ptr of class pointcloud of type PointT. The cloud_filtered is an instance of this Ptr and is instantiated dynamically.
- We then create an object from the PassThrough class of Pcl and name it pass. This will be used to filter our point cloud data along x y and z directions.
- We pass our point cloud as an argument to the setInputcloud function, pass the field name as an argument to the setFieldName function and pass lower and upper limits for our filtering window as arguments to setFilterLimits function to allow filtering along the direction of given field name. The function filter() takes dereferenced pointer to cloud_filtered as an argument to perform actual filtering. The output of filtering in one direction is given as input to the setInputCloud function for filtering along the next direction and the process repeated. Our GUI has the option to vary lower and upper field limits in real time.
- An if condition checks if our enable capture feature is true and the elapsed time has exceeded that of a predefined value given in the GUI, the current filtered point cloud is appended to our vector using the pushback function. Simultaneously, our current filtered point cloud is also stored on our hard drive using the function savePLYFILE defined in the IO class of the Pcl library. The timer is restarted to start counting again. The current point cloud is also added to the visualization window using the addPointCloud function invoked on viewer object. It Updates the XYZ data for an existing cloud object id on screen and returns false if no cloud with the specified id was found.
- The function setPointCloudRenderingProperties on viewer object takes in 3 arguments; property type, its value and cloud id. The next step was to remove any undefined point clouds stored in the filtered_data vector using the function removeNaNFromPointCloud from pcl library and moving them to the vector cleaned_data. The last step was to update our Data class by moving the cleaned point cloud to its member m_poincloud_data. Finally, the callback function is disconnected from the connection object using the function disconnect().

7 Point Cloud Registration (ICP)

Iterative Closest Point provides a base implementation of the Iterative Closest Point algorithm as defined.it is an optimization algorithm, that is trying to minimize the distance between the correspondent points of the two input pointclouds. It may work either with the coordinates of the points in a 3D space or other data such as normals to the surface of the point cloud or the RGB color data. There are multiple different approaches as for the data ICP is working with.

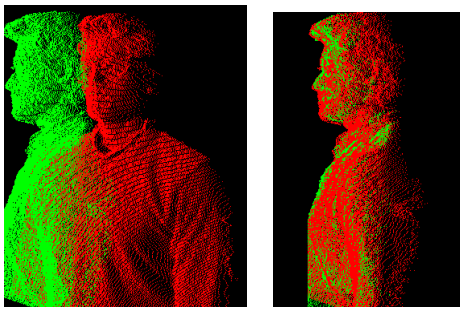

Figure 5: Point Cloud Rigid Transformation

7.1 Available Techniques in General

7.1.1 Surface normals

If we consider taking 3 or more points (that are not on the same line), we can estimate a surface, which is closest to the one that these points make. And we can always build a normal vector to this surface in order to characterize the position of the plane that we have built. Then we can work not only with the point coordinates, but also with the normal vectors, which in some cases may cause more accurate surface matching of the two pointclouds. However, using this approach requires either the point clouds to have no noise at all or to filter the normals, that we aqcuire. Calculating normals may cause absolutely unexpected results if the point cloud has outliers, or if the normal is calculated near the border of the point cloud. Both cases create quite a problem to deal with, because in order to create a filter, that will leave only the normals, which only represent the real surface, we need to define some very complex criterea. This approach rather creates other problems, than solves one. So after trying it out, we decided to avoid using surface normals for ICP.

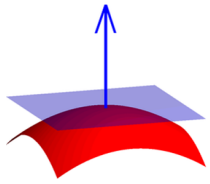

Figure 6: Surface Normals

7.1.2 Correspondences

If the two point clouds represent the same object, and they have a distinguishable overlap, there is always an option to find correspondences between some features of the object, that appeat to be represented in both point clouds. In order to do that we can determine the keyfeatures of the clouds by the means of various algorithms. There is no need to dive deeply into the principals of these algoritms, but better to mention that there is always a need to filter the aqcuired correspondences (checking if they are not a false positive) and this is also quite an issue. We have tried multiple algorithms from the PCL library for filtering the correspondences, however none of them did their job well for the data we had. Of course, that means that if we would experiment with various smoothing filters, outlier rejectors and others, we could have achieved a good result of even determining the proper correspondences. However then we face the question of computation time, that is required to achieve the desired result. At some point we may face the fact that the time that is required additionaly to filter points, find correspondences, use correspondence rejectors will be the same as the additional time that will be required for ICP to process more data. So, this approach was also rejected.

7.1.3 Filters

- In order to filter the point clouds before applying ICP, we have tried multiple different filtering techniques. The goal was to reject the outliers, which most of the time may be the noise of the depth camera and to smooth the point clouds, so that the representation of the surface does not contain any regions, that are not as smooth as the real surface.
- For rejecting outliers at first we used the radius based algorithms, that controls the densityof the pointcloud by itteratively counting the number of points that are contained inside a sphere of a certain radius.
- If the density is less than a given threshold, the points within this sphere are removed. This algorithms is quite straightforward, however it was shown that it requires a significant amount of time to process the data, we had.

- Therefore, we decided to use Statistical outlier removal (SOR) filter, which also controls the minimal density of the point cloud within its regions by the means of the statistical deviation properties of the point cloud. This method has been proven to be fast and reliable enough for us to use it in our final approach.

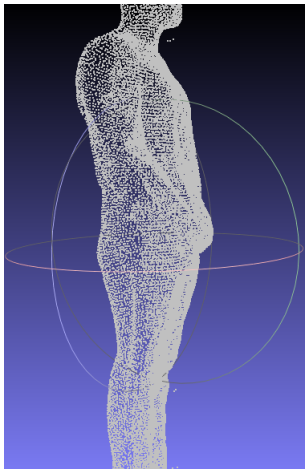

Figure 7: Filtered Point Cloud

7.1.4 Downsampling

When it is about ICP, there is always the question of stitching speed, and, as we know it strongly depends on the amount of data that is being processed. So at some point there is a goal to reduce the amount of data as much as possible without significantly loosing the quality and details of the point clouds. In order to achieve that we have used downsampling. Its main principal can be described as iteratively reducing the number of points in the cloud by replacing the points that are contained inside a cube of a given size(leaf size - it corresponds to the level of the downsampling) with the centroids of these points. It is also clear that the stronger downsampling we perform, the more details are lost of the initial data. So at the end we decided to use a soft downsampling, that did not really affect the quality of the result.

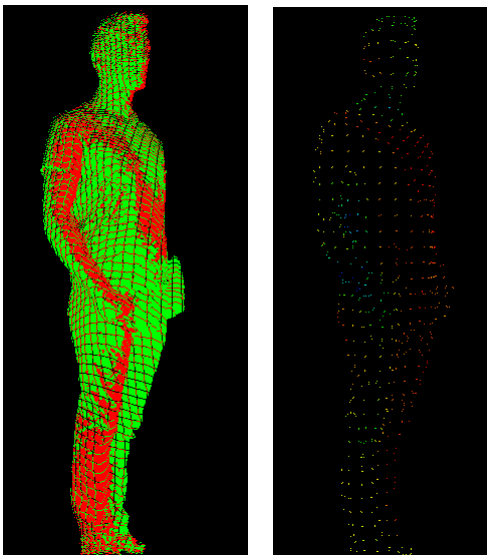

Figure 8: Left : Orignal Pt Cloud , Right : Downsampled Pt Cloud .

7.2 Chosen technique

7.2.1 Iterative Closeset Point(Our Approach)

The technique we used for Aligning Point Clouds (Finding Optimal Rotation and Translation between Corresponding 3D Points) is on the basis of iteration and Maximum Correspondence Distance between points of the Point Cloud.We Register the Point Cloud after Initial alignment and then we perform Concatenation which allows to join the point clouds, It is done in a loop so we use global Registration, Local registration can also be done to get better results but with different techniques of ICP ot the one we proposed, Once all point clouds are registered and concatenated we create one point cloud that is the combination of all the point cloud that is the later used for 3D reconstruction.We use light Statistical outlier removal filtering in between concatenation of point clouds in order to remove outliers and get rid of noise around the point clouds,which appeared during grabbing of point clouds or during concatenation of point clouds.This is the summary of the concept we used in ICP .Below here, we explain the whole process in detail for better understandings with solutions to the problems we explained above.

7.2.2 Technique used (Our Approach)

The basic working of Iterative Closest Point (ICP) in our case is to takes a vector of point clouds which are stored after acquisition through kinect v2, It can be a vector or directory of point clouds.

