Master MAIA Software Engineering Project

u2.cloud - A 3D Scanning Program in C++ for Watertight Surface Reconstruction

---

Authors:
Mahlet Birhanu
Brianna Burton
Oleh Kozynets
Doiriel Vanegas

Supervisors:
Prof. Yohan Fougerelle
Dr. Cansen Jiang
Mr. David Strubel

January 7, 2018

Contents

Nomenclature

The next list describes several abbreviations that will be later used within the body of the report.

BF matcher	Brute-Force Matcher
FLANN	Fast Library for Approximate Nearest Neighbors
GUI	Graphical User Interface
ICP	Iterative Closest Point
IR	Infrared
MLS	Moving Least Squares
OpenCV	Open Source Computer Vision Library
PC	Point Cloud
PCL	Point Cloud Library
SIFT	Scale Invariant Feature Transform
SNR	Signal to Noise Ratio

1 Introduction

One of the main tasks in computer vision consists of reconstructing a scene or object based on three dimensional (3D) point cloud (PC) data. In the last decade, 3D digital scanning devices have become available to the public making scanning, and virtual and physical replication, a practical reality in a wide range of fields including industrial design, healthcare and art. 3D scanning produces a high-precision digital reference document that records features, provides a virtual model for replication, and makes possible mass distribution of digital data [1].

One of the most popular 3D scanners is the Microsoft Kinect. It contains an infrared (IR) projector, a color camera, and an IR camera, which together provide both color (RGB) and depth data. Kinect's depth sensing technology is based on the structured light principle, in which the depth sensor consists of the IR projector combined with the IR camera. Although it was first developed as a complement for the Xbox game console to track movement and gestures, it has become widely used by developers due to its low price [2].

3D scanning and reconstruction of the human body is relevant for many fields such as cosmetic and medical applications, but is often challenging and computationally expensive. Students of the Masters in Computer Vision and Medical Imaging at the University of Burgundy, have developed a software to perform acquisition, registration and reconstruction on 3D point clouds of the human body. The aim of this project is to optimize their methods and generate more accurate 3D models.

2 Background Theory

2.1 Provided Data

As the code from four different groups was provided in this project, it was imperative to determine which parts of the code were best for use in a new, improved project. A brief analysis of the provided projects is done below.

2.1.1 Group 1: 3D Korn

Beginning with the name, the 3D Korn project was quickly realized to be the best documented project. The group detailed their progress throughout the project well and had a clear structure. A class diagram was provided, useful for programmers inexperienced in their software. The structure of the algorithm includes registering PCs using ICP point to point and filtering the registered PC. The project was the first one to be built, when testing, and when compared with other projects, was most robust.

A few problems exist with this project including a class called TDK_PointOperations that is created to handle filtering and meshing because the filtering algorithms are rewritten in the ScanRegistration class, defeating the purpose of creating a class to handle them. Also, many filters such as PassThrough filtering are included, but never implemented in the code. Although the group claims to have written three meshing algorithms, only Poisson meshing is successfully implemented in their project. Lastly, the program was annoying to use as there was no indication when the program was registering or meshing PCs.

2.1.2 Group 2

The second group has an unclear report, compared with the first group. They implement filtering before registration, but it is hard to see what the group achieved from their results. This project was chosen to be disregarded.

2.1.3 Group 3

The group 3 project, vtk, contains well written code and good documentation, as well as provides good task division. In the project, the algorithm includes filtering, registration using ICP point to plane (unique to their group), and surface reconstruction with smoothing. They implemented all of the filters and smoothing they describe in the classes cfilter and regmesh. They also achieved the most impressive results. Their project was able to be built, but the program crashed upon opening. Therefore, this was decided an unsafe basis for the new project, but the impressive results caused this project to become a source of inspiration for the u2.cloud program.

2.1.4 Group 4

Group 4's project utilized only one class, of which all others are members. Although this could be seen as a possible form of improvement, all of the other given projects had more optimized code, and this project was chosen to be disregarded, since the task would be so large and would not concentrate on improving the generated mesh.

2.2 Image Feature Detection and Matching

Regarding the fact that the Kinect sensor creates PCs by combining information from depth and color images, we can reconstruct these images from the given Kinect generated PCs. As a result, methods developed for processing of ordinary images can be applied with some constraints for analysis of such PCs. In our case, feature detection can be utilized for improving scan registration, it was decided to use *Scale Invariant Feature Detection* (SIFT) for extracting distinctive image features. The method was proposed by David G. Lowe in his article 'Distinctive image features from scale-invariant keypoints' [3]. Because these features are invariant to image scaling and rotation operations, changes in the 3D viewpoint, alterations of the *signal to noise ratio* (SNR), and illumination fluctuations, they can be used to estimate a transformation matrix for the registration of PCs obtained from Kinect sensor. According to [3], the following major stages are used to obtain the set of image features:

1. **Scale-space extrema detection:** The first stage of computation searches over all image scales and locations. It is implemented efficiently by using a difference-of-Gaussian function to identify the potential interest points that are invariant to scale and orientation. Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and $k\sigma$. This process is done for different octaves of the image in Gaussian Pyramid, after this local extrema are searched in the images. Regarding different parameters, the paper gives some empirical data which can be summarized as, number of octaves is 4, number of scale levels is 5, initial $\sigma = 1.6$, $k = \sqrt{2}$ as optimal values.

