Software Engineering Project Report

3D SCANNER

Authors:
Ama Katseena Yawson
Fakrul Islam Tushar
Md. Kamrul Hasan
Lavsen Dahal

Supervisors:
Dr. Yohan Fougerolle
Dr. Cansen Jiang
David Strubel

January 7, 2018

Contents

List of Figures

CHAPTER 1

1 Introduction

1.1 Objectives

- Improvement in accuracy of previous software developed by our predecessor's.
- Creation of an improvement software using OpenCV.

1.2 Specific Objective

The main objective of this project is to design an acquisition and processing software which has ability to perform 3D reconstruction.

1.3 Overview

3D scanning has opened a world of opportunities. 3D scanners peruse the real world objects and acquire its geometrical data which is used to reconstruct 3D model of the object in digital format. These scanning devices have made remarkable impact in various fields such medicine and health care, research and education, visual arts and many more. Due to the fact that this project depends on a sensor, it is important to have a data acquisition module that produces desired output. In this project, the Microsoft Kinect V2 scanner was used for data capture.

1.4 Microsoft Kinect V2

Microsoft Kinect V2 (as shown in Figure 1) is a time of flight sensor primarily used for gaming. This device features an RGB camera (1920 x 1080p), Depth sensors, IR emitters and multi-array microphone with software that provides full-body 3D motion capture, facial recognition and voice recognition capability. Both cameras stream at 30 frames per second. The sensing range of the depth sensor is adjustable between 0.5m and 4.5m.

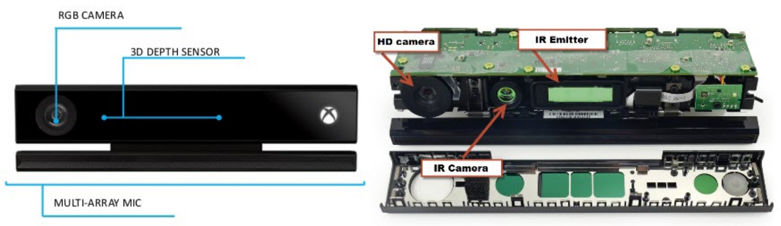


Figure 1: Microsoft Kinect V2

There are 4 main procedures (as shown in Figure 2) to be followed for 3D reconstruction using the Microsoft Kinect V2. These include raw data acquisition, registration, segmentation and object detection.

- Raw data acquisition: 2D depth images from the Kinect;
- Registration: it is the process of transforming different sets of data into one coordinate system. The key idea is to identify corresponding points between the data sets and find a transformation that minimizes the distance (alignment error) between corresponding points;
- Segmentation: it is the process of partitioning an image into segments. It is an important procedure in 3D reconstruction because captured image contain too much information and hence the region of interest must be identified to make analyses easier. It is challenging procedure because of high redundancy, uneven sampling density, and lack explicit structure of point cloud data;
- Object detection: 3D visualization of input data.

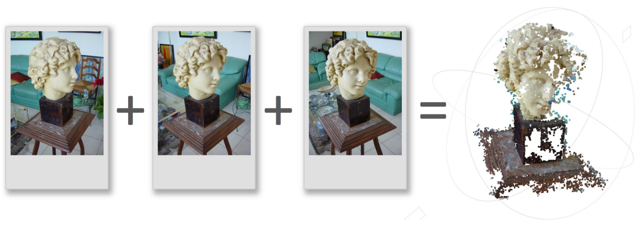


Figure 2: Kinetic Procedure using OpenCV

1.5 Software tools

The softwares used in the implementation of this project are as follows:

- Microsoft Visual Studio
  The Visual Studio interactive development environment (IDE) is a creative launching pad from Microsoft which is used to develop computer programs, as well as websites, web apps, web services and mobile apps.[1] Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. Its integrated debugger works both as a source-level debugger and a machine-level debugger. Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.[2]
- QT Compiled with MSCV 2015
  Qt is a cross-platform application framework 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 codebase.[3]

- 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 makefiles and workspaces that can be used in the compiler environment of ones choice.[4]

- VTK 7.0.0
  The Visualization Toolkit (VTK) is an open-source for 3D computer graphics, image processing, and visualization. It is made up of a C++ class library and supports a wide variety of visualization algorithms such as scalar, vector, tensor, texture, and volumetric methods. It can also support advanced modelling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. The toolkit supports parallel processing and integrates with various databases on GUI toolkits such as Qt and Tk. VTK is part of Kitwares collection of commercially supported open-source platforms for software development. [5]

- 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.[6]

- Intel RealSense SDK
  The Intel RealSense 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.

