Software Engineering - MSc CV/VIBOT/MAIA:

Project Report - 3D Scanner

By: Nitin ANAGHAN, Flavien DAVID, Akshay DUDHAGARA, Bhargav SHAH

Professor: Yohan FOUGEROLLE, Cansen JIANG, David STRUBEL

January 7, 2018

Contents

I. ACKNOWLEDGEMENT

We would like to express our special thanks to our supervisors Dr. Yohan Fougerolle, Cansen Jiang and David Strubel who gave us the opportunity to do this project on 3D scanner, which also helped us in doing a lot of research and we came to know about so many cutting edge technologies and novel algorithms. We are really thankful to them. Secondly we would also like to thank all of our team members and classmates who helped us a lot in finalizing this project within the limited time frame.

II. ABSTRACT

Nowadays 3D acquisition devices (3D scanner) allow us to build accurate, and objects of 3D model with less time consumption. We can construct digital 3-dimensional models by the data that we have collected. Here we use hardware and software assembly for digitizing the different properties with shape and colour of large objects. The developments of 3D scanning techniques are to introduced in this report. A 3D reconstruction software using kinect and point cloud library (PCL) for scanning, reconstruction and exporting a 3D model of a person into a 3D printer is explained in detail. First, a brief introduction to 3D reconstruction and an explanation of the depth measuring technology utilized by the kinect v2.0 is given. Some fundamental concept necessary to understand the procedure of how the kinect can perform a digital reconstruction of an object is also given. This project is based on open source libraries and frameworks. So next a presentation of the tools and libraries utilized during the design of the software is provided. The main source used for this project is the PCL being an open source project free for commercial use and with wide contents of online documentation PCL is one of the key core elements for the development of this project. As for the implementation of the reconstruction, the process is divided in four tasks: surface measurement, sensor pose estimation, smoothing and surface extraction. Surface measurement consists on the scanning and obtaining several point clouds from the kinect. For this scanning process a turntable was used in order cloud scan. This data is filtered by a pass through filter and some outliers removal procedure. For the sensor pose obtained point clouds from the kinect into one. From removing noise and sharp edges from the surface, smoothing is performed by moving least squares and laplacian smoothing. Finally for reconstructing the mesh,

greedy triangulation and grid projection is performed. The poisson reconstruction method was also implemented but as the algorithm was the slowest and required more extra cleaning of the mesh, this method was discarded.

III. INTRODUCTION

3D reconstruction is a technique used in computer vision which has a wide range of applications in areas like object recognition, city modelling, virtual reality, physical simulations, video games, special effects many more. To perform a 3D reconstruction specialized hardware is required. This equipment was often very expensive and available only for industrial or academic purposes. Nowadays, high quality 3D scanners are now available at a low cost, making research available for everyone at home. 3D painting technology has also made a breakthrough in recent years. By taking a digital concept and placing the physical result of that concept into the users hand, 3D printing is without a doubt a promising technology that will only become more commonplace as it continues to be refined The combination of technologies, 3D reconstruction and 3D printing, into a general 3D copy machine bears an enormous economical potential for the future.hence, the urgent need to obtain affordable and meaningful 3D content. The main objective of this project is to develop human 3D scanner software able to fully with a scanner rig composed of a turning table and a stationary depth sensor. A friendly, interactive graphical user interface provides simple control and outputs watertight mesh results that can be used mainly but not limited to 3D printing.

IV. DEVELOPMENT PLATFORM

A. Software requirement

Qt 5.7.0 (MSVC 2013, 32 bit)
MSVC 2015 build tools
PCL 1.8
Microsoft Kinect V2 SDK
Intel R200 SDK

B. Hardware requirement

Microsoft Kinect V2

Intel R200

V. SOFTWARE HARDWARE TOOLS

A. QT

The software is implemented using c++ programming. Our team choose to use QT creator, the cross-platform IDE to realize the software. The QT GUI module provide classes for a variety of GUI tools such as windowing system integration, event handling and 2D graphics, an ideal consolidation for all functions of 3D scanning

B. PCL (Point Cloud Library)

1. Installation PCL

There are few basic steps required for the installation of point cloud library its easy to install if right version of PCL is chosen else it will take you a lot of time even months to figure out the problem and interface it correctly. In the state when we starting interfacing QT it PCL we choose version 1.8.0 as it is the open source available official version online tried installing it by building libraries individually such as open NI2, Boost, Eigen, FLANN, Qhull, VTK which can also be done all together but there were many problems in installing them individually

2. Installation Instructions

Prerequisites PC windows 10 and a USB 3.0 port for Kinect V2. There is file attached here for process of installation PCL

3. 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 Install PCL Readme 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 les 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 insystem variables and add path to the variable PATH in system variables by clicking and edit and then new.

C. MICROSOFT KINECT V2

The Kinect, is a light laser scanner that generates real time depth maps. It obtains a colored 3D point cloud, also known as RGB-D image, which provides information about the colour and the depth of each scanned pixel.

The Kinect consists of three main elements:

- IR light emitter: beams IR rays light into the scene.
- Monochrome CMOS sensor: captures the reflected IR light.
- RGB camera: traditional colour camera.

The Kinect converts the pattern obtained by the CMOS sensor into a depth map. The depth map is aligned to the colour picture, obtained by the RGB camera, and they are combined into a single RGB-D image that is sent towards the USB port.

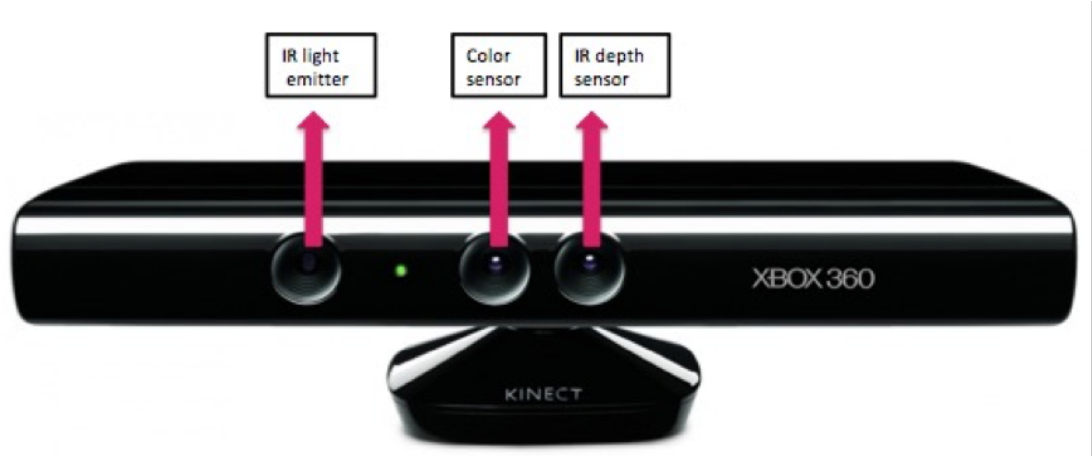

FIG. 1: Main element of the kinect

D. INTEL R200

R200 sensor was not selected although it has some extra features which kinect v2 doesn't have like portability, light weight 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. The Intel R200 is an infrared-projector assisted stereo camera. It uses a known pattern projected using an infrared laser projector to assist the stereo vision system to identify depth.

It has a color camera with 1920 x 1080 pixels and a depth/IR camera with 640 x 480 pixels. The color camera can stream at 30 fps and the depth camera can stream at 30 and 60 fps. We hence ask the sensor for aligned capture at 30 fps.

The angular field of view of the sensor in horizontal and vertical planes is approximately 50 and 60 degrees respectively and its sensing range is from 50 cm to 2m. The sensor has problems perceiving dark textures, as well as depths of brightly lit scenes. The R200 is highly portable because of its small size, and uses a single usb for power as well as data. This is unlike the kinect which is bulky in comparison, and requires a separate power block.

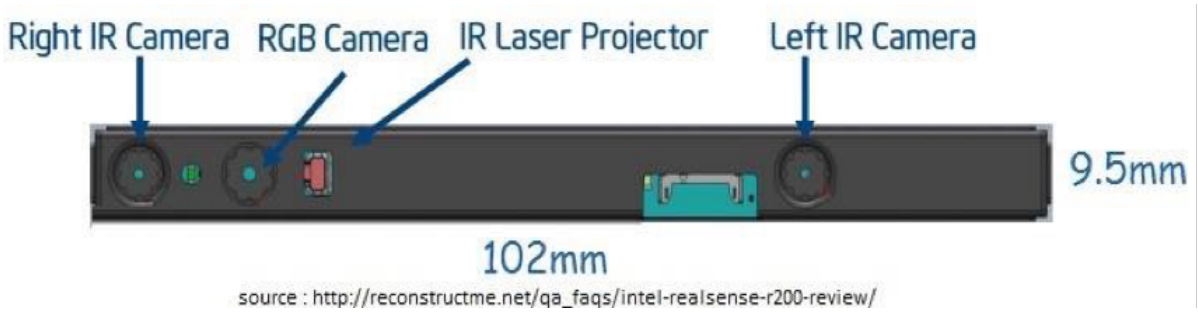

FIG. 2: Configuration of sensors in Intel R200

E. GIT

Git is an open source control system to manage and share cooperative work, especially for programming. Git is secure as it protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable. It is much optimized as it allows committing changes, creating branches, merging codes, and comparing it to past versions of itself. It recognizes the attributes of real source code file trees, how they are modified over time and which are the access patterns. It is also flexible as it supports various kinds of nonlinear development work flows.