In our case after data acquisition the point clouds was stored in a directory and we took all the point cloud data we had in that directory as an input to our ICP alignment function,

- The first point cloud of the vector/directory is taken as source and the second point cloud of the vector/directory as Target. We downsample both the source and the target in order to speed up the process as been explained above downsampling Algorithm. Downsampling is done very lightly on each of the point cloud we had , if the downsampling is done at larger leaf size, although it will increase the speed of execution but at the same time it will remove most of the information from the point clouds which will result in losing key feature of the final Point cloud.So once the source and targets are downsampled lightly .
- ICP is performed using iterative closest point approach, this approach is used to minimize the distance between all the points in cloud 1 to all the points in cloud 2 in order to minimize the distance between two point cloud after every iteration once the minimum distance is been found depending on the termination criteria the transformation is estimated based on Singular Value Decomposition (SVD).

7.2.3 Termination Criteria

The algorithm has several termination criteria such as setting maximum iterations.

- If the number of iterations has reached the maximum to the user imposed number of iterations (via setMaximumIterations) the algorithm will stop as it exceeds the set termination criteria .IterativeClosestPoint provides a base implementation of the Iterative Closest Point algorithm.
- The algorithm have also the epsilon(difference) termination criteria if the epsilon (difference) between the previous transformation and the current estimated transformation is smaller than the user imposed value (via setTransformationEpsilon) it will terminate else it keeps running till it reaches to the user defined value.
- We can also compute euclidean distance for finding best transformation so we apply ICP algorithm and we get the best transformation , We then apply the estimated transformation to the source in order to align source with the target, once the clouds are aligned simple step is to perform concatenation which add up both the point cloud and store them in another temporary point cloud of the same type as one cloud, which can be called as result we filter the resulting point cloud
- In order to remove outliers and noise which appears during concatenation. We use statistical statistical outlier removal for this purpose which works on the principal of deviation from the mean we set a mean value and standard deviation value if the point cloud is in between sum of mean value and standard deviation ($80 + 7 = 87$) it will remain as it is, if the sum of both lies outside the threshold we set it will take those points as outlier and remove it.

- Once we have the result we take the result and pass it as in input source cloud in next iteration and we take another point cloud at index three as target we perform same operation we get result of both point clouds we concatenate them, again we get one result which we take as source and take other point cloud as target do the same operation in a loop till the end of the size of point cloud datasets we loaded from vector/directory ,at last we get one point cloud which is combination of all point clouds we had,
- In order to tune/smooth the point cloud a little we apply Moving Least Squares (MLS) filter which is not only used for smoothing the point cloud having noisy data but also remove occlusion the place where kinect was unabloe to take point clouds so on the basis of resampling depending on the neighbour hood points missing points are created specially where we have occlusion problems so we resample the missing points in the point cloud and get a smooth output.

7.2.4 Technique Results

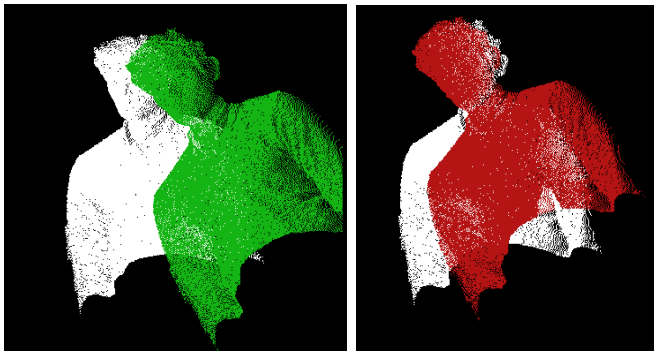

Figure 9: Left :iterations zero , Right : iterations 2

7.2.5 Best Transformation

We get the best Transformation between two point Clouds which can be checked by convergence value.If the convergence value between previous and current value is less then the set criteria of convergence it keep iterating once the best convergence value is received we proceed with the loop , for next source and target. As seen below we get a perfect overlap.

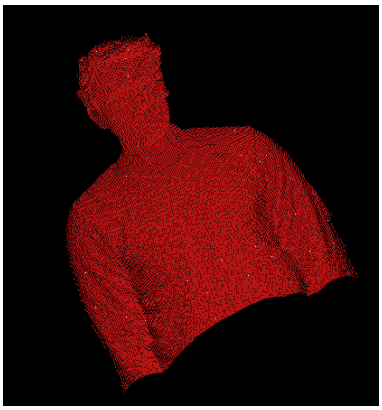

Figure 10: Full overlap between Source and Target(iterations 50)

There exist problem on shoulders, we get better Transformation for Front and back side of the Person Point Cloud as there is more overlapping but on shoulders we get worse results as can be seen below even after more then 200 iterations we don't get a proper overlap of point cloud on target as seen in below figure.

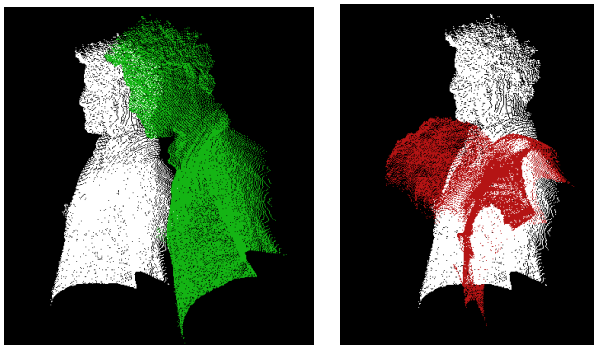

Figure 11: Left : iterations = 0 , Right : iterations = 200

7.2.6 Technique Solutions

The solution we propose to the above mentioned problem after analysing results is as under.

- More datasets/Point Clouds should be taken on Shoulders will result in More overlap of point cloud with More Correspondences and better transformation.
- Removing Noise before applying Transformation.

- The data should be taken with Better Resolution not further than 2m.Else for complex/minute object details will appear like noise.
- Less movement of the person while acquiring the data from kinect. V2.
- Aligning or registration done locally.
- We also remove NAN points from the point cloud as it can gave worse output results.

7.2.7 Filtering

Depending on the results filtering can be done before or after ICP, Reason for selecting filtering before and After are as under.

- Filtering can be done in between Point Clouds before each loop iteration .
- We can filtering source and target Before finding Transformation between them and we apply ICP for alignment.
- It can be performed After getting the final Concatenated Point Cloud as we do in our case as well.
- Moving Least Squares (MLS) is a special filter that could be used specially for removing occlusion.
- Outliers could be removed using a Statistical Outlier Removal filter.

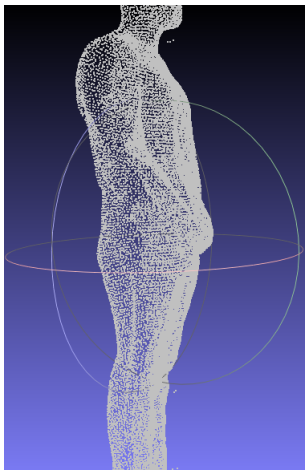

Figure 12: Filtered Point Cloud

Figure 12 above shows the result we got after filtering the Point Cloud.

7.3 Technique improvement

The technique we used is basic one it can be improved in many ways , but the more the improvement the more the complexity we have worked on all improvement parts but we were not getting our desire results so we stick to the basic approaches and we got better results, The technique can be improved by the following points below

- Feature estimation on the basis of colour as we use PointXYZ type of point cloud so it doesn't have colour, So feature estimation could be used with PointXYZRGBA type of Point Cloud .
- We also got some problem of extra layer on the final concatenated point cloud , which can be solved by Explicit Loop Closing Heuristic (ELCH) .
- Choice of sensor was 2 better sensor could be used in order to get better results.
- The algorithm itself could be improved by adding few more functions for better alignment .
- Normalization could be used to better results and Transformation.

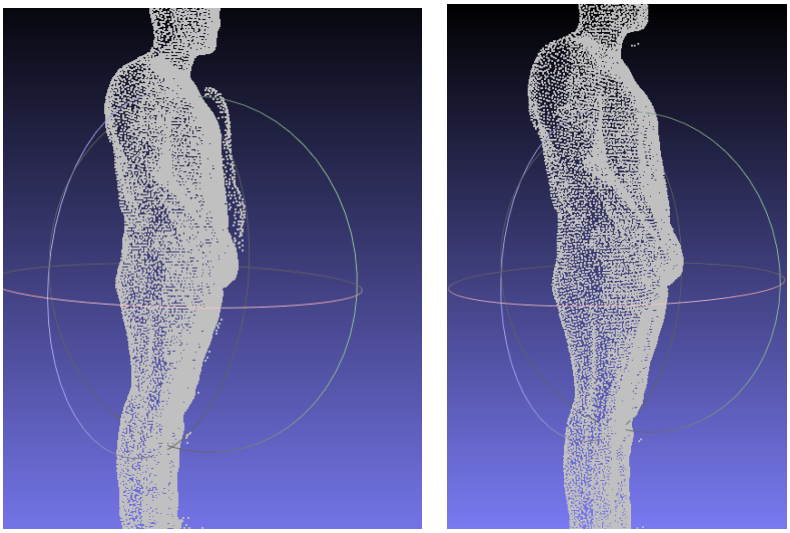

Figure 13: Left : Orignal Pt Cloud with Extra layer Right: Pt Cloud without Extra layer.