- PCL 1.8.0: The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing. The pcl framework contains many algorithms for point cloud data, filtering, feature estimation, segmentation, surface reconstruction and model fitting.

- 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.[7] 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.

- Git: It's a distributed version control system that allows group of people to work on the same documents (often code) at the same time.

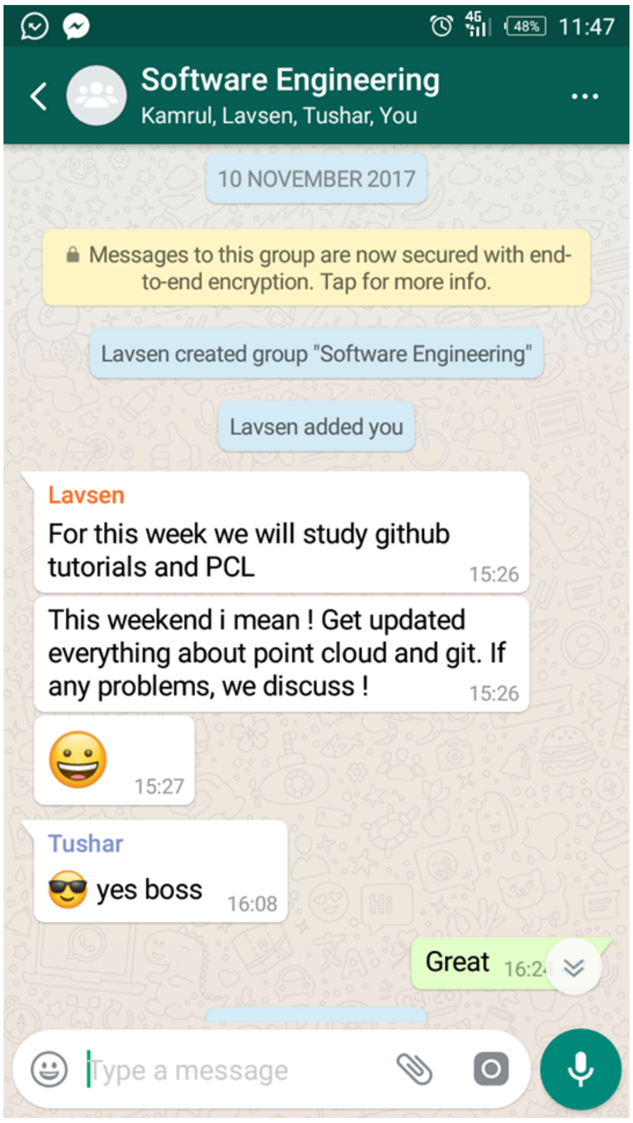
- LaTeX: It is a document preparation system for high-quality typesetting which gives a good appearance. It is most often used for medium-to-large technical or scientific documents as well as any form of publishing.

CHAPTER 2

2 Project Management

A successful project requires a detailed and well-planned project management plan. On our first meeting, we created various communication platforms such as;

- Whatsapp Group: This communication platform (Figure 3) was mainly used for setting up meeting times, to keep track of the contribution and evaluating progress on research of every member.



Messages to this group are now secured with end-to-end encryption. Tap for more info.

Lavsen created group "Software Engineering"

Lavsen added you

Lavsen
For this week we will study github tutorials and PCL 15:26

This weekend i mean ! Get updated everything about point cloud and git. If any problems, we discuss ! 15:26