Github is a web based Git repository hosting service. It was used in this project as a free central repository to store all the data in a cloud. It allowed each developer to always have a full local copy of the repository with the development history. Even if a local hard drive fails or if anyway the data is lost, all the data can be recovered. In Github it is possible to store projects in the Git format and from.

VI. HIGH LEVEL DESIGN

The design of the application is the integral part of the software life cycle process where the application starts to take shape. If the design is not robust enough and fails to meet the requirements then the application developed will become instable. Hence lot of research was done before arriving at the final design for the application which will be explained in the coming chapters.

A. Use case diagram

The diagram depicts the different features provided by the application at a high level and the types of users who can have access to the system.

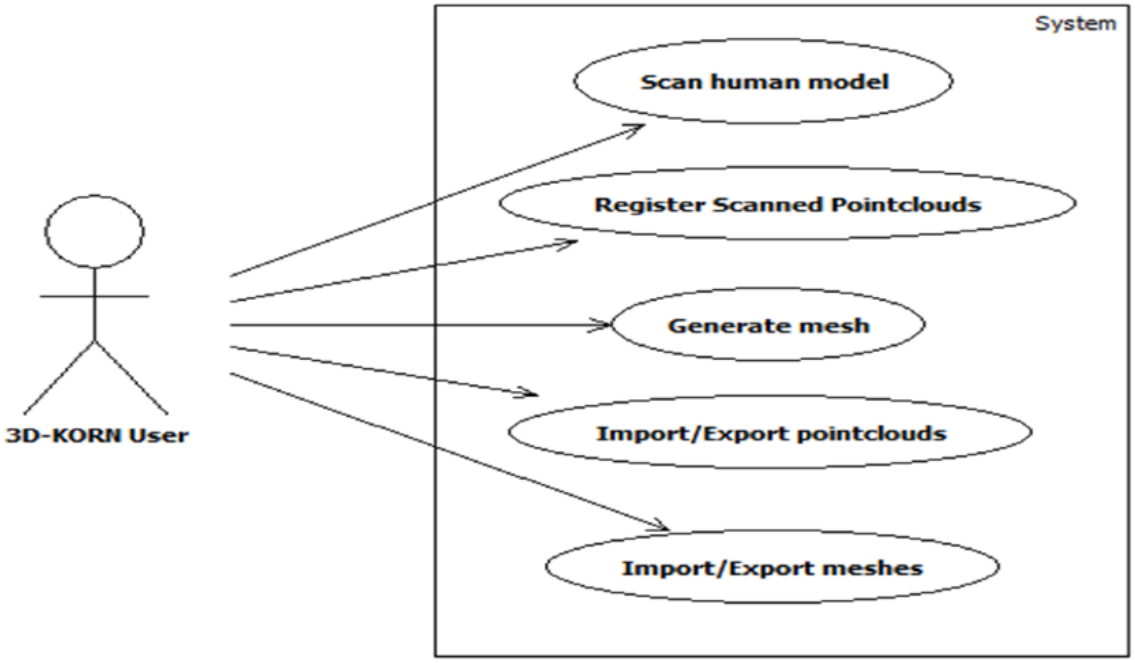

FIG. 3: Use case diagram

B. Class diagram

The main factors taken into account when designing the class diagram are listed below:

- Reduce complexity
- Increase modularity
- Create reusable classes
- Ease of maintenance
- Team size

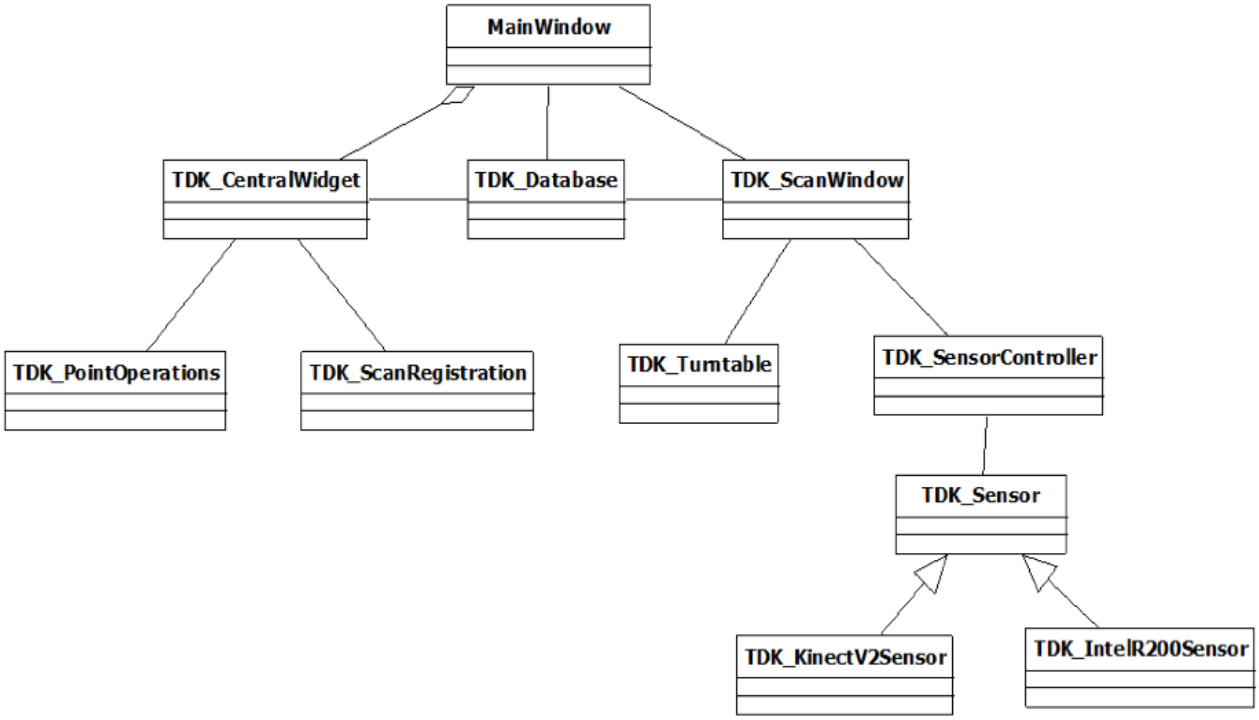

FIG. 4: Class diagram

VII. REGISTRATION

A. The registration problem

Once multiple frames from the 3D object have been acquired, we are faced with 3D point clouds that are in different frames of reference relative to the scanned object. To transform all 3D point clouds to the original frame of reference we need to apply 3D transformations to each point cloud to compensate for the rotations and translations occurred between several scans.

The process of 3D data alignment is generally known as registration. Its corresponding algorithms addresses the problem of finding the optimal transformation that maps a pair of point clouds between them, considering that both data sets are exposed to camera noise during acquisition. Overall, there are two main families of registration methodologies: rigid and non-rigid ones. While the first family assumes that the mapping transformation can be modelled by using six degrees of freedom, the latter one considers the problem of softer bodies whose shape can change

during data acquisition [Bellekens 2014].

Registration algorithms can work, independently on the way a pair of point clouds is registered, pair wise or by multiplier registration approaches. Although pair wise registration allows easier implementation, generally suffers from the problem of drifting due to the accumulation of small alignment errors. This kind of problems can be addressed and corrected using loop closing algorithms.

B. First approach: General Case

Our first approach to the registration problem used pair wise registration based on feature correspondences. The original point cloud was initially down sampled to around 40% of the original data size in order speed up computation by means of data dimensionality reduction. Then, a transformation matrix was estimated with respect to the previous down sampled point cloud using features descriptors. Once pre-alignment was obtained, a final aligning step was conducted and the obtained transformation using down sampled versions was applied to the original raw point cloud to finish the pair wise registration. A flowchart of this registration approach is shown down below.

