西 南 交 通 大 学

本科毕业设计（论文）

Monocular Visual Odometry on the GPU

年　　级：	2012 级
学　　号：	20122779
姓　　名：	侯佳维
专　　业：	自动化
指导教师：	汪晓宁
	Sören Schwertfeger

二零一六年六月

院系	信息科学与技术学院	专　业	自动化
年级	2012 级	姓　名	侯佳维

题目 Monocular Visual Odometry on the GPU

指导教师

评　　语 ______________________________

指导教师 ____________(签章)

评 阅 人

评　　语 ______________________________

评 阅 人 ____________(签章)

成　　绩 ____________

答辩委员会主任 ____________(签章)

年　　月　　日

毕业设计（论文）任务书

班　　级	自动化 2 班	学生姓名	侯佳维	学　　号	20122779
发题日期：	2015 年 11 月 30 日			完成日期：	2016 年 6 月 4 日
题　　目	Monocular Visual Odometry on the GPU				

1、本论文的目的、意义

One of the main challenges of mobile robotics is to know the location of the robot as it moves in the environment. This is important to build a map of the environment, to follow a pre-computed path and to assess the progress towards the high level task goals of the system. Visual data from cameras provide lots of data with a high update rate. Using this visual data to estimate the odometry of a moving system and thus estimating its location is a widely investigated approach in robotics. Achieving a high update rate, preferably with real time performance, will allow for more accurate localization. Utilizing the Graphics Processing Unit (GPU) for this computation yields in high processing speeds and high efficiency. Using and improving state of the art open source software this project will enable robots to utilize a single camera for efficient and real-time localization.

2、学生应完成的任务

1. Read and understand state of the art literature on visual odometry and write the respective part of the thesis document.
2. Use available open source software to quickly reach the state of the art.
3. Devise and perform experiments to test the localization accuracy of the system.
4. Improve the theoretical and/ or implementation aspects of the used software packages with respect to accuracy, usability or speed.
5. Use experiments to measure and document the achieved improvements.

3、论文各部分内容及时间分配：（共 18 周）

第一部分 Literature research. (3 周)

第二部分 Develop monocular visual odometry system on the GPU using available open source software. (4 周)

第三部分 Develop and perform experiments. (2 周)

第四部分 Improve certain aspects of the system. (5 周)

第五部分 Document improvements using experiments and finish the thesis. (3 周)

评阅及答辩 (1 周)

备　　注

指导教师：　　2015 年 11 月 30 日

　　2015 年 11 月 30 日

审 批 人：　　2015 年　　月　　日

Abstract

Monocular visual odometry is an important topic in computer vision, which can apply in computing the location and path of the robot as it moves in the environment.

This thesis is a study and research about monocular visual odometry as well as practical implementation and experimentation of its theory. With monocular visual odometry we can estimate the camera pose transformation between images taken by a single camera at different points in time.

The algorithm starts by describing the SIFT (Scale-Invariant Feature Transform) features and using SiftGPU for feature extraction and feature matching. On the basis of a depth study and research on its mathematical foundation, epipolar geometry, the fundamental matrix and essential matrix are described and derived as a highlight. Besides, there is an introduction to camera calibration and the image coordinates normalization.

The main algorithm of this thesis is the five-point algorithm with RANSAC (Random Sample Consensus) to exclude outliers. The five-point algorithm, which is calculating the essential matrix, is using the matched feature points which contain outliers. Also this calculation does not exclude the pseudo-solutions. Therefore, using a single five-point algorithm to get the correct essential matrix to get the camera pose transformation is not reliable. This thesis's code is written by c++ language contained Eigen library, programming to achieve the five-point algorithm with RANSAC to exclude the outliers of the feature matching and the wrong roots of the essential matrix. This thesis describes the algorithm with a lot of space, and explains how to use the epipolar constraint and Sampson distance to check outliers, this method make the results greatly improved.

The theories and methods introduced in this thesis have been verified by the implementation, as can be seen in the tables of the experiments, using the five-point algorithm with RANSAC can calculate the relative pose transform between two images taken by a single camera at the same scene reliably.

Keywords: Monocular Visual Odometry; Epipolar Geometry; Five-point Algorithm; RANSAC; SiftGPU

摘 要

单目视觉测程是计算机视觉中的一个重要课题，它可应用于计算机器人的位置和路径，对机器人所在的环境进行测绘。

本论文是对单目视觉测程的学习研究，以及对理论的实践与实验，即是利用单个相机在不同瞬间拍摄的图像估计相机的姿态变换。

论文首先介绍了 SIFT（Scale-Invariant Feature Transform）特征，并且使用了 SiftGPU 对图像进行特征提取与匹配。论文在深入学习研究了其数学理论对极几何的基础上，重点推导描述了基本矩阵和本质矩阵。并介绍了标定相机以及对图像坐标进行归一化的方法。

本论文最主要的算法是采用 RANSAC (Random Sample Consensus) 排除异常值的五点算法。由于五点算法使用包含异常值的匹配特征点进行本质矩阵的计算，并且对计算结果没有进行伪解的排除。因此，要使用单次五点算法来获得正确的表示相机姿态变换的本质矩阵是不可能的。本论文采用包含了 Eigen 库的 c++代码，编程实现了使用 RANSAC 进行异常特征点和本质矩阵伪解的排除的五点算法。文中使用了大量篇幅介绍了该算法，并且介绍了如何使用极线约束和 Sampson 距离进行异常值判断，这种判断使结果正确率大大提高。该论文所述的理论及方法经过本人代码实践，通过验证实验的数据表格可以看出，使用带 RANSAC 的五点算法可以以高正确率准确地计算出使用单个相机在同一场景拍的两幅固定图像间相机的相对姿态变换。

关键词：单目视觉测程；对极几何；五点算法；RANSAC；SiftGPU

Contents

1 Introduction........ 1
1.1 The Significance and Background of the Thesis........ 1
1.1.1 The Significance of Monocular Visual Odometry........ 1
1.1.2 Current Research Situation at Home and Abroad........ 2
1.2 Monocular Visual Odometry........ 4
1.2.1 Fundamental........ 4
1.2.2 Method and tools........ 5
1.2.3 The Overall Algorithm Flowchart........ 7
1.3 The Main Content of the Thesis........ 7
1.4 Thesis structural arrangements........ 8
2 Theory and Preparation........ 10
2.1 Overview of ROS........ 10
2.1.1 What is ROS?........ 10
2.1.2 What do we do on ROS?........ 11
2.2 Features Detecting and Matching........ 11
2.2.1 What is SIFT?........ 11
2.2.2 The Purpose of Using SIFT........ 12
2.2.3 Introduction of GPU........ 14
2.2.4 Introduction to SiftGPU........ 14
2.3 Epipolar Geometry........ 16
2.3.1 Epipolar Geometry........ 16
2.3.2 Epipolar Constraint and Triangulation........ 17
2.4 The Fundamental Matrix........ 18
2.4.1 Overview of Fundamental Matrix........ 18
2.4.2 Geometric Derivation of Fundamental Matrix........ 19
2.4.3 Algebra Derivation of Fundamental Matrix........ 20

2.4.4 The Necessary and Sufficient Condition of Fundamental Matrix 21
2.4.5 The Camera Matrix Obtained from the Fundamental Matrix...... 22
2.5 The Essential Matrix........ 23
2.5.1 Camera Calibration........ 23
2.5.2 The Essential Matrix........ 27
2.5.3 Recovering R and t from E........ 29
2.5.4 Chapter Summary 31
3 Five-point Algorithm with RANSAC 32
3.1 Five-point Algorithm........ 32
3.1.1 Basic Knowledge........ 32
3.1.2 The Method of Calculating the Essential Matrix........ 33
3.1.3 The Flowchart of the Five-point Algorithm 36
3.1.4 Result of Five-point Algorithm........ 36
3.2 RANSAC 38
3.2.1 Overview of RANSAC........ 38
3.2.2 Fundamental of RANSAC 39
3.2.3 Parameters........ 39
3.2.4 Algorithm Complexity Analysis of RANSAC 42
3.3 Five-point Algorithm with RANSAC 43
3.3.1 Description of the Algorithm 44
3.4 Chapter Summary 44
4 Experimtal Setup and Results........ 45
4.1 Experimental Steps for Each Part 45
4.2 The Verification Experiment with Different Groups of Relative Motion 46
4.2.1 Experiment with Images At Same Position........ 46
4.2.2 First Rotation: Pitch 48
4.2.3 Second Rotation: Roll........ 53

4.2.4 Third Rotation: Yaw 57
4.3 Testing Experiment with Different Parameters in RANSAC 61
4.3.1 Change Parameters of the No Motion Group 61
4.3.2 Change Parameters of the Second Yaw Group 63
4.4 Chapter Summary 64
Conclusions 65
Acknowledgments 66
References 67

1 Introduction

1.1 The Significance and Background of the Thesis

1.1.1 The Significance of Monocular Visual Odometry

In the research of mobile robots, to know the real-time position of the robot is very important. But, most of methods have their limits. Using visual information to locate the robots has become one of the current hot topics. We can get the current environment information from the robot cameras, and then calculate the motion information when the robot moves (translation and rotation). Then we can compute the related location of the robots.

This has many applications, for example in Simultaneous Localization And Mapping (SLAM). Here we need to know the location of the robot (Localization) in order to create the map (Mapping). Other applications include the following of a precomputed path or the assessment of the progress towards the high level task goals of the system. Visual data from cameras provide lots of data with a high update rate. Using this visual data to estimate the motion (odometry) of a moving system and thus estimating its location is a widely investigated approach in robotics.

In visual odometry we use the information provided by the camera by comparing the previous camera image (frame) with the current frame. The goal is to compute how the camera pose (position and orientation) changed between the two images. If the camera is rigidly mounted on the robot we can then easily deduct the motion of the robot from the calculated motion of the camera, thus achieving visual odometry.

Achieving a high update rate, preferably with real time performance, will allow for more accurate localization. Utilizing the Graphics Processing Unit (GPU) for this computation yields in high processing speeds and high efficiency. GPUs are nowadays available on a wide range of devices: from supercomputers, over desktop and laptop PCs to mobile phones.

Using and improving state of the art open source software, this project enables robots to utilize a single camera for efficient and real-time localization. We will test this software using real robots. The experiments will demonstrate the performance and shortcomings of monocular visual odometry.

1.1.2 Current Research Situation at Home and Abroad

Visual Odometry is an approach to estimate the agent motion which only uses the visual data with a single or several cameras. There is a good tutorial [1] for beginners, from which I summary some good introduction in my thesis. Its application includes robotics, augmented reality, wearable computing, and automotive. Visual odometry, this term, is created by Nistèr in 2004 in his paper [2]. How can visual odometry estimate the motion? Its principle is to detect the differences between adjacent frames from the camera, which are different due to the the motion of the agent.

In the following the state of the research on visual odometry between 1980 and 2015 will be presented. Most visual odometry implementations in the first-twenty years are offline. Some real-time implementations began to appear in the third decade. What is this real-time approach for? visual odometry estimates the poses of the argent frame by frame. After accumulation of time, the track of the camera can be computed. For this great progress, visual odometry was first used on another planet, Mars, by the Mars explorations[3, 4]. More precisely, Visual Odometry is a special part of SFM (Structure From Motion). The algorithm for extracting the camera poses and 3D structure from a series of images from calibrated or calibrated or uncalibrated camera is called SFM in Computer Vision. Compared to visual odometry, SFM is more generic. It deals with the three-dimensional reconstruction of the poses and the structure of the camera from a set of images - in order or not in order. But visual odometry focuses on real-time and sequential (once a new frame arrival) computation to estimate the three-dimensional movement of the camera.

Using only the visual data to estimate the egomotion of the vehicle began in the early 1980s. There is a description by Moravec[5]. What is interesting is that we find that most the study of visual odometry in beginning is for planet explorations, especially motivated by NASA (National Aeronautics and Space Administration) Mars exploration program.

As we all known, wheel odometry always encounters the problems with the accuracy. For example, wheel sliding on the floor or just wheel rotation can lead to non-uniform movement, which can produce distance errors. And the problem can be more complex if the wheels travel on a non-smooth surface. What's worse, the error accumulates as the time goes. Compared to unreliable wheel odometry, visual odometry can provide much higher accuracy trajectory estimates. Therefore, it is widely required by some applications where the usage of wheel odometry is unreliable, such as space exploration project, underwater project, and so on. It strives to provide with calibrated all-terrain detectors to measure their six degrees of freedom of movement when wheels travel upon uneven and rough terrain.

Most of research on visual odometry is based on stereo camera systems. It is common among these works that, for every stereo pair, the 3D points are triangulated. And the relative motion problem is seen as a 3D to 3D point location problem to solve. Comport et. al.[6] introduced a different approach of motion estimation. Unlike the before scheme to use 3D to 3D point registration or 3D to 2D point camera poses estimation technology, it depends on the quadrifocal tensor. This method allows motion to be computed in 2D to 2D point image matches, which doesn't need to triangulate 3D points in stereo pairs. Using original 2D points directly instead of triangular 3D points results in more accurate calculations.

Instead of stereo visual odometry we want to use monocular visual odometry in this project. In this case, only the location information is available. Therefore, it's disadvantage is the motion can only be restored to a scale factor. Its absolute size can be obtained from direct measurements (for example, measuring a scale factor of the scene), motion constraints, or other sensors integrated, such as IMU and distance sensor. Under certain situations, stereo visual odometry degrades in performance and we have to use monocular visual odometry. When the distance to the scene is much longer than the distance between the two cameras of the stereo system, stereo visual odometry does not work anymore. Different to the stereo scheme, both relative motion and 3D structure in monocular visual odometry should be computed from 2D image data.

In the last ten years, a successful outcome over long distances with single camera has been obtained using perspective and omnidirectional camera[7–12]. There are three methods for the work: feature-based method, appearance-based method, and hybrid approaches. The first method is the work by the authors of [2, 8, 9, 11, 13–15]. It bases on the feature which are obvious and can be tracked in the frames. The second method makes use of all the pixel intensities or a part of pixels. The hybrid approach is a combination of both of them.

The first real-time, large scale monocular visual odometry was performed by Nister et. al.[7] They use RANSAC to reject outliers and estimate camera pose to compute the coming poses. The five-point minimal solver[7] is used in this thesis to calculate the motion with RANSAC, which made the five-point algorithm results became exactly in visual odometry[7, 9, 11]. To compute the visual odometry data, five-point algorithm will be used in our project.

An important tool that I will use in my project is SiftGPU(SIFT on GPU). Andrea Vedaldi's sift++[16] and Sudipta N Sinha et al's GPU-SIFT[17] make great contribution to SiftGPU's development. So far, many parameters in SiftGPU are inherited from sift++ (for instance, number of DOG levels, number of octaves, edge threshold, and so on). And Sift on GPU is based on feature detecting and matching[17]. In my project, we use SiftGPU to detect and match features.

1.2 Monocular Visual Odometry

1.2.1 Fundamental

Monocular visual odometry is a method that use the images taken by a single camera fixed on a agent as the input to calculate the relative motion information between frames, and finally obtained the agent poses by superimposing the motions. For the single camera, assume that the image sequence photographed at various times is donated as $I_{0:n} = I_0, ..., I_n$. The main task of the visual odometry is to calculate the relative transformations T_k between the frames I_k and I_{k-1} for $k = 1, ..., n$, then recover the current pose of the camera C_n through series the relative transformations.

There are two main implementations can be used to calculate the relative motion: the appearance (or global) based method, which uses the intensity information of all pixels of two input images, as well as feature-based approach, which uses only the features extracted (or track) from the images. The appearance-based approach is not as accurate as feature-based method, and more expensive computationally. Feature-based method requires a robust matched (or tracked) features between the frames, but is faster and more accurate than the appearance-based approach. Thus, most of the visual odometry programs are on feature-based method.

In this thesis, I use the feature-based implementation as well. With this method, to estimate the relative motion between the instant k and $k-1$, that the relative transformation T_k, first of all, after we obtain the frames I_k and I_{k-1}, we need to extract the features from these two images then match the corresponding features. Note that I chose to use feature matching approach rather than feature tracking, both have been mentioned at section 2.2.2. Feature tracking means to find the feature in one image and then in the next one image uses a local search technique to track them; and feature matching means to independently identify the features in each frame, and then based on some similarity to matching them between frames. Then we calculate the transform between the corresponding image points as the transform between two frames. However, there are must some outliers in these matches. For a accurate motion estimation, the matching features, as the input, can not include outliers, so we need to use some outliers excluded method to select the inliers as much as possible to be the input of the motion estimation. Calculation the relative transformation of two frames actually is to calculate the mapping between the corresponding image points, expressed as the essential matrix (when the image point coordinates are normalized coordinates). The matching features coordinates for motion estimation are calibrated, by the knowledge of the epipolar geometry, using the normalized image coordinates (that is say have been calibrated) can directly work out the essential matrix. The essential matrix contains the translation and rotation information, so the camera matrix between two images (assume camera matrix of the image I_{k-1} is $[I|0]$, the rotation matrix R and translation vector t of the image I_k is the relative rotation and translation between two images) can be extracted from the essential matrix. The camera matrix extracted from essential matrix is not unique (mentioned in section 2.5.3). Therefore, we triangulate the corresponding points to select the correct solution, that choose a solution which image points are both in the front of the two cameras. After getting the camera matrix, we can also use triangulation to re-projection to exclude some of the wrong essential matrix.

After obtaining the relative transformation between every two frames, we can get the final camera pose through seriesing the transforms. The camera pose in the moment k, donated as C_k, is the integration of C_{k-1} and T_k, that $C_k = C_{k-1}T_k$. Suppose the camera pose in the moment $k = 0$ is C_0, the camera's current pose is the integration of all the relative transforms $C_n = C_0(T_1...T_n)$.

1.2.2 Method and tools

We chose the Asus Xtion to take pictures for us. It is a equipment with depth camera (there are three cameras on it), but in view of our project needs, we only use the normal camera. The package openni2_launch contains the driver OpenNI-compliant for the camera on ROS. Tbe openni2_launch start command can replace roscore command to start ROS, drive camera, then publish the topics /camera/rgb/image_raw. In this way, we can subscribe this topic in the code to get the images captured by the camera.

After receiving the image, we use powerful visualization tools Rviz to view real-time images captured by the camera. We need to start the Rviz, and then add the image display in Rviz and subscribe the topics /camera/rgb/image_raw in order to see the image captured by the camera. In addition, we can also use the tool image_view to see the camera images taken in real time, and right-click it to save images at a time. In this way, we can use the fixed frames as the test data.

Visualization tools are just a way to view the image, which is not necessary. In my code, one approach is to use the fixed frames to test the code, in which I take pictures and tell the pictures' path to the code, then the code can find the pictures. When we want to use the successive frames, we can not save each moment frame (which consume a lot of memory, but not necessary). In this order we can use another method, in the code directly subscribe the topic to get images' information, and then calculate the transformations directly.