2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine the location and scale. Keypoints are selected based on measures of their stability. The Taylor series expansion of scale space is used to get more accurate location of extrema, and if the intensity at this extrema is less than a threshold value (0.03 according to [3]), it is rejected. The difference of the Gaussian has higher response for edges, so edges also need to be removed.

3. **Orientation assignment:** One or more orientations are assigned to each keypoint location, based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby rendering these transformations invariant. In detail, a neighbourhood is selected around the keypoint location depending on the scale, and the gradient magnitude and direction are calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. It is weighted by the gradient magnitude and Gaussian-weighted circular window with σ equal to 1.5 times the scale of keypoint. The highest peak in the histogram is taken, and any peak above 80% it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contributes to stability of matching.

4. **Keypoint descriptor:** The local image gradients are measured at the selected scale, typically a 16x16 neighbourhood is used, in the region around each keypoint. The gradients are transformed into a representation that allows for significant levels of local shape distortion and change in illumination. For that purpose, the 16x16 neighbourhood is divided into 16 sub-blocks of 4x4 size, for which an 8

bin orientation histogram is computed. As a result, a total of 128 bin values are available, which can be represented as a vector to form a keypoint descriptor.

After finding keypoints in the two provided images, we have a need to determine feature descriptors and match them against each other. As stated in [3], the keypoints between two images are matched by identifying their nearest neighbours. In case the second closest-match lies near to the first match, due to noise etc., the ratio of first-closest distance to the second-closest distance is taken. When it is greater than 0.8, both matches will be rejected. Such a procedure eliminates around 90% of false matches while discarding only 5% correct matches [3].

2.3 Filters and Smoothing

Because the data acquired from the Kinect scanner is noisy, it is necessary to remove points which do not add information to the PC. All code for filters was written in the filters class and applied throughout the Scanregistration and meshing classes.

2.3.1 PassThrough Filtering

Pass through filtering is performed using the pcl::PassThrough method, where the PC is filtered along certain dimensions, and points are cut off outside a chosen range. It first filters in the x direction, then the y direction, and lastly the z direction.

2.3.2 Statistical Outlier Removal

Statistical outlier removal filters the data by removing points in a neighbourhood that falls out width a certain distance. The function called pcl::StatisticalOutlier removal accomplishes this by applying a Gaussian distribution to the entire PC, and removes the points that are further than a predefined number of standard deviations from the others.

2.3.3 VoxelGrid Downsampling

To reduce the sheer amount of points processed when each button is pressed, down sampling is imperative. pcl::VoxelGrid is used, and it filters the PC by choosing the average point in a voxel of size x,y,z, where the parameters are provided by the user. A square voxel was chosen.

2.3.4 Moving Least Squares Smoothing

Moving least square (MLS) smoothing approximates a surface as a spatially varying low-degree polynomial [4]. First, the tangents to points are found, and they are projected into tangent space. Then, in the tangent space, the initial PC is approximated by a low-degree polynomial. This improves the resulting cloud's normals. MLS is performed using the function pcl::MovingLeastSquares.

2.3.5 Laplacian Smoothing

Laplacian smoothing moves the vertices of the input mesh to find a more smooth final mesh [5]. It uses information from the nearest neighbours, such as their positions, to

find a new position for each vertex. The function pcl::MeshSmoothingLaplacianVTK was used to perform this smoothing.

2.4 Registration

Registration algorithms are able to estimate the motion of the sensor by calculating the transformation that optimally maps two PCs, each of which is subject to noise [6]. These registration algorithms can be classified coarsely into rigid and non-rigid approaches. Rigid approaches assume a rigid environment such that the transformation can be modeled using only 6 Degrees Of Freedom (DOF). Non-rigid methods on the other hand, are able to cope with articulated objects or soft bodies that change shape over time [6].

Rigid registration can be approached by defining a cost function that represents the current matching error. This cost function is then minimized using common optimization techniques. If the distance between corresponding points in each 3D PC needs to be minimized, this can be simplified to a linear least-squares minimization problem by representing each point using homogeneous coordinate [6]. Two of the algorithms that perform pairwise registration are Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).

Both PCA alignment and SVD are pairwise registration methods based on the covariance matrices and the cross-correlation matrix of the PCs. There are also iterative algorithms that intend to minimize the cost function used to estimate matching error [6]. These algorithms are named, Iterative Closest Point (ICP) algorithms consisting of many variants.

2.4.1 Principal Component Analysis

PCA essentially projects data on a new orthonormal basis in the direction of the largest variance [7]. It solves

$$det(A - \lambda I) = 0 \tag{1}$$

to obain the eigenvectors of the PCs. Once the eigenvectors are known for each PC, registration is achieved by aligning these vectors.

Since PCA based registration simply aligns the directions of the largest variance of each PC, it does not minimize the Euclidean distance between corresponding points of the datasets [8].

2.4.2 Singular Value Decomposition

Given that point correspondences between the PCs are available, it is possible to directly minimize the sum of the Euclidean distances between these points. This corresponds to a linear least-square problem that can be solved robustly using the SVD method [7]. Based on the point correspondences, the cross-correlation matrix M between the two centered PCs can be calculated, after which the eigenvalue decomposition is obtained as in Equation 2

$$M = USV^T \tag{2}$$

The optimal solution to the least-square problem is then defined by rotation matrix R as in Equation 3

$$R = UV^T \tag{3}$$

The SVD algorithm directly solves the least-square problem assuming perfect data, which is not actually the case in real life.