Figure 13 above shows the final result we get in ICP after solving outlier and extra layer problem.

8 3D Reconstruction

8.1 3D Reconstruction Methods

In this section, the theoretical as well as the implementation details of 3D reconstruction will be presented. This section is divided into two subsections, each dedicated to the explanation of the two meshing methods: Greedy Triangulation and Poisson Method.

We decided to implement both the meshing methods in the project and give user the choice. The user has the choice to implement both the algorithms and save the better result.

8.2 Greedy Triangulation Method

One of the reconstruction methods implemented in our software is called the Greedy Triangulation. As the name suggests, this method is based on the construction of triangles in a plane and in our case, these planar triangles are made on a set of point clouds. A formal definition of the GT is: "The triangulation is obtained by starting with the empty set and at each step adding the shortest compatible edge between two of the points"[**?**]. This means that the algorithm passes through all the points on the point cloud in order to create edges between two points, chosen such that they do not intersect other edges.

This algorithm is focused to connect the majority of the points to be able to build triangles and then create a mesh. One of the advantages of GT is that it allows us to work with unordered/unorganized data. In our case as well, we work with unordered points taken from several scans of a subject. However, the big disadvantage is the lack of "watertight-ness" (a condition in which the mesh has no holes) as opposed to the watertight mesh produced by the Poisson algorithm (it will explain on the next section).

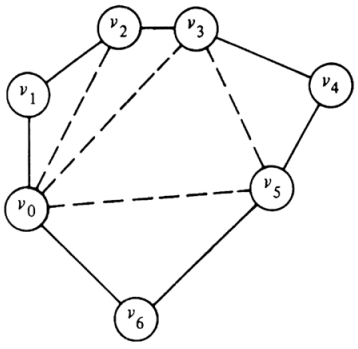

Figure 14: Illustration of the creation of triangles with GT

In the figure above, it is shown how the triangles are generated from the GT. By finding the edges (Lines -) and the vertices (V0 to V6), it is creating new edges to form the best triangulation in order to obtain the reconstruction of the surface.

Moreover, the quality of the reconstruction can be improved be proper manipulation of the variables that we can modify in the algorithm. For our GT class, we use the following variables (listed with the default parameters and their respective definition obtained from PCL documentation [PCL-documentation]:

- SearchRadius (0.8).-Maximum allowable length of an edge of a triangle.
- Mu (10).- Density parameter.
  Comment: It is multiplied with the distance of the query vertex with the nearest vertex to get a new search boundary for the algorithm. This enables the algorithm to cater for the sparesly populated areas in the mesh. These two parameters together control the size of the neighborhood.
- MaximumNearestNeighbors (20).- maximum neighbors taken in account for creating the triangles.
- MaximumSurfaceAngle (π).-The angle between the normal of subject point and the self point
- MinimumAngle ($\pi/10$).- It refers to the minimum angles in each triangle.
- MaximumAngle (π).- It refers to the maximum angles in each triangle.
- NormalConsistency (false).- It determines if the normals are oriented consistently

The default values were taken from a point cloud of a person. These values vary according to the size and form of the point cloud. The results of this method without applying any filling hole algorithm is presented on the section 4.

8.3 Poisson Method

The second reconstruction method we implemented in our software is the Poisson algorithm. This techniques was created by Michael Kazhdan, Matthew Bolitho and Hugues Hoppe in 2006, which makes use of the Poisson equation. $\Delta\chi = \nabla\cdot\vee$. This equation is used in many different research fields e.g. for studying propagation etc.

As the explanation of the equation is extensive and a bit complex, and it is not the major approach of this paper, we will briefly define this method as: Given a set of oriented points sampling the boundary, a watertight mesh can be obtained by transforming the oriented point samples into a continuous vector field in 3D, finding a scalar function whose gradients best match the vector field, and extracting the appropriate isosurface. PUT HERE THE REFERENCE. To understand it better, the following figure illustrate the phases of Poisson algorithm in a series of 2D point sets.

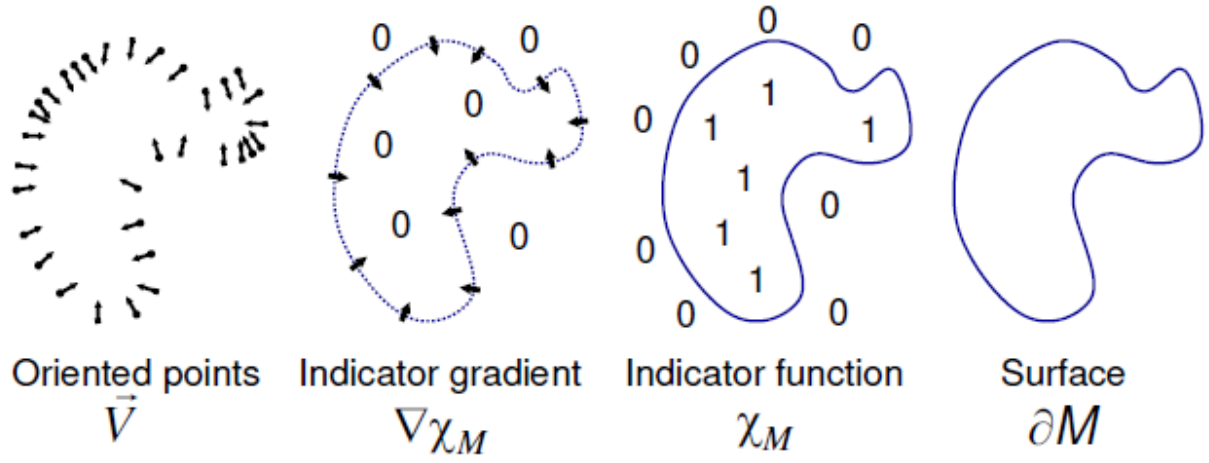

Figure 15: 3D Illustration of the Poisson reconstruction in 2D.

So, as it can be seen, this algorithm is famous for the creation of a watertight surface reconstruction with the use of oriented sets of points.The key parameter in this algorithm is the depth of the octree (number Poisson equations used in the reconstruction), this parameter is a value between 1 and 14, that increase exponentially the number of reconstructed faces but also the processing time. A good compromise between them is to have a value around 9 or 10, that give a decent result in few seconds. In the following figure, the effect of using different parameters can be observed.

Tree Depth	Time	Peak Memory	# of Tris.
7	6	19	21,000
8	26	75	90,244
9	126	155	374,868
10	633	699	1,516,806

Figure 16: Description of the effect regarding the value for the Depth of the octree. The time is in seconds and the peak memory in megabytes.

However, the Poisson algorithm is quite sensitive to the noise and to complex forms of the point set creating over-smoothed mesh in some areas. A good estimation of the normals is needed to have a well-reconstructed mesh and a noisy point cloud can orient the normals in a wrong way. The problem with the complex form is linked to the narrow spaces that are often regroup as one single space without holes by the Poisson equation. In section 4 is presented the comparison with 2 different forms.

9 Software Implementation: 3D Reconstruction Class

In this part of the report, we will present how the 3D reconstruction class in our project is built . Like almost all the other classes in our software, we are using the PCL Library and its functions, which provide us many functionalities needed to create this class. Moreover, this class is derived from the Database class as explained before (See its description, page). The code for this class can be found on Github at this address:

Link Github Link for 3D reconstruction

9.1 Description of Functions

In the class, we create a "global" function and three "tool" functions. The "tool" functions can only be called by the "global" function, depending on what the user want to do with his point cloud. These three "tool" functions are 'protected' because we don't want to let them be available for calling from other parts of the software since it may corrupt the data. Additionally, as we might also want to upgrade the functionalities of this class using these functions, we may need to derive another class from this class. The "global" function is 'public' so that we can use it in the GUI class.

Below is an explanation of the "tool" functions, followed by that of "global" function.