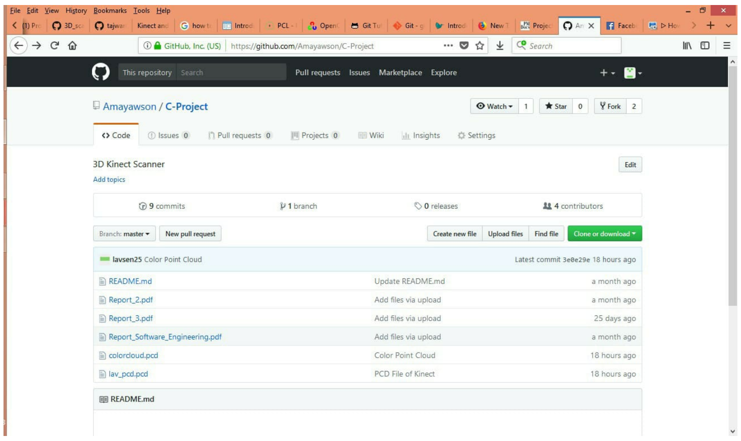
😀 15:27

Tushar
😎 yes boss 16:08

Great 16:2

Figure 3: Whatsapp group

- Github Repository: This platform was used for version control of documents and code to enhance collaborative programming amongst our group(Figure 4).



File Edit View History Bookmarks Tools Help
GitHub, Inc. (US) https://github.com/Amayawson/C-Project
This repository Search Pull requests Issues Marketplace Explore
Watch 1 Star 0 Fork 2
Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings
Edit
Add topics
Branch: master New pull request
Create new file Upload files Find file Clone or download

Amayawson / C-Project

3D Kinect Scanner

9 commits | 1 branch | 0 releases | 4 contributors

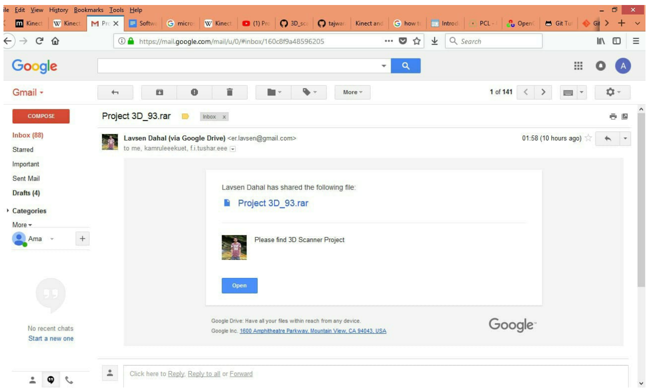
lavsen25 Color Point Cloud — Latest commit 3e0e29e 18 hours ago

README.md	Update README.md	a month ago
Report_2.pdf	Add files via upload	a month ago
Report_3.pdf	Add files via upload	25 days ago
Report_Software_Engineering.pdf	Add files via upload	a month ago
colorcloud.pcd	Color Point Cloud	18 hours ago
lav_pcd.pcd	PCD File of Kinect	18 hours ago

README.md

Figure 4: Github Repository

- Google Doc: Google Docs is a web-based document management application for creating and editing both private and public, word processing and spreadsheet documents as shown in Figure 5. These documents can be stored both online on the Google cloud and/or on the user's computer. This platform was used to track our individual research on the project.



Project 3D_93.rar Inbox

Lavsen Dahal (via Google Drive) <er.lavsen@gmail.com> 01:58 (10 hours ago)
to me, kamruleeekuet, f.i.tushar.eee

Lavsen Dahal has shared the following file:
Project 3D_93.rar

Please find 3D Scanner Project

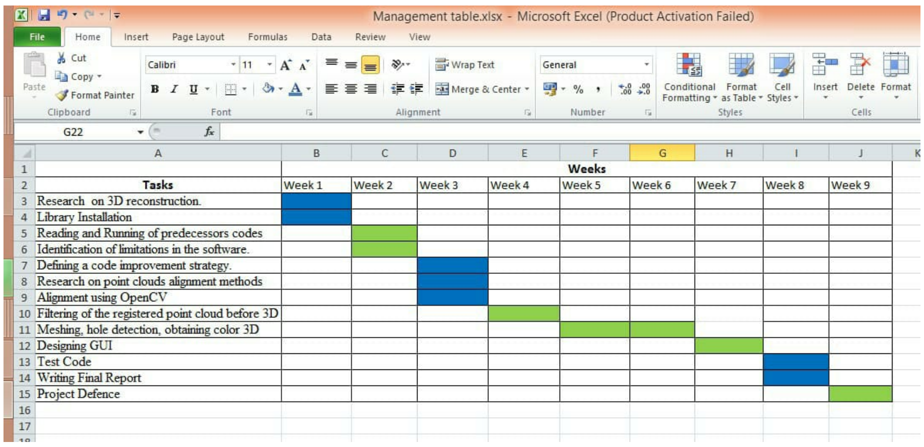
Google Drive: Have all your files within reach from any device.
Google Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

Click here to Reply, Reply to all or Forward

Figure 5: Google Doc

2.1 Management Structure

The whole project works was divided on weekly basis as shown in Figure 6. For tracking our whole project management, Gantt chart was used. Gantt chart is a type of bar chart that demonstrates a project schedule with starting and finishing dates of the terminal elements as well as it summarizes the elements of a project.