After incoming the image information to the code, we need to extract the features in each frame. And then to match the features between the frame and the previous frames, we use use SIFT to detect and match features for its outstanding robustness and accuracy. But we do not need to write code to detect features by outselves, we used SiftGPU open source code, as long as we use its interface function in our code. Then it will output the coordinates of the matched features between frames, with these we can calculate the relative transformation between frames.

First we need to get the internal parameters of the camera, the image point pre-multiplied by the inverse of the internal parameters matrix K can be converted to the normalized coordinate form. We use the camera_calibration's cameracalibrator.py node over ROS to calibrate the camera. Then we can get the camera internal parameter matrix K and radial distortion vector D. And the camera will publish a distortion corrected topic after use the cameracalibrator.py node. Thus, we can use the distortion corrected topic directly, without using the distortion vector D in the code.

After obtaining the normalized image points set, we use a five-point algorithm to compute the essential matrix. Since the matching features may include some wrong matches, we can use RANSAC algorithm, with five-point algorithm, to exclude outliers. With this algorithm, we can get the essential matrix calculated by normal matches. This is the most important part of our project, in section 3 has a very detailed introduction. Then, we use SVD decomposition to obtain relative motion, rotation matrix R and translation t, from the essential matrix E. Finally, we can use bundle adjustment to optimize the results.

In the next chapter the flowchart for the whole process will be drawn.

1.2.3 The Overall Algorithm Flowchart

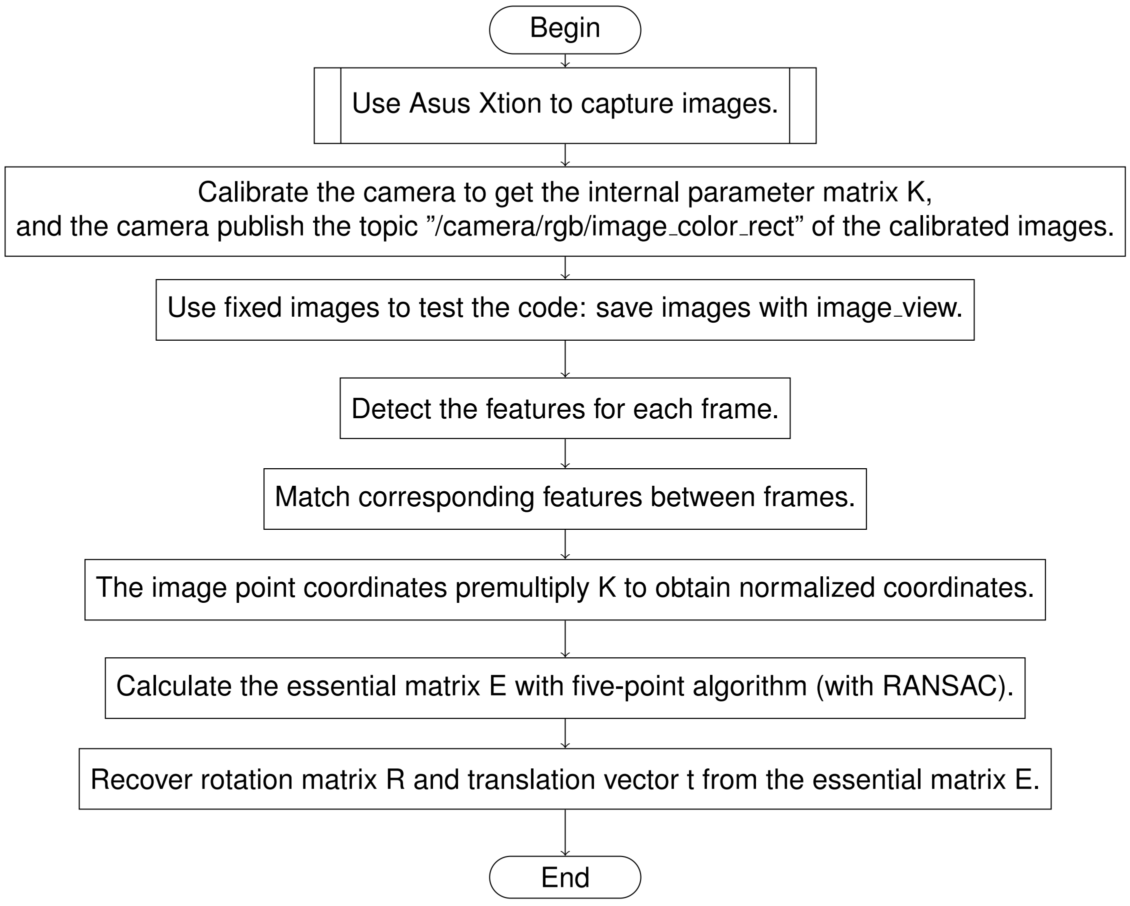

Begin
Use Asus Xtion to capture images.
Calibrate the camera to get the internal parameter matrix K, and the camera publish the topic "/camera/rgb/image_color_rect" of the calibrated images.
Use fixed images to test the code: save images with image_view.
Detect the features for each frame.
Match corresponding features between frames.
The image point coordinates premultiply K to obtain normalized coordinates.
Calculate the essential matrix E with five-point algorithm (with RANSAC).
Recover rotation matrix R and translation vector t from the essential matrix E.
End

Figure 1-1 The Overall Algorithm Flowchart

1.3 The Main Content of the Thesis

This thesis introduces the monocular visual odometry project, namely using the images to estimate the relative motion of the camera. Realization of this project includes the following parts:

1) Use Robot Operating System[18] (ROS) to capture (live) images from cameras.

2) Use ROS to calibrate the camera and obtain the camera parameters.

3) Use available open source software in SiftGPU to extract features and descriptors and to match the images' features.

4) Use Nister's Five-Point Motion Estimation Algorithm to compute the visual odometry.

5) Use five-point algorithm combining with RANSAC to exclude outliers and make the results more accurate.

6) Collect data with ground truth motion.

7) Do experimental validation of the algorithm under different configuration parameters.

Before all the principles being introduced, it will use a chapter explains the basic principles of monocular visual odometry, and the overall algorithm, as a overview of all the algorithms and principles. As the prepare part of the thesis, there is a introduction of the operating system my project based on, ROS, and a simple description of the GPU. In the project implementation process, before the beginning of each part, read the relative papers while recording there principle. After achieving the code of this part successfully, it's necessary to write its principles and algorithms in my thesis. The thesis uses a lot of space introduces the main algorithms used in this project, the five-point algorithm, and the fundamental it mainly based on the epipolar geometry, and the algorithm combining with RANSAC. Besides, there is a reasonable space for SiftGPU, the method of detecting and matching features. Incidentally, analyze the advantages and disadvantages of these algorithms. In the final part of the thesis, there is the detailed process and result of my experimental, after which the experimental results were analyzed and evaluated.

1.4 Thesis structural arrangements

This thesis is structured as follows:

The chapter 1 is the introduction, including the background and significance of this thesis, after that there is a overview of monocular visual odometry. In this part, it introduces the principle of monocular visual odometry and the research tools(including algorithms and fundamental knowledge) involved. Then a flowchart of its implementation is given to let readers understand what the thesis is doing, and the details of every research parts will be included in the following chapters. Finally it introduces the main content and structure of the thesis.

The chapter 2 is the prepare part, which includes two parts. First it gives an introduction of ROS and SIFT(including SiftGPU). The first part is about what is ROS and about the features' extraction and matching, and introduction of the features of SIFT and GPU. At last of this part, it gives an introduction of tools of extraction of features and matching tools—SiftGPU. The second part is the introduction of epipolar geometry, fundamental matrix and essential matrix. Epipolar geometry is the basic mathematical principle of monocular visual odometry, and we gives the details of the mathematical mapping relationship between the process of the camera transformation obtained by the image points of the two images. Besides, we find the fundamental matrix of the transform by epipolar geometry. After introducing the calibration of camera and coordinate normalization, we introduce the essential matrix and explain how to extract rotation and shift information from essential matrix.

The chapter 3 is the five-point algorithm, RANSAC algorithm and five-point algorithm with RANSAC. The five-point algorithm is the main algorithm of the thesis, while RANSAC algorithm can greatly exclude outliers. So, the five-point algorithm with RANSAC is used in the thesis.

The chapter 4 is the experimental setting and experimental results. Two experiments are included in the thesis. The first one is using different rotation transform as input to verify the correctness of the code. The second one uses the fixed images as input, and changes the parameters of RANSAC, to test the correctness and calculation time of the code. For the two experiments, this chapter has a detailed introduction of the experimental procedures with corresponding codes listed. At last, we have analysis the every results of the experiments.

The chapter 5 is the conclusion of the whole thesis.

2 Theory and Preparation

Before calculating the motion of the camera, we need to first get the camera frame of every moment. Estimating camera motion is actually calculating the camera relative position and pose transforms between each of two frames, which is decided by the image features points conversion between views. Thus, after obtaining the frame from the camera in every moment, it needs to extract the features from the frame and match features between this frame and the last frame. Then, it can output the matching feature points for the next step calculation.

The most important arithmetic in this project is five-point algorithm, and the basic theory of the algorithm is epipolar geometry. Thus, to facilitate describing the five-point algorithms in next chapter, this chapter is actually a introduction of the five-point algorithm basic theory, the epipolar geometry. When we get the calibrated matching features between the images, we can use five-point algorithm to obtain the essential matrix, which can be easily described after introducing the fundamental matrix.

2.1 Overview of ROS

2.1.1 What is ROS?

The Robot Operating System (ROS) is an open-source meta-operating system for robot[18]. It provides the necessary services for operating system, including the hardware abstraction, the underlying device control, commonly used functions, message passing between processes, and package management. It also provides tools and library functions for obtaining, compiling, writing codes, and running code across computers. In certain aspects ROS equivalent of a robot frameworks.

ROS running process is a loosely coupled peer-to-peer network based on ROS communications infrastructure. ROS implements several different means of communication, including those based on synchronous RPC-style communication services mechanism, based on asynchronous data streaming topics server mechanisms and parameters server. Our project communication is based mainly on the topic mechanisms.

2.1.2 What do we do on ROS?

The main objective of ROS is to provide code reusing support for robotics research and development. ROS is a distributed process (i.e. nodes) framework, these processes are encapsulated in easily share and publish packages and feature packs sets. ROS support a joint system similar to code repository, which can also achieve project collaboration and publishing. This design allows the development and implementation of a project from the file system to the user interface completely independent decisions (without ROS limitation). At the same time, all projects can be integrated by ROS basic tools.

In this project, it mainly uses openni2_launch to drive the camera Asus Xtion. This package contains a launch file, using OpenNI-compliant to drive the type of kinect camera over ROS, and publish the topics. A detailed introduction can be found in [19].

In addition, ROS has many useful tools. For example, in this project, we used rviz visualization tool to view the input images, use the bag file to record data as input and used rqt to draw the waveform of the output data.

2.2 Features Detecting and Matching

After getting the frames from the camera, we should detect and match the features in frames. Then we can use these matched features pairs to calculate the relative motion between frames. In my project, the SiftGPU, SIFT for GPU, was used for detecting and matching features. Therefore I will introduce SIFT and GPU, and discuss why I use SIFT compared with other detectors/discriptors.

2.2.1 What is SIFT?

SIFT (Scale-Invariant Feature Transform) is a local feature extraction algorithm, looking for extreme value in scale space, extracting location, scale, rotation invariant. It is a kind of outstanding blob detector. SIFT was proposed by D.G.Lowe in 1999[20], and he developed and summarized the algorithm in 2004[21]. Later Y.Ke improved it through using PCA to replace histogram in descriptor part[22].

SIFT features are local features in the images, keeping invariance of rotation, scaling, brightness changes, also maintaining a certain degree of stability for viewpoint changing(up to 60 degrees), affine transformation and noise. Based on these characteristics, they are highly notable and relatively easy to be captured. Large amount of information of SIFT descriptors makes it suitable for fast and accurate matching in massive database. When SIFT features are used under the conditions of small features database, it is accessible to instant computation on recognition speed.

In Mikolajczyk's invariance comparative experiments[23] for 10 kinds of local descriptors, SIFT and its expansion algorithm have been confirmed that have the most robust in the same type of descriptors.

2.2.2 The Purpose of Using SIFT

Given a sequence of images, generally speaking, there are two methods that can be used to find the corresponding features between two frames. One method is to find the features in the first frame, and then track these features in the next frames. The second method is detecting features in each frame, then doing feature matching between every two adjacent frames.

The first method is suitable for the case that most features of the first frame can be traced in the second frame, that is, the viewpoint changing between the two frames is small, i.e., during the time between taking two photos, the camera only made a little movement. Most of early visual odometry works[5, 24–26] use this approach.

The second method is more suitable for the camera made large movement or large viewpoint changed between two frames. In this case, the features from the first frame are difficult to be trace in the second frame. It is used more frequently in the last decade[2, 9, 11, 13, 14]. This is because the early VO researches are based on small scale movements, and the main visual odometry work in recent years was in large-scale environments. It makes it possible to use images with large movements in between. The advantage of this method is to avoid feature drift in the tracking process based on cross-correlation.

Some properties a good feature detector should have are:

* good localization accuracy (including in location and scale),
* reproducibility (majority of the detected features can be detected in the next frames),
* computation efficiency,
* robustness (noise, blur and compression artifacts),
* distinctiveness (these features can be accurately detected in the different frames),

* invariance (under different illumination, scale, geometric transformations and fluoroscopy distortion).

There are many points feature detector described in the VO literature, for example, point detector[5, 27–29], blob detectors[21, 30–32] and various detectors, each detector has its advantages and disadvantages.

Table 2-1 Properties and performance comparison of feature detectors[1]

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	Y	N	Y	N	N	+++	+++	++	++
Shi-Tomasi	Y	N	Y	N	N	+++	+++	++	++
FAST	Y	N	Y	Y	N	++	++	++	++++
SIFT	N	Y	Y	Y	Y	+++	++	+++	+
SURF	N	Y	Y	Y	Y	+++	++	++	++
CENSURE	N	Y	Y	Y	Y	+++	++	+++	+++

From Table 2-1, we know that blob detector performs better than corner detector in term of invariance(scale invariance and affine invariance). Though it is not as strong as corner detector in localization accuracy, its repeatability and robustness performance was phenomenal.

SIFT is a kind of outstanding blob detector. And it's widely used for the following advantages:

a) SIFT descriptor has a excellent robustness as local features in invariance.

b) Besides, it have well distinctiveness, and rich information.

c) A mount of SIFT features can be obtained from a few objects .

d) It provides high computation speed. The optimized SIFT algorithm can be seen as a real-time method under current computer hardware.

e) Its outstanding scalability makes it can easily be combined with other forms of feature vectors.

For these excellent properties, we can confirm that SIFT is really a good feature detector. And it's worth to use SIFT to detect and match features.

2.2.3 Introduction of GPU

NVIDIA first proposed the concept of GPU when they announced the GeForce 256 graphics processing chips in 1999. From then on, NVIDIA use this new name to call the chip of graphics[33, 34]. GPU makes the graphics reduce the dependence on CPU, and it replaces CPU to do some works, especially in the 3D graphics processing. GPU is no longer confined to the 3D graphics processing now, its general computing technology has caused quite a lot of attention. It turns out that in the aspects of the floating point computing, parallel computing, etc., GPU can provide dozens of times or even a hundred times better than the CPU in performance. General standard in computation for GPU are: CUDA, OpenCL and ATI STREAM.

Computing industry is developing from using only the CPU central processing to the CPU and GPU co-processing. Graphics card maker NVIDIA introduced CUDA (Compute Unified Device Architecture) programming model that is wanted to take advantage of the respective advantages of CPU and GPU in the application. CUDA is a a universal parallel computing architecture introduced by NVIDIA, the architecture makes the GPU can solve complex computational problems. It includes the CUDA Instruction Set Architecture (ISA) and the GPU parallel computing engine. Developers can use C language to write programs for the CUDA architecture, the program can be run on the CUDA processor to support ultra-high performance.

OpenGL Shading Language is used in OpenGL shader programming, that developers write short custom programs. They are executed on the GPU, instead of the fixed part of the rendering pipeline, so the rendering pipeline has programming types in different levels. For example: view transformation, projection conversion. GLSL (GL Shading Language) shader code is divided into two parts: Vertex Shader and Fragment, and sometimes there is Geometry Shader.

2.2.4 Introduction to SiftGPU

SiftGPU is an implementation of SIFT[21] (Scale Invariant Feature Transform) for GPU[35]. It provides two implementations: CUDA and GLSL. And it runs on GLSL by default. SiftGPU can processes pixels/features in parallel to do the work in following step:

1) SiftGPU converts color input images to intensity and up-samples or down-samples them.

2) Builds Gaussian pyramids and detects DoG Keypoints.

3) Generates compact feature lists according to GPU list generation[36].

4) Processes features in parallel and determines their orientations and descriptors.

During this process, SiftGPU uses the GPU/CPU mixed method, which improves the calculation speed considerably.

We can get the SiftGPU codes of Changchang Wu and its manual from [35]. The program package includes some functions about reading the pictures, converting colour image to intensity image, detecting features from images, and matching features between images, etc. In addition, it requires that the GPU has a large graphics memory and supports dynamic branch. It defaults to using GLSL, and CUDA is not used by default. If you want to use CUDA, you should set CUDA_SIFTGPU_ENABLED in compiler and recompile. But my computer doesn't support CUDA, so GLSL is used.

In my code, it use SIFT::RunSIFT function, than it is easy to get features from the image by using SIFT::GetFeatureVector function. After getting features from both two images, match these features with SiftMatchGL::GetSiftMatch function. As before shown to you, the program use Changchang Wu's code to detect and match features instead of writing the codes to do this by myself.

The following figure is a feature matching instances. In the figure 2-2, I used circle represents the detected feature points and the straight lines connecting the corresponding features. In the figure, there are some matches are wrong, but not many (looks not more than 10%).

