3D Scanning Project

Partners: Fahad khalid, Ali Berrada, Kenechukwu Henry NGIGE, and Basel Alyafi

January 8, 2018

Contents

List of Figures

Chapter 1

Introduction

1.1 Objectives

The objective is to study, analyze and implement an improved version of the previous submitted projects. Identification of limitations in the software developed by predecessor's and creation of an improvement software using OpenCV. Our main purpose was to design an acquisition and processing software which has ability to perform 3D reconstruction.

1.2 Background

Development and integration of scanners for 3D reconstruction, prototype printing, augmented and virtual reality had made 3D scanning an increasingly common part of the world we live in. With more and more affordable depth sensors around such as Microsoft Kinect, 3D scanning is no longer restricted to industrial or research. Among 3D scanning applications, human body scanning has especially gained immense popularity due to its versatile use in the movie, video game, clothing and medical industry, as well as anthropometry and ergonomic applications. As new technologies and tools were continuously developed with massive cost reduction, more realistic figures can be obtained under brief period of scanning. As the number of non-technical user increase to employ these high-end technologies, a solution that allows user to perform minimum amount of operations to acquire final scan result is not only ideal but also demanded. This can be achieved using a 3D scanning rig, which combines point-and-shoot functionality with portability. The main objective of this project is to develop a human 3D scanner software able to fully interface with a scanner rig composed of a turning table and a stationary depth sensor. The software is aimed to perform full body scan under 90 seconds. 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 [5].

1.3 KINECT sensors V2

The Kinect, see Figure 1.1 , it 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 color and the depth of each scanned pixel [6]. 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 color camera.

The Kinect converts the pattern obtained by the CMOS sensor into a depth map. The depth map is aligned to the color picture, obained by the RGB camera, and they are combined into a single RGB-D image that is sent to the USB port. [7]

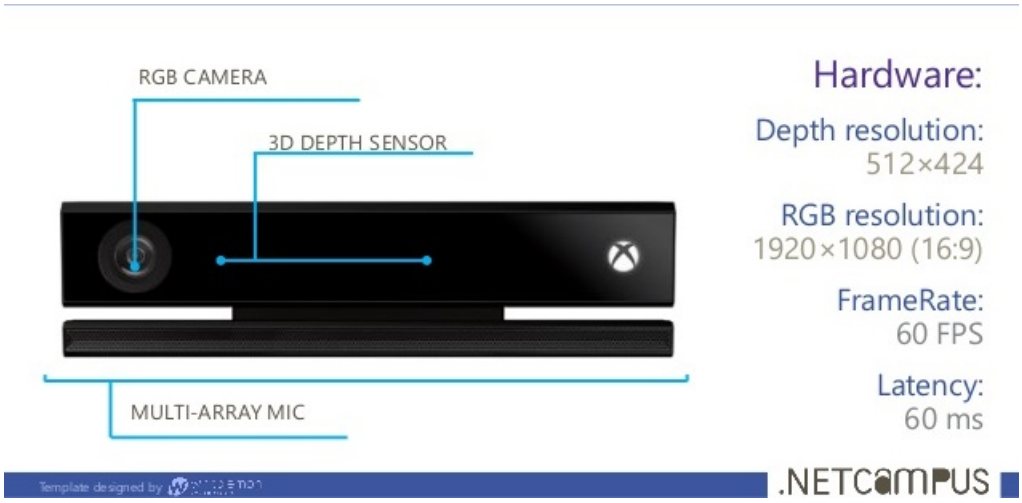


Figure 1.1: Kinect2 sensor

1.4 Microsoft Visual Studio

It is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation

Foundation, Windows Store and Microsoft Silver light. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code re-factorizing. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every levelincluding adding support for source control systems (like Subversion) and adding new tool sets like editors and visual designers for domain-specific languages or tool sets for other aspects of the software development lifecycle QT Compiled with MSCV 2015. Qt is a cross-platform application frame-work that is used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying code base.