	Weeks								
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Research on 3D reconstruction.	🟦								
Library Installation	🟦								
Reading and Running of predecessors codes		🟩							
Identification of limitations in the software.		🟩							
Defining a code improvement strategy.			🟦						
Research on point clouds alignment methods			🟦						
Alignment using OpenCV			🟦						
Filtering of the registered point cloud before 3D				🟩					
Meshing, hole detection, obtaining color 3D					🟩	🟩			
Designing GUI							🟩		
Test Code								🟦	
Writing Final Report								🟦	
Project Defence									🟩

Figure 6: Gantt chart

CHAPTER 3

3 THEORY AND METHODOLOGY

3.1 Overview

3D scanning is the visualization of real-time object in a 3D digitalized format. This visualization requires a combinational computation and analysis of different methods. To achieve the desired project goal, some methods have been used in this project to perform this various operations i.e. computing depth image, detection of keypoints, extraction of descriptors, feature detection and matching, and finally computing and visualization of the point clouds. Overview of the operations performed in the project shown in Figure 7.

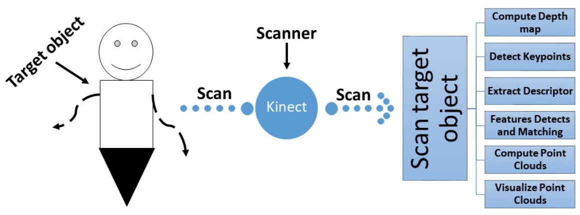


Figure 7: Overview of the operations performed in the project

The workflow used to the acquire desired outcome are stated in steps below:

- Scanning of the target object using Kinect V2.
- Detection of the keypoints using SURF detector.
- Feature Detection and Matching using FLANN based Matcher.
- Computation and visualization of point clouds.

Speeded up Robust Features (SURF) is a most suitable local feature detector and descriptor (Figure 8). It can be implemented for object recognition, image registration, classification, 3D reconstruction or and to extract points of interest and this method was derived from the Scale-invariant Feature Transform (SIFT) descriptor.[11] One key advantage of using SURF was its fast response than SIFT. SURF approximates SIFT schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. This special features are achieved by relying on integral images for image convolutions that was based on detectors and descriptors. Specifically, Hessian matrix-based measurement(for the detector) and a distribution-based descriptor are commonly used. To detect interest points, SURF was used for integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using a pre computed integral image.The key steps that followed by the SURF are mentioned below:

- Integral Image
  Integral Image $I_{\sum}$ (x) at any point (x,y) is the sum of all the pixels above and to the left of (x,y). Mathematically,it is written as;

  $$I_{\sum}(x) = \sum_{i=0}^{\imath \leq x} \sum_{j=0}^{\jmath \leq x} I(x, y)$$

- Interest Points based Hessian Matrix, H(x,σ)
  In order to have good performance with accuracy, Hessian matrix was used selected. For the given point (x,y) of an image I, the Hessian Matrix, H(x,σ) where σ is the scale factor in x is calculated by using the following matrix;

  $$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}$$

  where L_{xx}(x,σ), L_{xy}(x,σ) and L_{yy}(x,σ) are the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial t^2}$ g(σ) with the image I. The eigenvalues of H(x,σ) are proportional to the principal curvatures of D. The ratio of the two eigenvalues is given by;

  $$r = \frac{a}{b}$$

  .

  with a greater than b. The trace of H(x,σ) is:

  T(r)=L_{xx}(x,σ)+L_{xx}(x,σ)

  This gives the sum of the two eigenvalues. On the other hand, the determinant is given by:

  D(r)=L_{xx}(x,σ)*L_{yy}(x,σ)-L_{xy}(x,σ)*L_{xy}(x,σ)

  The ratio R can be shown;
  R =$T(r)^2$/D(r)
  R =$(r+1)^2$/r

This R depends on the ratio of the eigenvalues rather than their individual values. R is minimum when the eigenvalues are equal to each other. Therefore, the higher the absolute difference between the two eigenvalues, which is equivalent to a higher absolute difference between the two principal curvatures of L, the higher the value of R. It follows that, for some threshold eigenvalue ratio r_{th}, if R for a candidate keypoint is larger than $(r_{th}+1)^2/r_{th}$, that keypoint is poorly localized and hence it is rejected.