2.4.3 Iterative Closest Point-to-Point

The input of the ICP algorithm consists of a source PC and a target PC [9]. Point correspondences between these PC are defined based on a nearest neighbor approach or a more elaborate scheme using geometrical features or color information. SVD, as explained in the previous section, is used to obtain an initial estimate of the affine transformation matrix that aligns both PCs. After registration, this whole process is repeated by removing outliers and redefining the point correspondences.

The ICP point-to-point algorithm simply obtains point correspondences by searching for the nearest neighbor target point q_i of a point p_j in the source PC. The nearest neighbor matching is defined in terms of the Euclidean distance metric in Equation 4

$$i = \arg\min_i \left(\|(q_i - p_j)\|^2 \right) \tag{4}$$

where i ∈ [0, 1, ..., N], and N represents the number of points in the target PC.

2.4.4 Iterative Closest Point to Plane

The ICP point-to-surface algorithm assumes that the point clouds are locally linear, such that the local neighborhood of a point is co-planar [10]. This local surface can then be defined by its normal vector n, which is obtained as the smallest eigenvector of the co-variance matrix of the points that surround correspondence candidate q_i.

Instead of directly minimizing the Euclidean distance between corresponding points, it minimizes the scalar projection of this distance onto the planar surface defined by the normal vector n in Equation 5

$$R, t = \arg\min_i \left(\left\| \left(\sum_{i=1}^{N} \mathbf{RP}_\mathrm{i} + \mathbf{t} \right) \right\| \right) \tag{5}$$

2.5 Surface Reconstruction

Surface reconstruction, or the creation of a 3D mesh, consists of determining the topology of the surface from a PC, fitting a mesh to the data, and filling holes in the mesh [11]. An array of surface reconstruction algorithms exist, divided between global and local methods. Global methods consider the entire PC, and each point's distance to the center, while local methods use the distance from tangent planes to the nearest points. Poisson reconstruction combines both local and global methods, while grid projection and greedy triangulation are local methods.

The aforementioned meshing algorithms were implemented in this project as mentioned in the Group 3 project, as well as a new meshing technique called Marching Cubes, in the TDK_meshing class.

2.5.1 Poisson Reconstruction

The Poisson method reconstructs a surface as a Poisson problem, using a PC and its inward facing normals [11]. The Poisson problem computes a scalar function χ, whose

Laplacian equals the divergence of the vector field $\overrightarrow{V}$ such as in Equation 6

$$\Delta\chi = \bigtriangledown \cdot \bigtriangledown \chi = \bigtriangledown \cdot \overrightarrow{V} \tag{6}$$

To perform Poisson reconstruction, a 3D indicator function is defined, χ, by assigning 1 to points inside the PC and 0 to those outside. Then, using the fact that the gradient of this indicator function is zero except near the surface, the gradient function of χ is the inward facing normals. Using Equation 6, the Poisson problem is solved. In this problem, the PC and its normals are related. Poisson meshing works best with noisy data, to recover final details.

Parameters of the Poisson algorithm include: the input cloud, depth, scale, solver divide, iso divide, samples per node, confidence, manifold, and output polygons. The output is a polygon mesh.

2.5.2 Grid Projection

Contrasting to the meshing algorithm using Poisson's equation, another method, called *grid projection* uses polygons to approximate an external surface [12]. The method as explained by Li [12] requires PCs and unoriented normals, and owing to this, it produces a continuous surface. To project a grid onto the given PC, the grid edges must first be identified, namely critical and external edges. Critical edges are determined where the scalar function, *g(x)*, in Equation 7 evaluates to zero.

$$g(x) = \overrightarrow{n(x)} \cdot \bigtriangledown s(x) \tag{7}$$

where *s(x)* is a two parameter function that depends on a direction, and $\overrightarrow{n(x)}$ is the oriented vector function. Since n(x) is continuous across any edge, the critical edges occur when $\bigtriangledown s(x)$ is zero. Next, the extremal edges, the critical edges are local minimum of s restricted to $\overrightarrow{n(x)}$, are calculated by linear interpolation along the critical edge. After the edges are determined, a polygonal surface is projected, keeping in mind the extremal edges.

Grid projection is important, as it estimates a surface using polygons, which is the idea behind a triangular watertight mesh. In addition, it is seen to create much more smooth, continuous meshes from PCs, than other methods. Parameters of the grid projection algorithm include: the input cloud, resolution, and padding size. The output is a polygon mesh.

2.5.3 Greedy Projection Triangulation

Compared to both the Poisson and grid projection methods, *greedy triangulation* is most similar to the grid projection method. Greedy triangulation can produce a surface reconstruction for any given convex shape. First, a PC is given and empty set initialized. Then, the set is grown by adding the shortest compatible edge between two points, without an edge crossing any previously added edges. The PCL function greedy triangulation is based upon Dickerson's 1994 paper [13]. In Dickerson's method, the set is initialized, all point to point distances are computed, and the lengths are sorted and examined. Then, an edge is drawn between two points of the shortest distance, that does not cross another line.

Parameters of the greedy triangulation algorithm include the search radius, standard deviation, maximum nearest neighbours, maximum angle between surfaces, minimum and maximum angle, and normal consistency. The output is a polygon mesh.