9.1.1 The Normal Estimation Function

The first "tool" function we have created is designed to get the normal estimation of all the points from a point cloud. The normal of the point represents the orientation of each point. This estimation is mandatory for the two reconstruction techniques we implemented in our project (Greedy Triangulation and the Poisson algorithm).
This function requires a PCL PointCloud templated for a XYZ PointCloud (a point cloud with only the position parameters) in input and it return a PCL PointCloud too but templated to return a Normal PointCloud (same as the XYZ one, but with more parameters to store the normal values).
We composed thos function by using the NormalEstimationOMP function [Link] from PCL. We fixed the search parameters for the estimation and we computed the centroid of the point cloud too, to have a better approximation. When we have obtained the aforementioned values, we compute the normals. Then, we reverse them (changing the sign of all the normals) to have them all in the good direction. At the end, we just have to concatenate the input point cloud and and the normals to form a Normal PointCloud, the datatype needed for the two reconstruction methods.

9.1.2 The Greedy Triangulation Function

The second "tool" function is built to get a 3D reconstruction of the input Point Cloud using the Greedy Triangulation method, a basic algorithm used to create triangles in order to build the mesh.
This function requires the Normal PointCloud, obtained by the Normal Estimation function, and it returns a smart pointer containing the reconstructed mesh in the PolygonMesh datatype from the PCL library. The smart pointer used in the algorithm is declared as "boost::share_ptr". It is important to use this type of pointer because it leaves multiple smart pointer that can point to the same object at the same time, this is well known as a share of ownership, and also we are sure that it will delete the object only when the object is not used anymore.
Like the previous "tool" function, we composed it using an instance from the GreedyProjectionTriangulation class [Link] from PCL. We create many parameters to adjust the final mesh according to the input. All the parameters are declared in the Database class (mother class of the 3D reconstruction one). This is the list of all of them:

- SearchRadius parameter: maximum allowable length of an edge of a triangle
- Mu: density parameter
  Comment: It is multiplied with the distance of the query vertex with the nearest vertex to get a new search boundary for the algorithm. This enables the algorithm to cater for the sparesly populated areas in the mesh.
- MaximumNearestNeighbors parameter: maximum neighbor taking in account for creating the triangles
- MaximumSurfaceAngle parameter: angle between the normal of subject point and the selected point
- MinimumAngle parameter: minimal angle for a triangle
- MaximumAngle parameter: maximum angle for a triangle
- NormalConsistency parameter: set a flag if the normals are oriented consistently

These parameters can be modify on the GUI by the user (See the "Global" Function section). We return the pointer to the mesh when the reconstruction is done. Some resulting meshes are presented in the Results section, later in the report.

9.1.3 The Poisson Algorithm Function

The last "tool" function is the Poisson Algorithm one. It computes the 3D reconstruction of a point cloud using the Poisson method, a way to build a watertight mesh thanks to the Poisson distribution.

Like the Greedy Triangulation function, it also requires the Normal PointCloud, obtained by the Normal Estimation function, as input and returns a smart pointer containing the reconstructed mesh in the PolygonMesh datatype.
We use also for this function a class from the PCL library too, the Poisson one [Link]. There is only one parameter that can be modified by the user because we set the other ones to the default value from PCL, because they are less important for the reconstruction or they are already tuned for our applcation. This parameter taht we can change in our software is the octree depth which proportionally increases the quality from the mesh and the processing time, (see the presentation of the algorithm to have more precision).

9.1.4 The "Global" Function

The "Global" function is the function called when the user wants to reconstruct his mesh using our software. It will call the "tool" functions needed to respect his choice.
This function needs a Boolean as input to choose the reconstruction method (Greedy Triangulation = false or Poisson = true), because it can be linked very easily with the simple button on the interface to select the method we want. This function will return the same output as the two "tool" functions for reconstructing the mesh, a smart pointer containing a PolygonMesh datatype from PCL. This datatype can be shown on an GUI interface without difficulty or exported in a file as a PLY or PCD.
When the function is called, it takes the resulting point cloud we get after ICP stored in the Database class as the input. This point cloud will be used as input for the Normal Estimation "tool" function, detailed before. When we have the Normal PointCloud, we can use a reconstruction function. We have a "if-else" loop to check the method selected by the user and apply the corresponding "tool" function. Then, we return the smart pointer to the PolygonMesh containing the mesh.
All the parameters from the functions can be modify, thanks to a special interface we designed (See the Following figure).

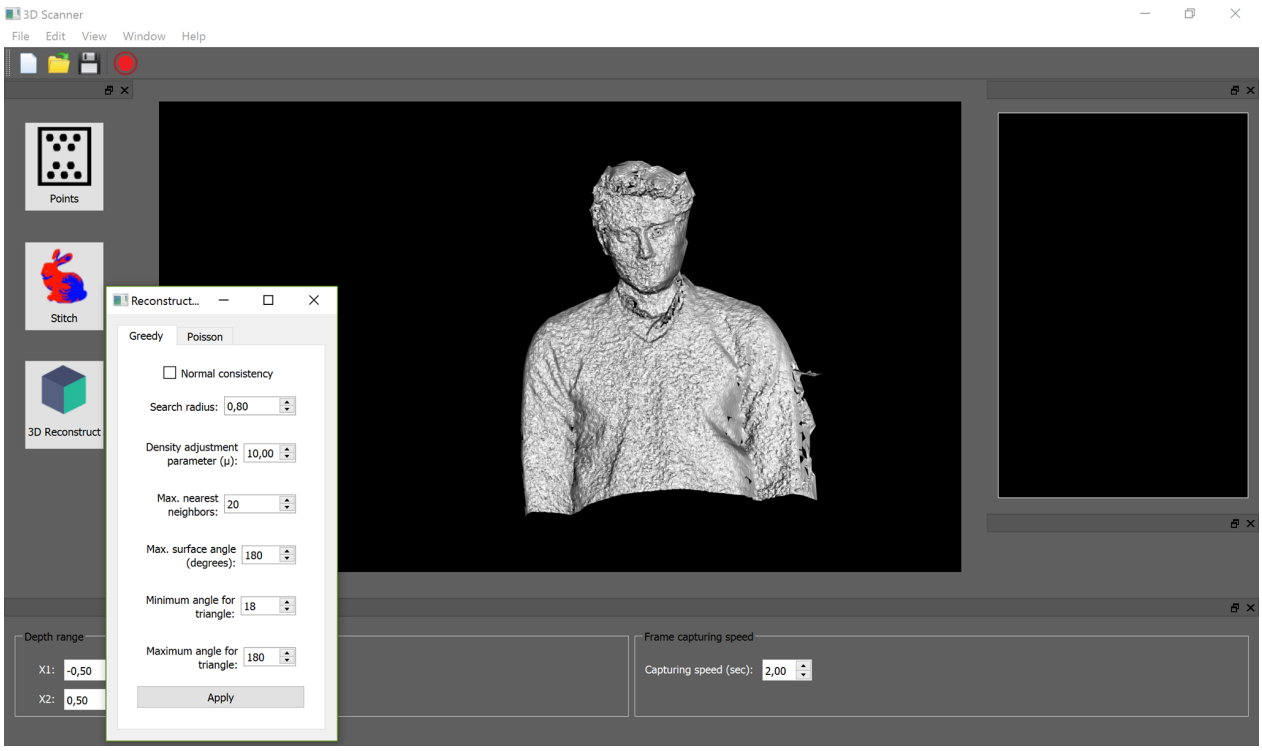

Figure 17: 3D Reconstruction Interface design by Yamid Espinel. You can easily choose your reconstruction method and the parameters to get the best result as possible.

9.2 The Constructor

This class contains only one default consructor. Here, the defualt constrcutor is a parametric constructor, since it takes an instance of the Database class as the argument. The constructor initializes all the parameters needed for the Greedy Triangulation function and Poisson function. We have set default values for these parameters which give a good result on thepoint cloud of a human form.

10 Presentation of our Results

Now that you have sufficient knowledge about the methods we are using for the reconstruction, we will present you the results we get from each methods. Afterwards, we will also introduce how we can improve them. For fair comparison, we will be using the stitched point cloud of the same person as input of the algorithm.

10.1 Results: Greedy Triangulation

On the following figures we will compare the importance of the manipulation of the GT variables. It is important to have a good understanding of their definition in order to get the best approximation to a good reconstruction depends.

Figure 18: Comparison of the usage of values on the GT variables.

Finally, we present the reconstruction result applied to a set of point cloud of a person using the default values given previously.

Figure 19: Final result of the Greedy Triangulation algorithm.

As it can be noticed, we have fairly good results in terms of the details and form of the

reconstruction . While referring generally to this method, we face with the problem of missed space i.e. holes. Therefore, it is needed to implement a hole filling algorithm to obtain the desired results.