1.5 CMAKE

It is also a cross-platform application of tools designed to build, test and package software. It controls software compilation process using simple platform and compiler independent configuration files that generate native make files and workspaces that can be used in the compiler environment of ones choice.

1.6 VTK

The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing and visualization. VTK consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java, and Python. Kitware, whose team created and continues to extend the toolkit, offers professional support and consulting services for VTK. VTK supports a wide variety of visualization algorithms including: scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as: implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. VTK has an extensive information visualization framework, has a suite of 3D interaction widgets, supports parallel processing, and integrates with various databases and GUI tool kits such as Qt and Tk. VTK is cross-platform and runs on Linux, Windows, Mac and Unix platforms. At its core VTK is implemented as a C++ toolkit, requiring users to build applications by combining various objects into an application. Kinect SDK: A Software Development Kit (SDK) is a set of tools used to develop software for an operating system that has been determined. Kinect for Windows Software Development Kit (SDK)

version 2.0 enables developers to create applications that support gesture and voice recognition, with Kinect sensor running on windows version 8 upwards.

1.7 Intel Real Sense SDK

The Intel Real Sense SDK is a library for pattern detection and recognition algorithm implementations exposed through standardized interfaces and aims at lowering barriers by shifting the application developers focus from coding the algorithm details to innovation based on the usage of these algorithms[10].

1.8 PCL 1.8.0

The PCL is an open source project for point cloud processing written in C++. It is aimed to be a cross Platform library which is why it is successfully compiled and deployed on Linux, Windows, Mac OS and Android [8]. The PCL framework contains numerous state of the art algorithms for point cloud processing, including filtering, feature estimation, surface reconstruction, registration,model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract key points and compute descriptors to recognize objects based on their geometric appearance, create surfaces from point clouds and visualize them [9].

1.9 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source library which is written in optimized C++ and takes advantage of multi-core processing. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Adopted all around the world, it usage ranges from interactive art to mines inspection, stitching maps on the web or through advanced robotics.[10]

1.10 LaTeX

LaTeX is a system where you can prepare documents for professional usage like scientific articles, books, and presentations, to name a few only. In addition, LateX encourages its users to concentrate on having the right content in their documents instead of focusing on how their documents

look like. LateX has been used for this report as it provides many advantages over using other traditional software like Microsoft Word, for instance, using LateX, handling mathematical equations and expressions is done in a very good way which makes it very comfortable for the user. Moreover,

1.11 Git

It 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 very optimized as it allows to commit changes, create branches, merge codes, and compare 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.

Chapter 2

Registration

2.1 Problem stating

Once multiple frames from the 3D object have been acquired, 3D point clouds are in different frames of reference relative to the scanned object. To transform all these point clouds to the original frame of reference, 3D transformations to each point cloud to compensate for the rotations and translations occurred between several scans are needed [1].

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 datasets 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 modeled by using six degrees of freedom, the latter one considers the problem of softer bodies whose shape can change during data acquisition. [1]

2.2 Iterative Closest Point (ICP)

ICP is an algorithm employed to minimize the difference between two point clouds. This is done by finding a transformation matrix, T, that aligns the source point cloud to the target one with maximum overlap.

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.3 Correspondence estimation

registering large point clouds is considerably more computationally expensive than registering clouds of smaller cardinality. registering only subsets of the original point clouds can yield sufficient results while saving computation time. In principle, two methods of data reduction can be distinguished: automatically extracting a small set of unique and repeatable key points and sampling of the original data with respect to a desired target distribution. Correspondence estimation is the process of pairing points pi from the source point cloud P to their closest neighbors qj in the target cloud Q [3].

2.4 Termination Criterion

1. Maximum number of iterations: Exceeding the number of iterations means that the optimizer diverged. This threshold has to be tuned depending on the complexity of the registration problem; expect that registering a pair of scans that are far away from each other will require more iterations, than two scans that have a good initial alignment.