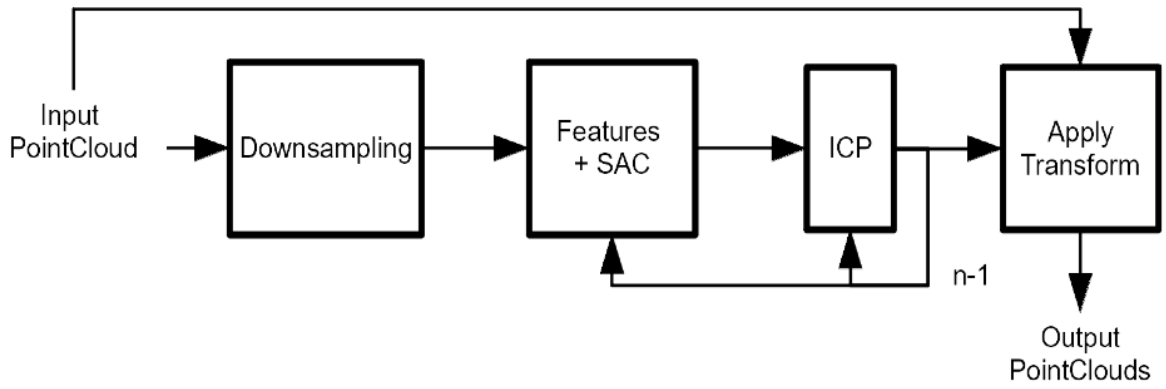

FIG. 5: Flowchart for the first registration approach

In the first step, a voxelized grid approach was conducted using the VoxelGrid class in order to downsample the input point cloud. Here, a custom size 3D voxel grid is created over the input cloud and all data inside each voxel is represented by means of its voxel centroid. As result, the input point cloud is described by a lower dimension 3D cloud that still preserves the high-dimension cloud morphology.

After downsampling, the correspondence matching between point clouds was eased by using Fast Pointer Feature Histograms (FPFH), which consists in informative and pose invariant local features describing the surrounding surface properties of a point [Rusu 2009]. The method receives the downsampled point cloud achieved in the previous step as well as its estimated point normals. A searching radius is custom set in order to define the region of analysis. The output of this step consists on a set of informative features describing the point cloud.

For pre-aligning two point clouds based on its FPFH descriptors, Sample consensus Initial Alignment (SAC) was conducted. The method consists in an exhaustive search in the correspondences space (FPFH descriptors of the clouds) in order to find the best transformation that aligns data based on its features [Rusu 2009].

Finally, once a pre-alignment was achieved using FPFH and SAC methods, the Iterative Closest Point (ICP) algorithm was used to register the last point cloud with the previous one. The method uses singular value decomposition to estimate the affine transformation that better aligns data. Iteratively, outliers are discarded and point correspondences are redefined after finding the best transformation matrix [Bellekens 2014]. The search distance for the algorithm is custom determined. Since pre-alignment with SAC was conducted, a short distance was chosen (maximum correspondence distance of 5 cm). Once the best ICP-based transformation for the clouds was found, it was applied to the original high dimensional cloud jointly with the SAC obtained transformation.

FIG. 6: Registration obtained with the first approach

Problems and Limitations with the first registration approach.

The results shown in Figure 6 were not as good as we expected and later analysis led to several conclusions:

- SAC feature matching was not working as expected. An initial pre alignment was expected after this step. However, in some cases the transformation obtained deviated the point cloud

much more than the initial input, adding more complexity for ICP.

- The poor initial pre-alignment was making ICP obtain non-desired results. Mainly, the algorithm gets stuck in local minima in the optimization process.

- An important observation after this approach was that only using pairwise registration is not possible for global registration. The main problem underlies in drift multiplication, providing very poor global alignment for the last processed clouds.

- Another conclusion regards data pre-processing. Downsampling step only reduces the size and complexity data, but still remain noise and outliers in the cloud. This suggests the use of outlier removal techniques in order to discard non-informative points.

It was clear the need of severe modifications in the registration approach accounting with the problems and limitations herein found. Hence, an improved registration algorithm was proposed as explained as follows.

C. Second approach: Controlled Environment

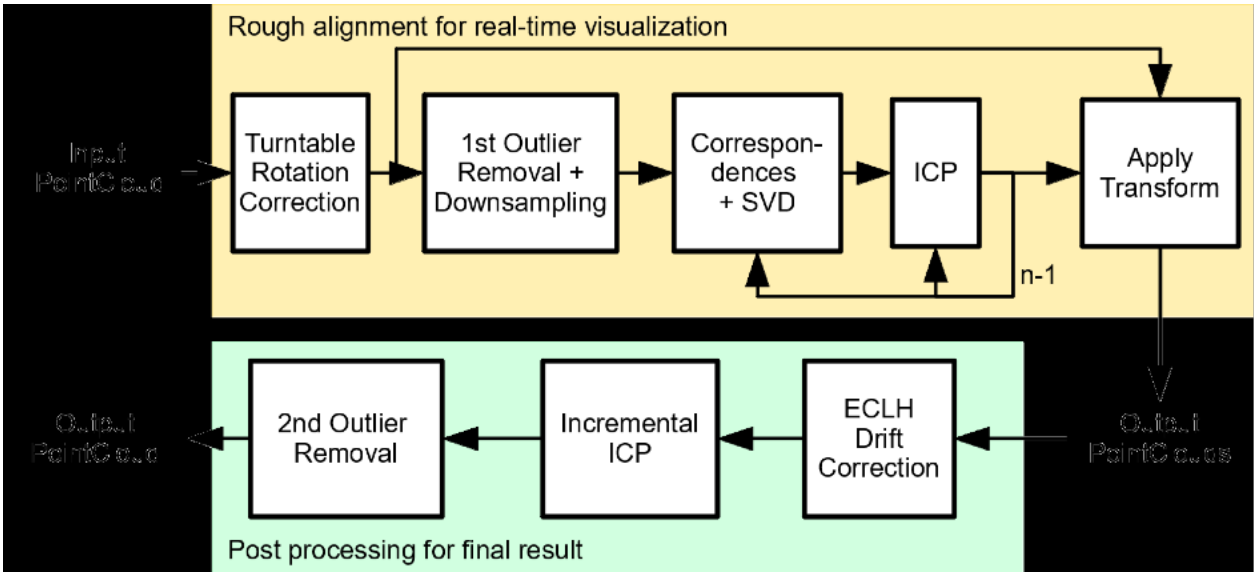

FIG. 7: Flowchart for the second registration approach

This second approach aimed to solve ICP getting stuck at local minima by providing a better initial alignment and have two types of registration: real-time registration designed to be fast to show the result in real-time to the user and a post-processing registration slower but much more robust, dependant on having all views from the object, also using the results from real-time registration process.

1. Pre-alignment knowing turntable parameters

Seeing the bad results the general approach gave, and after some analysis, we reached the conclusion that the main problem was poor alignment estimation. We thought that we could get a good estimate of the transformations between point clouds since we know the acquisition process: the general transform from one point cloud to the next is a rotation in the height axis of some degrees around the axis of rotation of the turntable.

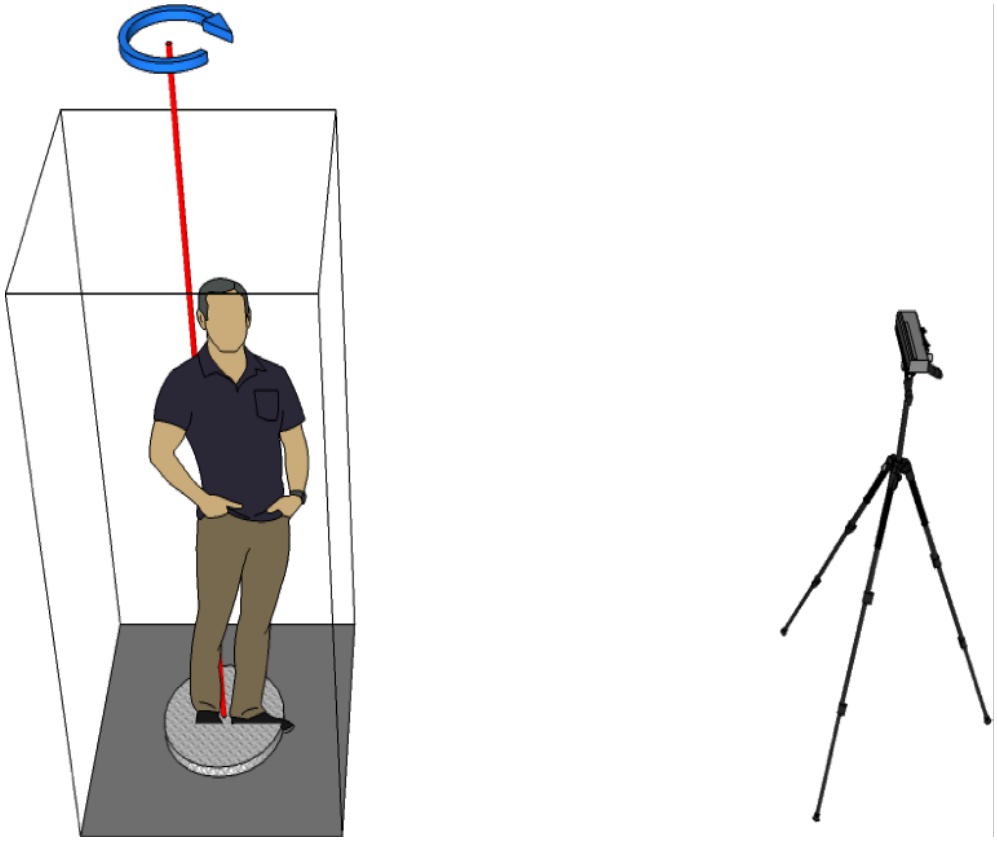

FIG. 8: Scanner turntable with drawn axis of rotation

2. Scanner turntable with drawn axis of rotation

For figure 8, if we could have these parameters in our algorithm, we could manually pre align the point clouds for improved robustness. Several tests revealed this strategy to be a really good idea, as shown in Figure X, even with rough estimates of the axis of rotation of the turntable and angle of rotation between scans. They certainly gave a much better starting position for the rest of the registration process.

