Software Engineering Project
Final report

3D Scanning and Reconstruction

Project
Sculptura

Submitted by

Program	Names of Students
MAIA	Elizaveta Genke
MAIA	Daria Zotova
MSCV	Mohamed Ali
MSCV	Karim Botros

Under the guidance of
Professor Yohan Fougerolle
Tutor: Dr.Cansen Jiang
Tutor: David Strubel

Université de Bourgogne
720 Avenue de l'Europe, 71200 Le Creusot, Tél : 03 85 77 00 70

Contents

List of Figures

Introduction

Nowadays, 3D scanners have become widely used in different fields, such as robotics, medicine and manufacturing. One of the most interesting applications is scanning of real life objects and people in particular. The goal of this project is to create a scanning system for getting a 3D model of a human body.

For this year we already have source code of the previous year projects and our goal is to improve performance and especially achieve better results in 3D mesh reconstruction.

Basic requirements for the project

- Acquisition duration must not exceed 90 seconds in total;
- As the output the system must provide watertight triangular meshes;
- The system must allow for the import and the export of 3D textured meshes;
- The user interface must allow to preview the scan and individual acquisitions.
- The acquisition has to be done with Kinect V2 or Intel R200 sensors (or both).
- Implementation must be in C++ only. IDE can be Qt or Visual Studio. OS must be Windows only.
- Extra libraries can be used once agreed with tutors.

Critics of the previous year projects

We have the code of the previous year projects and we might use it for our project development. For this, we decided that it's important to study their reports and code and find out what can be improved and what should we work on.

The first thing that we thought about is Data Acquisition. Students of the previous year were using Kinect not efficiently; they were scanning the whole body, so point clouds they got were noisy and poorly detailed. Better idea would be rather move Kinect and scan from closer distance to get better quality RGBd data.

The other (probably the main) issue is that they were using signal from encoder to calculate rotation. This decision has some disadvantages: firstly, it neglects translation component in point cloud transformation. Secondly, the signal from encoder can be corrupted by environment parameters and rotating table is not the common thing. We would rather suggest calculating

translation and rotation using just images from camera using feature extraction and feature matching with OpenCV library.

From not satisfactory alignment of point clouds derives the problem of 3D meshing. Quality of 3D models is not that high as desired. Besides, we noticed that mostly 3D models created with Poisson method were not really good in sense of detalisation (faces, for example, were poorly reconstructed). Therefore, good idea might be to use greedy triangulation algorithm instead, as it was giving the best results out of all. 4th group was working with hole filling, but it still can be improved. None of the group built colored 3D model – that is also what we can implement.

Last but not least, we can mention that code they were writing contains small amount of comments and that makes difficult to understand it. Some projects contain code that is not used for 3D model building at all. We would try our best to be as clear in our code as we can and make it easy for user. We also are going to make our own graphical user interface using Qt Designer to make it more user- and developer-friendly.

Objectives

After we have read the reports of the previous year teams and tested their programs, we proposed a list of possible improvements and set further goals for our project:

1. Implementation of alignment methods using OpenCV.
2. Perform data acquisition with Kinect in order to save depth and color images in order to make it work in OpenCV.
3. Propose new filtering method of the registered point cloud before 3D reconstruction.
4. Work on possible solution for hole detection.
5. Provide resulting mesh in color.
6. Design our own GUI for usage easiness.

Project management

In order to organize our work and use time efficiently, we created a Gantt diagram considering the importance of mentioned above tasks.