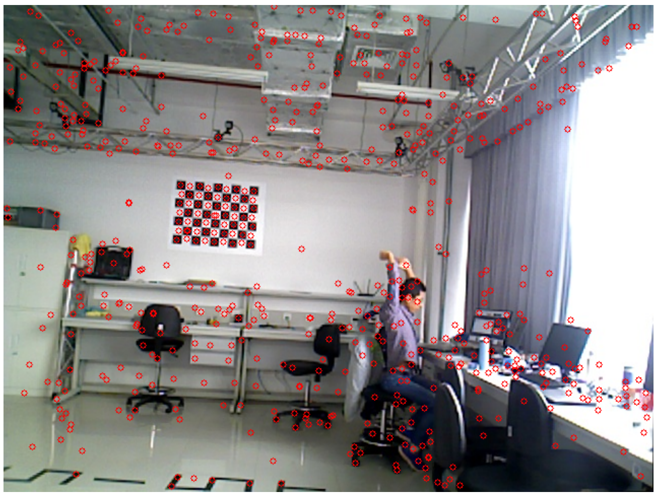


Figure 2-1 SIFT Features in the Image

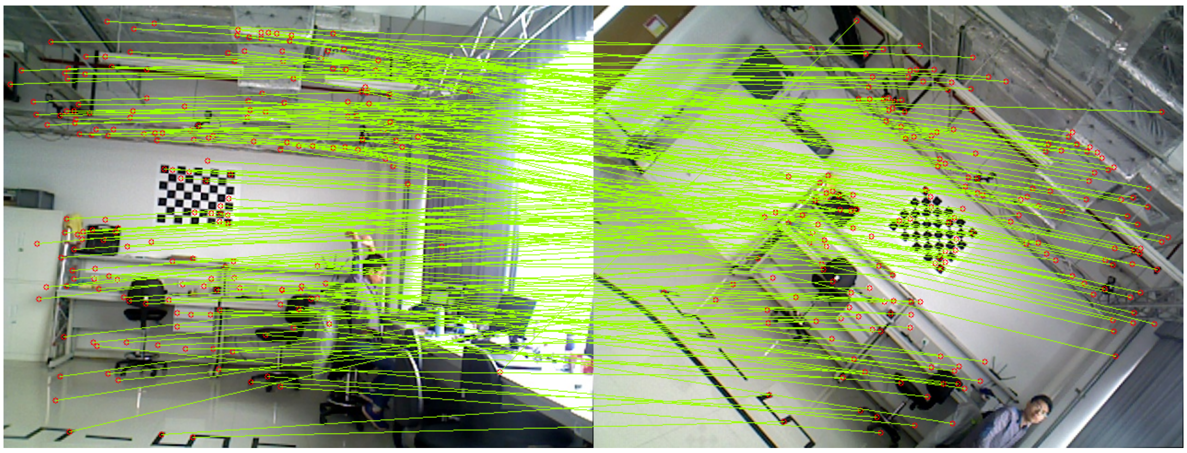


Figure 2-2 Match the Features Between Two Images

2.3 Epipolar Geometry

In the beginning introduction of this chapter, we have known the relationship between the five-point algorithm, essential matrix and the epipolar geometry. It found that the essential matrix is deduced from the epipolar geometry. Therefore, it's essential to telling the fundamental and the derivation of the epipolar geometry before the description of the essential matrix. In this subsection, it highlight the fundamental matrix F, and its associated mathematical relationship will be derived in detail, in order to give the relationship between the essential matrix E and the fundamental matrix F in the next subsection.

2.3.1 Epipolar Geometry

Epipolar geometry is the intrinsic projective geometry between two views[37]. It does't depend on scene structure, but only decided by the camera internal parameters and the relative pose between two cameras. Due to the reason that it uses the images which were taken by a single camera at different instants, it can be seen as that it uses two camera to take two images respectively and calculate the relative pose of these two cameras from these two images.

The geometric entities included in epipolar geometry are shown in Figure 2-3. For stereo vision system(two cameras), definite the optical center of the two cameras are C and C'. Assume that there is a 3D scene point X in 3D space. The plane made up by the space point X and camera centers C, C' is called epipolar plane. Every camera has its own view, denoted as $Image1$ and $Image2$. The line CX intersects Image1 at point x, and the line $C'X$ intersects Image2 at point x'. Two camera centers

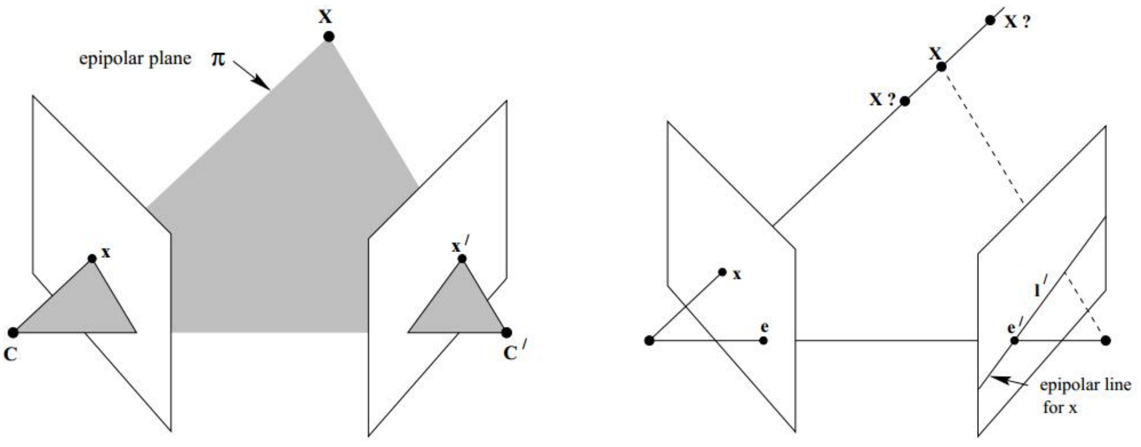


Figure 2-3 Epipolar Geometry[38]

intersect $Image1$ and $Image2$ at points e and e' respectively. These two points are called epipoles(or epipolar points). The line connecting two camera centers CC' is baseline. The epipolar plane intersects two image planes($Image1$ and $Image2$) at the two epipolar lines l and l' separately. What's the relation between the epipolar lines l and l'? They are corresponding to each other, on which the point x, e and x', e' lie correspondingly. The rays from x and x' to the space point X is coplanar, which tells that more significance of the epipolar plane is to find the correspondence between x and x'.

2.3.2 Epipolar Constraint and Triangulation

The epipolar plane is constituted by the ray from x to X and the baseline, and x' is on the plane, which implies that we can find the x' take advantage of this relationship. If we have know the projection point x in $image1$ and the relative pose between two cameras, it's convenient to find the projection point x' in $image2$ with epipolar constraint. The definition of epipolar constraint is that, for every space point X, if the projection point x in $image1$ is known, its corresponding epipolar point x' must lie on the only corresponding epipolar line l' of x. Because x is known, then its corresponding epipolar line l' can be calculated. According to the epipolar constraint, we can just find the other projection point x' on the epipolar line l' but not finding it on the whole image2. The epipolar constraint between two cameras can be described as the essential matrix E or the fundamental matrix F. However, it uses a single camera in this thesis, so the relative pose between two views is not fixed that the epipolar constraint is not used for finding the corresponding points between two views. We use it in another aspect, verifying the essential matrix E. When the essential matrix E is worked out, and we get a number of pairs of corresponding epipoles x and x' in *image*1 and *image*2 respectively, then we can use epipolar constraint to verify the relative pose, the essential matrix E, between two views.

Triangulation has a opposite usage against epipolar constraint, with which we can get space point X from its two projection points x and x'. If the x, x' and their projection lines are known, these two lines must intersect at X in 3D space. Therefore triangulation is useful in structure recovery.

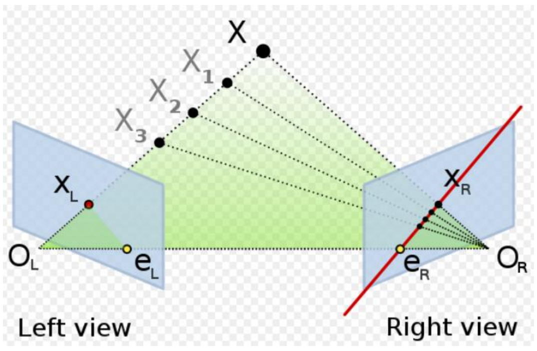


Figure 2-4 Epipolar Constraint[37]

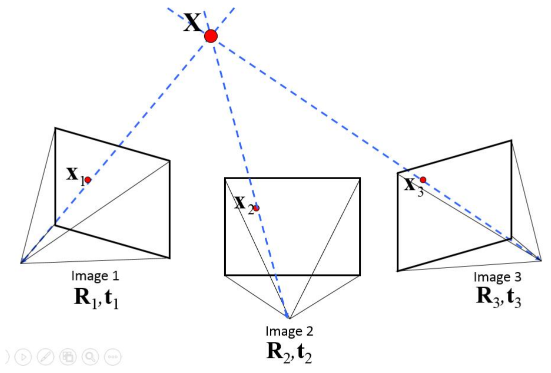


Figure 2-5 Triangulation[39]

2.4 The Fundamental Matrix

2.4.1 Overview of Fundamental Matrix

Fundamental matrix is a algebra representation of epipolar geometry, encapsulating this internal geometry relationship, which is a 3*3 matrix, denoted as F. It can be computed from the scenes image points without knowing the cameras internal parameters or their relative poses. There introduces the fundamental matrix very specifically in [38]. According to Figure 2-4, to every projection point x, there is the only corresponding epipolar line l'. This mapping is denoted as $x \mapsto l'$, which can be described by F. For a space point X, if x is its projection on image1 and x' is that on image2, then their corresponding fundamental matrix satisfy the condition

$$x'^T F x = 0 \tag{2-1}$$

Its derivation will be showed in Section 2.4.4. Now let's derived the fundamental matrix from two directions.

2.4.2 Geometric Derivation of Fundamental Matrix

As described above, fundamental matrix F is a mapping from a point on one image to a line on another image. This mapping can be decomposed into two steps: in the first step, there is a point x' on image2 corresponding to the point x, lying on the epipolar line l'; in the second step, the epipolar line l' is a line joining the points x' and epipole e'.

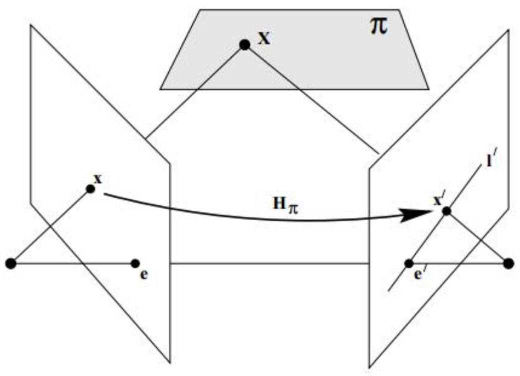


Figure 2-6 The point transform through a plane H_π

The first step is a point transform through a plane. As the Figure 2-6 illustrated, suppose that the plane H_π doesn't through two camera centers C and C', the line passing C and x intersects the plane H_π at space point X. And the projection point of X on $image2$ is x'. This process is a point transform through plane. For every pair of x and x', there is a homography H_π, then the mapping can be described as

$$x' = H_\pi x \tag{2-2}$$

Then we construct the epipolar line l' in the second step. Because the epipolar line l' join the point e' and x', it can be described as

$$l' = e' \times x' = [e']_\times x' \tag{2-3}$$

The cross product of e' and x' can be written as $[e']_\times x'$, the definition of $[e']_\times$ can refer to [40].

Substitute the formula (2-2) into the formula (2-3), we have

$$l' = [e']_\times H_\pi x = Fx \tag{2-4}$$

where F is defined as

$$F = [e']_\times H_\pi \tag{2-5}$$

The rank of $[e']_\times$ is 2, and the rank of H_π is 3, thus the rank of F is two. In other words, F is a mapping space from a 2D image point to a 1D epipolar line l' passing the epipolar point e', so it is a matrix of rank 2.

2.4.3 Algebra Derivation of Fundamental Matrix

From the fundamental matrix F to the camera projection matrix P,P' form can be obtained from the algebraic derivation. The relationship of x and X can be described as $PX = x$(it will be deduced in the Section 2.5.1). The solution of the ray can be expressed as

$$X(\lambda) = P^+x + \lambda C \tag{2-6}$$

where P^+ is the pseudo-inverse of P ($P^+P = I$), and C is the camera center, which is a null-vector. Thus, $PC = 0$. The equation (2-6) shows a ray parameterized by λ. There are two points: when $\lambda = 0$, the point is P^+x; when $\lambda = \infty$, the point is C(the camera center). According to $x' = P'X$, there are two corresponding points on the ray back-projected from x' by P': $P'P^+x$ and $P'C$. The epipolar line l' joining these two point, which can be written as

$$l' = (P'C) \times (P'P^+x) \tag{2-7}$$

where $P'C$ is the projection of the first camera center on the $image2$, denoted by e'. Thus, we have

$$l' = [e']_\times(P'P^+x) = Fx \tag{2-8}$$

Then, we obtain the fundamental matrix

$$F = [P'C]_\times P'P^+ = [e']_\times P'P^+ \tag{2-9}$$

Compared the formula (2-9) with the formula (2-4),

$$H_\pi = P'P^+$$

which shows that the homography H_π can be obtained from the two camera matrices.

Assume there are two calibrated camera matrix

$$P = K[I|0] \qquad P' = K'[R|t] \tag{2-10}$$

then we have

$$P^+ = \begin{pmatrix} K^{-1} \\ 0^T \end{pmatrix} \qquad C = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2-11}$$

Combine the equation (2-9) with the equations (2-10) and (2-11), F can be written as

$$\begin{aligned} F &= [P'C]_\times P'P^+ \\ &= [K't]_\times K'RK^{-1} = K'^{-T}[t]_\times RK^{-1} = K'^{-T}R[R^Tt]_\times K^{-1} \\ &= K'^{-T}RK^T[KR^Tt]_\times \end{aligned} \tag{2-12}$$

Because the two epipoles are

$$e = P\begin{pmatrix} -R^Tt \\ 1 \end{pmatrix} = KR^Tt \qquad e' = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = K't \tag{2-13}$$

Substitute the equation (2-13) into the equation (2-12), we can see that the fundamental matrix F can also be written as

$$F = K'^{-T}RK^T[e]_\times \tag{2-14}$$

2.4.4 The Necessary and Sufficient Condition of Fundamental Matrix

A 3×3 matrix F is the fundamental matrix corresponding the transform between x and x' if and only if the matrix F satisfy the condition(2-1):

$$x'^T F x = 0$$

We can verify that it is the necessary and sufficient condition of fundamental matrix as follows.

If x' is the correspondence epipole of x, and lies on the epipolar line l'. That $x'^T l' = 0$, combine with formula(2-8), we get $x'^T F x = 0$ as required. Conversely, if the matrix F satisfies the condition (2-1), the back-projection rays from x and x' is coplanar, thence it's the necessary condition for the two points corresponding to each other.

The significance of this conclusion is that it gives a expression of F without going through the camera matrix P or P' that F can be directly obtained from the corresponding image point calculated.

2.4.5 The Camera Matrix Obtained from the Fundamental Matrix

One of the important usefulness of fundamental matrix is that we can extract the pair of camera matrices from it. However, the pair of camera matrix is not unique due to the projection invariance of fundamental matrix. In this case, we can obtain the canonical form of the pair of camera matrices from the fundamental matrix, which is unique. Now, let me introduce the projection invariance of the fundamental matrix F and the canonical form of the camera matrix to you.

As we can see from the formula $x'^T Fx = 0$ and $l' = Fx$, the fundamental matrix F is the mapping from a image point to a image line or point, that means that F is only measured by the image coordinate, but not any world units(such as meter or degree). That is to say, the relationship between x and l' is just projection relationship, for which we can say the image relationship is projection invariance. When we make a projection transform for x and x': let $\hat{x} = Hx$ and $\hat{x'} = H'x'$, there is their corresponding mapping: $\hat{l'} = \hat{F}\hat{x}$, where $\hat{F} = H'^{-T}FH^{-1}$, it is a rank 2 matrix.

The camera matrix P does not only depend on the image coordinate frame measuring, but also on world coordinate frame measuring for the reason that it connects the image measuring and the world measuring forms. Different from the camera matrix, the fundamental only depends on the projection property of the camera matrices, which make it have no matter with the world frame measuring. For example, a projection transform on camera matrices (P, P') does not affect the fundamental matrix F. More specifically, let's suppose a 3D space projection transform H, which is a 4×4 matrices, impose the projection transform H on the camera matrices (P, P'), the fundamental matrix corresponding to the camera matrices (P, P') and the camera matrices $(PH, P'H)$ are same.

In fact, different projection transform between camera matrix pairs is the only ambiguity. For this ambiguity, we specify the canonical form of the camera matrix pair (P, P') corresponding to the fundamental matrix F. In the canonical form, we define $P = [I|0]$, where I is a 3D identity matrix and 0 is a 3D null-vector, then the pair of camera matrices corresponding to the F is unique.

To prove it is feasible, we can expand the camera matrix P to a 4×4 matrix P^*, let $H = P^{*-1}$, then $PH = [I|0]$ as required.

The projection invariance can be use in turn: if both camera matrices pairs (P, P') and $(\hat{P}, \hat{P'})$ correspond to the same fundamental matrix F, there must exists a 4×4 nonsingular matrix H such that $\hat{P} = PH$ and$\hat{P'} = P'H$.

There is a useful theorem: Support a pair of camera matrices $P = [I|0]$ and $P' = [M|m]$, to which the fundamental matrix corresponds is $F = [m]_{\times}M$. If and only if $P'^{T}FP$ is a skew-symmetric matrix, F is the fundamental matrix corresponding to (P, P').

Camera matrix is a mapping from 3D world points to 2D image points, represented as $x = PX$[41]. The specific definition of the camera matrix will be introduced later. If the image points have been calibrated, $P = [I|0]$ is the camera matrix of the first image. If we use the image points calibrated in advance, we can compute the essential matrix directly, and get the camera matrix in canonical form from the essential matrix without doing projection transform(finding the transform matrix H). Therefore, in my project, it's necessary to calibrate the camera and use the calibrated image points to compute the essential matrix, with no necessary to compute the fundamental matrix, then get the camera matrices in canonical form. I will introduce the camera calibration and the relationship between the essential matrix and the fundamental matrix in the section 2.5.

2.5 The Essential Matrix

In the previous subsection fundamental matrix F have been derived according to the epipolar geometry, which is calculated under the image uncalibrated case, as the representation of transform between views. If we have calibrated the camera, and use the calibrated image points (using normalized coordinates) to calculate the relative transformation between views, which can be represent as the essential matrix E. Therefore, this subsection will focus on how to calibrate the camera, and the deduction of use normalized coordinates of the image points to get the essential matrix E. After obtaining the essential matrix E, we'll show you how to extract the rotation matrix R and translation vector t from essential matrix.

2.5.1 Camera Calibration

The purpose of camera calibration in my thesis is to get the internal camera parameters matrix K, which can calibrate the image, and let us compute the essential matrix directly. In detail, camera calibration can help us to find:

* The image center (x_0, y_0). It is generally not the point at ($\frac{width}{2}$, $\frac{height}{2}$) of the image, but should be the optical center of the camera.

* The focal length of the camera f.