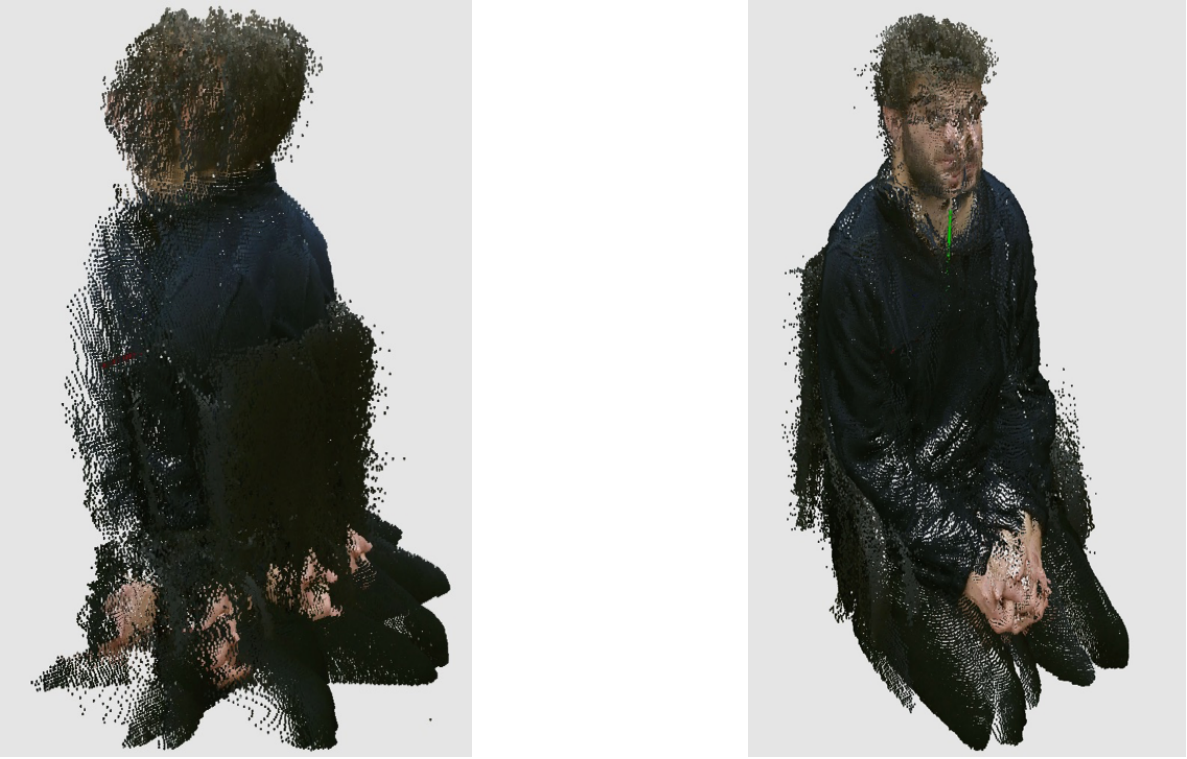

FIG. 9: Results on the application of rotation compensation

3. Denoising and Downsampling

In order to improve the registration step performance, a noise reduction and outlier removal step was added in the algorithm. The main goal in this step implies the identification and discarding of those samples from the point clouds that can mess with 3D registration distance error computations. These points are generally associated with different kinds of noise and boundary and steep surfaces distortion. In Figure 10, an example of a noisy point cloud it is shown.

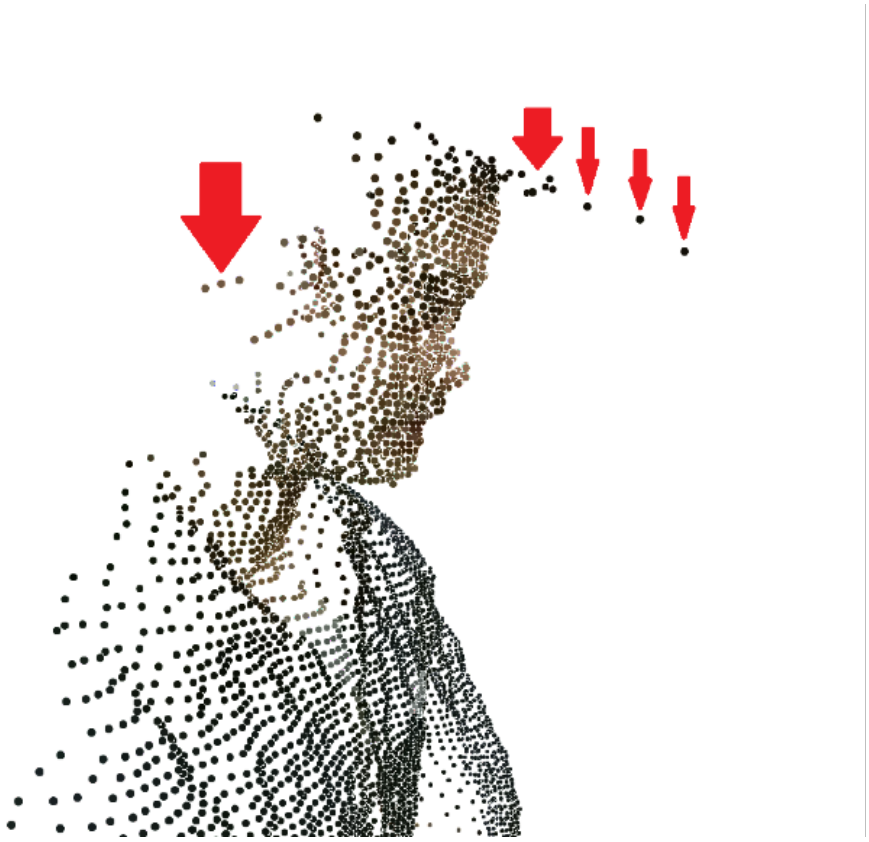

FIG. 10: Noisy point cloud. Red arrows highlight outliers.

To tackle this problem, statistical outlier removal was used by means of a nearest neighbours approach. Here, k-points surrounding a specific point of the cloud are analyzed. The statistical parameters of the sample distribution are estimated (mean and standard deviation), and those points falling out of m-times the standard deviations are removed. Hence, the algorithm requires setting the values k and m for the thresholding. In our implementation, values k=8 and m=2.5 were used.

After cleaning the raw data, point clouds were downsampled using Voxel Grid, the same strategy used in registration approach 1 and described above (section 2).

The result of these two consecutive steps consists on an informative, free from noise downsampled point cloud. In our implementation, values k and m were tuned based on empirical results, accounting to a correct balance between boundaries definition and point-cloud density homogeneity.

4. Pairwise Initial Rough Alignment

After these pre-processing steps, the rough registration transform is estimated with a similar setup than that of the first approach. We first get a list of correspondent points between the new downsampled point cloud and the previous registered downsampled based this time solely on the normals as descriptors. Then a consensus correspondence rejection algorithm is applied to discard possible outliers. Afterwards, a linear system is solved that gives the best approximate transform to map each correspondence to the same frame of reference using SVD for efficient computation.

Finally, ICP is applied to the result of the previous step to get a finer alignment. The obtained transform is multiplied with the one from the previous step and applied to the original not-downsampled point cloud. At this point the rough initial alignment is done and we either wait for another pointcloud to register or for the user to start the post-processing.

5. Post Processing

- Closed Loop Drift Correction

  The first stage of the algorithm implies pairwise registration of the consecutive point clouds that capture all views of an object. As stated earlier, this way of registering accumulates small errors that create errors when the rotation of the turntable is finishing the full rotation.

FIG. 11: Results on the application of loop closing

The render on the left picture is roughly aligned and the right one is a closed loop. As we can see in Figure 11 the small error in pairwise registration has induced a significant error when the view returns to the initial position. We can see that the legs are significantly misaligned, causing the appearance of having three legs and two pairs of hands.

This is a common problem in robotics and to solve this issue we used a variant of the well-known SLAM (simultaneous localization and mapping) technique through the ELCH (Explicit Loop Closing Heuristic) implementation in PCL. The result by slightly modifying the initial transforms until the last point cloud matches with the first one is a better global registration. However, this small modification induce at the same time small local misalignments as a consequence, which need further processing since ELCH goal is to provide good global registration and not so much good registration at the local level.

- Incremental ICP

  To solve inaccuracies and misalignment from the relaxation of the bounds from the loop closing algorithm and to avoid again the typical drift of pairwise registration we used the PCL implementation of Incremental ICP, which performs ICP between the new point cloud and a merge of all previously aligned clouds. Since the global registration provided by ELCH is quite good we can put a really short correspondences search distance to be certain that ICP will land on the global minima and give the best registration possible.