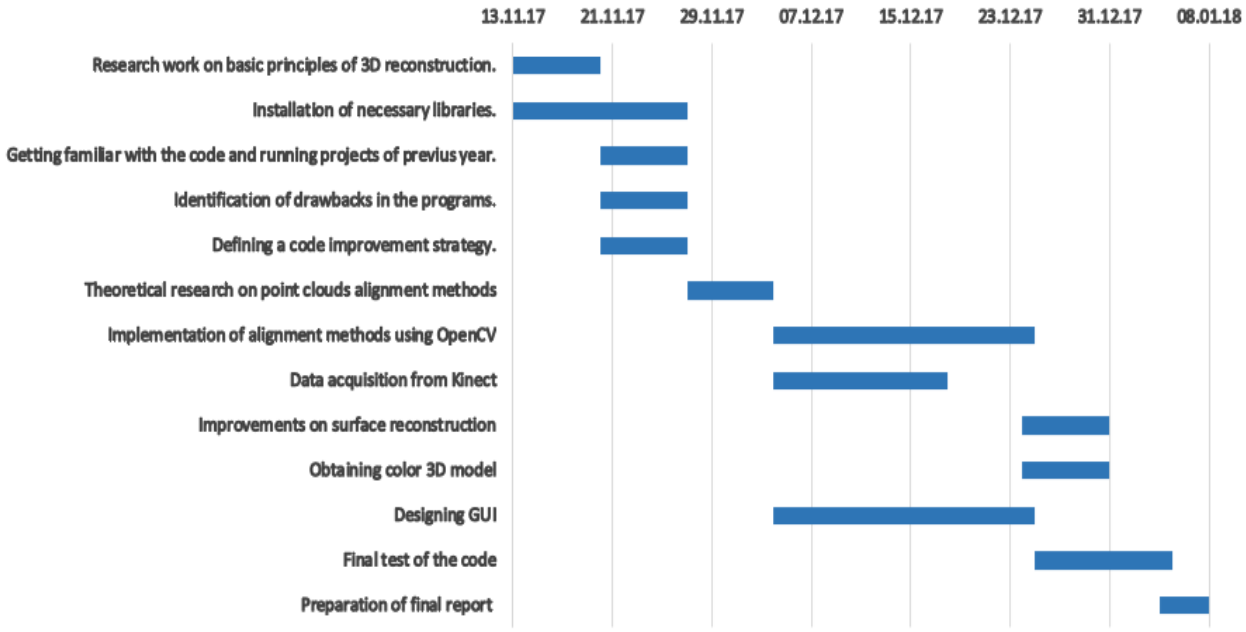

Figure 1.1: Gantt diagram

We had weekly meetings to discuss the progress of our project, define difficulties we face, propose solutions and make plans for the next week. We were writing weekly reports where we put theoretical overviews, summary of meetings and personal achievements. Besides these meetings we also were gathering for collaborative work on project and were attending tutorials that our tutors held.

For project purposes we used GitHub repository, Slack and different messengers.

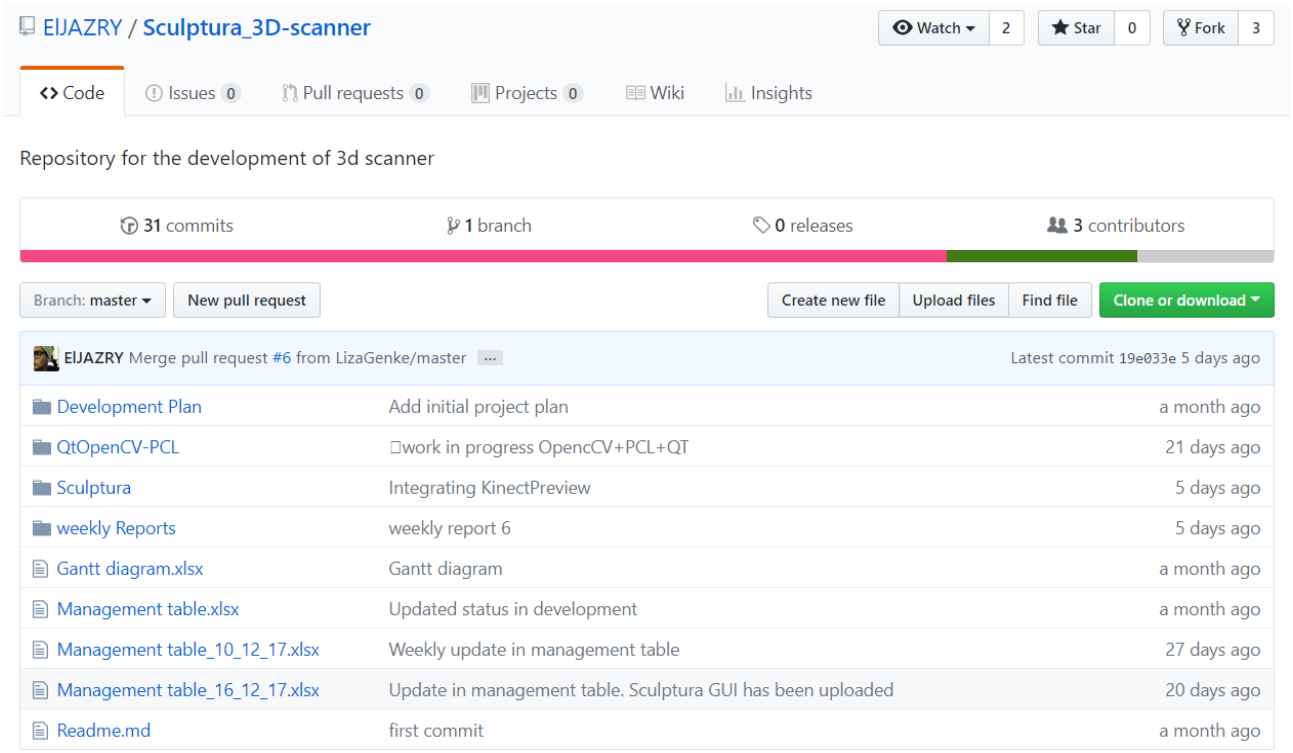

Figure 1.2: Github repository

Roles in the team:

- Mohamed was working on feature detection using OpenCV and ICP alignment algorithm;
- Karim was working on Kinect data acquisition;
- Elizaveta was working on 3D meshing of pointclouds;
- Daria was working on GUI implementation.

