UNIVERSITE DE BOURGOGNE

Software engineering

December 26, 2017

Reports

Team Members:

BATERIWALA, Malav

OCHOA, Eduardo

VAISHNAV, Mohit

VALENCIA, Liliana

Project Supervisors:

FOUGEROLLE Yohan

CANSEN Jiang

STRUBEL David

1 Report week 1

1.1 Setting up of the system

To be able to run the projects in which we must work we should install the next items:

- QT Creator 4.0.3 with MSVC 2015
- PCL all in one 1.8 installer
- VTK 7.0.0
- Cmake
- VisualStudioCPPTools

This has been one of the most difficult task and the time taking one too. We have followed various instructions set and hit and trial method to make or PCL program run. Though we are not taking into consideration all those into this report but the simplified version of the output. Two most helpful of the approaches were obviously the one contained in the project folder of the Group 3-D scanner project by group 3 and the installation guide upload to Edmodo for our schoolmate Roger Pi. One trick used while installing the software is to copy the PCL folder after setting up of VKT into the system as a way to skip this part of installing the Visual Studio and build the VTK for using it further.

1.2 Division of work

In order to have a better understanding about the work we have to do, we split the stages followed to build the previous project in the next order:

- Filtering: Malav
- Registering: Mohit
- Denoising and smoothing: Liliana
- Meshing: Eduardo

1.3 Setting up of tools for work

As part of the project and also as a better way to work, during one of our meeting we created a github repository https://github.com/bigmb/HakunaMatata.git and a slack's group. So far, we are woring mainly through slack to set up our meeting and to create lists about what we need to do.

1.4 Tasks for the week - November 27 to December 3

For the next week, we need to fix the problems that we still have with the laptops of some of us. As we want to implement OpenCV to do matching feature in the registration section, we will start with the installing of the software and the understanding to implement this in the code.

2 Report week 2

2.1 OpenCV integrated with PCL

During this week we were trying to explore the avenues for integrating the aspects of OpenCV with the earlier project which was purely based on PCL. OpenCV has many efficiently working algorithms which could be integrated into the project to have better performance as well as the results. Some of them are feature matching, registration with the help of OpenCV, etc. We have created a bibliography of various other related projects using the OpenCV platform which we shall include in the final report for the future reference of others.

2.2 Feature matching

During this week we were focused in ICP – Iterative Cloud Point. Specially in feature matching.

This part is very important because a good alignment depends on it to the result of the final model. Also, a high run time expended in this stage, determining the correspondence between the points, can be saved if we find a good way to determine it.

Last year groups have some issues with the alignment of the point cloud. As we said before one important part here is determine the correspondence (Feature matching), Next, we can see the methods that each group use for it:

- Group 1. Correspondence rejection algorithm + SVD. They first list the correspondence points between a new downsampled Point Cloud and the previous registered downsampled Point Cloud. Then they applied the correspondence rejection algorithm to eliminate the outliers, and after they apply SVD.

- Group 2. Iteration and Maximum Correspondence distance between points of the Point Cloud. They registered the Point Cloud after Initial alignment and then they perform Concatenation which allows to join the point clouds, it is done in a loop, so they use global Registration. They use light Statistical outlier removal filtering in between concatenation of point clouds to remove outliers.

- Group 3. RANSAC - Random sample consensus – and K-d trees. With RANSAC method, they estimated the transformation of subset of the given subsets of correspondences based on the Euclidian distance between the points after the computed transformation is applied to the source Point Cloud. They found that this method is effective keeping ICP algorithm from converging local minima and it gives good initial parameters for the transform estimation in ICP. With this they also performed the filtering of the outliers. K-d trees was used whenever they wanted to find nearest neighbor.

Through the research, we found that the nearest neighbor algorithm could be good to determine correspondence and this is implement in a K-D tree structure. It is very good improving run time efficiency and giving a better starting point for the ICP which would help in the alignment of the Point Clouds (Feature Matching). Group 3 applied this method with PCL library and as it is stated before every time that they wanted to find nearest neighbors.

But there is an issue using just k – d tree structure because this works well for exact nearest neighbor search in lowdimensional data, but quickly loses its effectiveness as dimensionality increases.

So that improved structures of k-d trees can be found as FLANN (Fast Approximate Nearest Neighbor Search Library)

2.3 ICP Registration algorithm

As we have stated one of the main ideas in our project is to implement OpenCv together with PCL to do the registration part. With this and the ideas of the last group project we need to have a clearly idea of the ICP process and with this find the functions of OpenCv and PCL that can make this algorithm functional.