D. Class structure and operation

1. Public interface and using the class

There exist few basic functions through which all the interaction and operation with the class are made:

- To construct the class two basic parameters are needed:

  boolregisterInRealTime:

  Specifies if the rough initial registration will be made at the time of adding the point clouds or in the post processing stage.

pcl::PointWithViewpoint scannerRotationAxis:

Which encodes the Cartesian point coordinates and rotation euler angles around each axis that define a line, which spatially coincides with the turntable axis of rotation. The value of this parameter is calibrated through visual feedback once the scanner is deployed, before starting to scan.

- Then the consecutive rotated views to register are input with:

  addNextPointCloud(inputCloud, degreesRotatedY)

  Where inputCloud is the next rotated cloud and degreesRotatedY is the degrees rotated in Y axis with respect to the previous pointcloud.

- Get the current progress of roughly aligned point clouds for display with:

  getRoughlyAlignedPC()

  This returns the merged roughly aligned point clouds.

- Finally, when all pointclouds have been acquired to close and finish the registration, the post processing of the scan is made by calling:

  postProcess and getAlignedPC()

  This post-processes and returns the final merged aligned result.

Extended operation diagrams

The final implementation of the described algorithm for rough registration and the finer post-processing registration are summarised in the diagrams depicted below.

E. Results and conclusions

1. Results

The results obtained are really good at the global registration but with a lot of room for improvement in local registration, having a good deal noise and artifacts. The artifacts and noise in the attached figures may probably be due illumination issues and the noise in the sensor itself, they will be removed with smoothing filters and other post processing for reconstruction.

FIG. 12: Results on the application of rotation compensation

2. Conclusions and Future work

We consider the results of our registration algorithm to be good enough considering the technical difficulty, initial skills and knowledge, available time and mentoring and organisation overhead in the project. We have provided an algorithm that can successfully register 3D objects with as few as 10 frames while providing a rough registration preview in real-time. It is also robust to rough parameter estimation and errors in the pre alignment of some frames. The developed class is modular and flexible for ease of future development, while offering a simple use interface and configurable parameters.

Future work should focus mainly on reducing the noise and misalignment derived from sensor noise and non-rigid object registration. The severity of the sensor noise in the registration pro-

cess has only been noticed in the final stages of development and it hasnt been fully addressed. Consequently, the outlier removal step and point cloud denoising should be further studied and developed. However, even with really good outlier removal and noise reduction, if the scanned person makes small movements between frames (such as breathing), it will still impact the quality of registration for mesh reconstruction. It is suggested to study the integration of non-rigid fine registration algorithms that can account for these misalignments and can adjust them to create a smooth yet detailed final result.

VIII. FILTERING

Before going into the implementation of the algorithms greedy triangulation, marching cube and Poisson it is necessary to look at the input data is provided to these algorithms and how it is processed so before give the point cloud this algorithm we have to do some operation on the pointcloud.

Because the received point it was oversampled first of all we decide to elaborate on it some filtering operation.

Pass-through this filter iterates through the entire input point cloud, and automatically filtering the non finite point and the point outside to one presented interval specified.

The voxelGrid class creates a 3D voxel grid, as a set of tiny 3D boxed in space, over the input point cloud data. Then, in each voxel all the points present will be approximated with their centroid. This approach is a bit slower than approximating them with the centre of the voxel but its represents the underlying surface more accurately.

IX. SMOOTHING

The first step taken after preparing the cloud is smooth is using moving least squares (MLS). the idea behind MLS is to have a data set of point p = pi that contracts a smooth surface SP, however, instead of using the original data set of point P, a reduced set R = ri is created and a new surface SR is defined.

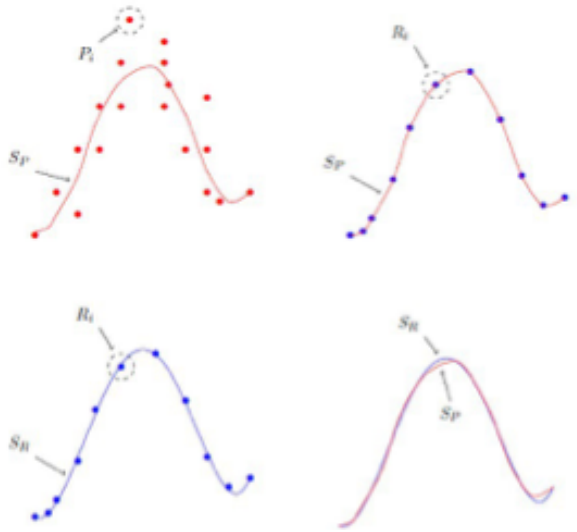

FIG. 13: MLS

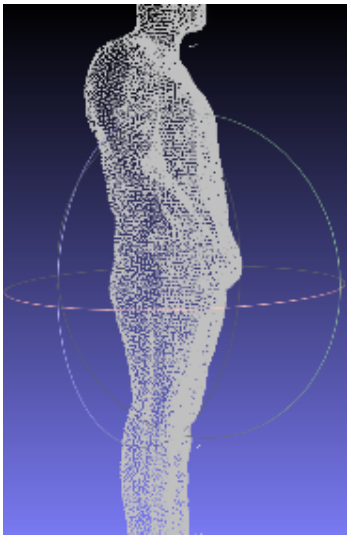

FIG. 14: Point cloud after MLS

The MLS smoothing is done before any reconstruction algorithm is applied. Another smoothing method is also used, however only applied after the reconstruction is complete.

X. NORMALIZATION

In order to use the cloud properly for the reconstruction, a cloud with both x,y and z values and normal information is needed. Many methods of surface reconstructions require normal associated with the point clouds, the normal can either be oriented or unoriented. Normal that do not have a direction are called in oriented normal, which means it is to be expected the normal to be pointing either on the inside or outside of the surface. This sort of information is useful to determine planar regions in a point cloud, projecting a point onto an approximated surface or achieving covariance matrix. An oriented normal on the other hand have consistent directions and it is known which point outside or inside of the surface. The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plea tangent to the surface which is turn becomes a least square plane fitting estimation problem. The solution for estimating the surface normal is therefore reduced to an analysis of the eigenvectors and eigenvalue of a covariance matrix created from the nearest neighbours of the query point.

XI. MESHING

Will introduce the concepts regarding 3D reconstruction which will lead to the process of implementation. Firstly, an introduction about surface reconstruction using point cloud data. Finally, a short discussion of the criteria leading to a successful reconstruction in the implementation will be described.

In the point clouds the points refer to the X, Y and Z geometric coordinates of a surface. Here for the project purpose point clouds are acquired by using kinect v2 device. It is a time of flight laser scanner which measures the elapsed time between a light emission and sensor detection. The data acquired from kinect v2 can be used to recreate certain objects digitally represent shapes or detect particular surfaces and their topology. With the point data it is possible to create a mesh of the surface.

There are four main steps to be considered when using the point cloud data to reconstruct a surface:

- Once the point cloud data has been gathered it is important to eliminate irrelevant data and sample point to reduce the computational time.

- Finding the relations between the data point their connectedness continuity and boundaries.
- Start the process of generating the surface using polygons and create the meshes.
- Edit the model created to perfect the polygonal surface resulted.

A. Greedy Triangulation

This is an iterative algorithm developed by zoltan csaba marton. The focus of fast triangulation is to keep a list of possible point that can be connected to create a mesh.

FT is obtained by adding short compatible edges between point of the cloud where these edges cannon cross previously formed edges between others point. Fast triangulations are non-planar.

The method works by maintaining a list of points from which the mesh can be grown and extending it until all possible points is connected. It can deal with unorganized point coming from one or multiple scans, and having multiple connected parts. It works best if the surface is locally smooth and there are smooth transitions between areas with different point densities.

In the point cloud library, fast triangulation works locally, however it allows the specification of the required features to be focused on. Such parameters are neighbourhood size of the search, his search radius and the angle between surfaces. The surface coated by this algorithm is not continuous and can contain holes; this was one of the reasons that lead us to not use FT for reconstruct the meshes. In the image below is illustrated the result obtained:

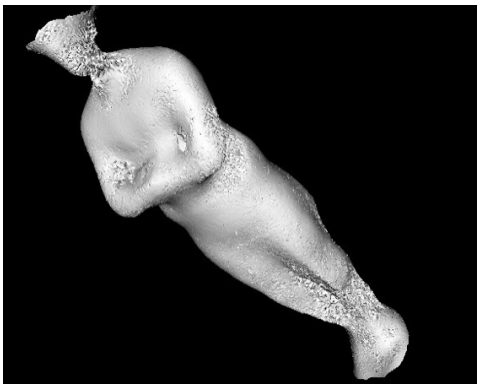

FIG. 15: Triangulation

B. POISSON

There are a multitude of surface reconstruction techniques, however each one of them poses a number of difficulties when it is applied to data points. But because watertight mesh was one of the prefixed settled point at the end we decided to use Poisson algorithm reconstruction. The Poisson reconstruction uses the Poisson equation which is also known to be used in systems that perform tone mapping, aid mechanics and mesh editing.