10.2 Results: Poisson algorithm

The result for the Poisson algorithm are different from the Greedy Triangulation. First, the main difference is that we get directly a watertight mesh (without any holes) without applying a hole filling algorithm. Nevertheless, as explained before, the texture obtained is very smooth too. However, there is a huge drawback: the algorithm creates extra surfaces in the noisy part and in the narrow space of the point cloud. This is a huge problem because the type of mesh we are working on, a human body, contains a lot of narrow spaces (between the two legs, the neck etc.). The point cloud taken as the argument can also be a bit noisy depending on the result of the ICP. This method is not well adapted to our application, but we can improve the result by removing those extra surfaces via different methods which are explained later in the "Future Work and Improvement" section. We decided to keep this method on our software because if we are scanning something else than a human body, it can give very good results (see the result on the dwarf statue) and it can be improved later, if we decide to continue to develop the software.

In the following figures, we present the reconstructions of 2 different types of point clouds and the final result for the reconstruction of a person.

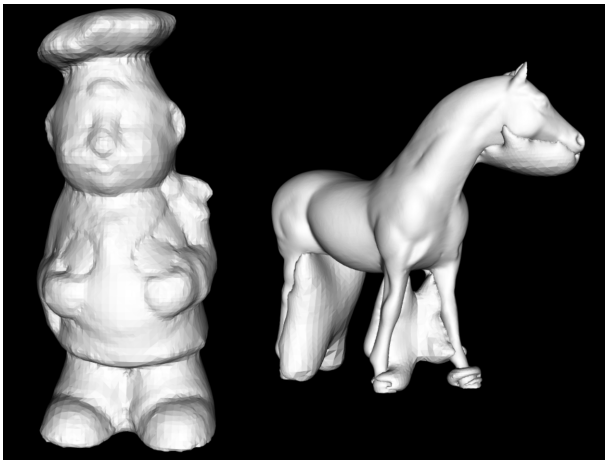

Figure 20: Comparison of two different types of point clouds.

Figure 21: Final result of the Poisson algorithm.

11 Challenges and Critique

In this section, we will explain some of the difficulties that we faced along with some critique of our work.

11.1 Management

3D reconstruction class of this software was developed by a team of three students. In the beginning of the project, four students were working on it but we soon realized that it wasn't efficient for the fourth person to be in our group and that he/she could help the group in some other respect, we decided to work in a group of three.
From the beginning of the project, we were an autonomous group with the objective of providing a functional class to do 3D reconstruction. The main problem we faced was how to take decisions (for example: which reconstruction algorithm should we used). We wasted a lot of development time by switching from greedy triangulation to Poisson and inversely. We started by developing both Poisson method and Greedy Triangulation method in parallel. Wehn we started to get good results with one, we used to leave the other and focus on one method. However, we later figured something that enhanced the other method and gave good results too, so we switched our focus again. This way, we were not able to focus much on one algorithm. At the end, we decided to quit preferring one over the other and develop code for both the techniques and let the user choose the method he prefers (the two methods have their advantages and weaknesses). Because of this mismanagement and the time lost as a result, we were really pressed for time to code a "good" hole filling algorithm to make the mesh from the greedy triangulation watertight.

11.2 Complexity of algorithms

The second difficulty we faced in the development was that we did not fully understand the algorithms in the beginning, especially the Poisson Algorithm. For us, the Poisson method is very complex to study because it's based on very advanced mathematical techniques that require a lot of work to understand how it works and how tune it to get good results. We think we didn't study too much the Poisson algorithm to get the best results (depending on the mesh, we can have extra reconstruction in the narrow space of the point cloud, see the Section on the Poisson algorithm for more explanation).
This is link to a lack of a research at the beginning where we were just eager to search for and implement the 3D reconstruction implemented in the PCL library, without taking in account the way it works. We had to study the techniques to fix the bugs but it wasted a lot of precious time.
We believe that we didn't make use of the "tool" that are our teachers as we just saw them twice to discuss about our difficulties. Although we did meet them in the later part of the project, we believe that our progress could have been expedited at a much earlier time of the project.

12 Future Work and Improvement

Finally, we will present the work that is still incomplete. These functionalities will help to get better reconstructed meshes on our two algorithms and to obtain a watertight mesh.

12.1 Improving Greedy Triangulation method

12.1.1 The Holes filling algorithm

It is impossible to ensure that the reconstructed mesh obtained after Greedy Triangulation (GT) is watertight i.e. the surface of the mesh has no holes. If there are holes in the mesh, it is impossible to 3D print it. Therefore, it is essential to perform a watertight-ness check after GT and, if there are holes, apply a hole-filling technique to get a model suitable for 3D Printing.

We read some research papers and information available on the internet regarding Hole-Filling techniques. Of all, we found the article "A robust hole-filling algorithm for triangular mesh" by Wei Zhao, Shuming Gao and Hongwei Lin extremely helpful and derived our implementation from it.

Below, we have presented two methods, the first one for checking watertight-ness and the second one is for filling hole.

12.1.2 Check for Watertight-ness

The proposed method to check Watertight-ness is as follows:

1. Read the .PLY file created after applying Greedy Triangulation algorithm.
2. Extract information of the polygons that is in the form "$a \quad b \quad c$", where a, b and c , respectively, are the index number of the three vertices of the triangle present in the same .PLY file.
3. From the polygon information, extract the edges in the form of "$d \quad e$", wher d and e are, respectively, the index number of vertices present in a single edge. In this case, we will get three edges from each polygon, which, in this case, is a triangle. Afterwards, save the edge information in another vertex for further processing.
4. A Boundary Edge is defined as the edge which does not lend itself to two triangles. Clearly, if the 3D model is watertight, we will have no Boundary Edges. To look for Boundary Edges, check for those edges in the edge information as the ones that occur only once. Save there Boundary Edges in a new vector which we will call as "Boundary Edge Vector". If there are no Boundary Edges, the 3D model is declared as watertight. Otherwise, we get to know that we shall have to apply some hole-filling algorithm to make it watertight.

The proposed Hole-Filling procedure is as follows:

1. Check for connectivity of boundary edges that make up one hole in the mesh. Parse the Boundary Edge Vector and group those edges in which any of the two vertex indices in an edge is repeated. This group shall contain those edges that surround a single hole.
2. Take a single group of edges that surround a hole and apply the Advancing Front Meshing (AFM) technique to fill that hole, as it is presented in the article "A robust hole-filling algorithm for triangular mesh" by Wei Zhao, Shuming Gao and Hongwei Lin.

12.2 Improving Poisson Algorithm

As you can see in section where we show the results of the Poisson algorithm, we get extra surfaces that are created by the Poisson method. To improve the resulting mesh, we should develop a function that remove all the surfaces that are too far from the original point cloud. When these surfaces will be removed, we may also have to use the Holes filling algorithm to make it watertight again. It will be the last part of the development of the class when the Holes filling method will be implemented.

12.3 Literature Survey and Research Work

Poisson surface reconstruction:

We studied the two paper given [1] and [2] on Poisson surface reconstruction and tried to get the clear understanding of the algorithm. For a given point cloud first step in Poisson reconstruction is to find an indicator function M(x,y,z) = a F(x,y,z) + (1-a)B(x,y,z), where M is the model of boundary, F is point inside the model and B is points out side the model. To estimate the indicator 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: S- S*G = M, and then we optimise the model M using Poisson Equation $\hat{2}$ M= $\hat{2}$ 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 size is large then the indicator function M will be so strong that optimisation using Poisson equation will not result in any improvement on the other hand if size of Gaussian kernel is small then then the indicator function will not be appropriate to get the actual surface, 2. If the point cloud data is missing some where the approximation of indicator function will not be appropriate.

13 Graphical User Interface

13.1 Detailed Process of making of GUI

We came up with a sample GUI at start of this project. We designed a user interface that is able to divide all the principal functionalities, taking as reference other 3D scanning software's such as MeshLab, 3D builder and Kinect.

The distribution of the functionalities is done in such way that it follows the basic steps of performing data acquisition and processing, as follows (this is just a sketch, more functionalities will be added and some of the already included can be modified.