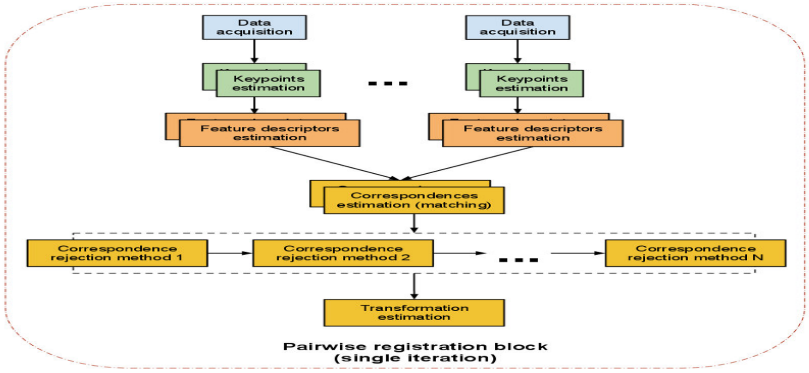

Figure 1: ICP algorithm workflow

In Figure 1 we show the main idea or the work flow for the Iterative Closest Point algorithm consist on the following steps:

- **Data acquisition.** First we gave to the algorithm an input and a reference cloud. For this we have to figure out how to read the data with OpenCv and what kind of information is useful to us.
- **Estimation of Keypoints.** A keypoint of a cloud is a point that: has a clear definition, has a well defined position in space and and the structure around the point is rich in terms of local information. This points characterizes the point clouds and make different parts distinguishable.
  This will speed up the whole ICP process and improve accuracy.
- **Using descriptors.** For every key-point we need to compute the descriptors that are a compact representation of a point's local neighborhood. The descriptors try to resemble the shape

and appearance of the local neighborhood around a point and are useful in terms of matching.

- **Correspondence estimation.** This step consist on finding the correspondences between key-points. In this step the most useful thing is to take advantage of the local descriptors and match each one of them to his corresponding counterpart in the target point cloud.
- **Correspondence rejection.** Normally the two point clouds don't have the same amount of information, resulting in getting more features in one point cloud than in another. For this we have to run an algorithm to get the matches that really corresponds one to another. One of the most common approches is to use RANSAC.
- **Transformation estimation.** After making a robust correspondences between the clouds we match the two point clouds and with this we can start to do ICP iteratively to find the best match of the point clouds.

One of our main task would be to analyze the example code of OpenCv to find the functions that we can use to improve the registration process. And with this we can take some ideas of the previous codes to obtained then the 3D model.

2.4 Tasks for the week - December 4 to December 10

As we received the kinect, we will start to acquire and load images using OpenCV and try to start the registration process.

3 Report week 3

During this week we have been working in the data acquisition from the kinect. We have tested the Kinect and the SDK application to see if the Kinect was working in our computers and we managed to see the Depth map and RGB data that we are going to use in our project. But when it came to the implementation between QT, OpenCv and SDK we have had some troubles. Despite of all our efforts, so far it has not been possible because we have not found

the right connection between Qt, OpenCV and the grabber of SDK. From the research we have been doing to solve the problem, we found some codes that links these three tools through openNI 1. But one of the disadvantages is that the projects from last year were using openNI 2 so, our objective is work with the same version because doing a downgrade to work with openNI 1 might cause problems with the functions that are already implemented in the previous projects.

Right now our main objective is to make some data acquisition to start doing the pre-processing of our data and do the link within OpenCV and PCL. The pipeline that we are going to follow then will be: a pre-processing of the RGB data and depth map given from Kinect, then extracting some main features of every frame and do the matching within every Point cloud. We know for sure that this will make a better implementation in the ICP algorithm.

4 Report week 4 and 5

During these week we could acquire images from Kinect in Qt using OpenCV. To do this we are using different Microsoft Kinect API [**KinectApi**]. These methods are included in a public class that were developed by Microsoft to be used with the kinect sensor. The most relevant are:

- ColorFrameSource
- DepthFrameSource
- CoordinateMapper
- IsAvailable
- IsOpen

The basic idea of the code is to set manually the three main parameters: colour frame, depth frame and the coordinate system and then create a dynamic array to display the information acquired. It is important to mention that in order to get the information from the kinect, first we have to link it with the program and check if it is available. We make this possible using *IsOpen* and *IsAvailable*

properties. Then, we create a dynamic array that let us store one frame coming at a specific time and display it. So far, we are able to get three images:

(a) RGB Frame

(b) Depth Frame

(c) Combined depth and color image

Figure 2: Acquired Kinect Frames

For the following weeks we will work on saving and transform the images in point clouds to use them in the previous projects.