By its nature, this algorithm can only be used with watertight or closed object. This can be problematic when we get the points cloud through kinect, in fact the acquisition that we make not always respect this property as we can see in our model the head is not a close object. In our specific case at the head and feet can be noticed that Poisson reconstruction connects the regions which poses some limitation in the use of Poisson reconstruction even more Poisson reconstruction has a high noisy sensitivity.

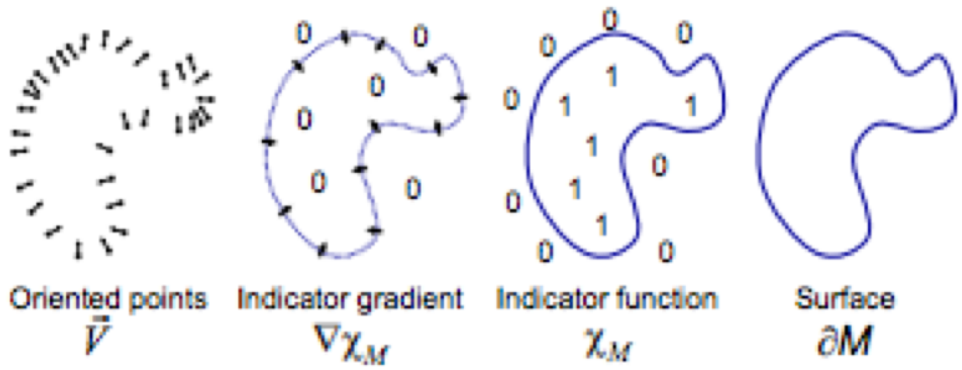

FIG. 16: General overview of poisson method

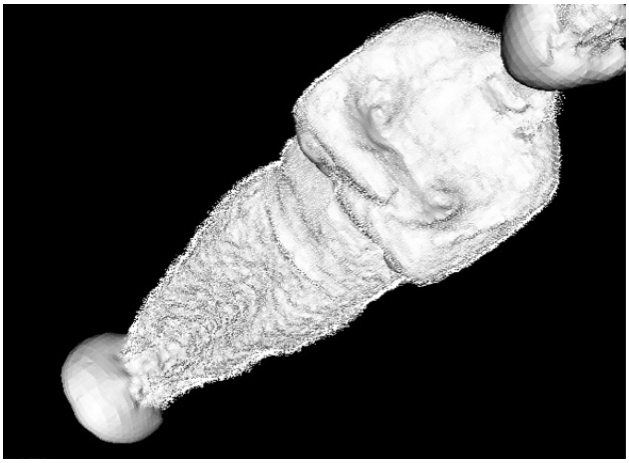

FIG. 17: Poisson Reconstruction

C. Post processing Algorithm Laplacian

The obtained result was still a lot noisy, for having a better result in teams of surface problem we decide to apply a post smoothing method.

The smoothing method used is mesh smoothing laplacian VTK. This is a laplacian smoothing method from the VTK library. As described in the documentation for the original VTK smoothing method: the effect is to relax the mesh making the cell better shaped and the vertices more evenly distributed.

FIG. 18: Laplacian Filtering

D. Holes Filling

Holes filling can be performed as a post processing operation applied after surface reconstruction or it can be integrated into a surface reconstruction algorithm. If integrated we can further distinguish between reconstruction algorithms that operate on connected meshes of range samples and algorithms that operate on clouds or unorganized 3D point hole filling as a post process and mesh bash reconstruction with hole filling.

The origins of these holes ranged from early scanning issues, where some parts of the body cant be scanned properly, to later stages, where filtering and meshing algorithms can distort the surface of the model and create additional holes issues. In case of usage of grid projection amount of holes drastically decreases, but at the same time orientation of the squares which are created by the algorithm can create small holes between a tiny numbers of squares, ranging from one. This can be fixed by usage of the built in VTK library which can fill holes with following types of edges:

- Boundary or a line cell
- Non manifold
- Feature edges
- Manifold edges

But nevertheless this library is rendered useless for big holes. Most of the time, hole issue occur nearby and inside the seams and joints, therefore apart from the hole detection which can be seen on the pictures, we propose a pipeline approach, which sums up to extraction of the detected edges and filling them with created patches which can be passed through vtk triangle filter in case of usage of the grid projection method. This pipeline approach potentially solves the problems of the vtk fill holes filter, closing the big holes on the feet and head. It can be a suitable direction for the development of our project.

Pipeline approach comprises four steps:

- Extraction of the edges
- Create patches

- Use contours as an input and fill holes with created patched
- Append change to the model

XII. GRAPHICAL USER INTERFACE

GUI is a type of the user interface that allows users to interface with electronic devices through graphical icons and visual indicators such as secondary notation instead of text based use interfaces typed command labels or text navigation. GUI were introduced in reaction to perceived steep learning curve of command line interfaces. Which require commands to be typed on a computer keyboard.

The implemented GUI has a user friendly interface and simple operation. The software design framework is based on the class diagram show in figure. For this approach a diagram network structure is used to show the system built by a combination of different programs or processes. In the diagram, each box represents a class and the member functions. The dashed arrow represent unidirectional dependencies between two classes, meaning that the main class my vtk widget depend on instances from the other classes.

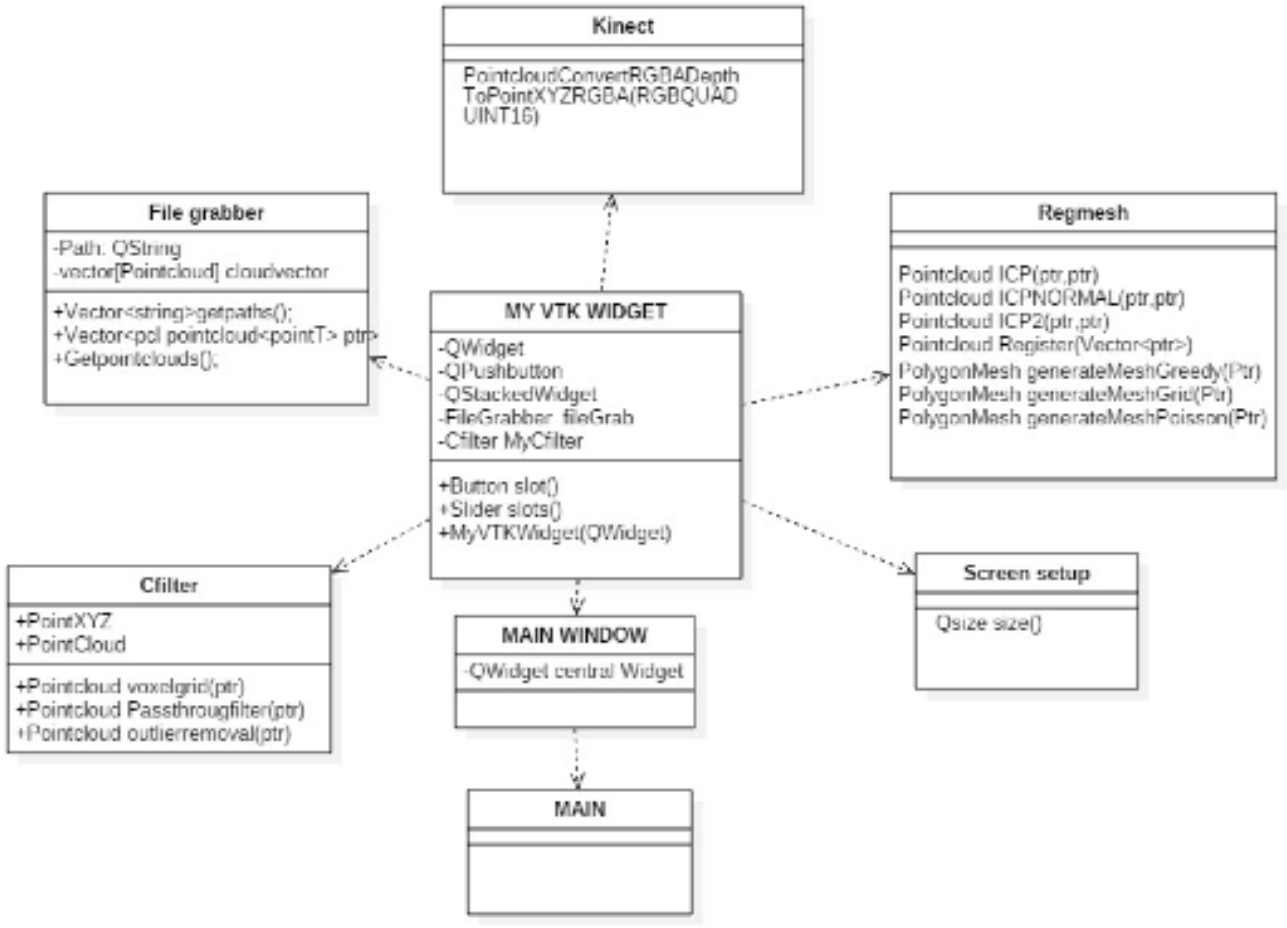

FIG. 19: Class diagram of the software

As shown in the class diagram the main window class contain the central widget me vtk widget with several subclasses from where the entire program is executed sequentially. The vtk widget class include the following widgets:

QPUSHBUTTONS used for executing a command QRADIOBUTTONS used for choosing one of the offered options QSLIDERS used for adjusting parameters

The screen setup class adjusts the size of the display screen according to the dimensions of the users screen.