* Different scaling factor of column pixels and row pixels (Figure differentscaling.jpg), k_x, k_y. It is because the pixels maybe not square. Its imaging process is shows in Figure 2-7, which is the reason why the scaling of pixels in columns and rows is different.

* The skew factor s, imaging skew is shown at Figure **??**. The scaling factor transforms pixel units to length of image coordinate.

* The len distortion D, which results pin-cushion effect Figure 2-8, it happens seriously on fish eye camera.

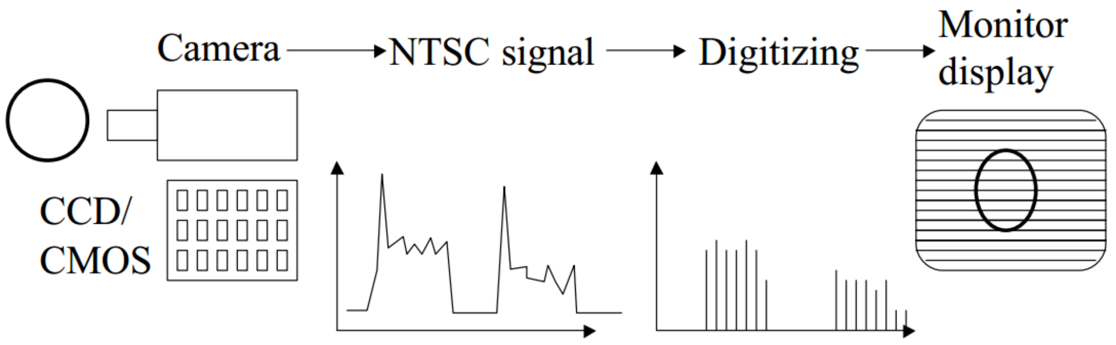


Figure 2-7 Scaling factor of column pixels and row pixels may be different[42]: The output from camera may be analog, then AD converter samples NTSC signal, and digitizing may make the pixels not square.

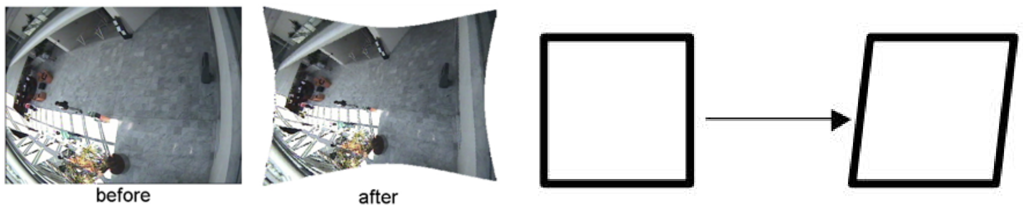


Figure 2-8 Distortion[39] and skew[42]

If both image point and space point were represented in homogeneous vectors, the cental projection is a linear projection(Figure 2-9).

According to the principle of similar triangles,

$$x_i = f\frac{x_s}{z_s} \qquad y_i = f\frac{y_s}{z_s} \tag{2-15}$$

After camera calibration, we get the image center(x_0, y_0), and the scaling factor k_x, k_y. In general, the matched features coordinate we got from SiftGPU is in pixel units

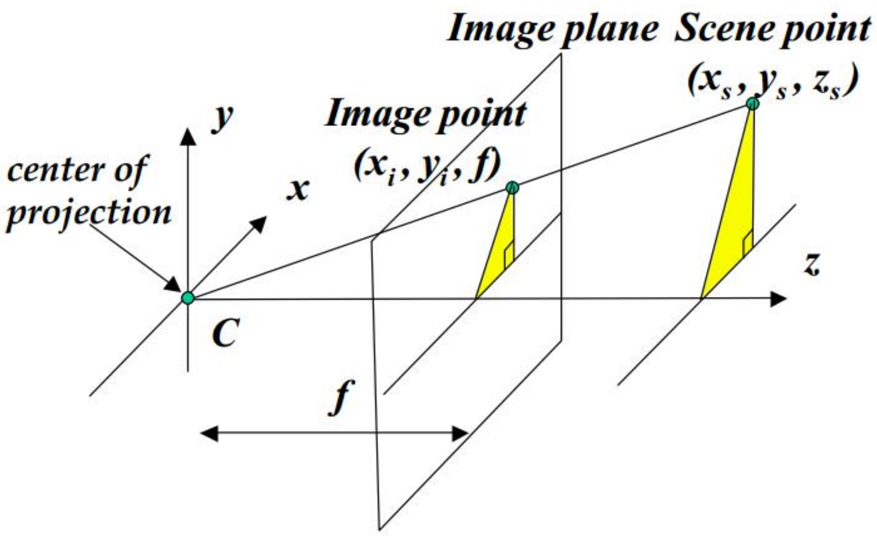


Figure 2-9 Cental projection[39]

($x_p ix$, $y_p ix$), its coordinate origin is the pixel at the lower left corner of the image, and the scaling between the pixel and the cental projection coordinate of width and height is k_x and k_y separately. Their relationship is shown in Figure 2-10.

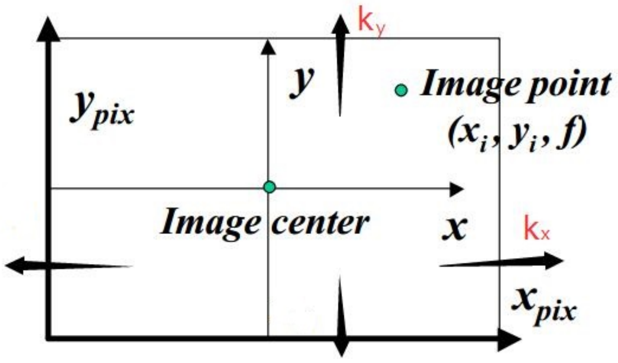


Figure 2-10 Transformation from pixel to image coordinate[42]

Obviously, this transformation can be written as

$$x_{pix} = k_x x_i + x_0 \qquad y_{pix} = k_y y_i + y_0 \tag{2-16}$$

Combine (2-15) with (2-16), there is

$$x_{pix} = f k_x \frac{x_s + z_s x_0}{z_s} \qquad y_{pix} = f k_y \frac{y_s + z_s y_0}{z_s} \tag{2-17}$$

let

$$\alpha_x = f k_x \qquad \alpha_y = f k_y \tag{2-18}$$

and

$$x_{pix} = \frac{u}{w} \qquad y_{pix} = \frac{v}{w} \tag{2-19}$$

written as the matrix form:

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \tag{2-20}$$

Add the skew factor s into this transformation,

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \tag{2-21}$$

then we get the calibration matrix K:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-22}$$

In our project, we don't need to write a code to calibrate the camera, but just use the cameracalibrator.py node of the camera_calibration package in ROS[43]. The camera_calibration's cameracalibrator.py node is a convenient tool to calibrate a monocular camera with a raw image over ROS, with which we need a large checkerboard with known size and number of its squares and th e camera output raw image through ROS. In our calibration, we use a 8*6 squares checkerboard, and we should move it to make it lie on different positions (as the Figure 2-11 shown) of the view. Finally, it will output the internal parameters of camera K and some other calibration parameters to us.

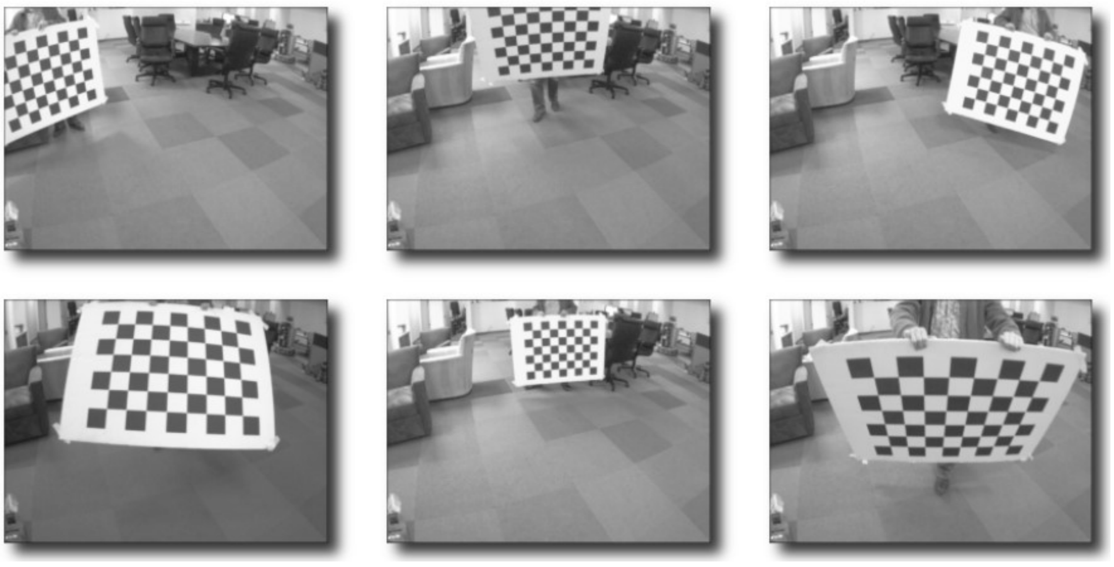


Figure 2-11 Checkerboard lies on the camera's left and right, top and bottom of field of view(the 1-3th picture); toward/away and tilt from the camera(the 4-5th picture); checkerboard filling the whole field of view(the 6th picture) [43]

My camera calibration parameters is as follows:

$$D = [0.047492863768895464,\ -0.1038925599268795,\ 0.01385209245928748,\ -7.160814536273587e - 05,\ 0.0]$$
$$K = [545.9860520978382,\ 0.0,\ 314.44397065234756;\ 0.0,\ 546.773154836032,\ 259.894806022969;\ 0.0,\ 0.0,\ 1.0]$$

Generally speaking, we need to use D to correct imaging distortion in code. However, we needn't due to the node will outputs another topic for calibrated images, which is a image message, in which the images distortion has been corrected. And we can use this topic directly and needn't use the D vector in code.

2.5.2 The Essential Matrix

Essential matrix is the special form of the fundamental matrix under the case of which is calculated in the normalized image coordinate form. In the history of the development of computer vision, essential matrix was presented (by Longuet-Higgins[44] in 1981) earlier than the fundamental matrix (by QT Luong[45]). Fundamental matrix shouldn't need to consider the calibration of the image, which is a generalization of the essential matrix. Compared with the fundamental matrix, the essential matrix has fewer degrees of freedom and more properties. Its properties will be given in the following description (the properties of the fundamental matrix have been described in the previous parts).

To introduce the essential matrix, I will describe what normalized coordinates is first. Consider a camera matrix $P = [R|t]$, and $x = PX$ is a point on the image. If the calibration matrix K of this image is known, then we can make the x normalized by premultiplying the inverse of K to x, there is $\hat{x} = K^{-1}x$. Then we obtain $\hat{x} = [R|t]X$, where $\hat{x}$ is the image point x representing in the normalized coordinate form. This expression can also be seen as $\hat{x}$ is the point on the image imaging with the camera matrix $[R|t]$ corresponding to the space point X. That is to say, the calibration matrix of it is a 3×3 identity matrix. The camera matrix $\hat{P} = K^{-1}P = [R|t]$ is called normalized camera matrix, in which the necessity of the calibration matrix K has been removed.

Let's support that there is a pair of normalized camera matrices $P = [I|0]$, $P' = [R|t]$, the fundamental matrix corresponds to which is denoted as essential matrix E. The form of the essential matrix is

$$E = [t]_\times R = R[R^T t]_\times \tag{2-23}$$

According to the formula(2-1), the necessary and sufficient conditions of essential matrix can be expressed as

$$\hat{x'}^T E\hat{x} = 0 \tag{2-24}$$

which is also the expression of the mapping between the corresponding points on the two images in normalized coordinates. Substitute $\hat{x} = K^{-1}x$ to the formula (2-24), then we have

$$x'^T K'^{-T} E K^{-1} x = 0 \tag{2-25}$$

Compared the formula (2-25) with (2-1), it's easy to derive the relationship between the essential matrix E and the fundamental matrix F is

$$E = K'^T F K \tag{2-26}$$

Essential matrix only has five degrees of freedom: the rotation matrix R and translation vector t both have three degrees of freedom, but there is a scale ambiguity of it, and it is a matrix with homogeneous quantity. The reducing degree of freedom is an additional condition the essential matrix should satisfy, compared to the fundamental matrix.

A 3×3 matrix can be an essential matrix if and only if the two values of its singular value decomposition are equal, and the third value is zero, the proof of which can be found in [38].

Therefore, in the SVD expression $E = Udiag(1,1,0)V^T$, both U and V have three degrees of freedom. However, since two values of its singular values are equal, the SVD of E is a family of one-parameter SVD family, in the other word, which is not unique. Thus, the singular value decomposition of the essential matrix can be written as

$$E = (Udiag(R_{2\times2},1))diag(1,1,0)(diag(R^T_{2\times2},1))V^T \tag{2-27}$$

where $R_{2\times2}$ can be any 2×2 rotation matrix.

2.5.3 Recovering R and t from E

When we get the essential matrix E (calculated according to the five-point algorithm[46, 47]), the camera matrix (containing the rotation matrix and translation vector) can be extracted from the essential matrix. The camera matrix extracted from the essential has four possible solutions, with a scaling ambiguity. Among them, exactly only one camera matrix is correct, this will be mentioned later. Let us assume that the first camera matrix is P = [I — 0], in order to calculate the second camera matrix, let E be decomposed into the product of a skew-symmetric matrix S and a rotation matrix R. Support that the SVD of E is $E = Udiag(1,1,0)V^T$, which can be decompose in two possibilities (including two probable R):

$$S = UZU^T \quad R = UWV^T \ \ or \ \ R = UW^TV^T \tag{2-28}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2-29}$$

It is easy to check that both of these two decompositions are true, and there is no another disintegration except them.

Obviously, the decomposition (2-28) of E determines the t in the camera matrix does not contain scales. Compare $E = SR$ with (2-23), we get $S = [t]_\times$, and the Frobenius norm of S is $\sqrt{2}$, that the scaling factor included in S is $||t|| = 1$, which is simply normalized with the baseline between two camera matrix. Since $St = 0$(according to the property of cross product), the translation vector can be obtained as

$$t = \pm U(0,0,1)^T = \pm u_3 \tag{2-30}$$

where u_3 is the last column of the vector U. But the translation vector still can't be determined exactly due to the fact that the sign of the essential matrix can not be determined.

In summary, for a given essential matrix E we can solve four possible camera matrix P ′, which contains two different R solution and two different symbols for t. Formally, for a given essential matrix $E = Udiag(0,0,1)V^T$, assume the first camera matrix is $P = [I|0]$, there four possible solutions for the second camera matrix P':

$$\begin{cases} P = [UWV^T|u_3] \\ P = [UWV^T|-u_3] \\ P = [UW^TV^T|u_3] \\ P = [UW^TV^T|-u_3] \end{cases} \tag{2-31}$$

Obviously, in (2-31) the difference between the first two solutions is just the opposite direction of the translation. And the relationship between the first and the third solution is shown below:

$$[UW^TV^T|u_3] = [UWV^T|u_3]\begin{bmatrix} VW^TW^TV^T & \\ & 1 \end{bmatrix} \tag{2-32}$$

where $VW^TW^TV^T = Vdiag(-1,-1,1)V^T$ is a 180 degree rotation regarding the baseline between two camera centers.

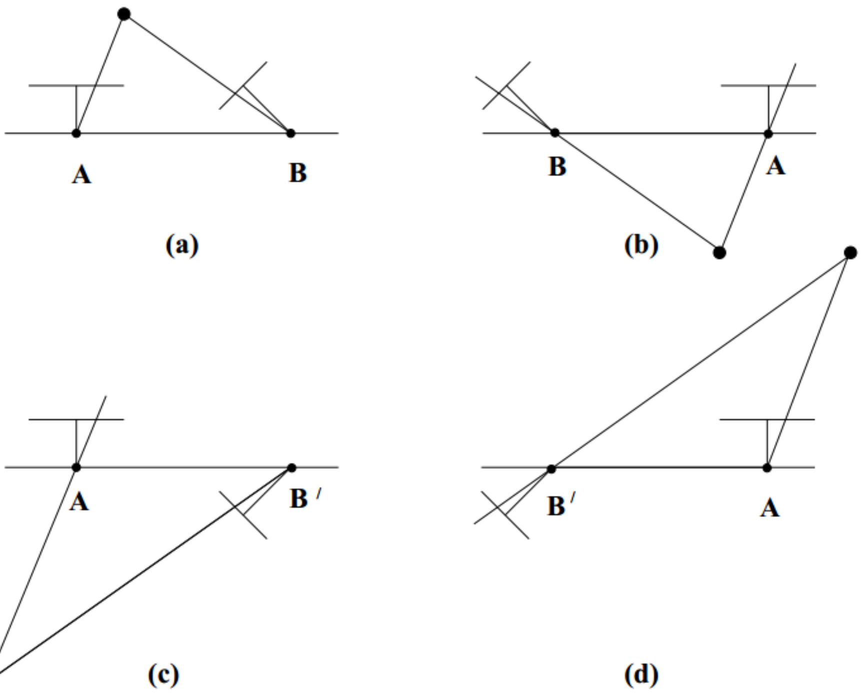


Figure 2-12 The four solutions for the second camera matrix P'[38]

Imaging significances of these four solutions represented are shown in Figure 2-12: the space point X position reconstructed corresponding to the four camera matrix solutions respectively. Only the reconstruction of point X in the Figure (a) is located in front of two cameras (point A, B represents the two camera centers), in Figure (d) the reconstruction of the X point behind two cameras, and in the corresponding figures at up and down made a 180 degrees rotation from each other. Therefore, only the Figure (a) is the correct solution for camera matrix P'.

2.5.4 Chapter Summary

In the first part of this chapter, the program running plat ROS and how to get the images from the camera by Rviz over ROS are introduced, which actually the tool for capturing frames. And then we introduce the SiftGPU, before which it introduces something about features extracting and matching and the overview of SIFT feature and GPU. It do the composition between SIFT feature and other feature detectors and descriptors.

As we known from the first chapter, the five-point algorithm is a method to solve the essential matrix between views with the corresponding image points, matching features. This section then gives a detailed description of epipolar geometry, in which a variety of mathematical relationships will be derived, and then a highlight introduction of fundamental matrix and essential matrix, both of which can be derived from the epipolar geometry. Especially, the essential matrix includes the transform information of the camera.