2. Absolute transformation threshold: The iterations are stopped when the currently estimated transformation is far away from the initial transformation. This is an early termination criteria for registration procedures that diverge. The intuition behind it is that the two clouds to be aligned are expected to be within a certain range of distances from each other, and so transformations that are outside that range need to be rejected.

3. Relative transformation threshold: It speci

   es the minimum transformation difference from one iteration to the next that is considered small enough for the optimizer to have converged.

[3]

Chapter 3

Mesh construction

3.1 introduction

After data acquisition, filters are used to remove anomalies due to the noise around the scanner. After getting rid of inaccurate data points, these points should be normalized then connected to create a surface. This process is referred to as surface reconstruction.

3.2 Filtering

two main steps are required:

- Passing Through filtering automatically removes points outside predefined intervals.
- A set of tiny 3D boxes in space are created over the input point cloud data, then in each voxel, all the points presented will be replaced by their centroid [1].

3.3 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 non-oriented. Norms that do not have a direction are called non-oriented normals, which means that the normal is pointing either to 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 has consistent directions and it is known which point is outside

or inside the surface.[Bergeret al.]. The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation problem. The solution for estimating the surface normal is therefore reduced to the analysis of the eigenvectors and eigenvalues (Principal Component Analysis) of a covariance matrix created from the nearest neighbour of the query point.

3.4 Reconstruction algorithms

There exist many meshing algorithms, in this chapter Greedy and Poisson methods are briefly described.

3.4.1 Greedy Triangulation (GT)

As the name suggests, this method is based on the construction of triangles in a plane where these planar triangles are made on a set of point clouds. A formal definition of the GT is: "The triangulation is obtained by starting with the empty set and at each step adding the shortest compatible edge between two of the points". This means that the algorithm passes through all the points on the point cloud in order to create edges between two points, chosen such that they do not intersect [2].

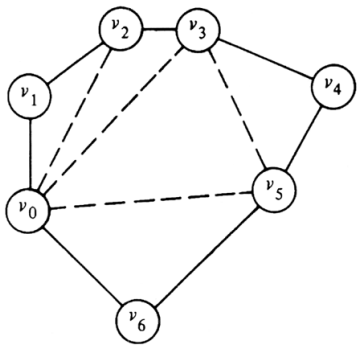


Figure 3.1: Greedy triangles [2]

In figure 3.1, it is shown how the triangles are generated from the GT. By finding edges (Lines -) and vertices (V0 to V6), it creates new edges to form the best triangulation in order to obtain the reconstruction of the surface.

-Advantages and disadvantages:

One of the advantages of GT is that it allows to work with disordered/unorganized data. However, the big disadvantage is the lack of "watertight-ness",a condition in which the mesh has no holes, as

opposed to the watertight meshes produced by the Poisson algorithm.[2]

An approach at computing the GT is to compute all distances, sort them and examine each pair of points in length and compatibility with edges created. The Greedy Triangulation algorithm is simple but not very reliable. With a point cloud of n points, the algorithm looks for the closest point where a compatible edge can be created between the two. A compatible edge is as an edge between two points that does not intersect with any other edge.[3]

3.4.2 Poisson method

This techniques was created by Michael Kazhdan, Matthew Bolitho and Hugues Hoppe in 2006. Given a set of oriented points sampling the boundary, a watertight mesh can be obtained by transforming the oriented point samples into a continuous vector field in 3D, finding a scalar function whose gradients best match the vector field, and extracting the appropriate surface.[2]