Everyone was involved in theoretical research, report writing and code implementation and testing.

Methods

KINECT V2

The Microsoft kinect v2 is a sensor. It uses one infrared camera coupled with an infrared projector system. The Kinect has a traditional colour video camera in it, similar to webcams and mobile phone cameras. Microsoft calls this an RGB camera referring to the red, green and blue colours it detects. It has a resolution 1080p (1920 x 1080) for RGB camera and a 512 x 424 for pixels depth camera, both cameras can stream at 30 frames per second. Kinect Sensor reaches a depth measurement accuracy of 4 millimeters at a relatively wide range from 0.8 to 3.5 meters. The Kinect is using structured light technique. Structured light is a process of projecting a known pattern of pixels on to a scene and then analyses the way these deform when hit a surface.

Figure 2.1: Github repository

Feature matching and feature tracking (for motion estimation)

One of the most important steps in 3D model reconstruction is alignment of point clouds. This task requires information about object or camera motion. Global alignment is sensitive to lighting, occlusion, noise and so on. That's why to track this motion we would like to have set of distinctive points that we can easily find in different frames and from them calculate the rotation and translation. In the ideal case, we would like these points to be invariant to image scaling and rotation, invariant in change of illumination and camera viewpoint and, of course,

be matched with high probability [1].

So, the steps for alignment using features are the follows: detect features → find corresponding pairs → align images.

Camera Motion Estimation and Tracking

- Given: two camera images f_0, f_1

Figure 2.2: Consecutive frames example

- Wanted: estimate the camera motion u

We will try to extract the essential matrix from the tracked feature from one frame to another, that will be used to align the set of points extracted from every frame To achieve this goal we will try to use Kanade-Lucas-Tomasi (KLT) Tracker:

1. Find image features to track in the first frame and initialize tracks
2. Track from frame to frame using RANSAC to eliminate outlier features
3. Delete track if error exceeds threshold
4. Initialize additional tracks when necessary
5. Repeat step 2-4

The first step is feature extraction. One of the best methods for feature extraction is the SIFT method (SIFT stands for Scale-Invariant Feature Transform).

Below are the main steps for SIFT feature extraction:

1. Scale-space extrema detection (using difference of Gaussians function)
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor extraction

Basic idea of SIFT descriptors extraction can be described like this:

- Take 16x16 square window around detected interest point (8x8 shown below)
- Compute edge orientation (angle of the gradient minus 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations (8 bins)

The next step is feature matching. To find the best match in two images we define distance function that compares two descriptors. Then we find pair with minimum distance. As the methods are not perfect, there always will be false matches. To avoid this, different methods to define similarity were presented.

The simplest approach is to minimize the sum of square differences(SSD). However, this method can result in bad matches. In order to avoid ambiguity we might look to ratio of SSD, one SSD of best matching feature and another of the second best matching feature. This approach gives small values for ambiguous matches. Another approach is to create certain threshold, and here we have a trade off: the threshold should be low enough to exclude false matches, but at the same time high enough to include good matches.

For comparing different feature matching methods Receiver-Operator Curves can be used. To align images one of the most popular approaches is to use Iterative Closest Point method.

SVD Method

Singular value decomposition (SVD) is method that can be applied to any matrix: M = $U\Sigma V^T$, where U - left singular vectors matrix, V - right singular vector matrix, $\Sigma-$singular values matrix. It has many applications for linear algebra problems: matrix rank, matrix inverse, solving linear equation systems etc. We can use it for shape matching problem.

Suppose, we have two point clouds of the same rigid object P and Q, but from different perspectives (points of view). We can say then, that q is the same p cloud (almost) but somehow rotated and translated. So we want to find rotation and translation matrices (actually, translation matrix is simply a vector). These two problems can be solved separately, as we can centralize both p and q clouds (get rid of translation component). So, first we find rotation matrix R and from this we can find translation vector t.

As described in the article [2], to find rotation matrix we should calculate "covariance matrix" S =XWY^T (X and Y are set of points centralized, W is weight matrix). Then we do singular value decomposition S =UWV^T. Due to some analysis we consider that the case when M = $V^TRU = I$ is the best for minimizing the error, from this R= VU^T. (regarding this source [3], we also need to transpose it, and this source [4] tells that if determinant of a matrix R is equal to -1 it's a special reflected case and in such case we need to multiply the third column by -1). After we find translation vector t = **p** - R**q** (**p** and **q** are centroids of these clouds).