3 Five-point Algorithm with RANSAC

This chapter is an introduction to the five-point algorithm and RANSAC. The most important mainly algorithm in this project is five point algorithm, use to calculate the essential matrix, that estimate the relative poses transformation through the corresponding points between the images of the camera. Because in the previous step matching feature points is not necessarily the real match points, even with the real match points to calculated the essential matrix there is a certain probability of error. As a result, simply calculating essential matrix E with five-point algorithm is substantially not qualified. The RANSAC algorithm was used here as an optimization algorithm of the five-point algorithm to improve the probability of getting correct essential matrix E.

3.1 Five-point Algorithm

3.1.1 Basic Knowledge

If the pair of images have been calibrated, their camera matrices is $P = [I|0]$ and $P' = [R|t]$. The essential matrix corresponds to this pair of camera matrices is

$$E = [t]_\times R \tag{3-33}$$

where

$$[t]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{3-34}$$

Therefore, the form of the fundamental matrix is

$$F = K_2^{-T}[t]_\times R K_1^{-1} \tag{3-35}$$

According to the epipolar constraint, the fundamental matrix should satisfy the condition:

$$x'^T F x = 0 \tag{3-36}$$

A rank 2 real non-zero 3×3 matrix F must satisfy the following cubic singularity condition[48], which can be a fundamental matrix:

$$det(F) = 0 \tag{3-37}$$

For the calibrated image points (x,x′), assuming they have premultiplied the inverse of K, the essential matrix E should satisfy the epipolar constraint:

$$x'^T E x = 0 \tag{3-38}$$

Compared with the fundamental matrix F, the essential matrix E should satisfy the additional condition, that its two non-zero singular values are equal, and the third singular value is zero. It gives the following equation[48] that the essential matrix must satisfy:

$$EE^T E - \frac{1}{2} trace(EE^T)E = 0 \tag{3-39}$$

3.1.2 The Method of Calculating the Essential Matrix

The most basic condition in this algorithm (3-38), the epipolar constraint, can be written as

$$\widetilde{q}^T \widetilde{E} = 0 \tag{3-40}$$

where (assume the image points (in homogeneous coordinates) are: $x = (x_1, y_1, z_1)^T$, $x' = (x_2, y_2, z_2)^T$)

$$\widetilde{q} = [x_1x_2 \quad x_2y_1 \quad z_1x_2 \quad x_1y_2 \quad y_1y_2 \quad z_1y_2 \quad x_1z_2 \quad y_1z_2 \quad z_1z_2]^T \tag{3-41}$$

$$\widetilde{E} = [E_{11} \quad E_{12} \quad E_{13} \quad E_{21} \quad E_{22} \quad E_{23} \quad E_{31} \quad E_{32} \quad E_{33}]^T \tag{3-42}$$

Compute $\widetilde{q}$ (3-41) for all pairs of image points (assume there are N pairs), we get a $N \times 9$ matrix Q. Assume there are four nine-dimensional vector$\widetilde{X}, \widetilde{Y}, \widetilde{Z}, \widetilde{W}$, which make up the right null space of the matrix Q. Since they correspond to the smallest singular values of the matrix Q, they can be computed by doing SVD decomposition. Then reshape them as four 3×3 matrix respectively: X, Y, Z, W. The expression of E, expressed by these four matrix, can be obtained from (3-40):

$$E = xX + yY + zZ + wW \tag{3-43}$$

where x, y, z, w are some scalars. This expression is derived as follows.

Since the four vector $\widetilde{X}, \widetilde{Y}, \widetilde{Z}, \widetilde{W}$ span the right null-space of the matrix Q, this relationship can be expressed as

$$\widetilde{Q}(x\widetilde{X} + y\widetilde{Y} + z\widetilde{Z} + w\widetilde{W}) = 0 \tag{3-44}$$

where x, y, z, w are some scalars. If $\widetilde{E}$ satisfies the constraint (3-40), according to the knowledge of SVD, we obtain

$$\widetilde{E} = x\widetilde{X} + y\widetilde{Y} + z\widetilde{Z} + w\widetilde{W} \tag{3-45}$$

where $\widetilde{E}$ is a nine-dimensional vector. Reshape $\widetilde{E}$ as a 3×3 matrix E, correspondingly, the formula (3-45) can be written as (3-43) $E = xX + yY + zZ + wW$ as required.

The equation (3-39) gives nine equations on the elements of E, but of which only two are independence on algebraic. Substituted E, expression (3-43), into the condition (3-37) can obtain another equation. Then we have enough equations to solve the essential matrix E.

Since these four coefficients in the expression (3-43) can be defined on any arbitrary scale, in order to facilitate the calculation, let $w = 1$. The nine equations from condition (3-39) gives form a coefficient matrix of 9×20, which corresponds to 20 terms (written as a vector space of the monomials):

$$[x^3, y^3, z^3, x^2y, x^2z, x^2, xy^2, y^2z, y^2, xyz, xy, xz^2, xz, x, yz^2, yz, y, z^2, 1] \tag{3-46}$$

In the David Nistèr's 5-point algorithm, Nistèr use Gauss - Jordan elimination method for the coefficient matrix of the equations system and form another coefficient matrix A (an upper triangular matrix). Then he performed some algebra operation on the matrix A and obtained two 4×4 (here we use CVPR version, because CVPR version is easier to understand, in the PAMI version it obtains a 3 * 3 matrix). Then he do a further elimination to give a ten-degree polynomial. Finally, solve the roots of z (ten roots). After that another unknowns x and y can be solved by back substitution method.

Gauss - Jordan elimination is really a complex method to solve the variables. The code referencing to a more simple method[47], which is based on hidden-variable resultant[49] but not on Gauss - Jordan elimination such a cumbersome method, to solve these three variables.

Hidden-variable resultant is a well-known algebraic elimination method, and easy to implement. It can easily and quickly eliminate the variable from polynomial equations. Its main principle is as follows. Given a equations system composed of M homogeneous polynomial equations (in which every homogeneous polynomial equation $p_i = 0$, for i = 1 ... M), containing N variables $(x_1, x_2, ..., x_N)$, we can see one of the variables as a parameter (i.e. x_1). Then the equation system can be rewritten as

$$C(x_1)X = 0 \tag{3-47}$$

where $C(x_1)$ is the coefficient matrix of X. X is the vector space only concluding the terms of the other $N-1$ variables. And the coefficients in the matrix $C(x_1)$ includes a parameter x_1.

If in the rewritten equation system, the number of equations is equal to the number of the terms in the X (i.e. $C(x_1)$ is a square matrix), then if and only if the determinant of $C(x_1)$ is zero the equation systems has non-trivial solutions. The condition gives a equation

$$det(C(x_1)) = 0 \tag{3-48}$$

By solving the equation (3-48), we can get the roots of the variable x_1, that can be seen as we eliminate the other $N-1$ variables at once.

We can apply this method on solving the variables x, y, z[47]. We can obtain 9 equations from condition (3-39). Adding another equation obtained from condition (3-37), there are totally 10 equations in the equation system. Let's make the variable z as a parameter, the equations system can be rewritten as

$$C(z)X(x,y) = 0 \tag{3-49}$$

where $X(x,y)$ is the vector space consisting of the homogeneous monomial terms of variables x and y

$$[x^3, y^3, x^2y, x^2, xy^2, y^2, xy, x, y, 1] \tag{3-50}$$

Obviously, the rewritten equations only has 10 terms, then the number of equations equal to the number of terms in X. For making this equation system have non-trivial solutions, it must satisfy the equation

$$det(C(z)) = 0. \tag{3-51}$$

The equation (3-51) is an univariate polynomial, which only consist the variable z. Therefore, we can easily solve the roots of z by solving this equation. Obviously, there are only at most 10 roots for z, so the five-point problems can have at most 10 solutions.

Next, we can use the substitution method to obtain the rest of the unknowns. But the substitution step will be done for many times, due to the fact that there are many z solutions, that's why it is somewhat cumbersome. We can use a more convenient way to get other variables, that is doing SVD decomposition for $C(z)$. The vector space $X(x,y)$ has included all the combinations of variables x and y up to degree three. Thus, from the equation (3-49), it's obviously that the solutions of x and y are the right null-space of $C(z)$. In the other words the solutions of x and y can be conveniently computed by doing a SVD decomposition for $C(z)$.

After computing the coefficients x, y, z, combining with the vectors X, Y, Z, W known before, we can obtain the solutions of the essential matrix E. Finally, we can recover the motion information, rotation matrix R and the translation vector t, from the essential matrix E.

3.1.3 The Flowchart of the Five-point Algorithm

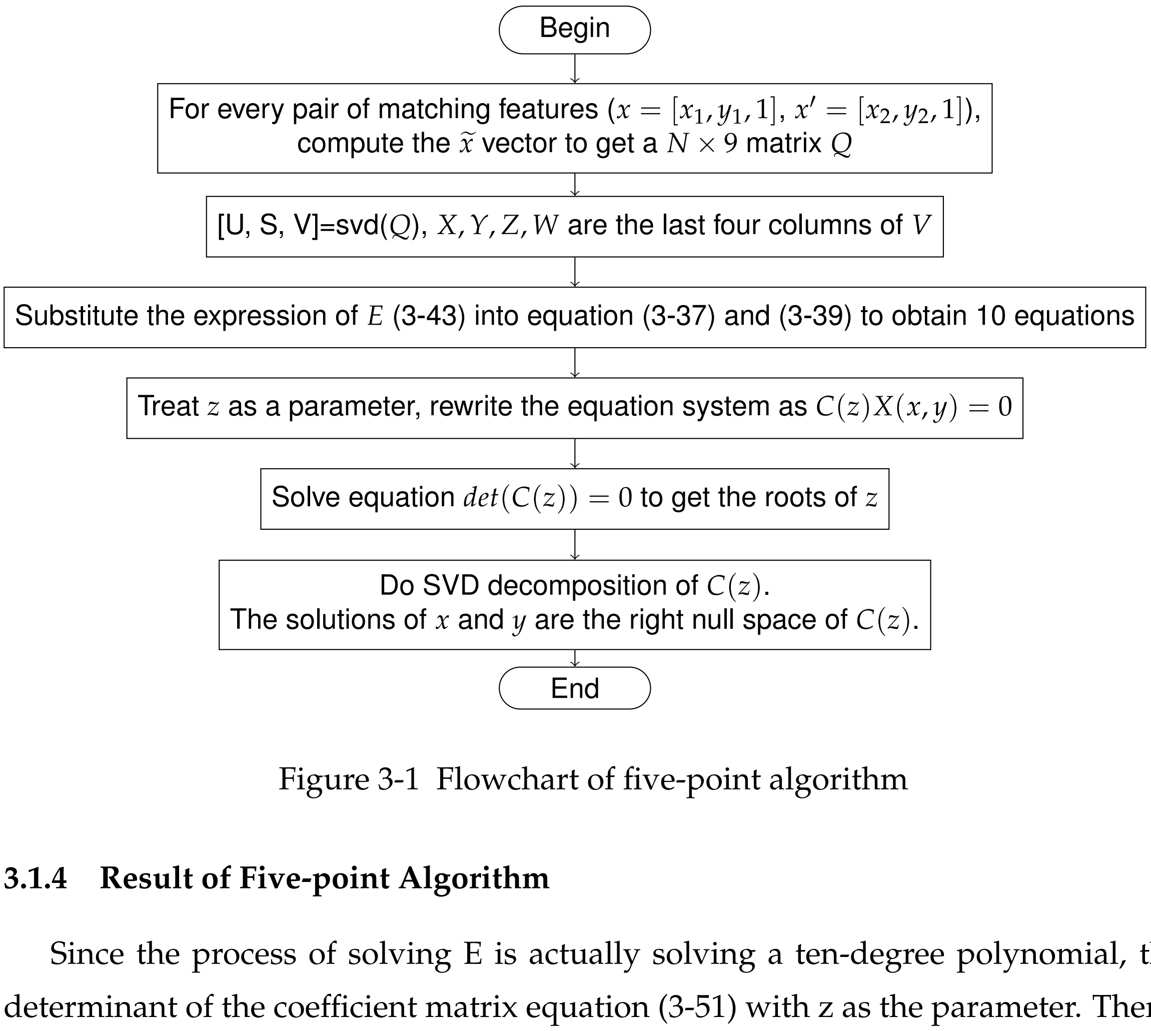
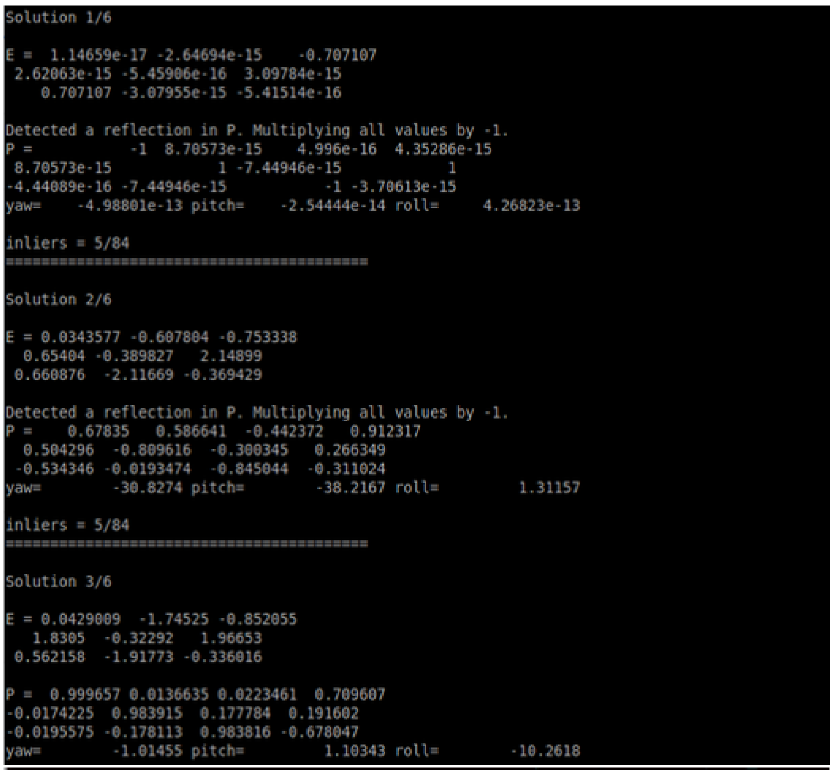

Begin
For every pair of matching features ($x = [x_1, y_1, 1]$, $x' = [x_2, y_2, 1]$), compute the $\tilde{x}$ vector to get a $N \times 9$ matrix Q
[U, S, V]=svd(Q), X, Y, Z, W are the last four columns of V
Substitute the expression of E (3-43) into equation (3-37) and (3-39) to obtain 10 equations
Treat z as a parameter, rewrite the equation system as $C(z)X(x,y) = 0$
Solve equation $det(C(z)) = 0$ to get the roots of z
Do SVD decomposition of $C(z)$. The solutions of x and y are the right null space of $C(z)$.
End

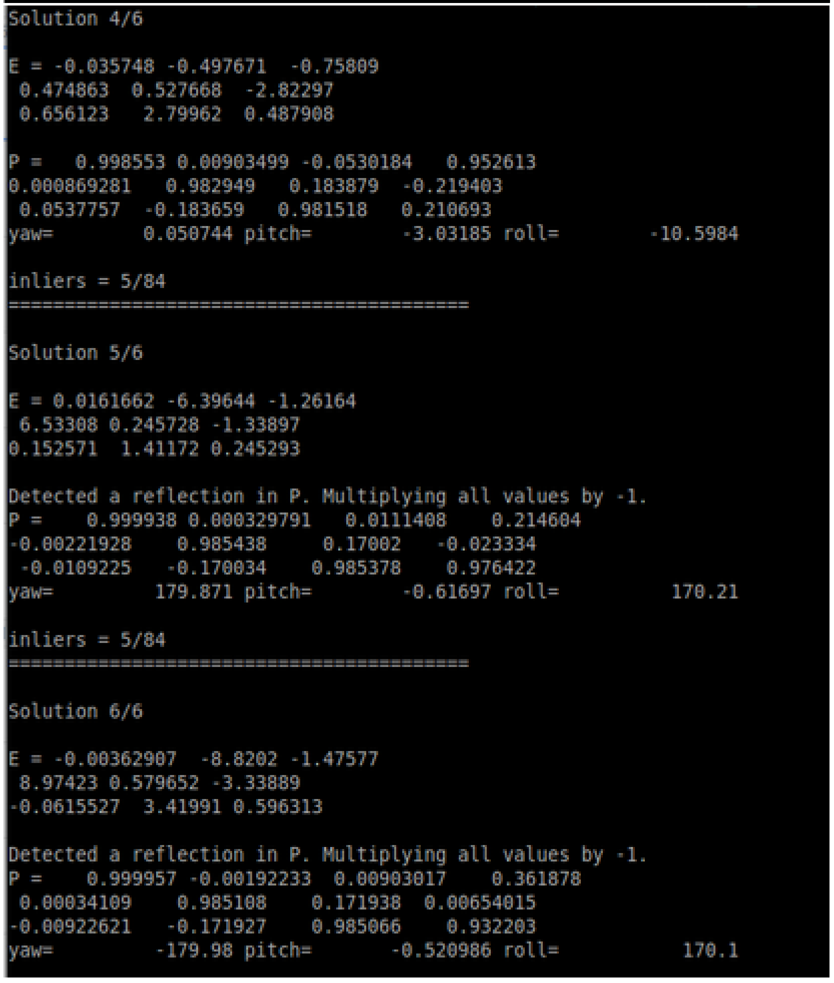
Figure 3-1 Flowchart of five-point algorithm

3.1.4 Result of Five-point Algorithm

Since the process of solving E is actually solving a ten-degree polynomial, the determinant of the coefficient matrix equation (3-51) with z as the parameter. Therefore, the roots of the E have wrong solutions (these solutions are not true represents of the camera pose). Here the results of a single five-point calculation with two fixed images (corresponding to Roll movement) are shown.



```
Solution 1/6

E =  1.14659e-17 -2.64694e-15    -0.707107
 2.62063e-15 -5.45906e-16  3.09784e-15
    0.707107 -3.07955e-15 -5.41514e-16

Detected a reflection in P. Multiplying all values by -1.
P =           -1  8.70573e-15    4.996e-16  4.35286e-15
 8.70573e-15            1 -7.44946e-15            1
-4.44089e-16 -7.44946e-15           -1 -3.70613e-15
yaw=    -4.98801e-13 pitch=    -2.54444e-14 roll=     4.26823e-13

inliers = 5/84
=========================================

Solution 2/6

E = 0.0343577 -0.607804 -0.753338
  0.65404 -0.389827   2.14899
 0.660876  -2.11669 -0.369429

Detected a reflection in P. Multiplying all values by -1.
P =    0.67835   0.586641  -0.442372   0.912317
  0.504296  -0.809616  -0.300345   0.266349
 -0.534346 -0.0193474  -0.845044  -0.311024
yaw=        -30.8274 pitch=        -38.2167 roll=         1.31157

inliers = 5/84
=========================================

Solution 3/6

E = 0.0429009  -1.74525 -0.852055
   1.8305  -0.32292   1.96653
 0.562158  -1.91773 -0.336016

P =  0.999657 0.0136635 0.0223461  0.709607
-0.0174225  0.983915  0.177784  0.191602
-0.0195575 -0.178113  0.983816 -0.678047
yaw=        -1.01455 pitch=         1.10343 roll=        -10.2618
```



```
Solution 4/6

E = -0.035748 -0.497671  -0.75809
 0.474863  0.527668  -2.82297
 0.656123   2.79962  0.487908

P =    0.998553 0.00903499 -0.0530184   0.952613
0.000869281   0.982949   0.183879  -0.219403
 0.0537757  -0.183659   0.981518   0.210693
yaw=         0.050744 pitch=        -3.03185 roll=        -10.5984

inliers = 5/84
=========================================

Solution 5/6

E = 0.0161662 -6.39644 -1.26164
 6.53308 0.245728 -1.33897
0.152571  1.41172 0.245293

Detected a reflection in P. Multiplying all values by -1.
P =    0.999938 0.000329791    0.0111408    0.214604
-0.00221928    0.985438     0.17002    -0.023334
 -0.0109225   -0.170034    0.985378    0.976422
yaw=          179.871 pitch=        -0.61697 roll=          170.21

inliers = 5/84
=========================================

Solution 6/6

E = -0.00362907  -8.8202 -1.47577
 8.97423 0.579652 -3.33889
-0.0615527  3.41991 0.596313

Detected a reflection in P. Multiplying all values by -1.
P =    0.999957 -0.00192233  0.00903017    0.361878
 0.00034109    0.985108    0.171938  0.00654015
-0.00922621   -0.171927    0.985066    0.932203
yaw=          -179.98 pitch=       -0.520986 roll=           170.1
```

Figure 3-2 The results of a single five-point calculation with two fixed images.