2.6 Marching Cubes

In addition to the algorithms mentioned above, a new method called *marching cubes* was implemented in the code, to test its robustness for reconstructing a watertight mesh. It was chosen to use another grid projection algorithm, as the above grid algorithms obtained a more watertight mesh than the Poisson algorithm. Marching cube reconstruction is a popular meshing technique and the Hoppe iteration was implemented in this project [14]. Hoppe's method first estimates a tangent plane for each data point using PCA. These planes are approximations of the surface. Then, contour tracing is used to extract a surface from the tangent planes, producing a continuous linear approximation. In this step, a cube size is set, to determine the size of the facets generated (this is set by the grid resolution).

Parameters of the Hoppe marching cubes algorithm include the isolevel, grid resolution, and percentage extended grid. The output is a polygon mesh.

3 Implementation

3.1 Program Design

In programming, it is ideal to work from the ground up in designing a software, in order to understand all the caveats of a program's functionality. Due to the limited time frame of the project and sheer scope involved in becoming learned in the topic of 3D scanning, it was decided to use one of the projects as a base for the 3D scanning program. From the initial analysis of the provided programs, presented in Section 2.1, it became obvious that the code of Group 1 3D Korn was most optimized for the project. In addition, the results of Group 3 were impressive, and with our research about the robustness of the ICP point to plane algorithm, the methodology of Group 3 was chosen to be implemented in the working program of Group 1. A UML diagram, representing our final design can be found in Appendix A.

3.2 Image Feature Detection and Matching

All the required functionality was implemented as TDK_2DFeatureDection class. The method TDK_2DFeatureDection::setInputPointCloud should be used to input the first PC, and another public method TDK_2DFeatureDetection::getMatchedFeatures should be used in order to get two PCs of matching features as an output. One should provide the sample PC that one wants to match against as reference (input) PC.

Internally, input PC's 3D coordinates are mapped to the pixels of the image. As a result, we get the initial image that the Kinect sensor utilizes for the creation of a PC. Such images can be processed by regular image processing libraries.

In our work, we used a realization of the SIFT algorithm provided by *Open Source Computer Vision Library* (OpenCV). The OpenCV library gives us two feature matchers: *Brute-Force Matcher* (BF matcher) and *Fast Library for Approximate Nearest Neighbors* (FLANN) based matcher. According to the OpenCV documentation [15], the FLANN based matcher may be faster when matching a large train collection than the brute force matcher, but in our case we had a small amount of features detected on every image, i. e. around one hundred. Moreover, the BF matcher implements a cross check of matching features, so only consistent pairs will be returned. This is an alternative to the ration test, used by D. Lowe in his paper [3]. As a result, we selected the brute force matcher as our main tool for determining matching keypoints.

3.3 Registration

As metioned above, we used the 3D Korn project as our base and improved registration implementation. 3d Korn only uses basic ICP operations on PCs. For comparison purposes, we integrated the implementations of both ICP point to point (TDK_ ScanRegistration::ICP) and ICP point to surface (TDK_ScanRegistration::ICPNormal) to register PCs. Because ICP point to surface gives more or less a better result, the program is set to use ICPNormal function (TDK_Scan Registration::ICPNormal) by default, unless told otherwise by the user.

Given that ICP (along with all its variants) is a piece-wise algorithm, it finds the best transformation between two PCs (source and target) and applies the transformation on the source to align it with the target. It outputs the result which will then be used as a source for the next PC in the vector.

ICPNormal also finds the best transformation of the normals of the PCs given as input. To achieve this, it calculates the normals first, and then iteratively minimizes the error between them. It uses SVD to obtain the initial affine transformation. It then applies the obtained transformation to the PCs, aligns them and returns the result. A short snippet of the code is shown below:

```
pcl::IterativeClosestPointWithNormals<pcl::PointNormal,
pcl::PointNormal> reg;
reg.setTransformationEpsilon (1e-8);
reg.setMaxCorrespondenceDistance (MaxDistance);
reg.setRANSACOutlierRejectionThreshold (RansacVar);
reg.setMaximumIterations (Iterations);
reg.setInputSource (points_with_normals_src);
reg.setInputTarget (points_with_normals_tgt);
reg.align (*normals_icp);
reg.getFitnessScore() << std::endl;
Eigen::Matrix4f transform_normals = reg.getFinalTransformation ();
pcl::transformPointCloud (*src, *cloud_norm, transform_normals);
std::cout<<cloud_norm<<std::endl;
return cloud_norm;
```

In order to register all the given PCs, a vector containing all PCs of interest is given to register function (TDK_ScanRegistration::Register) in the TDK_Scan Registration class, to iterate though the vector and call register previous and current PCs according to the algorithm chosen by the user (ICP or ICPNormal). The code snippet is shown below.

```
for (size_t i = 1; i < Data.size (); ++i)
    {
        cloud_src = final; // source
        cloud_tgt = Data[i]; // target
        if (mv_use2DFeatureDetection == true)
        {
            //code for registration with 2d detection
            //omitted because of its length
        }

        if (mv_ICP_Normals == false)
            result = TDK_ScanRegistration::ICP(src, tgt);
        else
            result = TDK_ScanRegistration::ICPNormal(src, tgt);

        *result += *cloud_tgt;
        final = TDK_ScanRegistration::mf_outlierRemovalPC(result);
    }
```

The registration implementation slightly differs if the user indicates the use of feature detection, which is also another new feature integrated to the project.