But SVD is just one of the point cloud registration. Good results also gives ICP especially when it's prealigned using SVD [5].

Iterative Closest Point (ICP)

ICP (Iterative Closest Point is one of the methods used for rigid point cloud registration. This method requires initial alignment (with SVD), then iteratively removes outliers, and redefines point correspondences. There are 2 most widely used variants of ICP: ICP Point-to-Point and ICP Point-to-Surface. The goal is to minimize Euclidean distance between corresponding points in two point clouds for Point-to-Point ICP, or scalar projection on planar surface for the ICP Point-to-Surface.

RANSAC

RANSAC (RAndom SAmple Consensus) is the method designed to cope with outlier problem (due to noise presenting in the images and change in the point of view there always will be outliers).

Algorithm of RANSAC [6]:

1. Select randomly the minimum number of points required to determine the model parameters.
2. Solve for the parameters of the model.
3. Determine how many points from the set of all points fit with a predefined tolerance ε.
4. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps from 1 to 4 (maximum of N times).

For choosing number of iterations we should solve the following equation for N[7]:

$$1 - (1 - (1 - u)^s)^N = p \tag{2.1}$$

where u - probability that point is an outlier, s - number of points in sample, N - number of iterations ensuring that we get at least one good sample, p - probability that we get a good sample.

From (2.1) we can deduce

$$N = \frac{log(1 - p)}{log(1 - (1 - u)^s)} \tag{2.2}$$

RANSAC divides points into inliers and outliers and yields estimate computed from minimal set of inliers. Then we improve this initial estimate using Least Squares minimization method.

Surface reconstruction

After alignment of point clouds we want to reconstruct the object that had been scanned, in other words, to create a watertight triangular mesh. There are different methods that allow us to do this, which can be divided into two groups: very fast triangulation methods and slower ones that also smooth the model and fill holes. [8]

For our project we have decided to use Poisson surface reconstruction method. This method is resilient to noise and misregistration artefacts [9], which can be really useful considering noisy scans we get from Kinect.

Poisson approach considers surface reconstruction as spatial Poisson problem. For surface reconstruction it requires points with associated normals. We estimate normals using PCL function *NormalEstimation* (it computes normals using PCA for covariance matrix over k-nearest neighbours). The isosurface is computed from the normals field assuming that points from pointcloud belong to surface and there is a relationship between the normal field and gradient of indicator function. So, Poisson method represents the points by a vector field $\vec{V}$ and it tries to find function χ whose gradient best approximates $\vec{V}$: $min\chi \mid\mid \nabla\chi$ - $\vec{V} \mid\mid$. Applying divergence operator this problem transforms into a standard Poisson problem[8].

$$\triangle\chi = \nabla * \nabla\chi = \nabla\vec{V} \tag{2.3}$$

After it reconstructs surfaces by using Marching Cubes algorithm. It builds an octree data structure for the representation of the surface. The marching cubes algorithm divides the pointcloud into a voxel grid by marching through the point cloud and analyzing which points define the isosurface of our object. By detecting which egdes of the voxel are intersected by the model's isosurface the algorithm creates the triangles of the mesh.

As our goal was to create colorful 3D mesh we found a snippet in the Internet [10] which we then applied to our method to add color to created polygon mesh.