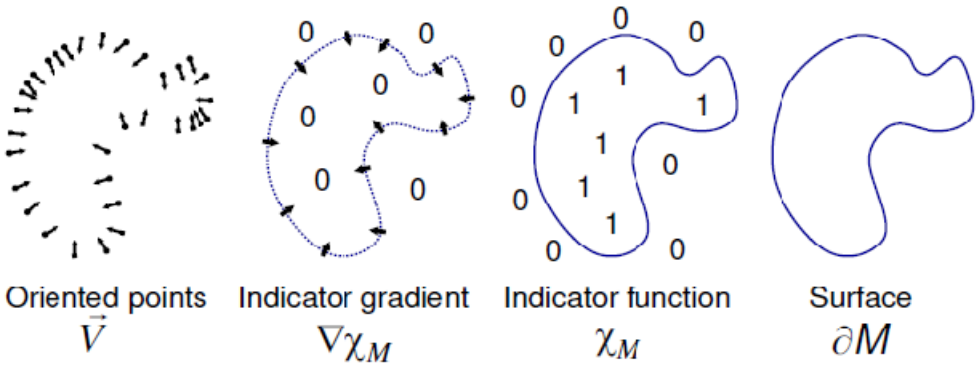


Figure 3.2: Poisson reconstruction process [2]

Figure 3.2 shows the phases of Poisson reconstruction process by which it is possible to construct a watertight surface. However, this algorithm is quite sensitive to noise and to complex forms of points set creating over-smoothed mesh in some areas. A good estimation of the norms is needed to have a well-reconstructed mesh, moreover, a noisy point cloud can orient the norms in a wrong way. The problem with the complex form is linked to the narrow spaces that are often regrouped as one single space without holes by the Poisson equation.

3.5 Greedy and Poisson algorithms comparison

A criteria for the success of the various reconstruction algorithms is that they are fast and reliable. Based on the used reconstruction methods accuracy level, time, and computational complexity, it

can be assessed that the Poisson reconstruction is a good solution, however, even if Triangulation gives good results in time, accuracy and speed, results were not watertight,i.e. they have holes. Therefore, based on the reconstruction method being applied using PCL, it can be concluded that Poisson can provide the most optimal mesh, even though computation time is still a little bit long [1].

Chapter 4

Technical Implementation

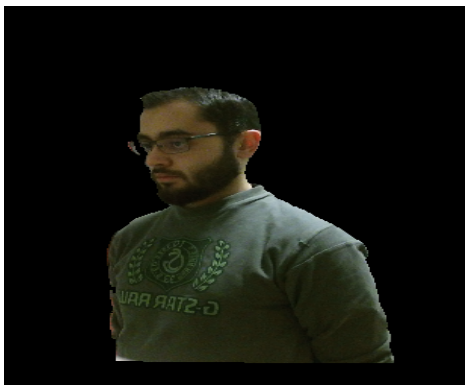
4.1 Kinect Cameras Calibration

Even the most expensive cameras have lenses that brings a level of deviations to the acquired images. Calibration is about mathematically relating camera measurements to the real worlds measurement. The relation between pixels and physical measurements (e.g. meters) is critical for 3D scenes reconstruction in the way we can obtain a model of the camera's geometry and a model of the lens' distortion. These two models define the intrinsic parameters of the camera, correct for lens distortions and interpret the whole geometry of the physical scene. For this project, our goal was to come out with a dedicated module that provides the user with the calibration parameters of a connected Kinect sensor. Unfortunately, this module was not implemented due to the limited time even though enough research on the matter was done. This can be achieved using OpenCV but for the time being we were contended with existing programs that does the same job. For this, we would like to refer to the GML Camera Calibration Toolbox and the Mobile Robot Programming Toolkit which provide very satisfactory results. One thing to add is that the intrinsic parameters of the Kinect (at least the Kinect v2) are added by the manufacturers and Kinect SDK knows about them and is making use of them when needed. Therefore, they are not explicitly used in the data acquisition module but will be needed for the registration module as will be discussed later.