The file grabber class captures depth map of the scene, which is converted to point cloud with the kinect class. They are saved as point cloud files which are used for the subsequent steps for obtaining the 3D model. The filters class is used to filter each of the captured frames, which

afterwards are aligned by using ICP after the filtering procedure is done. The class has three methods: voxelGrid for down sampling, pass through filter for filtering and outlier removal for removing the outliers from the point clouds.

The regmesh class is used for performing the registration and meshing of the point clouds after the filtering procedure is done. The filtered frames are aligned using different approaches for registration by ICP. The triangulation and obtaining the complete 3D reconstruction of the scanned object are done by three different meshing techniques. The class has six methods for registering and three methods for meshing. Each register method is used for registering sequence of point clouds. For aligning two point clouds the methods ICP, ICPNORAMAL and ICP2 are used. The meshing methods generate mesh greedy, generate mesh grid and generate mesh Poisson are used for performing greedy projection triangulation, grid projection and Poisson reconstruction, respectively.

A. GUI Design

The core of the GUI. It is the window from where the user has the options to open the scan window, import and export point clouds and meshes, register pointclouds and generate mesh.

FIG. 20: Window

The functioning of the window is based on its 4 different components which will be detailed below.

1. MENU BAR

The menu bar located in the upper left corner, give the possibility to the user to import and export points clouds and meshes. The point clouds, under the format .ply or .pcd, could be imported from the local computer or directly from the scanning interface The meshes, under the format .stl and .vtk could be also import from the local computer or generating using the module Point Cloud Operations. By importing from the local computer, the possibility to import files at the same time has been added. To export a point cloud or a mesh, its required to specify the format of the file by selecting it from the sub menu Export As.

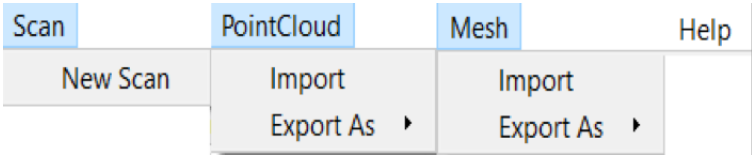

FIG. 21: Menu Bar

2. POINT CLOUD EXPLORER

The Point Cloud Explorer has been designed from a QTabWidget containing 3 QListWidget, each associate to a specific type of file: Point Cloud, Registered Point Cloud and Mesh. Each time a file is imported or generated, a QListWidgetItem is automatically added to the QListItem corresponding to the type of the file. In order to process operations on the point clouds or to show them on the Point Cloud Visualized widget, the QListWidgets give the possibility to the user to select multiples files by providing an associate check box to each QListWidgetItem.

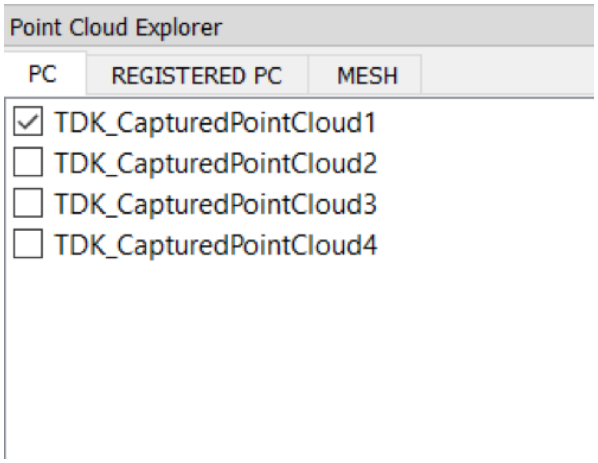

FIG. 22: Point Cloud Explorer

3. POINT CLOUD OPERATION

Different operations can be processed to the point clouds such as the registration between several point clouds and the generation of a mesh. The Point Cloud Operations module allows the user to choose the algorithms of registration and generation of mesh. By default, these algorithms are respectively Singular Value Decomposition and Iterative Closest Point for the registration and Poisson Surface Reconstruction for the generation of mesh. When one of these operations is applied, the generated file is directly added to the QListWidget of the Point Cloud Explorer.

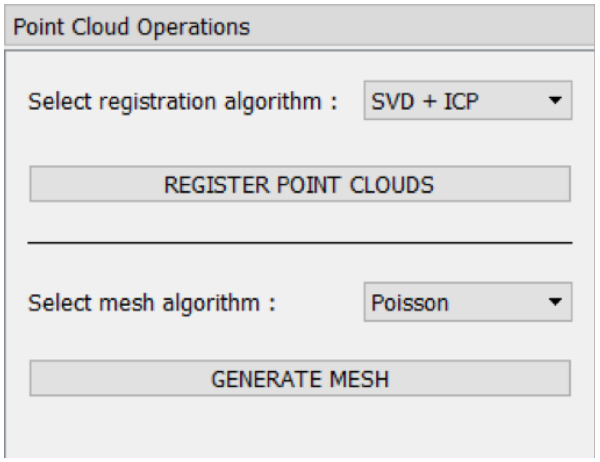

FIG. 23: Cloud Operation

The Point Cloud Visualize (figure 7.5 module is based on a QVTKWidget which give us the possibility to display a point cloud or a mesh. Each time a QListWidgetItem of the Point Cloud Explorer is checked or unchecked, the point cloud or mesh associated is added or removed to the QVTKWidget. Multiples files can then be displayed at the same time. In order to have a better visualization, its possible to zoom in / zoom out and rotate around the axis [x, y and z].

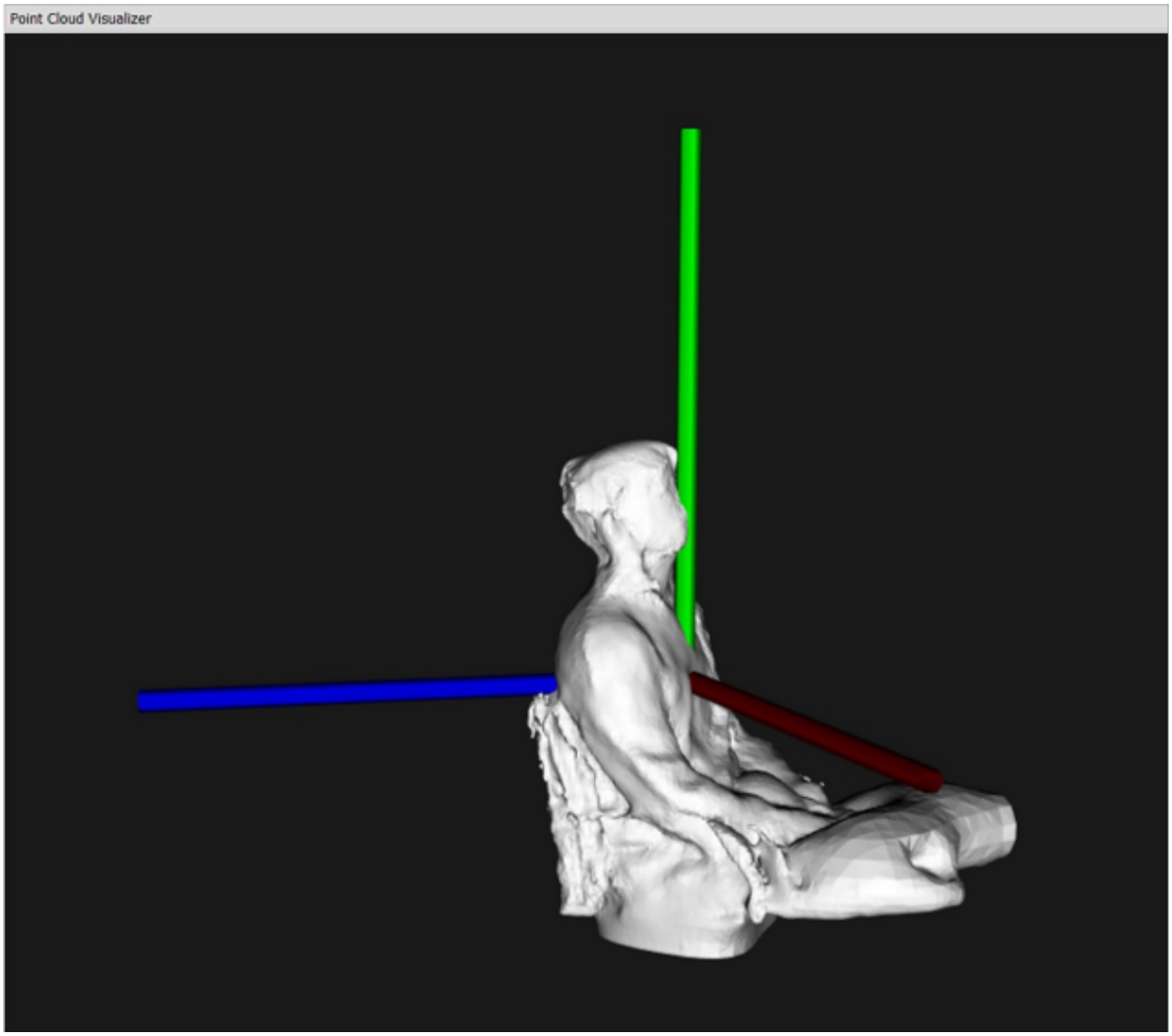

FIG. 24: Point Cloud Visualizer with displaying mesh

B. Scan Window

From the Scan Window (figure 7.6, the user will have the possibility to interact directly with the sensor and the platform to get a series of point clouds, which will be display on the Point Cloud Visualizer, similar to the one of the Edit Window (section 7.1.4