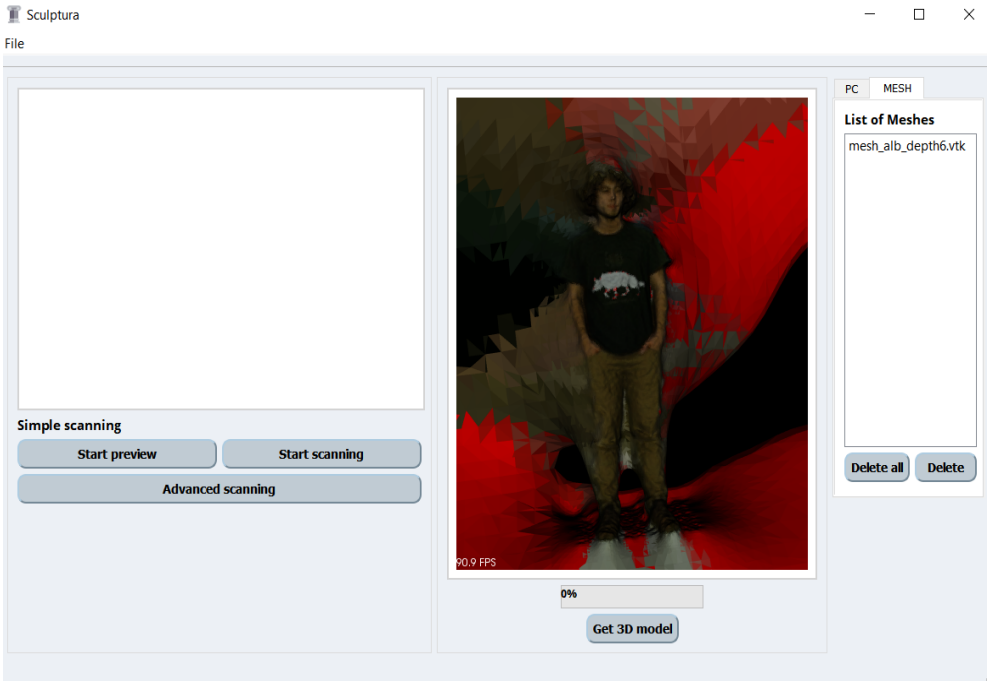

Figure 2.3: Result of surface reconstruction from one pointcloud

Design of the program

Workflow

First, we acquire RGB-d data with Kinect v.2 or Intel R200 sensors. Then we extract features from frames and do 'feature matching' operation for further alignment of point clouds. For 'feature' operations, we are going to use the OpenCV libraries functions. Then we calculate translation and rotation matrices with SVD method and refine this alignment with ICP (there is a possibility that we add RANSAC). After this, we calculate loop closure to exclude overlapping. Then we work with filtering/smoothing for denoising and better quality 3D model. After this we make 3D meshing. Finally, we do texture mapping to add color to our model. This algorithm is shown below in the figure 2.1.

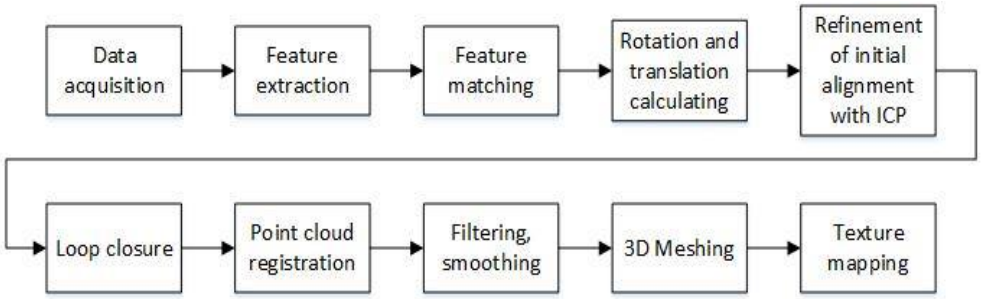

Figure 3.1: Data Processing Workflow

There are many decisions to take regarding the implementation (choice of libraries, choice of methods) and this will be more fully described in the next weekly reports as we reach these points. We have started designing classes for our program (fig.2.2).

Graphical User Interface

Designing Graphical User Interface (GUI) we focused on what ordinary users might need to do and ensuring that the interface has elements that are easy to access and understandable. If average user wants to scan human body, probably he or she does not have strong background about all processes behind. It would be very convenient to have a couple of buttons you need to press in order to receive final result. Basically, we need to see the preview to make sure that camera is placed correctly, then we start scanning. Once we have finished with scanning, we press one button in order to get final 3D model. Following such idea, we at first manually drew a desirable interface and then created a prototype using Mockplus – a free tool which is used by designers and developers to rapidly create software prototypes.