Figure 21(a) below shows options for importing and saving pcd data with capturing RGB Camera, 3D reconstruction and live point cloud data linking Kinect V2 with GUI with functionalities mentioned below. Figure 21(b) below shows options for setting Parameters and process which can be done in order to build a 3D model

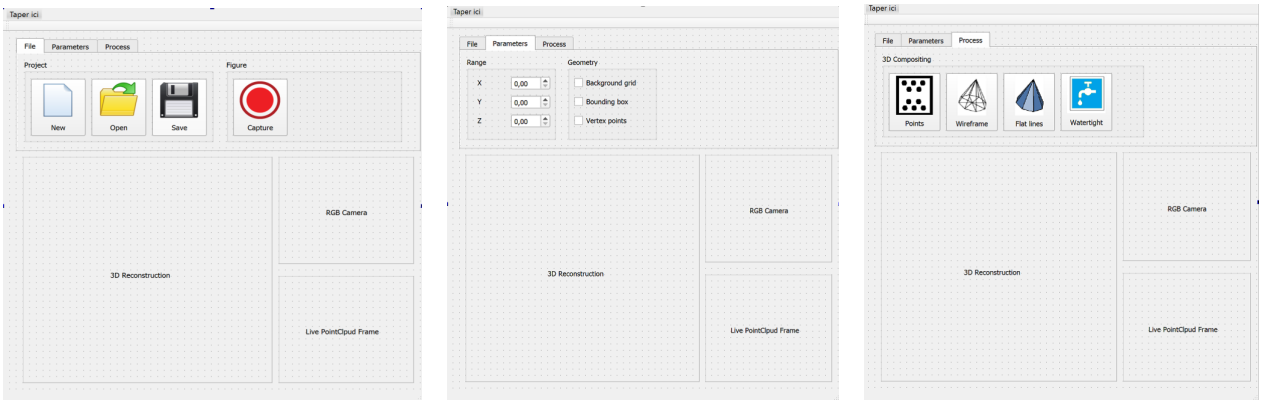

Figure 22: Initial GUI Design Functionalities

According to the sketch, the functionalities to be included into the program was meant to be classified according to the common order of work for 3D scanning applications. The idea also was to show in real time the 3D reconstruction of the object/person to be scanned, along with other useful information such as the bounding box, live point cloud, and different sensor information from the Kinect.

13.2 Streaming through GUI Qt

GUI Advancement VTK and Kinect Integration, we tried to integrate the VTK libraries inside the general GUI by means of the QVTK plugin provided by the VTK developers. This gave us the capability to show and manipulate from simple point clouds to complete 3D models inside the program's interface. However, such integration turned out to be difficult as the usage of those libraries required to compile the source code from scratch by using a full Visual Studio 2015 setup, the search of several external windows SDK dependencies, among other configuration issues. After several attempts we successfully compiled the VTK libraries with Qt support and of gathering all the numerous dependencies, we proceeded to integrate the QVTK plugin into the GUI, along with some controls to manipulate the loaded point clouds:

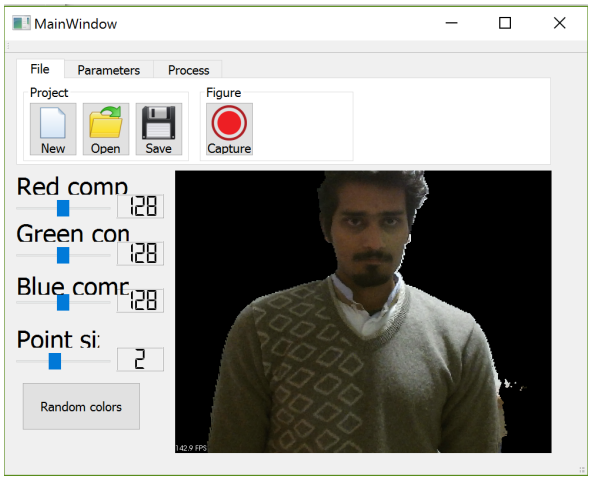

Figure 23: Point Cloud Loading QT QVTK Widget

As seen in figure 22, after clicking in the Open button and selecting a .PCD or a .PLY file, the point cloud data is loaded and shown through the QVTK plugin. By clicking on the figure and dragging it, the user is able to move, rotate or zoom the model.

As a next step, we tried to integrate the live Kinect streaming into the GUI. We make use of the available Kinect libraries as we intend to connect them also to the QVTK plugin in order to show the streamed data. The filtering-by-depth feature is also used, by setting a range in the Z edge from 50cm to 4.5m. After adding and configuring the corresponding libraries, we compiled the entire project successfully.

13.3 GUI Advancement VTK and Kinect Integration

We were now able to interface the Kinect with the GUI to perform live streaming of point clouds. The code was arranged in such way that the connection and streaming from the Kinect was activated by pressing the “Capture ” button. Once running, the live point clouds can be manipulated by using the controls on the left; for example, if the user moves the slider for Point size, the figure in the screen becomes bolder or lighter. Nonetheless, it was also possible to change other parameters in real time such as the filtering by depth, which lets the user to check what the best range of acquisition is.

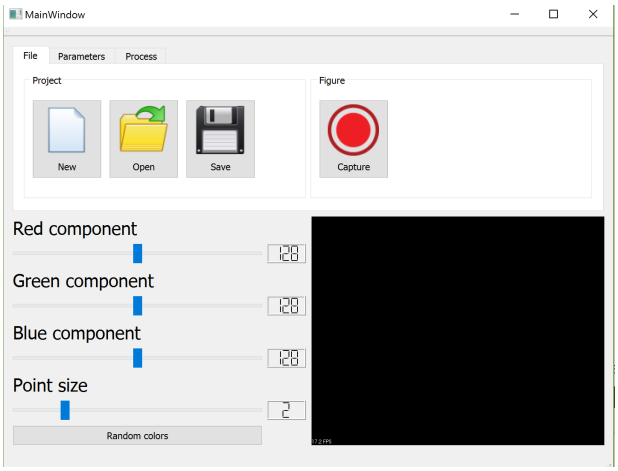

Figure 24: QVTK Widget QT

The application was also capable of exporting point clouds into .pcd files every certain time, files which are now used for performing stitching of the point clouds.

According to the suggestions from the group, the GUI was now 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 oat around the screen. The color of the interface was also been changed to gray, making it easier to work with the 3D models. In the Figure 24 It can be noted how the elements are grouped in the different docks attached around the visualization widget.

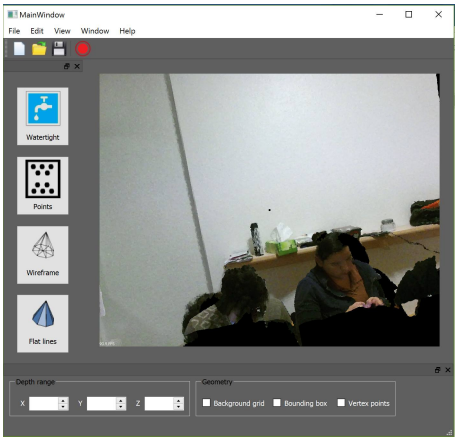

Figure 25: Real time Acquisition QVTK Widget QT

As mentioned before, these docks and toolbars can also be detached, leaving the visualization window alone to cover the whole space of the screen.

In Figure 25 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 resizable and also be definitely closed; a menu bar 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.

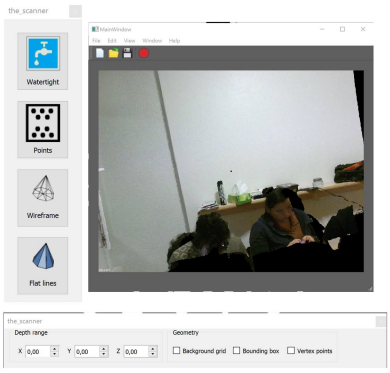

Figure 26: Real time Acquisition QVTK Widget QT

Now with the interfacing of the Kinect sensor within Qt. The goal now was to show the infrared sensor, the depth the RGB camera inside the program's environment, letting the user to know all the information about a scene and allowing him to take better scans of an object. There were two ways to display the sensor data: The first consists of two or three new dock able frames that will appear only when the user clicks the Capture button. Like this, the user will be able to see the state of the current scene along with the live point cloud acquisition. The second approach consist leaving one visualization window and giving the user the possibility to choose which information to display inside the widget. We also processed the redesigning of some of the icons for the post-processing dock, seeking to include the following techniques:

- Fusion through ICP
- 3D Reconstruction through Greedy Projection
- 3D Reconstruction through Poisson

For the 3D reconstruction techniques, we were designing a small popup window that will be displayed each time the user clicks on the respective button, and in which he can configure the different parameters for either algorithm. This will help the user to obtain better results according to the complexity of the scanned object. The application is now able to generate .pcd files from a variable containing a point cloud data set. As long as the acquisition process is complete (the live stream of point cloud data from the Kinect was stored in a variable), we integrated it into the general application.

13.4 Detailed explanation of each section of GUI

- **New Project:** This icon will erase all the data present in the dynamic memory of the project and will create a new project for a new 3D model
- **Open Project:** Using this icon we can open any existing project (3D model) and can edit it and save it in the existing project or as a new project.
- **Save Project:** Using this icon we can save the code for a new project or for an existing project whose code is being modified.
- **Play Icon:** Using this icon the computer starts receiving data from the sensor (Kinect) and starts displaying it on the QVTK Widget.

Figure 27: Buttons Layout

We can set the filtering of the object according to the size of object as shown in the figure 27. We can modify the length, width and height of the filtering box by just modifying information in this widget.

