Software Engineering Project
Report

3D Scanning and Reconstruction

Week (3) Report 04/12 - 10/12

Project
Sculptura

Submitted by

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

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

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

Contents

List of Figures

Introduction

Group Meetings Summary

Meeting Discussions and Work:
1) Meeting with David on UML and Qt Designer
2) Meeting for Kinect data grabbing

Objectives

For this week our goal was to grab data from Kinect using OpenNI. However, as the results were not satisfactory, we tried to do this with different libraries afterwards. Also, we wanted to do theoretical research on feature extraction and feature matching and write part of code for feature operations based on example provided by Cansen tutor.

Work Done

Reveiws Summary

Feature matching and feature tracking (for motion estimation)

One of the most important steps in 3D model reconstruction is alignment of point clouds. This task requires information about object or camera motion. Global alignment is sensitive to lighting, occlusion, noise and so on. That's why to track this motion we would like to have set of distinctive points that we can easily find in different frames and from them calculate the rotation and translation. In the ideal case, we would like these points to be invariant to image scaling and rotation, invariant in change of illumination and camera viewpoint and, of course, be matched with high probability [1].

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

Camera Motion Estimation and Tracking

- Given: two camera images f_0, f_1

Figure 2.1: Consecutive frames example

- Wanted: estimate the camera motion u

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

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

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

Below are the main steps for SIFT feature extraction:

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

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

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

- Create histogram of surviving edge orientations (8 bins)

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

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

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

Iterative Closest Point (ICP)

- Given: Two corresponding point sets (clouds)

$$P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$$
$$Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$$

- Wanted: Translation and rotation that minimize the sum of the squared error

$$E(R, \mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{p}_i - R\mathbf{q}_i - \mathbf{t}||^2$$

The goal is to find the most correct correspondences from two sets to match together, we start by

- Selecting and weighting source points

- Finding corresponding points
- Rejecting certain (outlier) correspondences
- Minimization

We will use the essential matrix which we obtained from the prepossessing of the input image frames to match the point cloud for better correspondence.

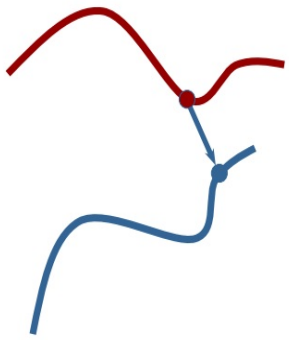

Figure 2.2: Point-to-Point ICP

OpenCV overview

OpenCV is an open source library for developing computer vision applications. It provides an image processing module that includes main functionality such as image filtering and enhancement, affine transformation, color space conversion and so on. For our project an essential advantage of the library is module called features2d. Using this module, we can detect features on set of images, compute descriptors, provide best feature matching.

All the OpenCV classes and functions are placed into the cv namespace. If we want to declare a variable that will hold an image, we can define it as follows:

```
cv::Mat image (240,320,CV_8U) //we declare a size of image 240x320, CV_8U stands for 8-bit unsigned integer matrix with 3 channels
```

When the `cv::Mat` object goes out of scope, the memory allocated is automatically released. This is very convenient because thus we avoid problems with memory leaks.

In order to retrieve data from Kinect (color image and depth values) we will use additional OpenNI library. Depth map and rgb image can be retrieved by using class `VideoCapture`. VideoCapture can grab data from depth generator:

```
CV_CAP_OPENNI_DEPTH_MAP // depth values in mm (CV_16UC1)
CV_CAP_OPENNI_POINT_CLOUD_MAP - XYZ in meters (CV_32FC3)
```

From BGR image generator:

```
CV_CAP_OPENNI_BGR_IMAGE - color image (CV_8UC3)
CV_CAP_OPENNI_GRAY_IMAGE - gray image (CV_8UC1)
```

For getting several data maps `VideoCapture::grab` and `VideoCapture::retrieve` are used.

For future 3D reconstruction we will need data in a format of point clouds. It is possible to make a conversion from `cv::Mat to pcl::PointCloud` if we know translation and rotation.

Results

Results for this week:

- First trials of grabbing data using OpenNI library → it seems that Kinect camera cannot be detected (though applications from SDK Browser were working). And the laptop camera was detected with OpenNI code. Trials with other library (SDL) → managed to acquire RGB data, but haven't succeeded in depth map grabbing.
- Trials of feature detection and feature matching provided nice results;
- Daria has written review on OpenCV library;
- Elizaveta has written review on feature extraction and feature matching for motion estimation;
- Mohamed has tested code for feature extraction and feature matching and has written review on ICP method;
- Karim was working on Kinect connection and data grabbing.

To do list

Plans for the next week:

- We will continue trials of Kinect data acquisition and try to scan a rigid object and test feature extraction;
- We will continue our work on GUI;
- We'll finish designing class for 2D image processing (for feature extraction, matching etc.)

References

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

[2] *Visual Navigation for Flying Robots* Lecture Notes, Technische Universität München (TUM) `https://vision.in.tum.de/_media/spezial/bib/visnav2013lecturenotes.pdf`