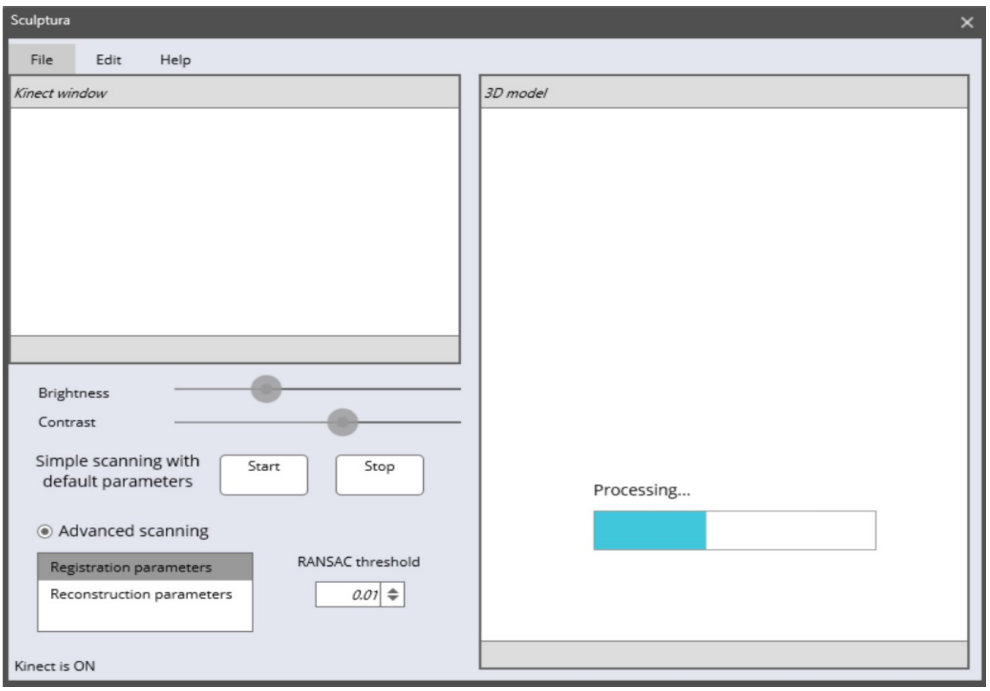

Figure 3.2: GUI prototype

The final user interface consists of main three parts: Kinect window, widget for showing point clouds and meshes, tab widget to display lists of point clouds and meshes. In the upper left corner there is a menu bar which provide further functionality:

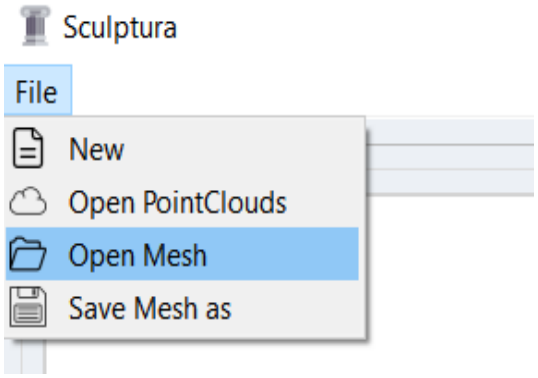

Figure 3.3: Menu bar

Once Kinect is detected, user can click on "Start preview" button, as a result, the color preview will appear in a QLabel. If user wishes to start scanning, the dedicated button should be pressed. PointCLouds in .ply or .pcd and meshes in .vtk or .stl formats can be downloaded, automatically files will be stored into vectors and appear as lists. User should double-click on the name of the file from the list in order to display it. Pressing "Get 3D model" button runs a process of creating a mesh from uploaded point cloud or from point clouds formed after the scanning.
We can say that our main target has been achieved: the program looks user-friendly, it is easy for usage and scanning for those, who are not familiar with technical aspects of the process, is clear and straight-forward.