Obviously, only the fourth solution is correct, the rest are spurious solutions, which happen in the case that the inputs are well enough, or all the solutions should be wrong. Visible, use a single five-point algorithm to obtain the correct essential matrix is unrealistic, so we use RANSAC to exclude spurious solutions.

3.2 RANSAC

3.2.1 Overview of RANSAC

Though we got match features from SiftGPU, they are not all the exact matches. That's say, there are some outliers. To solve the problem, we can use RANdom SAmple Consensus (RANSAC) algorithm, which can help us reject outliers[50] in visual odometry computation well.

RANSAC is short for RANdom SAmple Consensus[51]. It is an algorithm to compute the math model according to a dataset including outliers, and obtaining the effective sample data (inliers). It was first proposed by the Fischler and Bolles[52] in 1981. The RANSAC algorithm is often applied in robotics, for instance, extracting line from 2D image, extracting plane from 3D structure, and extracting structure from motion.

The basic assumption of the RANSAC algorithm is that a sample contains the correct data (inliers, can be described by the data model), but also contains abnormal data (outliers, deviation from the normal range very far, unable to adapt to the mathematical model of the data), that the dataset contains noise. And it also assumes that there exists a method that can calculate the model parameters which are consistent with this data. RANSAC is an iterative, non-deterministic algorithm. It estimates the model parameters by iterations. As far as to its non-determinacy, it gets different model in each iteration, and it has a possibility to get a reasonable model. For getting a good enough model, the number of iterations should be improved.

RANSAC is a learning algorithm, it gets model parameters through sampling points randomly. Given a dataset, which including inliers and outliers, RANSAC use vote scheme to find the optimal model. To put it simply, the elements in the data set vote for the models. There are two assumptions about the vote scheme. First, any outlier is impossible to vote for a single model all along. Second, there are enough features to fit a good model.

3.2.2 Fundamental of RANSAC

The RANSAC algorithm is given a set of observed data, the method to obtain a fitting model, and some reasonable parameters of the algorithm as its inputs. Then RANSAC repeats the steps as follows to get enough inliers in consensus set:

a) Choose minimal points from the observed data set to compose a subset, called hypothetical inliers.

b) Fit a model from the subset(compute its model parameters).

c) Test each point in the observed values data set to see whether they are consistent to the fitting model we got in step one. When the error between the testing point and the model is less than the threshold we set, add the point as an inlier of this model. If a point does not fit the fitting model and away from it more than a certain threshold we setted, it will be regard as a outliers, belong to noise.

d) We get a set of inliers from the fitting model, which was called ″consensus set″. If there are enough points in the consensus set, the model will be regarded as a good model.

e) Then we use all points in consensus set to compute a better model.

The algorithm repeats the steps above until there are enough inliers in the consensus set in certain iteration. In every iteration, a model with too less inliers will be throw out, or it can be insteaded by the model with more inliers in other iterations, which make us get the best model.

We can get a Matlab implementation of some kinds of model fitting and robust estimation using RANSAC algorithm from [53]. And some C++ implementations code also can be found in [54] and [55].

3.2.3 Parameters

Some important parameters we should set at the code beginning is as follows[51].

K is the maximum number of iterations. It can be caculated in section below (Algorithm Complexity Analysis of RANSAC),which is determined by how high the possibility we want to find the best model, the number of elements in the input dataset, and the number of inliers for the best model.

N is the minimum number of points to fit the model required.It is decided by what model you want to fit. For fitting a 2D line, we need at least 2 points. Therefore, for this example, N is 2. In this thesis, we need to use 5-point algorithm to fit a essential matrix, thus, N equal to 5 here.

T is the threshold measuring if the point fit the model. If the error between the point and the model less than T, the point can be add as a inliers to consensus set.

D is a threshold to decide whether a model is a good model. If the number of inliers in consensus set reachs the threshold, the model will be regarded as a good model. It also can be a rate of the numbers of inliers and all points.

The inputs and outputs of the algorithm is listed as follow:

Input:

InputData (A set of observed datas(points)),

K, N, T, D,

FittingFunction (A function can fitting a model with least points).

Output:

BestModel (Finally RANSAC returns a best model (it is possible to return NULL if it can not find any good model)).

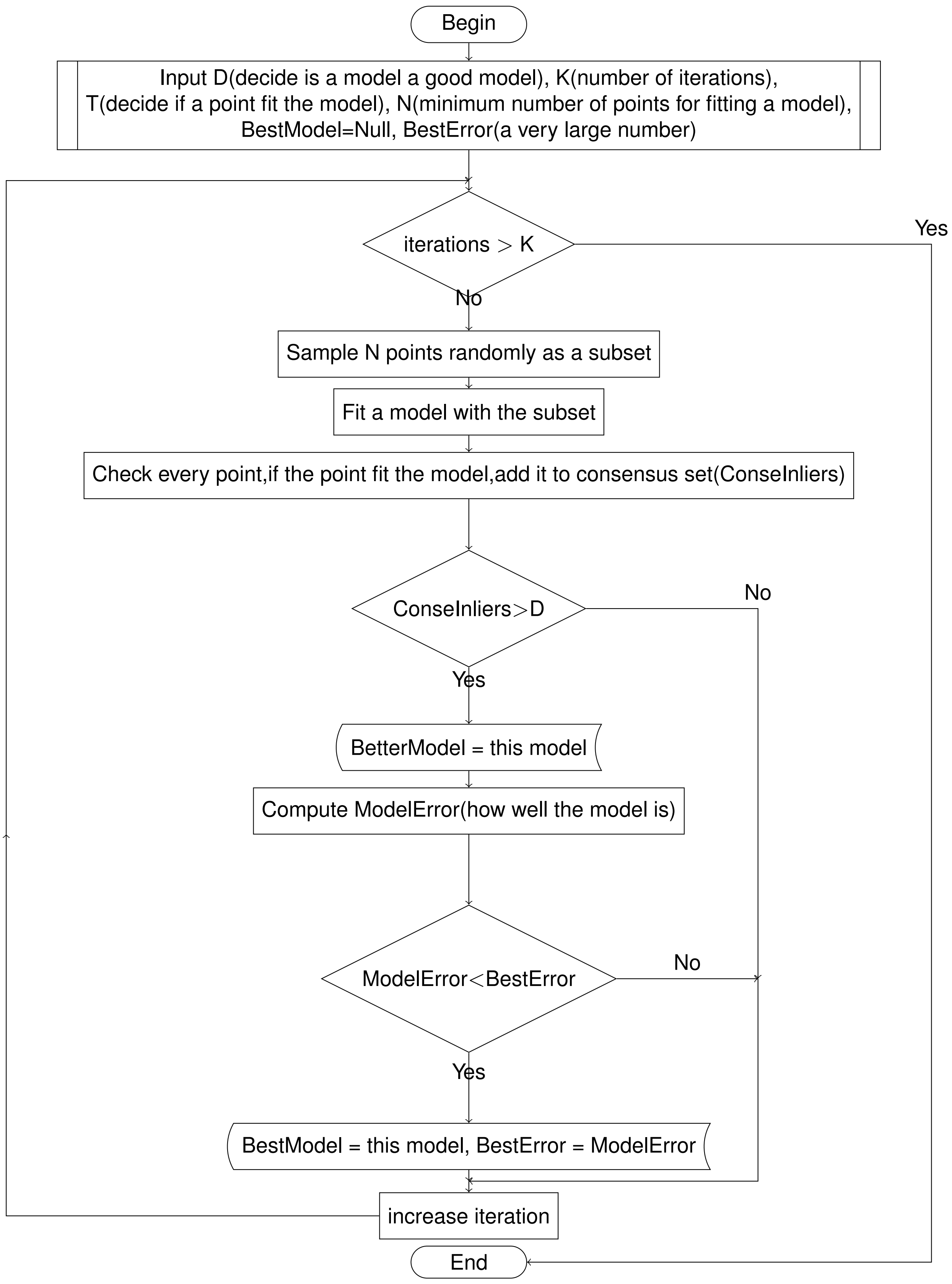

Begin
Input D(decide is a model a good model), K(number of iterations), T(decide if a point fit the model), N(minimum number of points for fitting a model), BestModel=Null, BestError(a very large number)
iterations > K
Yes
No
Sample N points randomly as a subset
Fit a model with the subset
Check every point,if the point fit the model,add it to consensus set(ConseInliers)
ConseInliers>D
No
Yes
BetterModel = this model
Compute ModelError(how well the model is)
ModelError<BestError
No
Yes
BestModel = this model, BestError = ModelError
increase iteration
End

Figure 3-3 Overall flowchart of RANSAC

3.2.4 Algorithm Complexity Analysis of RANSAC

In the RANSAC algorithm, the first layer of loop is K times iterations for all steps, besides initialization. Therefore, how large the K is really effecting how long the program runs. K can be decided according to our conputation.

When we get a dataset of observed data point, We can estimate the proportion of the inliers in this set based on experience. Assuming that there are $n0$ inliers in this input dataset, and n points in the dataset in total. Let W be the probability to choose an inlier in the dataset each time select a single point: $W = n0/n$.

In each iteration, we need to select N points to fit a model. Obviously, the possiblility W_N that we select N points and all of them are inliers is W_N=W^N. It is easy to infer, the probability of we get at least one outlier from these selected N points is $1 - W^N$. When this happened, the model fitting function will fit a bad model, and it is hard to get a good model in this iteration, not to mention a best model. If we get at least one outliers from N selected points in all K iterations, there is no doubt that the output of the algorithm, BestModel, will be null. The possibility to happen this thing is

$$P = (1 - W^N)^K \tag{3-52}$$

Therefore, the possibility of getting a good model is $1 - P$.

In summary, a reasonable K can be computed below:

$$K = \frac{log(P)}{log(1 - W^N)} \tag{3-53}$$

It's really not a large number. As to my RANSAC for fitting a essential matrix, $N = 5$, it has a 99 percent probability of the model can be obtained($P = 0.01$), and, according to experiment, there are about 60 percent of the input match features are inliers($W = 0.5$). Then K can be computed:

$$K = \frac{log(0.01)}{log(1 - 0.5^5)} = (int)145.05072 = 146$$

where the $K = 146$ will be used in the code, we can seen from the experimental (in the section 4.2) that these group of parameters are good enough in most cases.

As was described above, the parameter K can be decided by the inliers number and the possibility of getting good results, both of which are decided by experiment.

3.3 Five-point Algorithm with RANSAC

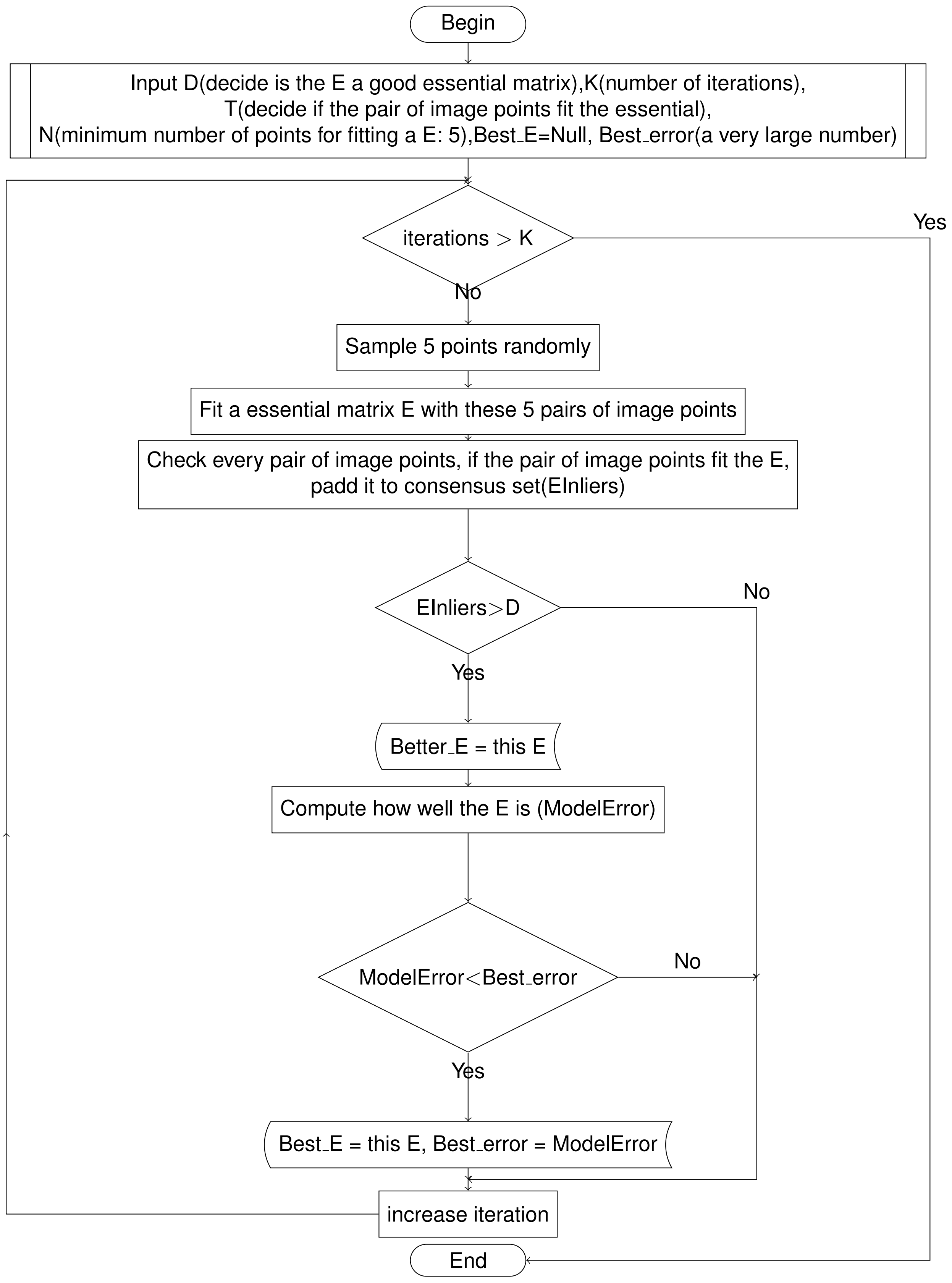

Begin
Input D(decide is the E a good essential matrix),K(number of iterations),
T(decide if the pair of image points fit the essential),
N(minimum number of points for fitting a E: 5),Best_E=Null, Best_error(a very large number)
iterations > K
Yes
No
Sample 5 points randomly
Fit a essential matrix E with these 5 pairs of image points
Check every pair of image points, if the pair of image points fit the E,
padd it to consensus set(EInliers)
EInliers>D
No
Yes
Better_E = this E
Compute how well the E is (ModelError)
ModelError<Best_error
No
Yes
Best_E = this E, Best_error = ModelError
increase iteration
End

Figure 3-4 Overall flowchart of five-point algorithm with RANSAC

3.3.1 Description of the Algorithm

In the previous two subsections, respectively, we described the five-point algorithm and RANSAC algorithm. In our project, we use the five-point algorithm with RANSAC algorithm. In the description we know, RANSAC requires some algorithm to build the model (get the model parameters). This model is in fact to find the essential matrix E, and we use five-point algorithm to find the matrix E. The algorithm requires the minimum number of points is five. So in each iteration of the algorithm, we need a minimum of sampling points is five. Then use the five-point algorithm to calculate the essential matrix E, and then use the E to test each pair of image points. For a pair of corresponding points, there are two method to check the inliers in my code. First is using the epipolar constraint ($x'^T Ex$). If the differential of $x'^T Ex$ and 0 less than a threshold T, then the points are considered as a inlier for this essential matrix. Second is using the Sampson distance[56]:

$$d = \sum_{i=1}^{N} \frac{(x_i'^T E x_i)^2}{(Ex_i)_1^2 + (Ex_i)_2^2 + (E^T x_i')_1^2 + (E^T x_i')_2^2} \tag{3-54}$$

Then we add all the well-fitting points of this essential matrix to the consensus set. If the number of points in the consensus set is bigger than a certain threshold D, this essential matrix can be seen as a good solution. During this process, we need to calculate how well the image points fit the essential matrix E. When the deviation of the points and the fitting model is less than the *Best_error* (the initial value of this variable is a very large number), we give the E to the *Best_E* variable. After all times of iteration, the finial *Best_E* is the best essential matrix.

3.4 Chapter Summary

My code is mainly according to this chapter. First it introduce the five-point algorithm, then the RANSAC, which is a outliers rejecting algorithm. As a combine, it introduces how to use five-point algorithm with RANSAC to select the inliers among the matching features to calculate the essential matrix finally. After that, it tells how to recover rotation and translation information from the essential matrix.

4 Experimtal Setup and Results

4.1 Experimental Steps for Each Part

For testing the code, it uses the fixed images to collect data, in which all groups (two images as a group) of image include the following relative motions:

a) Pure rotation: yaw.

b) Pure rotation: pitch.

c) Pure rotation: roll.