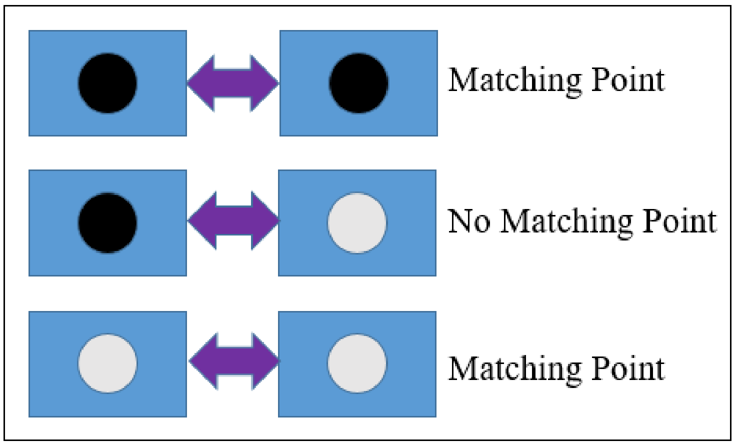


Figure 8: SURF procedure

In short, SURF adds a lot of features to improve the speed in every step. Analysis shows it is 3 times faster than SIFT while performance is comparable to SIFT. SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.[12]

The acronym for FLANN is Fast Library for Approximate Nearest Neighbours which contains a collection of algorithms optimized for fast nearest neighbour search in large datasets and for high dimensional features. One of the vital feature is that it works faster than Brute-Force Matcher (BFM) for large datasets. For FLANN based matcher, it is required to pass two dictionaries which specifies the algorithm to be used and it's related parameters. IndexParams is the first parameter and SearchParams is the second parameter. For various algorithms, the information to be passed is explained in FLANN docs.[13] Nearest neighbor (NN) graph methods build a graph structure in which points are vertices and edges connect each point to its nearest neighbours where the query points are used to explore this graph using various strategies in order to get closer to their nearest neighbours. [14] The user selects a few well separated elements in the graph as seeds and start the graph exploration from those seeds in a best-first fashion. They present recent experiments that compare favourably to randomized KD-trees and hence the proposed algorithm should be considered for future evaluation and possible incorporation with FLANN. The nearest neighbor graph methods suffer from a quite expensive construction of the k-NN graph structure. The scalable kNN graph construction for visual descriptors proposes improvement of the construction cost by building an approximate nearest neighbour graph.

CHAPTER 4

4 FUNCTIONS

4.1 Overview

The project depends on the acquisition and processing of the data from the kinetic sensor.For performing different operations in processing and data acquisition, different functions were implemented. These functions and their functionalities are discussed below.

4.2 Non-Static Function

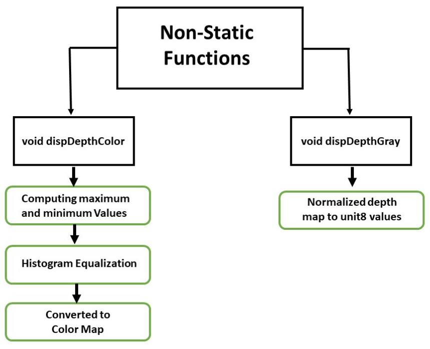


Figure 9: Non-static Function

- void dispDepthColor: This function was used to get the coloured depth map image. Depth map is a kind of image that consist of information about difference in distance of corresponding image points and their camera centers[8]. First this function compares the pixel intensity of the images and then the depth of a given pixel region is computed and stored in an array.

  cv::minMaxIdx function: This function was used to find the minimum and maximum element values and their positions. Afterwards, histogram equalization was performed and the depth map was converted to unsigned characters. The computation of the function ended by converting this depths into colour for maximum clarity. The closer the object to the camera, the higher the colour intensity and vice versa.