4.2 Data Acquisition (class SE3D::Recorder)

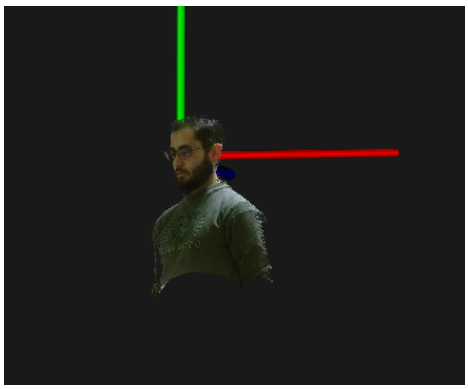
As obvious as it may seem, we need a module for acquiring frames from the Kinect sensor. This has been achieved by using the PCL OpenNI grabber interface as well as the Kinect SDKs libraries since the PCL's does not support the Kinect v2 through its current grabber. As a result, we are able to

save color and depth frames from the RGB camera and the IR camera respectively, see figures 4.1a and 4.1b below. However, the two sensors are at an offset which means they are not capturing the exact same scene. In fact, they even have different spatial resolutions with the RGB camera taking frames at 1920 x 1080 and the IR sensor at 512 x 424. This means the mapping from depth space to color and camera spaces is inherently a non-trivial task. This would imply reducing the resolution of the color frame by down-sampling and then aligning with the depth frame keeping in mind that calibration need to be done first. Luckily, we leveraged some of the multiple useful APIs from the Kinect SDK to handle the transformations for us. The color and depth frames are saved as PNG files at the resolution of 512 x 424. In addition, this module can render and save a colored point cloud of the frame, see figure 4.1c below. Finally, it was made such that it only captures within a range helping by that to remove any background or unwanted regions. This means that only the model, if standing at the correct distance from the sensor, is captured without any extra filtering or work as shown in the below generated files.



(a) colour frame

(b) depth frame (.png)



(c) point cloud (.pcd)

Figure 4.1: 3D scan

4.3 Model Registration (SE3D::Reconstructor)

This module deals with the rendering of a complete 360 model through the iterative alignment of multiple 3D point clouds each representing a different view. The approach was to compute the relative positions among the different views while setting a unique coordinate system for reference to optimize the overlap of the models parts that are common in two or more clouds. We will first describe how we went about reconstructing from two views and then with multiple ones. The pipeline starts with feature detection and description using OpenCVs implementation of the SURF algorithm. The SURF worked well and fast especially that we are dealing with thousands of points from each Kinect frame and hence the correspondence estimation is exponential. From two view images, see figure 4.2 below, the key points are extracted, and the descriptors computed (feature vectors). The descriptor vectors are then matched with FLANN (the brute force approach was tested but it was good enough with FLANN and subsequent filtering).

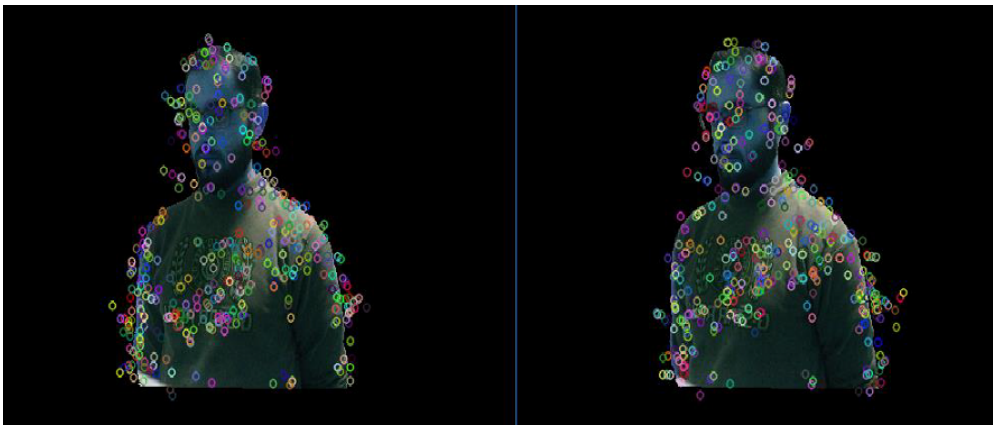
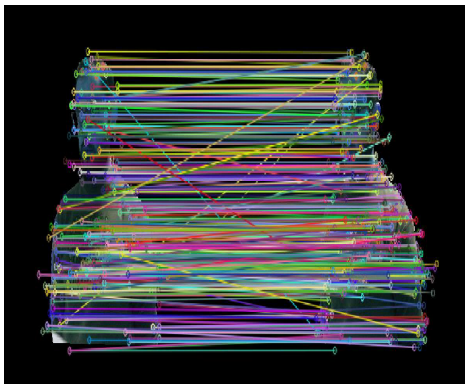