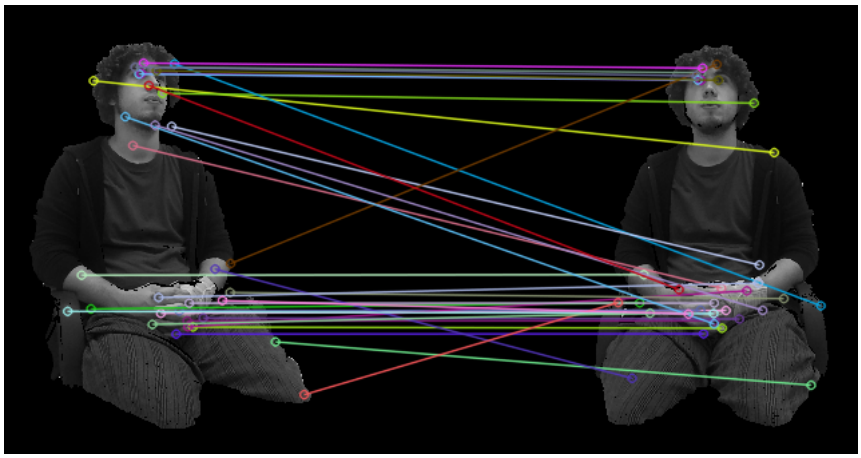
3.4 Registration using Image Feature Detection and Matching

In order to improve the quality of the PCs given to the registration algorithm, we propose the use of 2D features to provide an initial alignment. After computing 2D features, function TDK_ScanRegistration::MatchRegistration takes in the original PCs and their features and performs ICP on the features to obtain a rigid transformation matrix that will be used to align the PCs. As a final step a refinement takes place using the currently selected ICP method. The final output of the function corresponds to the registered PCs fused in one.

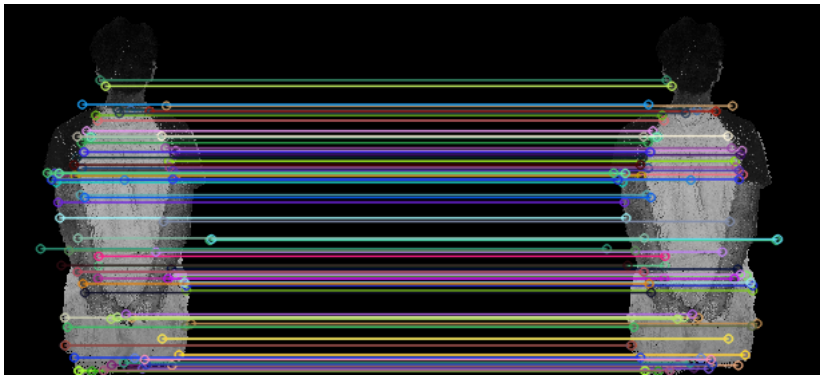
3.5 Surface Reconstruction

First, the user chooses a PC to perform surface reconstruction and a meshing algorithm from the drop down box in the GUI. Once the user presses G̈enerate Mesh,̈the class TDK_centralwidget calls the function for the chosen meshing algorithm from the TDK_meshing class (mf_Poisson, mf_Grid_Projection, or mf_Greedy Projection Triangulation). In each method's function, the selected PC is first filtered using statistical outlier removal (TDK_filters::mf_Filter Statistical Outlier Removal) and then smoothed using MLS smoothing (TDK_filters::mf_Filter MLS Smoothing).

Before the surface reconstruction is performed, normals must be estimated from the PC using the normal estimation function (TDK_meshing::mf_NormalEstimation). This function uses a KDTree to organize the cloud points into k-dimensions and produces information about the normals. Then, the function pcl::concatenateFields is used to store the data from the normals with that from the PC. A surface is reconstructed from the PC with normals and stored as a data type called pcl::PolygonMesh. This data type can store the vertexes of the mesh, the edges, and face data. The final mesh is smoothed with Laplacian smoothing (TDK_filters::mf_Filter Laplacian Smoothing). The resulting mesh can then be displayed in the GUI and exported for use with other 3D programs.