In addition, do experimental of the algorithm under different configuration parameters, and compare the calculate speed (time) and the correct rate.

For each set of relative movement (with fixed image), do the following steps respectively:

1) Capture two images and save them as the input when the camera do the designated motion.

2) Do feature extraction and matching for these two images.

3) Use five-point algorithm with RANSAC (with the set of parameters having higher correct rate) to calculate the essential matrix between two images, then extracts the rotation matrix R and translation vector t.

4) Use five-point algorithm implemented by OpenCV to calculate the essential matrix between two images, then extracts the rotation matrix R and translation vector t.

5) Draw the inliers and outliers of matches outputted by my code and OpenCV respectively.

After completing the above experiments, pick a group of fixed images as input, change RANSAC algorithm parameters for several times, and record the correct rate and the time of the calculation process.

4.2 The Verification Experiment with Different Groups of Relative Motion

Now, let's begin the first experiment, verify the code with different relative motions.

In this testing experiment, the parameters of the RANSAC algorithm are set as follows:
T=1e-04(To test if the pair of image point fit the essential matrix),
D=0.5*Match_Num (If there are 0.5 image points fit the essential matrix, which can be seen as a good essential matrix),
K=146(Through K iterations to find the best essential matrix), it means 99% probability of finding a reasonable essential matrix if there are more than 0.5 points are inliers.

4.2.1 Experiment with Images At Same Position

First, capture two frames when the camera stay at the same position:

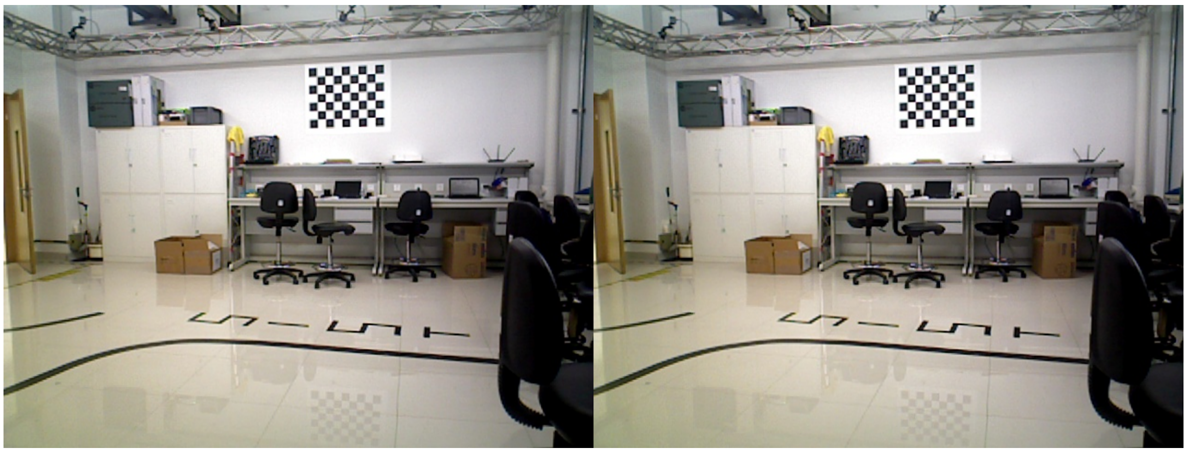


Figure 4-1 Two images at the same position

Secondly, extract and match the features between two images:

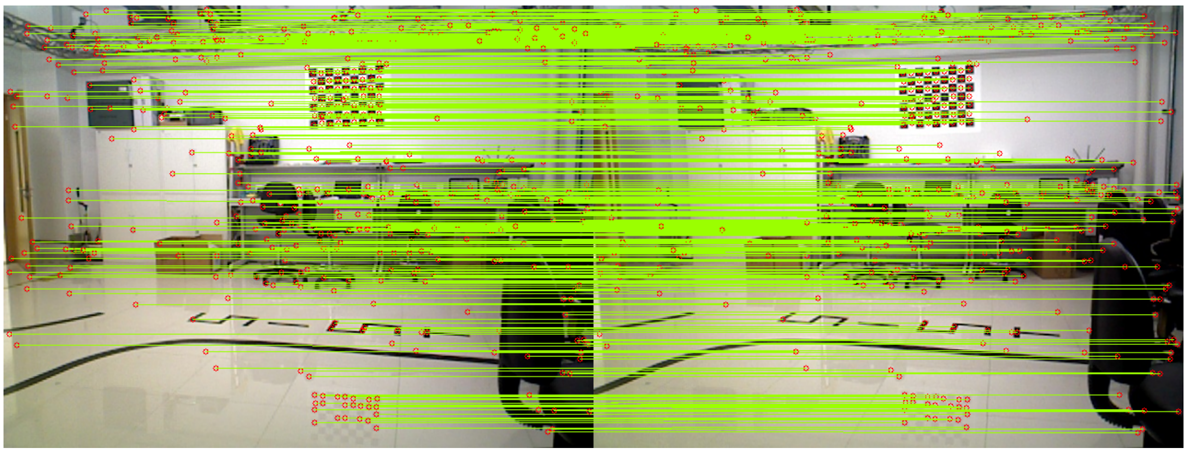


Figure 4-2 Extract and match the features between two images (Same position)

From the output of the code, SIFT extracts 673 features in the first image and 66 features in the second image. And there are 575 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

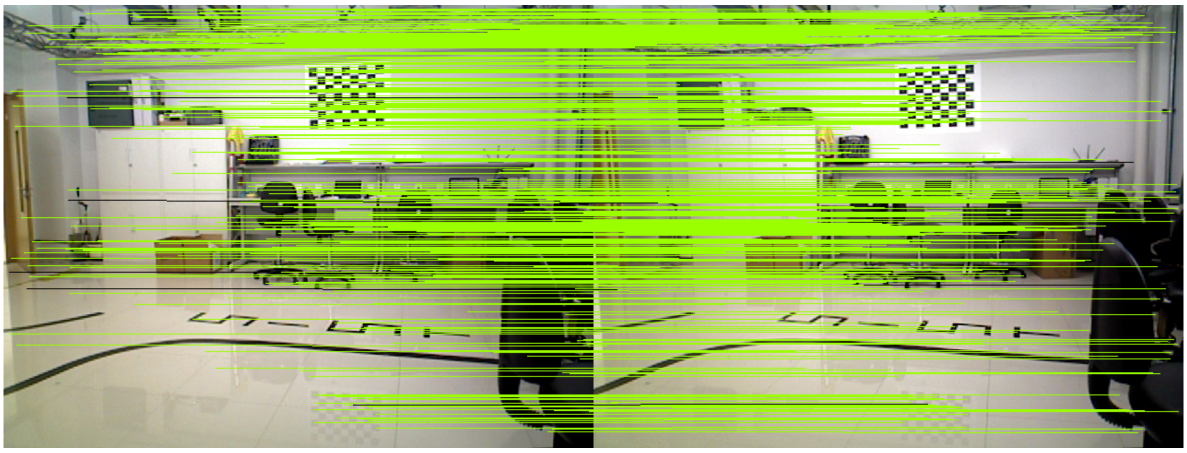


Figure 4-3 The inliers and outliers selected by RANSAC (No motion)

SiftGPU outputs 575 pairs of matches between the two images, which are used to calculate the essential matrix. After running the code for 50 times, there are 50 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-1 Result of no motion groups (rotation unit: degree)

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-1.71e-13	-1	1.87e-13	-1.75e-14	2.45e-15	-9.76e-15	573	7.6604	My code
2	-5.15e-14	-1	2.88e-14	-5.91e-12	-8.52e-26	3.29e-12	489	8.4228	My code
3	-1.71e-13	-1	-8.51e-13	-1.96e-11	2.82e-15	-9.75e-11	538	7.1157	My code
4	-1.65e-13	-1	2.95e-12	-1.89e-11	1.08e-15	3.38e-10	445	7.6277	My code
5	7.48e-14	-1	-2.31e-13	8.58e-12	5.38e-16	-2.64e-11	398	7.3972	My code
6	1.61e-13	-1	-3.37e-13	-2.71e-15	3.12e-24	-2.39e-15	572	6.7222	My code
7	1.21e-13	-1	-3.60e-14	1.39e-11	-3.62e-16	-4.13e-12	550	7.3098	My code
8	1.06e-13	-1	-2.84e-13	-7.67e-15	1.74e-24	-7.55e-16	558	6.7453	My code
9	5.59e-14	-1	-1.83e-13	-1.00e-14	1.33e-15	-6.17e-15	509	7.6162	My code
10	-1.37e-14	-1	-1.19e-13	3.18e-15	-2.02e-15	-1.58e-15	572	6.9295	My code
11	0.297	-0.028	0.954	-178.827	-34.570	-3.761	78	0.8051	OpenCV

In the table we can see that the rotation and translation values are all very very near to zero. Notice that the scaling factor $||t|| = 1$, so the translation values can not be all zero. The inliers in most group are more than 0.8 of the number of all matches. In fact, more than 90% of the matches are correct. Then we can find the reason why its computation time is very long. It is because the number of matches between these two images is quite big, and most of them are inliers, which make it easy to obtain a correct E from random 5 points. So most of E enter the following computations, computing a better essential matrix and recovering camera matrix P, and even join the comparing step. Obviously, if every iteration takes a long time, the total time will be very long.

In the second experiment, the number of iterations is reduced to 8 (when the inliers are more than 90%, it means 99% chance to get a correct essential matrix).

4.2.2 First Rotation: Pitch

a) Pitch: The First Group

First, capture two frames when the camera do the motion pitch:

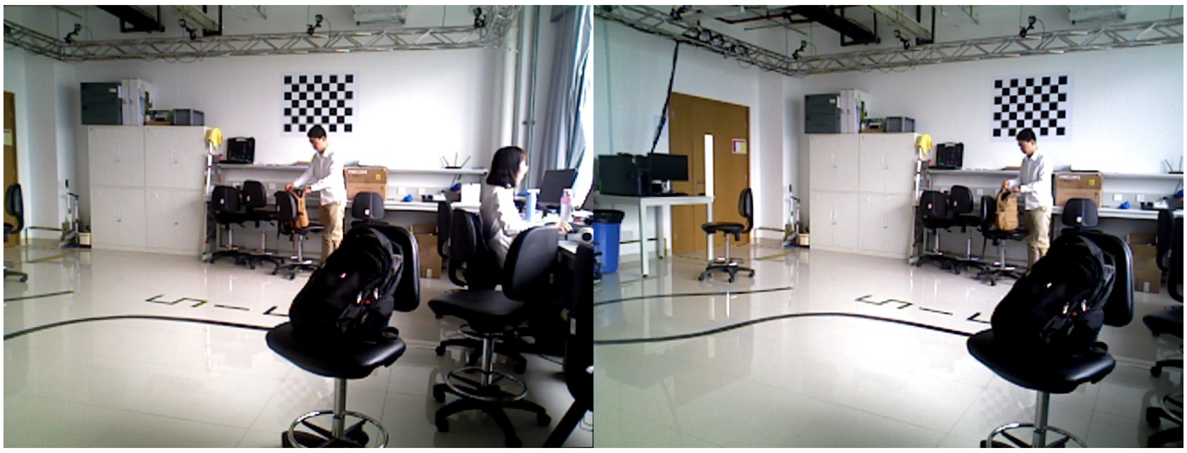


Figure 4-4 Two images of pitch

Secondly, run the code to extract and match the features between two images:

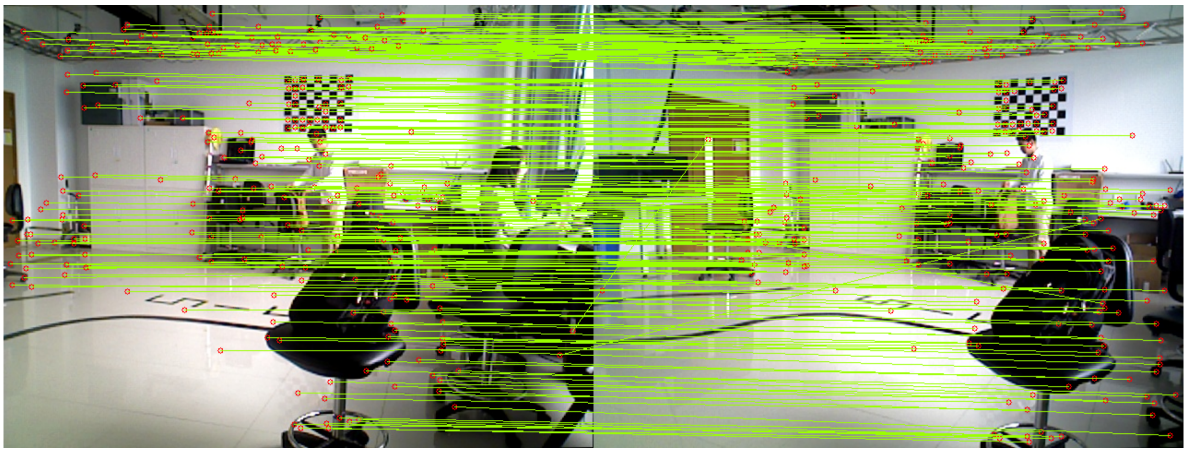


Figure 4-5 Extract and match the features between two images (Pitch)

From the output of the code, SIFT extracts 722 features in the first image and 677 features in the second image. And there are 285 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

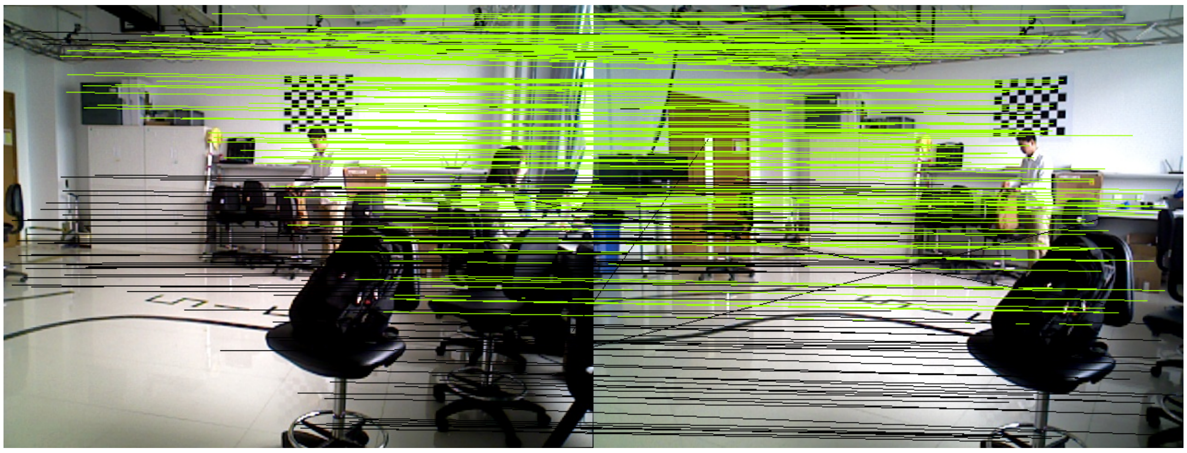


Figure 4-6 The inliers and outliers selected by RANSAC (Pitch)

SiftGPU outputs 285 pairs of matches between the two images, which are used to calculate the essential matrix. the parameters of the RANSAC algorithm are set as follows: T=1e-04, D=0.3*Match_Num, K=146. After running the code for 50 times, there are 49 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-2 Result of pitch

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-0.999	0.009	0.042	0.959	12.806	-0.323	88	4.9197	My code
2	-0.999	0.009	0.046	0.991	12.929	-0.309	102	5.1579	My code
3	-0.999	0.009	0.046	0.985	12.998	-0.292	103	4.9039	My code
4	-0.999	0.007	0.047	1.028	12.956	-0.324	92	4.8772	My code
5	-0.999	0.010	0.047	1.006	12.936	-0.292	97	5.1170	My code
6	-0.999	0.007	0.050	1.135	13.383	-0.293	86	5.3559	My code
7	-0.999	0.007	0.046	0.986	12.939	-0.337	94	4.9777	My code
8	-0.999	0.010	0.046	0.993	12.876	-0.286	96	5.3071	My code
9	-0.999	0.006	0.043	0.979	12.821	-0.276	86	5.3168	My code
10	-0.999	0.009	0.048	1.015	12.951	-0.292	87	4.9576	My code
11	-0.581	0.151	-0.799	0.257	12.091	-0.545	135	0.8273	OpenCV

Set $D = 0.4 * Match\_Num$ for other groups but set $D = 0.3 * Match\_Num$ for this two images. It's because when setting $D = 0.4 * Match\_Num$ for these group, the most results are wrong. Observe the inliers numbers selected by RANSAC in the table, we find in most time it's less than 0.4*Match_Num. Therefore it's hard to obtain a good E when $D = 0.4 * Match\_Num$. Meanwhile, it leads to the long computation time (about 5 seconds). It is hard to get 5 inliers in the 5 random sampling points, which make it take a long time to find all-inlier sampling points.