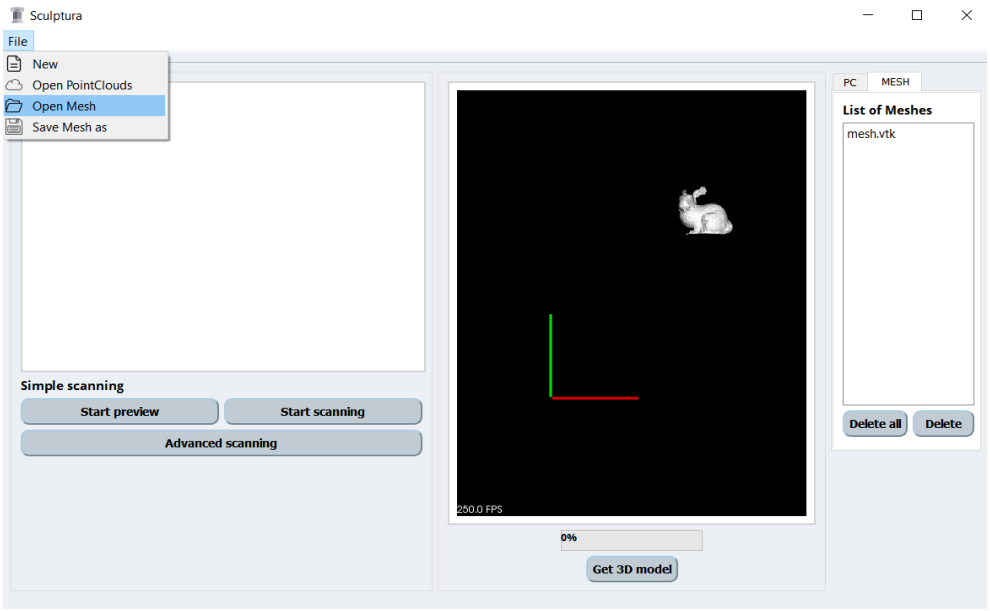

Figure 3.4: Final GUI

Data Acquiring Classes

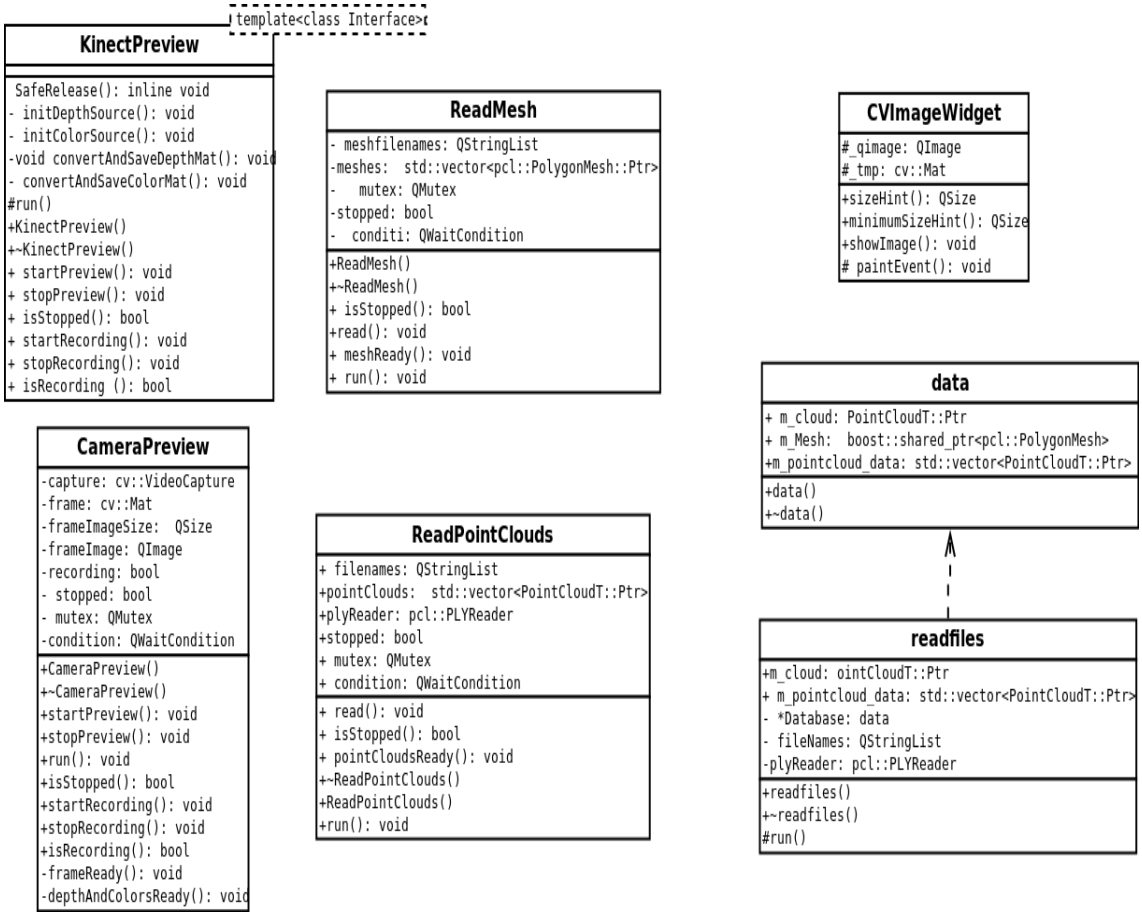

Figure 3.5: Diagram

Results

Overall results

At the very beginning we set very ambitious goals. In the process of work on the project we faced number of problems starting from linking parts of codes problems, pointcloud alignment and even Kinect data acquisition. Our level of debugging, coding and solving problems increased significantly since the start, but still was not enough to achieve all the improvements that we aimed for. However, we managed to improve some aspects comparing to the previous year project:

- At first we decided to work with OpenCV library in order to detect features on 2D images and receive better alignment. Only after this step we convert data into point clouds.
- Kinect successfully grab color images and depth images in a separate thread not to interfere with the main thread, so the program does not crash.

Figure 4.1: Color image

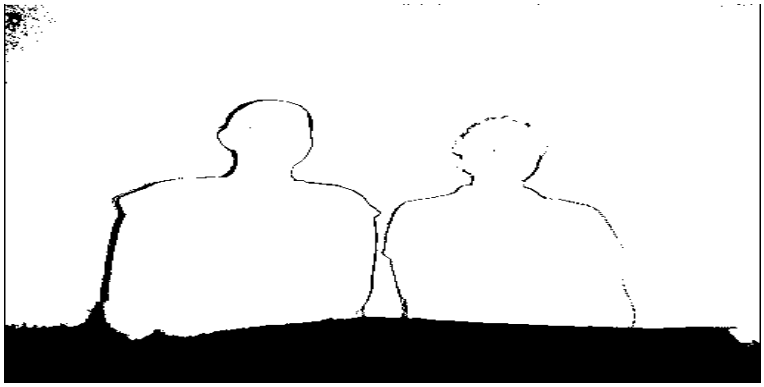

Figure 4.2: Depth image

- We built user-friendly GUI so the scanning should be easy now even for those people who are not familiar with the principles of 3D reconstruction.
- One of the greatest improvements in our project is implementation of 3D reconstruction in color.