(a) Poor quality PC



(b) Better quality PC

Figure 1: Feature matching

4 Results and Analysis

4.1 Feature Matching

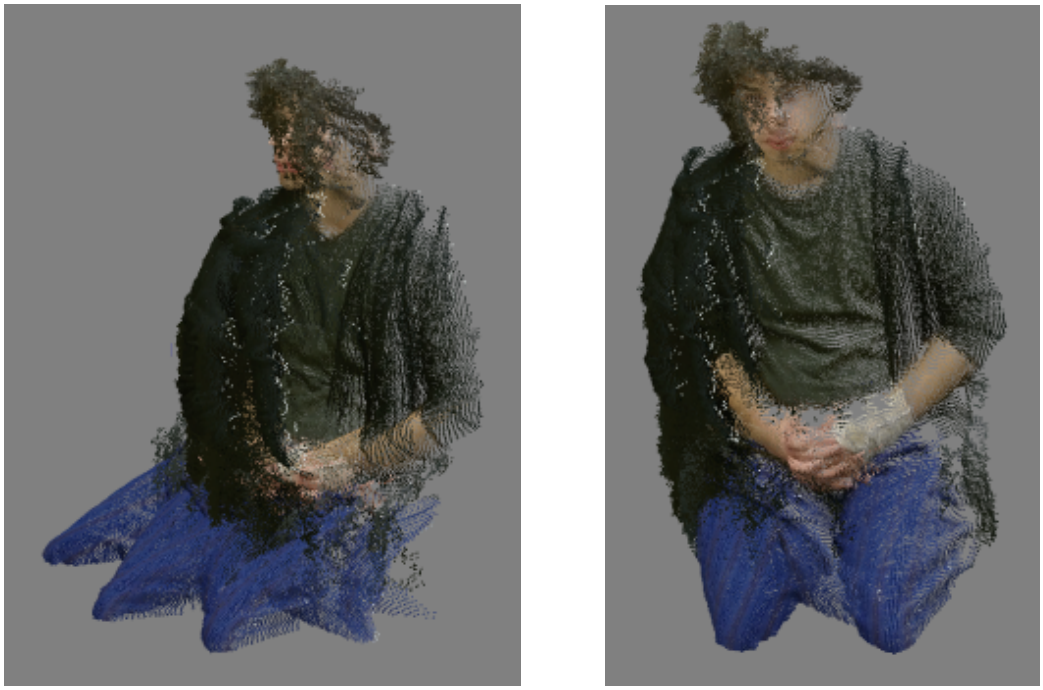
Our program successfully computes the common features between two images, that were obtained from provided PCs. As seen in Figure 1, the quality of detected features is directly related to the quality and similarity of the input PCs. Figure 1b shows the result for identical PCs and Figure 1a for PCs of a person rotated 45°.

4.2 Registration

Originally, the 3D Korn project used simple ICP to register two PCs. We added the computation of normals and applied the transformation to the original PCs as explained in the implementation section.

The results clearly show that registration improved significantly when ICP point-to-surface is used as shown in Figure 2. As ICP point-to-point is very sensitive to outliers, it can be concluded that ICP point-to-surface is a more reliable algorithm for noisy PCs.

Additionally, the results for registration using 2D feature detection on seven PCs are shown in Figure 3. As described above, feature detection is very sensitive to the quality of PCs, and despite filtering and down-sampling was performed, the registration could



(a) Original algorithm (b) Algorithm of ICP with normals

Figure 2: The original and improved meshing algorithms

not be improved. Detecting features in the human body is also challenging because of its complex geometry. In this project we use cropped PCs containing only the person, if some information about the surroundings was given, the feature detection algorithm could find more reliable matches in common geometries like wall edges and corners improving the final registration. Unfortunately, this would also make the process more time consuming. As the result obtained using ICP point-to-surface was clearly better and faster, we decided not to go further with this approach.

4.3 Surface Reconstruction

The results of all four surface reconstruction algorithms to a registered PC are shown in Figure 4. The meshing algorithm that worked best was determined to be greedy triangulation. Although this mesh has some holes, in general, the outline of the person is well defined. Poisson and grid projection also work work, but leave noise along the head and shoulders. The exploratory meshing algorithm marching cubes does not show a good result. This is due to the shape of the person and the small anomalies in the PC above the head.

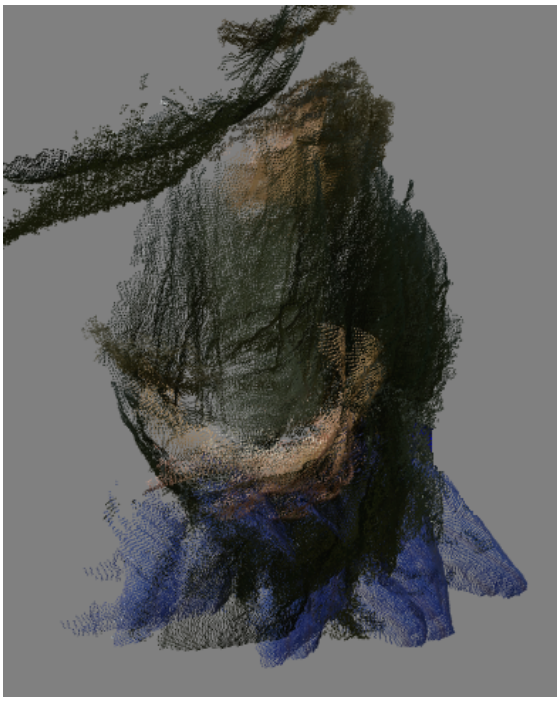


Figure 3: ICP with normals and feature detection for initial alignment

5 Project Management

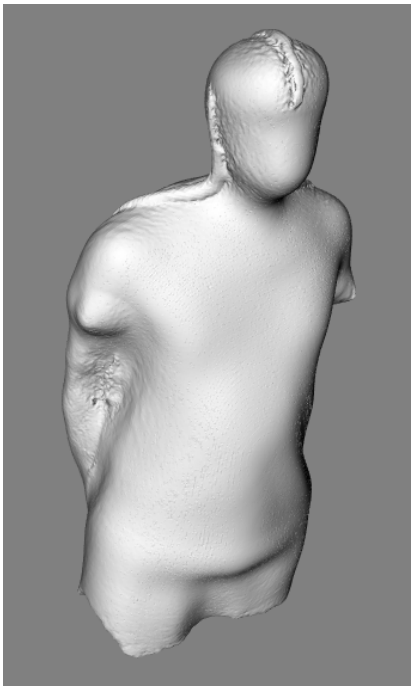
5.1 Team Planning and Task Division

Our project was based on the previous year students' projects. We received source code and supplementary information after 15th of October. The overall duration of the project can be roughly estimated as 10 weeks. Regarding the fact that at exactly the same time we had our mid-term exams started, this time range was quite limited. One of the requirements of the project was to restrict the team to four members.