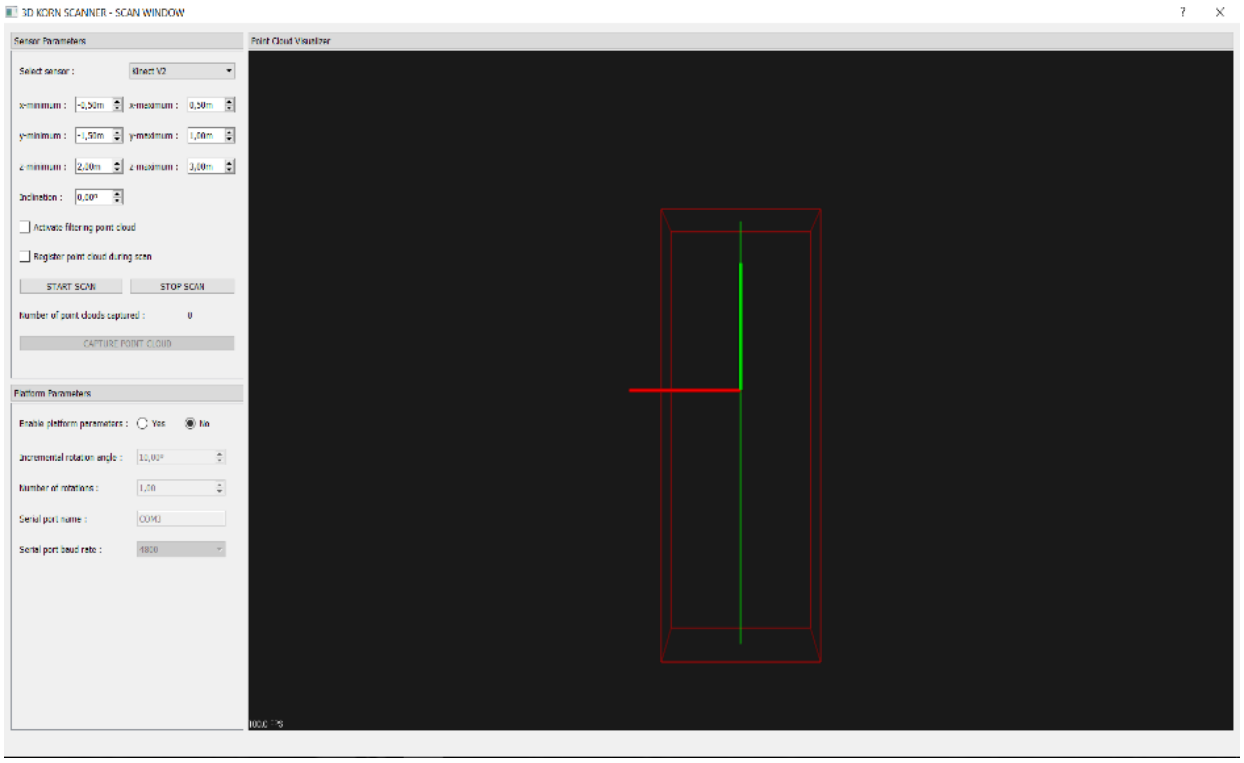

FIG. 25: Preview of the Scan Window

1. *Platform Parameters*

The user has the possibility to tune parameters relative to the platform such as incremental angle of rotation, the number of complete rotation, or to the mode of acquisition, the name and the baud rate of the serial port.

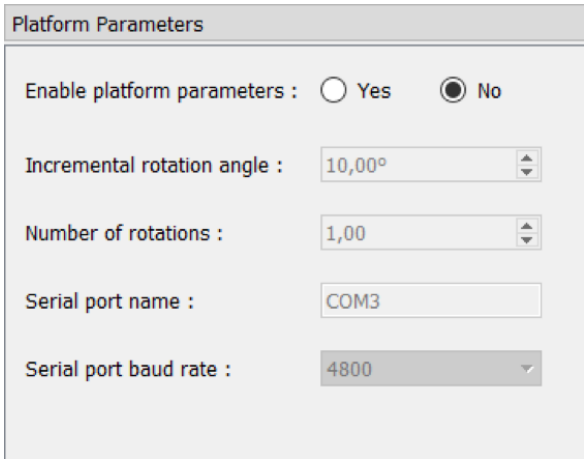

FIG. 26: Preview of the Platform Parameters

2. Sensor Parameters

Due to the range of sensor which could be used for the acquisition of point clouds, like Kinect V2 or R200, a module has been added to let the user tune the parameter of his sensor figure

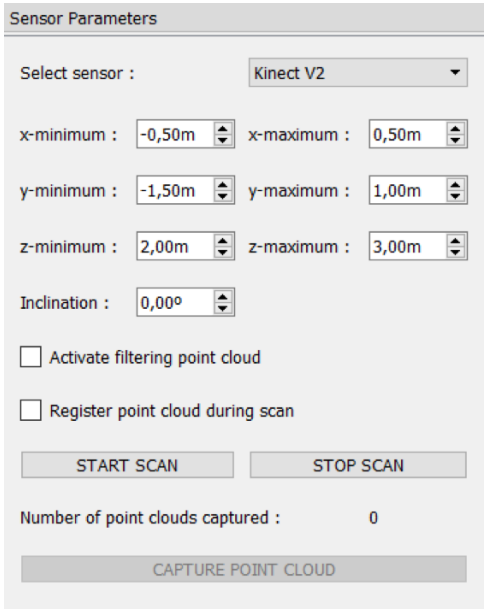

FIG. 27: Preview of the Sensor Parameters

Besides the possibility to select the sensor, the user can define the delimitations and orientation of the scanned area. He can also decide to apply directly the filtering and registration processes.

To perform a 3D scan, the user has to manually start the procedure. When the scan is completed, the series of point clouds are displayed on the Point Cloud Visualized module and the number of point clouds captured is updated. Using these informations, the user could verify the quality of the scan and decide to remake the scan or move to the Edit Window

XIII. CONCLUSION

The main goal of the project has been reasonably achieved. The application is considered to be robust to handle changes both in front end UI and back end algorithms and libraries. Additional efforts were made to achieve tasks such as to make the application compatible with both the sensors provided and automatic pointcloud capture by interfacing with the encoder.

[1] http://docs.pointclouds.org/trunk/group_lters.html.

[2] http://pointclouds.org.

[3] http://pointclouds.org/documentation/tutorials/passthrough.phppassthrough.

[4] http://pointclouds.org/documentation/tutorials/resampling.phpmovingleastsquares

[5] http://pointclouds.org/documentation/tutorials/statisticaloutlier.phpstatisticaloutlier-removal.

[6] http://pointclouds.org/documentation/tutorials/voxel grid.phpvoxelgrid.

[7] http://robotica.unileon.es/index.php/pcl/openni tutorial 2: cloud processing (basic).

[8] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva.
State of the art in surface reconstruction from point clouds
In EUROGRAPHICS star reports, volume 1, pages 161185, 2014.

[9] M. T. Dickerson, R. L. Scot Drysdale, S. A. McElfresh, and E. Welzl.
Fast greedy triangulation algorithms.
In Proceedings of the tenth annual symposium on Computational geometry, pages 211-220. ACM,1994.

[10] R. Fabio et al. *Point cloud to surface: the modeling and visualization problem*
International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34(5):W10, 2003.

[11] S. A. Goldman. *A space e_cient greedy triangulation algorithm*
Information Processing Letters, 31(4):191196, 1989.

[12] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke.
Registration with the point cloud library: A modular framework for aligning in 3-d
IEEE Robotics Automation Magazine, 22(4):110124, 2015.

[13] J. Hyvarinen et al.
Surface reconstruction of point clouds captured with Microsoft kinect 2012.

[14] M. Kazhdan and H. Hoppe.
Screened poisson surface reconstruction
ACM Transactions on Graphics (TOG), 32(3):29, 2013.

[15] C. Levcopoulos and A. Lingas.
Fast algorithms for greedy triangulation
BIT Numerical Mathematics, 32(2):280296, 1992.