- void dispDepthGray: This function was used to normalize the depth map into a depth gray image composed of gray pixel from 0 to 255 intensity level. Depth gray map was converted to an 8-bit unsigned integer image with 3 channels using the functiondepthGray.convertTo().

4.3 OpenCV Functions

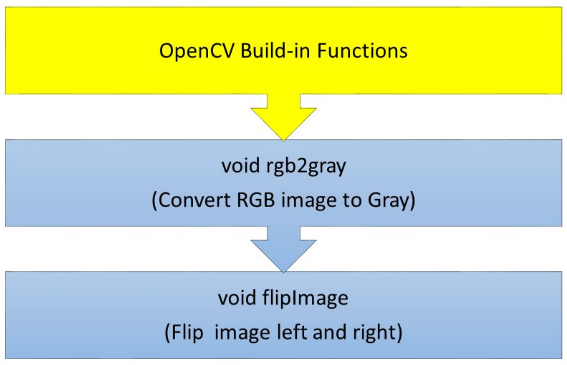

OpenCV Build-in Functions
void rgb2gray
(Convert RGB image to Gray)
void flipImage
(Flip image left and right)

Figure 10: OpenCV Building Function

- void rgb2gray(): This function was used to convert RGB images to gray images using OpenCVs built-in function cv::cvtColor().
- void flipImage(): This function flips the image from left to right. It was performed by using OpenCVs default function cv::flip().
- void featureExtraction():This function search for regions in the images that have maximum variation when moved by a small distance in all regions around it. This process is called feature detection.[9] The function was used for detecting the key points. SURF (Speeded-Up Robust Features) detector was used for detecting the key points and dictionaries were used to specify the algorithm to be used. Laplacian of Gaussian (SURF) is a speeded-up keypoint detection and dictionary algorithm, which approximate Laplacian of Gaussian (LoG) with Box filter.[11]SURF uses the determinant of Hessian matrix for both scale and location.
- void showKeyPoints(): After the detection of keypoints using featureExtraction() function, showKeyPoints() function was used to display keypoints. This function shows the keypoints on the RGB image by drawn the defected keypoints on the RGB image using a built-in function cv::drawKeypoints OpenCV.
- void commonFunc::rgbd2pointcloud (): This function convert the 2D- images to point clouds.
- void commonFunc::visualizepcl(): This function was used to visualize the point cloud in 3D using pcl library.

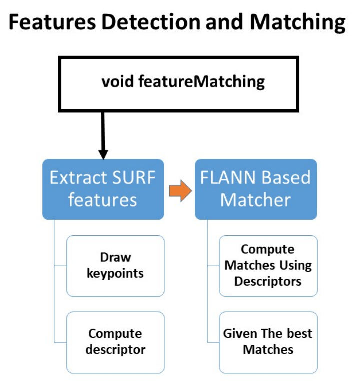


Figure 11: Feature Detection and Matching procedure

4.4 Image Matching

- void featureMatching(): Image feature detection and matching was done by this function. This function has many functionalities. Within this function, two algorithms SURF and FLANN have been implemented. SURF detector was used for extracting keypoints and computing descriptors. By putting Hassian threshold, key points of the two images were detected and drawn on the RGB image. Descriptors were computed using descriptor extractor given as SURF::create(). Most of OpenCV descriptors inherit DescriptorExtractor abstract interface. Afterward descriptors were calculated for each of the keypoint. The output Mat of the DescriptorExtractor::compute. FLANN stands for fast library for approximate nearest neighbors. It has optimized algorithms to compute fast search for nearest neighbour in large database related to high dimensional features.[10] Here FLANN based matcher was used for feature matching. Minimum and Maximum distance between the keypoints were calculated and considered. Only good matches with distance less than twice the minimum distance or small arbitrary value (0.02) were drawn and good matched number were given.

- void commonFunc::myDepth2meter: This function was used to convert uint8 depth to meter. Invalid points were rejected if the disparities were less than or equal to zero. Horizontal and vertical focal length ,the principal optical points, and scale factor were parameterized.

- void myDepth2meter: This function was used to convert the depth point to a 3D point and performed the calculation using the sensors defined parameters.