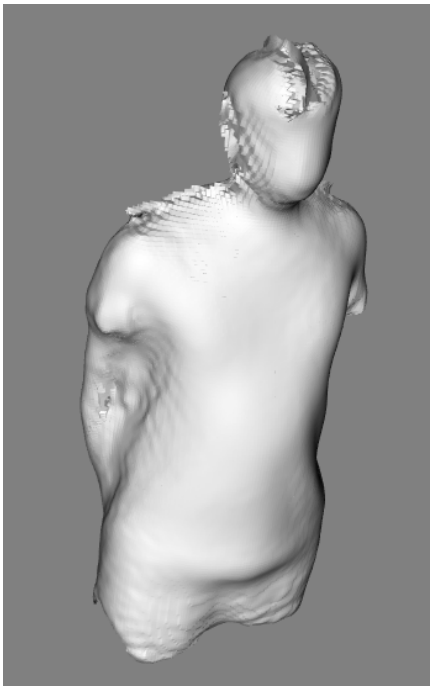
Subsequent analysis of our team skills set showed that we were facing several challenges. First of all, the majority of the team was unfamiliar with the C++ programming language prior to the Software Engineering course and had never participated in development of software with a *graphical user interface* (GUI) using Qt Framework. Secondly, half of the team members were unfamiliar with Git or had no experience of applications development with any other version control system. The same can be said regarding third party libraries used in the project, e. g. PCL, OpenCV. Finally, nobody from the team had any experience of working with Kinect sensor ever before.

In our situation, work arrived in an unpredictable fashion, and we had to pursue code improvement and our understanding of the background theory through evolutionary change. As a result, we were unable to fully organize our work using traditional phased approach, i.e. 'waterfall', which requires detailed planning and a straight time-line. Our development approach is more close to iterative and incremental project management, i.e. 'agile', which values early prototype delivery.

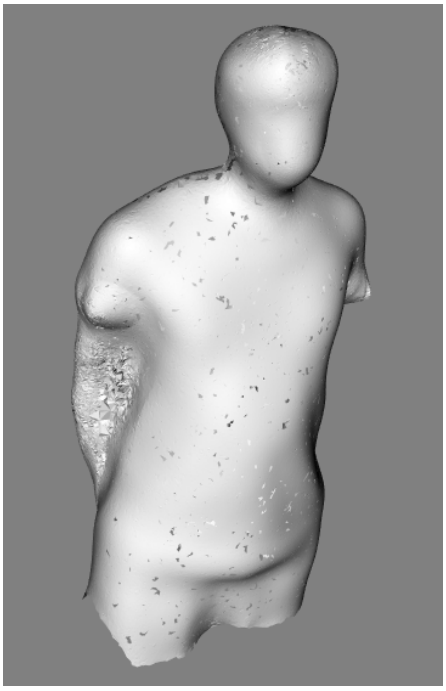
Regarding the fact that nobody had past experience in working with such kind of projects and in such a technical stack, the majority of decisions were a result of inner team negotiations and compromises. For this reason, we had weekly meetings in order



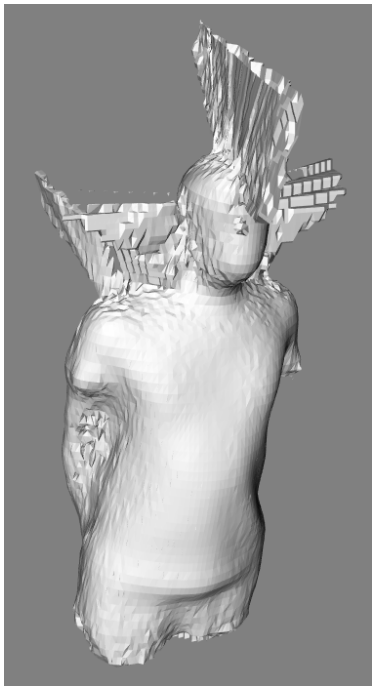
(a) Poisson



(b) Grid projection



(c) Greedy triangulation



(d) Marching cubes

Figure 4: Showing the four meshing algorithms implemented in u2.cloud

to discuss our current goals and problems faced, share opinions and try to find possible solutions together. During the last two months, our group created short weekly reports.

5.2 Project Management Tools

In order to organize our work as a team, we used several services. As a task tracking system, we selected Trello. The version control system was Git, as a result we also worked with the Github service for storing our source code on their servers. We also shared our weekly reports through Github. Members of the development team were encouraged to actively share information with other team members. Thus, team communication was organized through an established Facebook messenger group, and communication with our supervisors was made through the provided Slack server. We shared documents within our team by means of Google Docs and ShareLaTeX services, so that every team member had constant access to our documentation and final report.