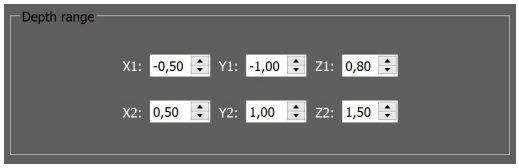

Figure 28: Filtering Coordinates Window Parameters Adjust

- Start recording: As soon as we press this icon, the code will show the live stream as well as it will start saving the images into a vector. We can even adjust the speed of capturing the image of the data using this widget shown in fig. 29.

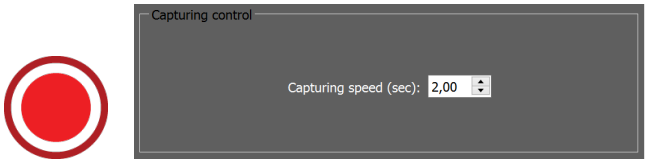

Figure 29: Recording Data Enabling Capture

- Show Point cloud: As we have the images now and we want to convert all the images into point clouds, so we will use this icon and convert all the images into point clouds
- Stitch point cloud: Now we have a number of point clouds as shown in figure 30 computed from pictures captured of the object from different angles. We can use those different point clouds of the same object and can stitch them together using the nearest point estimation technique (Furthermore of point cloud stitching is explained in the point cloud section).

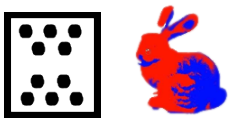

Figure 30: Performing ICP button

There are certain parameters which we have to set before starting the stitching process (As shown in fig 11).

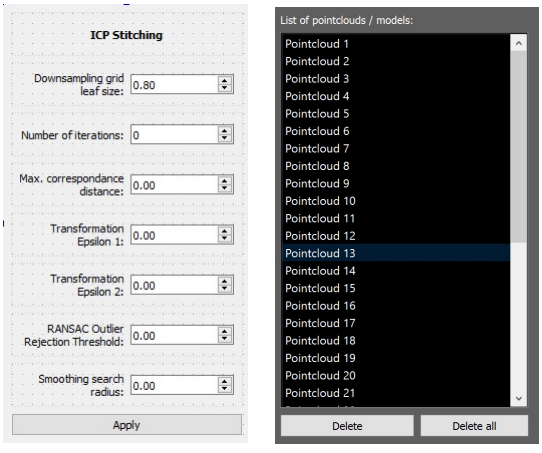

Figure 31: ICP Parameters Window

After we set the parameters and apply these settings the code will automatically start stitching the point clouds and after sometime a final 3D stitched point cloud will appear

on the screen whose point size can be varied from the slider shown in fig 32.

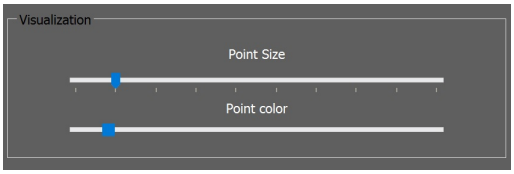

Figure 32: Point Cloud Size Adjustment Window

- 3D Reconstruction: In 3D reconstruction, we have to now fill the point cloud. We can do this in two ways.
- Using Poison Algorithm
- Using Greedy Algorithm

We have implemented both in our code as shown in fig 14, so whenever user has a stitched point cloud he can use any of the algorithm (either poison or greedy) and can create a 3D model of the object. The 3D model of the stitched point cloud will then appear at the QVTK Widget.

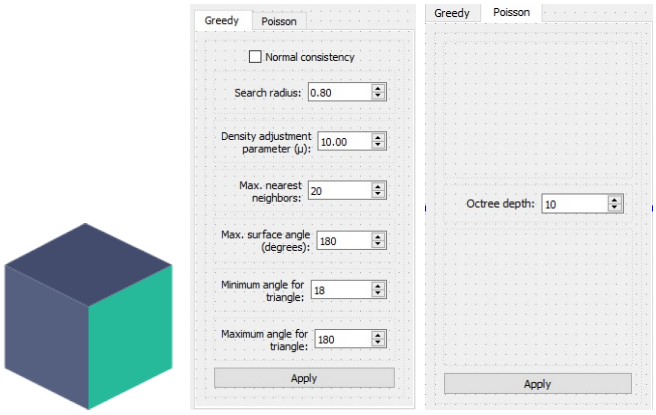

Figure 33: 3D Reconstruction Parameters Window

As seen above like ICP the parameter for 3D Reconstruction can also be set.

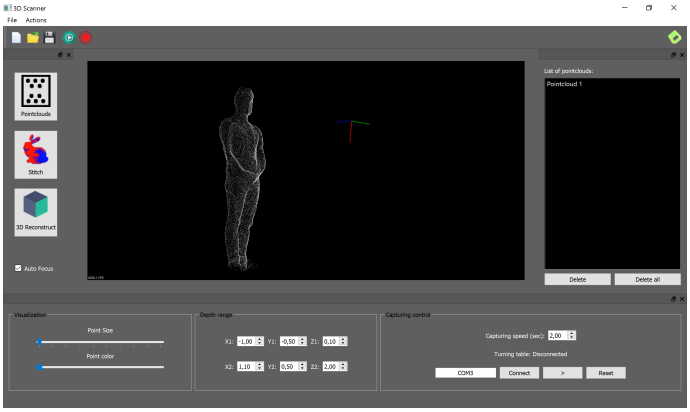

Figure 34: Arrangement of elements inside the GUI

14 Project Management and Skills improvement

14.1 General Transmission

By general transmission we are talking about answering the questions asked during the whole project and also being able to solve problems that include a special knowledge of the language or the general knowledge of coding.
The group had to develop programs and functions to have a consistent project and lot of questions were asked about the coding and the way of doing things, the possibilities or just explanations of the code available on Internet. This was not the biggest part of the transmission but it was really important to be able to answer almost every time to any questions possible without entering in big details (sometimes because it was not useful and not to overload the explanations but also simply because we hadn't the knowledge to answer the question).

14.2 Lectures

We proposed lectures for students of the group that was behind the others. Sometimes everything seems to be easy for lot of people but this is not the case for everyone. And the major problem was to identify the problem but also to convince people that it was useful for them but also for the group. Very simple notions that was already explained in class were explained such as Classes, Templates, Functions, Pointers, Objects...
We can say that the most difficult part was to find people interested in these lectures.

14.3 Point of view added to the project

In fact everyone were asked to add their ideas, their knowledge. From background we have a different point of view about project management but also and more than other on the code. Teachers and Companies always tell that the code is half of knowledge and half of rigor. And by that the rigor is induced. And this was also one of the biggest difficulty, trying to explain everyone that rigour is the most important things and we had to take in account that we were 12 and the rigor is also what avoid miss-understanding and the problem of joining the code.
A good code is not a working one, in fact this is secondary because something can work or can look like it works but this is not the case. The code should be optimized, understandable, adaptable and portable. Even now, part of the group still disagree with this point of view but this is the problem in group working.

14.4 Standard of Code

The standard and the usage of the code, this is a big part and also the most Misunderstood and not applied part.

- To recall briefly, standard of the code is like a language, to be consistent we have to all speak the same language. In fact the standards are this kind of process.
- It avoid problems with names, with formatting, indentation and it permits to be able in one check to understand the meaning of one function and / or one variable.
- These standards are useful because it is how every teams are working and there is a reason. This makes the code understandable. If 5 developers are coding in 5 different ways and writing in different manners, then at the end when we pack and join everything, there will be no coherence. That's why we proposed to adopt a standard of the code and to change all our names of functions / variables. We can't give the same prefix to a members of a class and an input of an user, and we have to differentiate them.
- We have to understand everything in one look, we have to comment everything that is not a logical function or that is too long. We need to understand the process of a function by the name and also by the comment before the function implementation.
- In a first look we can imagine that it is a waste of time, but if we have to take the code back in 5 years or if someone new to the project is added in the team, he will spare hours of work for decoding the way of seeing and coding of everyone.

- The other meaning of the standard of the code is to spare time also for the corrector, it is not easy and manageable to correct a code that is not compact and understandable with the same work flow.
- The usage of the code is an other part and this is more about the usage of the C++ language. Sometimes functionality is already implemented, some process are already optimized (the STL for example) and by sparing lines of codes, by avoiding some non-optimized lines, we think that the running time of the code is smaller. Sometimes it is useful to reinvent the wheel to understand the process but sometimes and especially in this kind of big projects, we don't have the time for this and also we had to keep in mind that we have a running time imposed.
- Some people don't get the usage of the standard of code, and more than half of the group isn't working with this way of thinking and coding. But we had to explain the standard of code and also suggested one, like this everyone had a model to work with. As example a member of a class should start by m and a user input by u .
- Avoiding the big characters to adopt the underscore.
- Concise and clear function names and variables.
- This was a way to have importance in the group but also to help the group. When we packed the groups function and codes, we had this problem still with standard of code and we tried to change every variables and function names to stick to this standard, for the respect of the chart of the group but also for the corrector and maybe fur future developer.