Figure 4.2: key points detected on two different views

The matches have been filtered to only those with good estimation relying on the distances, figure 4.3b. Another good filtering mechanism we came out with is to limit the matches to evolve within the model only and not outside (which obviously has no region of interest). Next is to perform a registration from the 2 views after rendering them as points clouds (reference and sample) along with their corresponding features. Yet, some additional purging of the correspondences is done using the RANSAC algorithm. Then, the transformation between the two features clouds is estimated based on SVD thanks to the PCLs ICP interface. The transformation matrix is then applied to the sample cloud before being fused with the reference cloud leading to the alignment. Now that we could register 2 clouds, see figure 4.4a below, we can incrementally register, in pairs of two, other clouds representing more views, figure 4.4b below. A general transformation is computed to transform all clouds to an initial clouds frame.

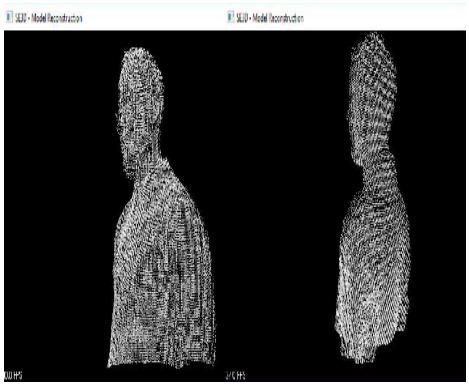


(a) All key points matched (erroneous matches)

(b) The matching after filtering

Figure 4.3: Features matching

(a) 3D registration from 2 clouds



(b) 3D registration from multiple clouds

Figure 4.4: 3D registration

Chapter 5

Project management

Project management is an important part in group team work. After we formed a group, we nominated a project manager. So to better manage our project we used the following management and communication tools:

1. Taiga
2. Skype enterprises
3. Facebook (project group)

5.1 Taiga

Taiga is a project management application that can handle both simple and complex projects for startups, software developers, and other target teams. It tracks the progress of a project. With Taiga, you can use either Kanban or Scrum template, or both. Backlogs are shown as a running list of all features and User Stories added to the project. Taiga integrates video conferencing functions with the use of third party services from Talky.io and Appear.in. Group and private chat is done via HipChat. The on-premise version of Taiga can be downloaded from its github repositories and used for free. This project management system can inteface with web-based version control repositories like GitHub and Bitbucket. Taiga also provides several importers to facilitate migration from other proprietary software platforms such as Trello, Jira, GitHub or Asana. As a promoter of the free software community, Taiga offers free accounts for those projects that are openly developed. All these projects are accessible to your open project explorer `https://tree.taiga.io/discover`

5.2 Skype

Skype for Business (formerly Microsoft Office Communicator and Microsoft Lync) is an instant-messaging client used with Skype for Business Server or with Skype for Business Online (available with Microsoft Office 365).[2] Skype for Business is enterprise software; compared to Skype, it has different features that target businesses. Skype for Business replaces Windows Messenger, which ran with Microsoft Exchange Server. We conducted regular skype conferences to make sure all the members were up to date, if any member needed extra help with a particular topic as figure 5.1 shows.