CHAPTER 5

5 GRAPHICAL USER INTERFACE(GUI)

5.1 Overview

This chapter is about the Graphical User Interface developed and its functionality. The GUI has two separated windows namely Main Window and Dialog Window which is responsible for computation of desired output.

5.1.1 Main Window

The main window is shown in Figure 10. This is the window from which the user connect the Kinect to acquire data by clicking on the Run Kinect as shown in Figure 11.

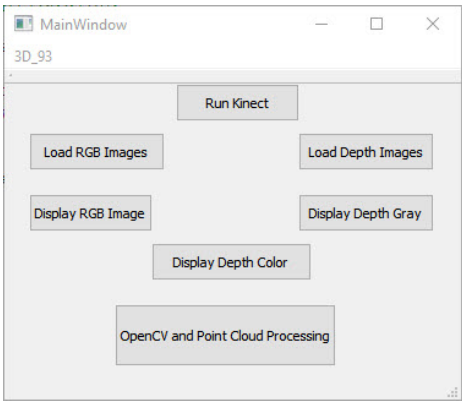


Figure 12: Preview of the Graphical User Interface of Main Window

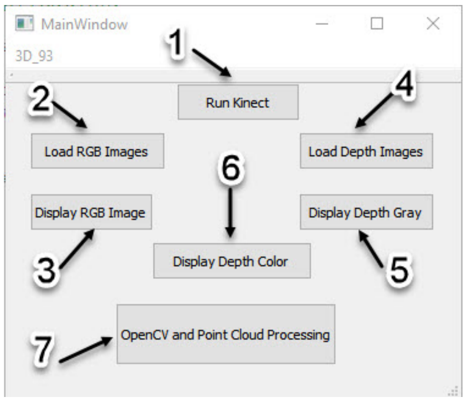


Figure 13: Shown the different Functionality of the Main Window

After connecting, the Kinect user loads the RGB image using section 2 and section 3 is used to view RGB image. Fig 11 is a demonstration of this operation.

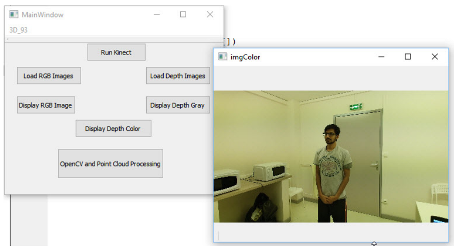
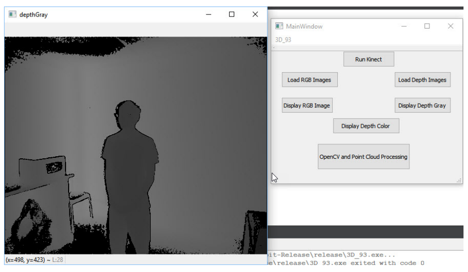


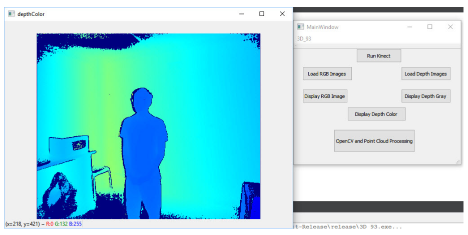
Figure 14: Preview of RGB image using the Main Window

User can load the depth image using the section 4 and depth Gray and depth image can be displayed using the section 5 and 6 respectively shown in Figure 11. This operation is shown in Figure 13 and 14.



```
it-Release\release\3D_93.exe...
e\release\3D 93.exe exited with code 0
```

Figure 15: Preview of Depth Image using the Main Window



```
t-Release\release\3D 93.exe...
```

Figure 16: Preview of Depth Color using the Main Window

To perform OpenCV and Point cloud processing, section 7 was used as shown in Figure 11 and this section pops up the second window which is the Dialog Window.

5.2 Dialog Window

Different OpenCV and Point Cloud functionality was performed using this Dialog Window Shown in Figure 14.