5.3 Individual Work

- **Week 1 - Week 2:** Everyone in the group performed a literature review on PCL and the main theories of PC registration. Each member of the group had to read one full project of the previous year.
- **Week 3:** Everyone installed PCL, VTK, Visual Studio, and miscellaneous software to build the projects.
- **Week 4:** We selected the source code to work on. Oleh was the first to build the 3D Korn project, and afterwards, each member successfully built the project. Mahlet and Brianna did research on registration and filters, which were used for the rest of the projects.
- **Week 5 - Week 6:** Everyone worked on implementing the method from Group 3 into Group 1's code.
- **Week 7 - Week 10:** Doiriel and Mahlet improved the registration using ICP with normals and worked together. They also cleaned the code, removing redundant, unused, and defective functions. Brianna corrected the filtering algorithms and wrote working functions for the meshing algorithms. She also implemented a new meshing function that was not implemented in any group's source code. Oleh integrated OpenCV and wrote a class for image feature detection and matching and integrated this within the code from Doiriel and Mahlet. Team members added small changes to the programs graphical user interface.

6 Conclusion

The designed software is based on 3D Korn project that successfully scans and registers a full body, and returns a watertight 3D mesh. As the goal of this project was to improve the implementation and assess the choice of parameters and algorithms used previously, we decided to focus on improving the registration and meshing methods.

As per our observation, we noticed the low quality of the registration results which hinted a direction of improvement. As a result, the use of a different variant of ICP that minimizes the error between the distances of the normals of points was proposed. Because this algorithm is more robust for tremendously noisy PCs, it resulted in a better registration output for the same input PCs previously used. In addition to using ICP with normals, the use of feature detection was proposed as a way to initialize transformations before the ICP operations.Theoretically speaking, a transformation from similar features in the input PCs is a good starting point for registration. Unfortunately, the results obtained were not any better than the result obtained from the previous method. Considering the fact that the feature detection algorithm used provides very good results in detecting features, it can be concluded that the geometry of PCs also plays an important role in having a good registration result.

The consequence of using filters repeatedly was also clearly studied as some important features were missing after the use of such method, even though the errors were significantly minimized. An optimum use of such filters was, therefore, significant to get a better result.

Moreover, other meshing algorithms that could work well with different PCs were explored, as the results vary significantly for different types of PCs and were integrated with the program providing a better control for the user.

6.1 Recommendation for future works

2D feature detection is clearly a working progress. Improving registration results with the feature detection can greatly change the output from noisy PCs.

As PCs are generally very noisy, registration and meshing algorithms have different results in different PCs. It could be interesting to figure out the best algorithm to use from the distribution and general statistical properties of the input PCs automatically without multiple trials. More research could be dedicated for this in order to automate the process of choice of algorithm for a given set of PC.

Considering the result of meshing, greedy triangulation provided the best result in comparison.Integrating a holes filling technique at the end could output an even better result.

References

[1] Wachowiak, M. J. & Karas, B. V. 3d Scanning and Replication for Museum and Cultural Heritage Applications. *Journal of the American Institute for Conservation* **48**, 141–158 (2009).

[2] Zhang, Z. Microsoft Kinect Sensor and Its Effect. *IEEE MultiMedia* **19**, 4–10 (2012).

[3] Lowe, D. G. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision* **60**, 91–11020042 (2004). `0112017`.

[4] Berger, M. *et al.* State of the Art in Surface Reconstruction from Point Clouds. *Proceedings of the Eurographics 2014, Eurographics STARs* 161–185 (2014).

[5] Herrmann, L. R. Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division* **5**, 749–756 (1976).

[6] Bellekens, B., Spruyt, V., Berkvens, R., Penne, R. & Weyn, M. A Benchmark Survey of Rigid 3D Point Cloud Registration Algorithms. *Int Journal on Advances in Intelligent Systems* **8**, 118–127 (2015).

[7] Marden, S. & Guivant, J. Improving the Performance of ICP for Real-Time Applications using an Approximate Nearest Neighbour Search. *Proceedings of Australasian Conference on Robotics and Automation* 3–5 (2012).

[8] Draper, B., Yambor, W. & Beveridge, J. Analyzing pca-based face recognition algorithms: Eigenvector selection and distance measures. *Empirical Evaluation Methods in ...* 1–14 (2002).

[9] Rusu, R. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, vol. 24 (2010).

[10] Low, K.-L. Linear Least-Squares Optimization for Point-to-Plane iICP Surface Registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2–4 (2004).

[11] Kazhdan, M., Bolitho, M. & Hoppe, H. Poisson Surface Reconstruction. *Proceedings of the Symposium on Geometry Processing* 61–70 (2006).

[12] Li, R. *et al.* Polygonizing extremal surfaces with manifold guarantees. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, 2, 189–194 (ACM, Haifa, 2010).

[13] Dickerson, M. T., Drysdale, R. L. S., McElfresh, S. A. & Welzl, E. Fast Greedy Triangulation Algorithms. *Computational Geometry* **8**, 67–86 (1997).

[14] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. & Stuetzle, W. Surface reconstruction from unorganized points. *ACM SIGGRAPH Computer Graphics* **26**, 71–78 (1992). `1011.1669`.

[15] OpenCV 2.4.13.5 Documentation, . Common Interfaces of Descriptor Matchers. URL `https://goo.gl/mMGwCt`.

A UML Diagram

The following UML diagram displays the relations between all of our classes and highlights our changes (green blocks in the diagram).