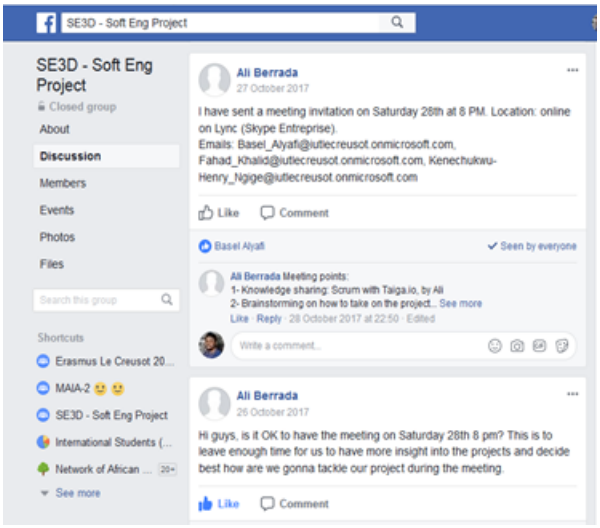


Figure 5.1: Facebook group

5.3 General tasks

- Functionality
- Implementation
- documentation

The task of functionality deals with the strengths and capabilities of various algorithms and/or devices. The task of implementation takes care of revising the existing source codes, libraries and improving available features. The documentation task is related to writing reports (weekly reports and the final report). To carry everyone in the team along, group discussion were organized of which each member of the team shares the knowledge necessary for the progress, success and implementation of the project.

5.4 Project flow

From figure 5.2, it can be stated that the group started the project smoothly and on time most of the work done was done during the 4th,5th weeks.

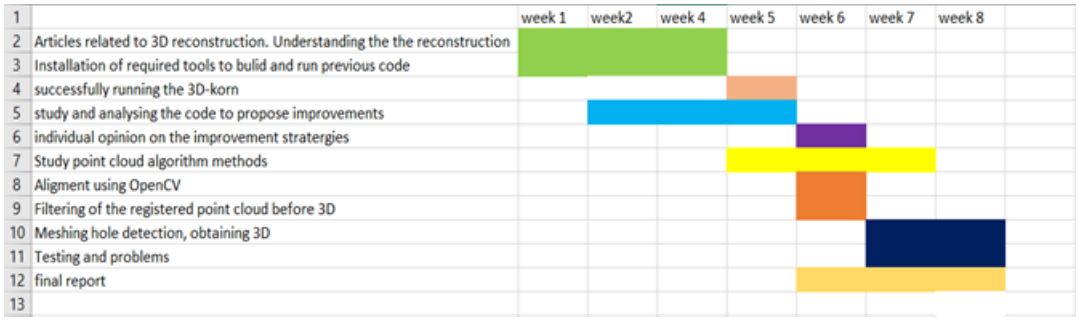


Figure 5.2: project flow

5.5 SE 3D advantages

- A software for 3D scanning that does the job
- Ultimate installation and setup guide
- OpenCV
- PCL
- CMake (even better, used within Qt with MSVC compiler)
- OOP

Bibliography

[1] https://github.com/umaatgithub/3D-KORN.

[2] https://github.com/WajahatAkhtar/Project-S.E.

[3] https://github.com/AnirudhPuligandla/3D_scanner.

[4] https://github.com/tazleef/Software-Engineering-Project.

[5] Jrgen Sturm, Erik Bylow, Fredrik Kahl, and Daniel Cremers, *Scanning and printing persons in 3d* in German Conference on Pattern Recognition, pages 405414. Springer, 2013.

[6] Song Tiangang, Lyu Zhou, Ding Xinyang, andWan Yi *3d surface reconstruction based on kinect sensor*. International Journal of Computer Theory and Engineering, 5(3):567, 2013.

[7] HGonzalez-Jorge, B Riveiro, E Vazquez-Fernandez, JMartnez-Snchez, and P Arias *Metrological evaluation of microsoft kinect and asus xtion sensors.* Measurement, 46(6):18001806, 2013.

[8] Juha Hyvrinen et al, *Surface reconstruction of point clouds captured with Microsoft Kinect* 2012.

[9] Radu Bogdan Rusu and Steve Cousins, *3D is here: Point Cloud Library (PCL)*. In IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 2011.

[10] https://www.microsoft.com/en-us/download/details.aspx?id=44561