Individual results

Daria
At the beginning of the project I did not have experience in C++ coding. It took a lot of time to read and understand the code of the previous year projects. Also the topic of 3D scanning was completely new, but exciting. During research phase I read about such registration methods as SVD, PCA, ICP, got familiar with basic functionality of PointCloud library and OpenCV. My main goal was to create GUI and provide connections between user controls and program execution. I was watching tutorials, searched information in the internet to achieve a desirable form of GUI. The difficulty started when I wanted to show preview from Kinect in a dedicated window (QLabel). It was successfully solved by applying new class inherited from thread. First test was made with built-in webcamera and then applied for the Kinect. The user may see a preview before starting to scan. After the scanning color images and depth images are saved for further processing. Getting familiar with threads helped to implement classes for pointclouds and meshes reading in parallel with the main stream. During the project I better understood the concept of classes and OOP.

Elizaveta

It's my first project requiring coding in C++. Before this semester I didn't knew about OOP at all, and knew just basics of C language (for MC programming), knew nothing about how software should be built, how to connect parts of code, how to use libraries. I wasn't familiar with 3D scanning and 3D model reconstruction as well, so the whole project was challenging. However, in the process of working on the project I tried my best to acquire necessary skills. I was doing my research, reading articles and writing theoretical overviews, which we regularly included in weekly reports, so everyone in our group also can read about theory behind the process of building 3D model from pointcloud. I was working on surface reconstruction and tried different methods (Greedy projection triangulation and Poisson method). As Poisson method gave better results, this approach was chosen by our team. I found out how to add color to resulting mesh, that's, probably, my main contribution to our project. This brought great improvement comparing to the previous year projects. Looking back, I see big progress in my knowledge and skills.

Karim

This was my first experience with c++ object oriented programming, this project was so interesting for me because I used my previous bachelor degree knowledge of mechatronics, since I have dealt with alot of senors, but understanding this sensor is quit hard, but me and my team were able to figure it and read about the kinect sensor, my tasks were to aquire data from the kinect provide RGBD image and PCL, but I was only able to get RGPD image during scan

with talking 1 frame second, in order to do so I grabed one frame from every 30 frames, in order to get the same speed of the kinect, and the application was running smoothly with my algorithm, I also faced another problem with my tasks was the release of the frame , Kinect required to release the frame in a safe mode, but eventually I was able to overcome all these diffculties, my team-mates were helpful, but we all had difficulties first with c++; but with the help of the previous lectures of Prof. Yohan, and the assistance of tutor David and tutor Cansen and their guidance, we were able to build a new program and get used to a very nice libraries such as OpenCV , PCL, BOOST, and other libraries. At last, I am happy that I learnt from this course, and I am NO longer thinking that programming is just a while loop or an if statement.

Mohamed

From my previous study Bachelor in computer vision I had background about using OpenCV library for image processing, for this year project I used more functions from OpenCV library as a pre-alignment method before generating the PointCloud to improve the Point Cloud Registration using PCL ,in this project my experience in software designing and project management increase from identifying the problem and the possible solution for it and also what is the most suitable development approach for our team and the complexity of the tasks and design different Pipelines for achieving our development goal from the project and my C++ skills mostly about using more OOP programming paradigm and get more practicing in using pointers and references , also as I get more involved in using different libraries and the doing research and learn from the documentations of the library .

References

[1] *Distinctive image features from scale-invariant keypoints* David G. Lowe https://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/lowe_ijcv2004.pdf

[2] *Least squares rigid motion using SVD* Olga Sorkine-Hornung and Michael Rabinovich https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

[3] ICP with SVD – Hands on http://www.markuszancolo.at/2015/12/icp-with-svd-hands-on/

[4] Finding optimal rotation and translation between corresponding 3D points http://nghiaho.com/?page_id=671

[5] *A Survey of Rigid 3D Pointcloud Registration Algorithms* Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn https://repository.uantwerpen.be/docman/irua/1ab789/4a6a3c9a.pdf.

[6] *Overview of the RANSAC Algorithm* Konstantinos G. Derpanis http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf

[7] *Computer Vision I: Introduction to Computer Vision* Lecture Notes, CSE Department, Penn State University http://www.cse.psu.edu/~rtc12/CSE486/

[8] *Screened Poisson Surface Reconstruction* MICHAEL KAZHDAN, HUGUES HOPPE http://www.cs.jhu.edu/~misha/MyPapers/ToG13.pdf

[9] *SSD: Smooth Signed Distance Surface Reconstruction* Fatih Calakli and Gabriel Taubin http://mesh.brown.edu/ssd/paper.html

[10] http://blog.csdn.net/sparta_117/article/details/78134819