14.5 Class Development

The separation of the code is an easy part but not useless. Separating everything in functions instead of keeping everything packed into a same file, this is not how we should develop codes. We need functions and this was also in a mind of adding the functions into a class at the end.

- Classes are very important, this is what differentiate good and bad code and this is what add the a project the three missing notions, optimization, portability and adaptability. We had to pack our code into one single project and it could be done in lot of different manners but the best one is the classes. Instead of having lot of things in one file for headers, one file for code and one file for running, it is better to have every big parts in dedicated files and group of files. Like this if we need one of our process in a future project, we just have to take it from the 3D Scanner and copy paste in or project, this will work. It's a way of divide and conquer, we divide the problem to solve it. We divided the project in small part, groups were asked to work on it, and after everything was done we had to stay coherent so we choose the classes.

- In one call and good inheritances, we are able to make the magic happen and the running code is very light, only call of classes.
- Class development can be separated in parts. The first and biggest task was to modify the code of everyone in classes and to make them communicate in each others. In this idea, we pushed interaction between classes, and restricted this by using a Database class which was the idea of the group. Like this the classes have no access to them but only access to the Database class and like this we created a kind of hierarchy that was optimized and understandable.
- For this, we started creating a Database class and we choose to give access to this database class to every classes in the code.
- The database contain all the variables that are needed in the code, starting from the more easy one to the most complex. And we can see something interesting, database class avoid constant or global variables. Like this we were able to initialize the variables independently and all the classes can have action to these members of the database class using accessors and mutators.
- We chooses also to put every access to public in every classes but this is not necessary, in fact we had in all the constructors of each classes as input, our database object created before every process. And their is a reason, we can link as many process as we want to any databases if more than one database is crated. Like this the usage of classes is really powerful. We can have multiple applications running in the same time, in different orders, just by creating different objects, making everything act on different databases.
- The other class don't act as main actor of the program, they just work as mutators and functions to modify the content of the linked database class.

14.6 Conversion of group programs into classes

As we had separated groups for tasks, we had different projects for each group. A big part of the development was to pack everything and by this we mean making every project become a class.

- In order to accomplish that, we formed a group to adapt the project into classes, by the same way we adapted in our standard of code many codes and we added inheritances and functions separations. Like this everything and every program can act as a partial mutator of the members of the database class.
- First task was to identify the members and the common variables of classes. We developed the database class in the same time as adapting the projects to classes. Like this we could establish the link between everything and add accessors and mutators to the database class and also the usage of them into the new class developed.

- This tasks require a good understanding of the code and also a backward view of the code, we had to make the code light, understandable, portable and the most optimized possible.
- As mentioned before, we chooses to include the database as link in every constructor of classes, like this we can have many process running in the same time on different databases, that's also why we use the name mutator or database member, because these classes have no real power without data to manipulate. We could just have one class 3D Scanner with all the functions and it would work perfectly but the difference here was to separate each parts of the process of a scanning, like this the code is more understandable and also more professional.

15 Log Class

- Even if this log class is not working with our program, it is still a part of the development. The principle is simple, having a class that manipulate a specified file, to output and write the process of the program including the time at each time, and in the same time, writing in a console the same messages.
- The logs are common in the development process as debugging tool or just to be able to keep track of the history of the program.
  Even if there are many possibilities to have already built libraries, the choice was to do a special one, developed by ourselves. So it uses processes to manage file, writing and overloading functions.
- As specified before the class is not working in our project for one specific reason, the computer that run the program have maybe wrong installations of SDK's and Visual Studio.
- The class have still many things that we can implement and maybe some bugs but we choose to stop the development of this class to be more focus on other things and function that can improve in a better way the 3D Scanner Project. This class was only an add of functions but till we haven't used it for the development process and we were building this class after building the whole project, there is not that many advantages to develop such this functionality.

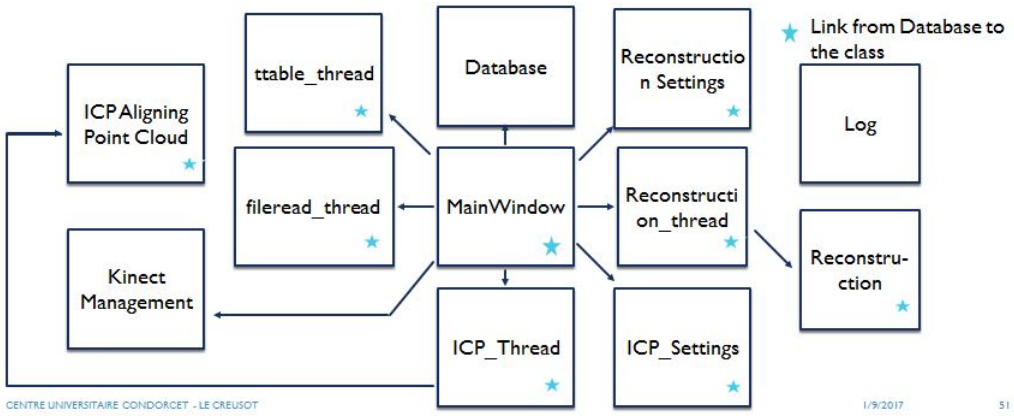

Figure 35: Database Structure

16 Management

Managementg is an important part of the project so it need lot of motivation and hard work, but was done properly with the help of hole team suggestions.

- Improvement in the Organization of Team Management during the Project by Setting weekly small goals and deadlines in order to achieve big goal.
- Assigning Weekly Tasks by arranging Weekly meetings (to reduce Communication gap between group members and to make them up to date about the task being done by the group members that every one remain on the same level). Weekly and Individual Progress Report to track the Performance of each member.
- Managing GitHub in order to have a backup of the code with every member knowing about the Progress.
- Forcing people to use Latex as it was a compulsion.

  Soocial communication networks were used for fast communication (Facebook, WhatsApp etc.)

16.1 Final Team

This team structure is the one on which each member worked during last one and a half month before that there were different teams different people in different tasks with different goals and deadlines.

Different Sub-Teams:

Team Manager : Wajahat Akhtar Acquisition and Filtering: Zain ICP : Lev and Wajahat 3D reconstruction : Maria del Carmen, Omair and Thomas GUI and Structure: Yamid, Mohit and Marc Research: Yuliia and Utpal we Prepared some backup plans when we saw someone is lacking behind we as group tried to help him out to help his solve the problem , if still that task was not complete more human sources were used to finish the task. Weekly meetings helped alot to finish our task in time and allowed us to write weekly reports. Problems we had were handled between the group by asking all the group members about there view.

17 Conclusion

In conclusion we would like to say that Main Goals were achieved , we were able to scan a full body and get its 3D mesh with a fully working software, Although there was room for improvement which is always the case but the targets we assigned, we were able to achieve them.

Results were good and can be improved by (smoothing,Explicit Loop Closing Heuristic and with watertight mesh).This Project helped us to improve as a team we also learned how to manage and work with different people with different backgrounds and knowledge which will help each of us alot in future, Not only managing techniques but also helped us improvement of individual skills (coding, team work, tool using etc).

18 User Manual

18.1 Gantt Chart and Presentation Link

Link https://github.com/WajahatAkhtar/Project-S.E/tree/master/Final-Report

18.2 Github link

Usage of the software is already explained but we would like to make a video of it and will add soon to the Github.

Link 1
https://github.com/WajahatAkhtar/Project-S.E
We have used many references but few of them are as under.

18.3 References

Links for the References :

- Link 1
- Link 2
- Link 3
- Link 4
- Link 5
- Link 6
- Link 7
- Link 8
- Link 9

Paper References :

- Poisson Surface Reconstruction- Michael Kazhdan, Matthew Bolitho and Hugues Hoppe
- Screened Poisson Surface Reconstruction by MICHAEL KAZHDAN, Johns Hopkins University and HUGUES HOPPE, Microsoft Research
- Automated Removal of Planar Clutter from 3D Point Clouds for Improving Industrial Object Recognition, T. Czerniawski a, M. Nahangi a, S. Walbridge a, C. Haas a
- Registration with the Point Cloud Library PCL Dirk Holz, Member, IEEE, Alexandru E. Ichim, Federico Tombari, Member, IEEE
- Dynamic Geometry Processing EG 2012 Tutorial Niloy J. Mitra University College London