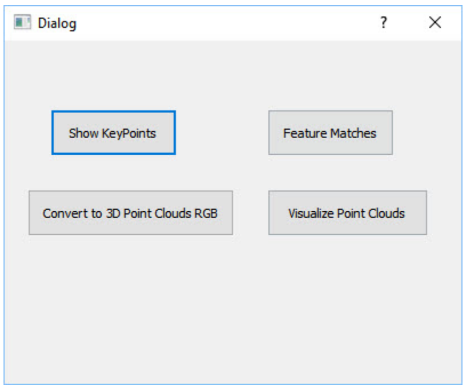


Figure 17: Preview of the Dialog Window

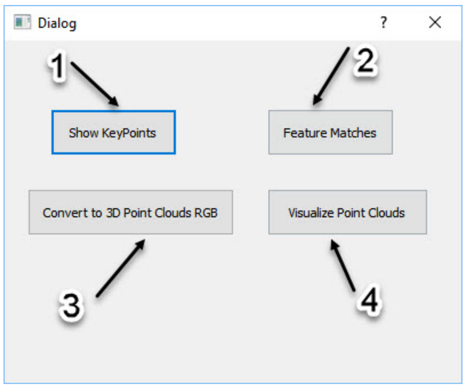


Figure 18: Different Functionalities of the Dialog Window

Section 1 shows the execution of keypoints as shown in Figure 16.

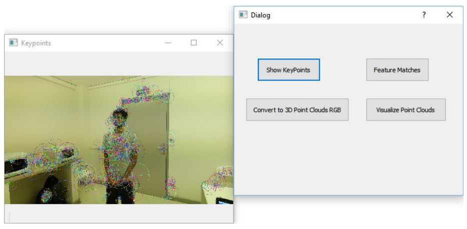


Figure 19: Execution of Keypoint

After this procedure features matched and converted to the 3D point clouds in RGB using the sections 2 and 3 respectively as shown in Figure 16.

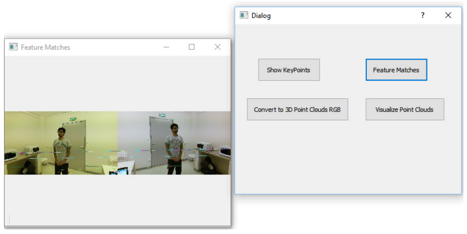


Figure 20: Preview of the Feature matching.

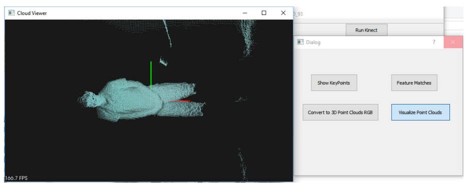


Figure 21: Visualization of the point clouds

CHAPTER 6

6 CONCLUSION

6.1 Improvements

The project has following enhancement scope:

- Mesh Generation.
- Feature matching for RGB stream and Depth gray images.
- An added functionality in Point Cloud Visualization

6.2 Discussion

The previous years' projects required many libraries and dependencies. So lot of time was required to make it run in our personal computers. Moreover we spent lot of time understanding the algorithms and implementing the codes.

6.3 Conclusion

This designed software can successfully scan an object using the Microsoft Kinect V2 sensor and then 3D point cloud are obtained. In order to obtain an optimal results, many research were made. In conclusion, the specific objective of this project was achieved and point cloud was visualized after feature matching using the pcl and OpenCV libraries.

References

[1] https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide

[2] Brenner, Pat (19 July 2013). ”C99 library support in Visual Studio 2013”. Visual C++ Team Blog. Microsoft. Retrieved 3 August 2014.

[3] Qt Project (14 August 2011)”Qt Wiki Support for OS X”.

[4] https://cmake.org/

[5] https://www.vtk.org/

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

[7] https://opencv.org/

[8] https://docs.opencv.org/3.1.0/dd/d53/tutorial_py_depthmap.html

[9] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#surf

[10] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

[11] H. Bay, A. Ess, T. Tuytelaars, and L. V. Goolab, Speeded-Up Robust Features (SURF), Computer Vision and Image Understanding, Vol. 110, Issue. 3, pp. 346-359, June 2008.

[12] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[13] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html#matcher

[14] M. Muja and D. G. Lowe, ”Scalable Nearest Neighbor Algorithms for High Dimensional Data,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 11, pp. 2227-2240, Nov. 1 2014.

[15] T. B. Sebastian and B. B. Kimia, Metric-based shape retrieval in large databases, in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2002, vol. 3, pp. 291296.