.

b) Pitch: The Second Group

First, capture two frames when the camera do the motion pitch:

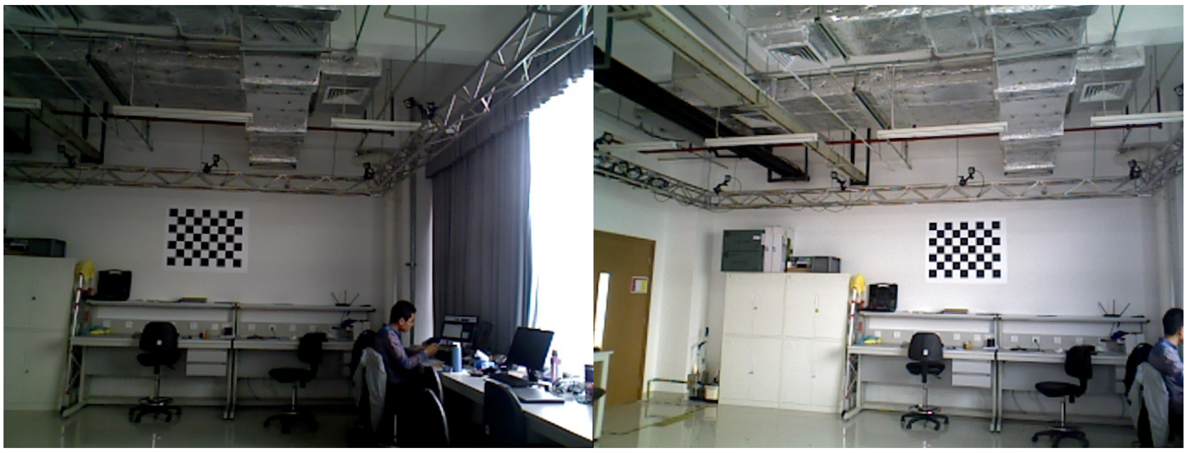


Figure 4-7 Two images of pitch

Secondly, run the code to extract and match the features between two images:

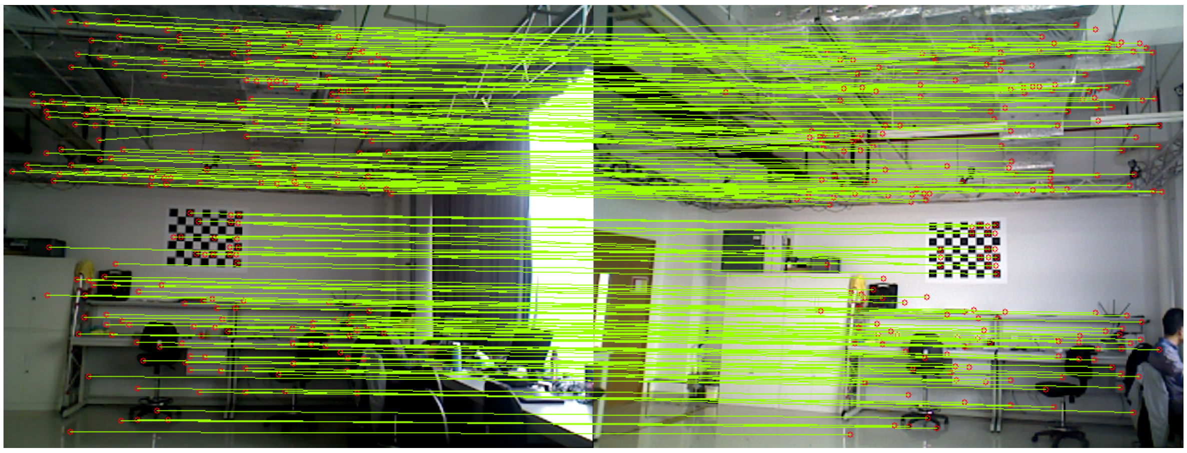


Figure 4-8 Extract and match the features between two images (Pitch)

From the output of the code, SIFT extracts 757 features in the first image and 854 features in the second image. And there are 240 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

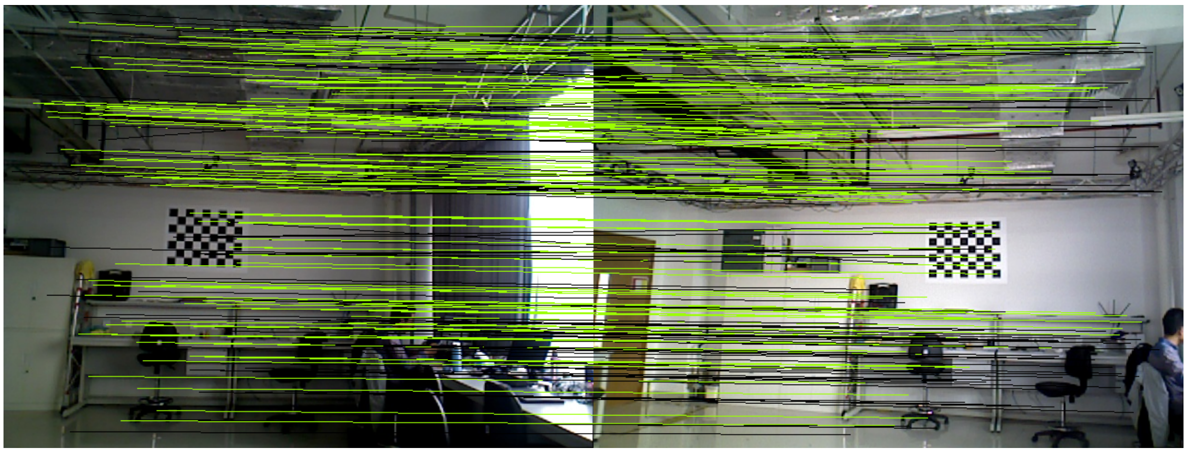


Figure 4-9 The inliers and outliers selected by RANSAC (Pitch)

SiftGPU outputs 240 pairs of matches between the two images, which are used to calculate the essential matrix. Set the parameters of the RANSAC algorithm as follows: T=1e-04, D=0.2*Match_Num, K=146. After running the code for 50 times, there are 50 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-3 Result of pitch

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-0.999	0.016	0.044	-3.627	18.317	-1.645	53	4.8628	My code
2	-0.999	0.014	0.045	-3.588	18.365	-1.652	58	4.1036	My code
3	-0.999	0.015	0.049	-3.581	18.470	-1.646	64	4.9039	My code
4	-0.999	0.013	0.042	-3.625	18.244	-1.691	70	4.0905	My code
5	-0.999	0.015	0.043	-3.626	18.262	-1.667	52	4.3203	My code
6	-0.999	0.016	0.043	-3.619	18.282	-1.648	50	4.4061	My code
7	-0.999	0.016	0.042	-3.617	18.126	-1.633	49	4.4866	My code
8	-0.999	0.016	0.043	-3.602	18.289	-1.649	59	4.6820	My code
9	-0.999	0.015	0.043	-3.609	18.341	-1.653	49	4.3048	My code
10	-0.999	0.016	0.042	-3.684	18.144	-1.645	53	4.2936	My code
11	-0.952	-0.168	-0.252	-2.894	15.043	-1.012	137	0.7963	OpenCV

Set $D = 0.3 * Match\_Num$ for last group but set $D = 0.2 * Match\_Num$ for this group. It's because when setting $D = 0.3 * Match\_Num$, the most results are wrong.

Observe the inliers numbers selected by RANSAC in the table, we find in most time it's less than 0.3*Match_Num. Therefore it's hard to obtain a good E when $D = 0.3 * Match\_Num$. Therefore, it also take a long time (about 4 seconds) to obtain a good E.

4.2.3 Second Rotation: Roll

a) Roll: the First Group

First, capture two frames when the camera do the motion pitch:

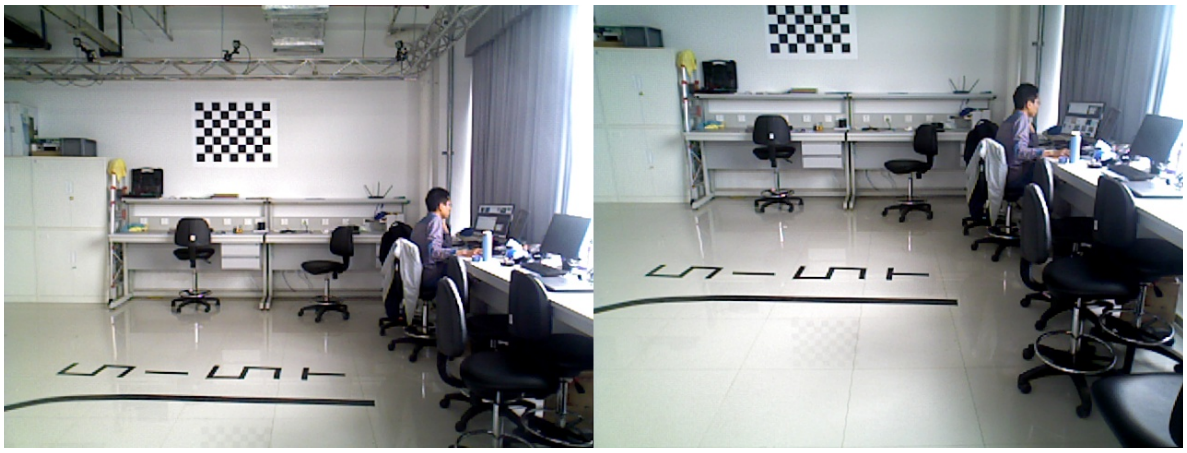


Figure 4-10 Roll

Secondly, run the code to extract and match the features between two images:

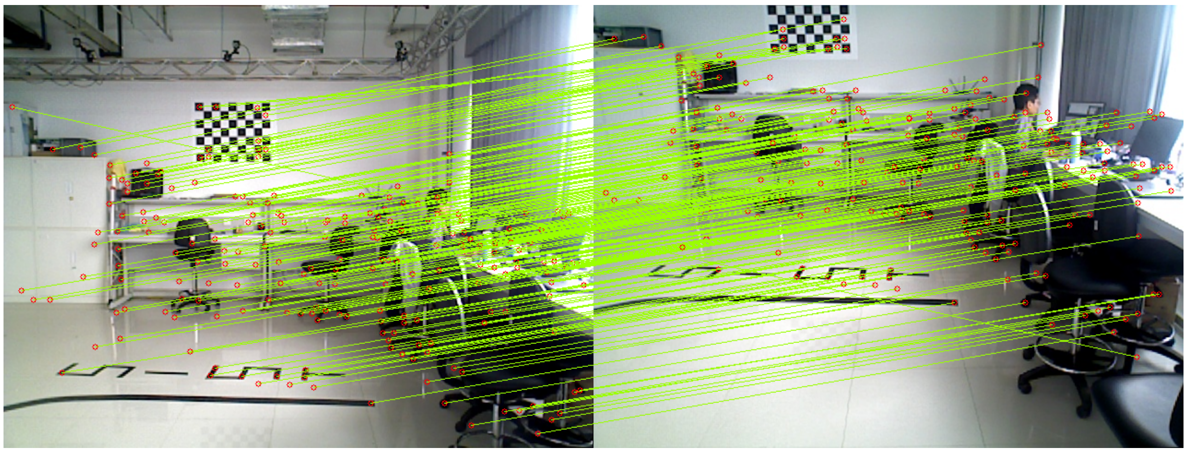


Figure 4-11 Extract and match the features between two images

From the output of the code, SIFT extracts 704 features in the first image and 542 features in the second image. And there are 234 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

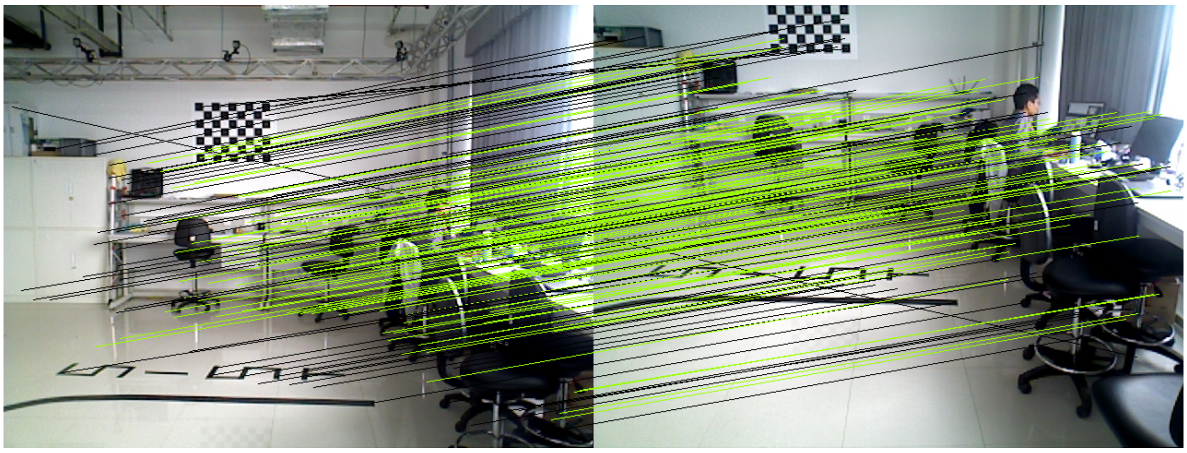


Figure 4-12 The inliers and outliers selected by RANSAC (Roll)

SiftGPU outputs 234 pairs of matches between the two images, which are used to calculate the essential matrix. After running the code for 50 times, there are 48 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-4 Result of roll

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-0.877	0.408	-0.250	0.158	3.382	12.721	118	3.6170	My code
2	-0.956	0.256	-0.142	0.338	5.472	12.6581	131	2.9162	My code
3	-0.882	0.238	-0.406	0.290	2.965	11.947	121	2.9411	My code
4	-0.977	0.119	-0.172	1.031	6.129	11.960	96	2.8901	My code
5	-0.882	0.442	-0.156	0.210	-3.647	12.994	124	2.8490	My code
6	-0.892	0.305	-0.331	0.150	3.768	12.424	128	2.6903	My code
7	-0.845	0.154	-0.510	0.339	2.921	11.599	123	2.8021	My code
8	-0.485	0.137	-0.863	0.012	0.820	11.300	95	2.8945	My code
9	-0.846	0.140	-0.512	0.670	2.748	11.542	111	2.7527	My code
10	-0.894	-0.232	0.381	-32.957	-47.833	26.408	124	2.6364	My code
11	0.468	0.811	-0.347	-0.456	-1.767	12.927	166	0.8150	OpenCV

The computation time is about 2.8 second, which is not a short time. The table tells us that the inliers are about 0.5 of the number of matches. Therefore it is also not easy to obtain a good E that can fit 0.4 of the matches. In general, the results are good enough so far. And if we reduce the parameter D, the rotation values can be more precise.

.

b) Roll: the Second Group

First, capture two frames when the camera do the motion pitch:

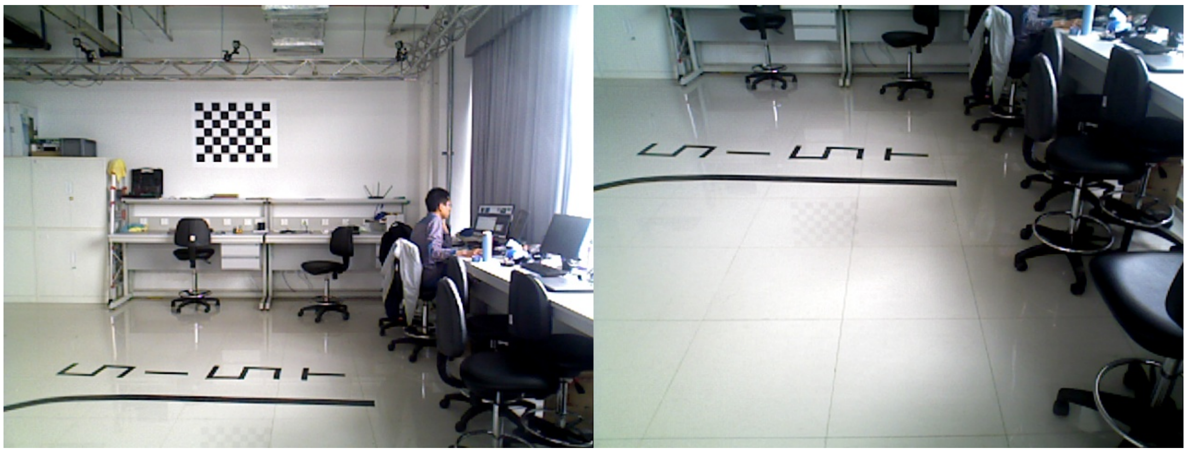


Figure 4-13 Roll

Secondly, run the code to extract and match the features between two images:

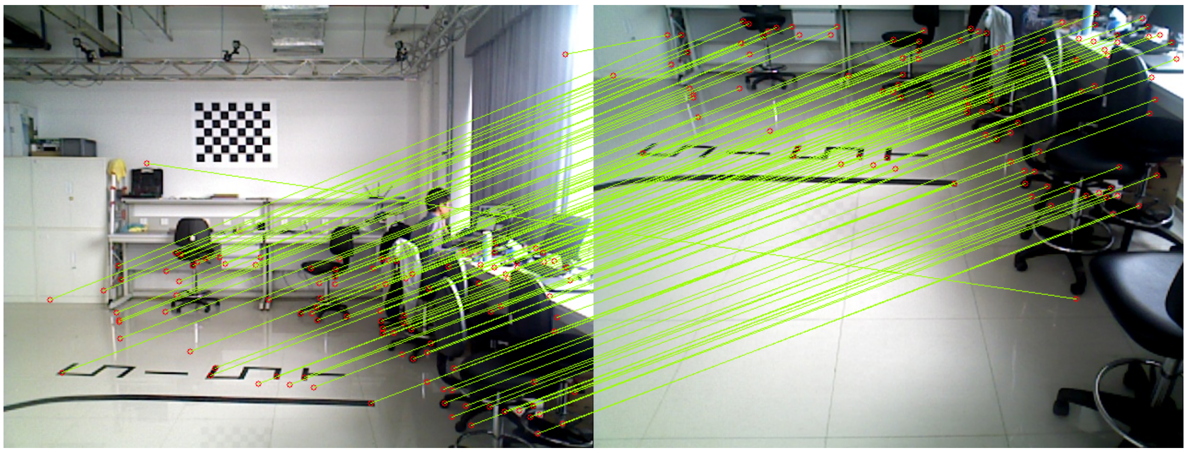


Figure 4-14 Extract and match the features between two images

From the output of the code, SIFT extracts 704 features in the first image and 353 features in the second image. And there are 125 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

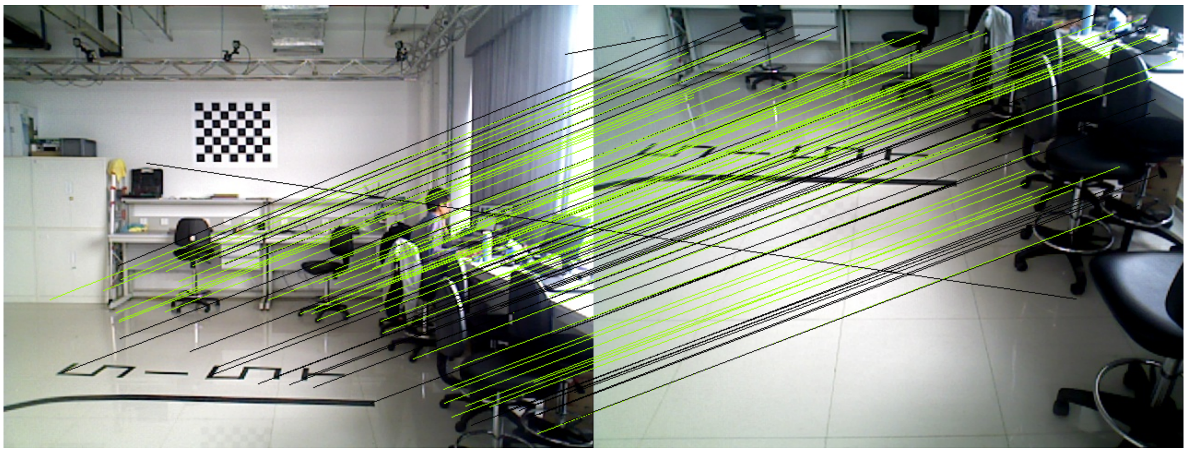


Figure 4-15 The inliers and outliers selected by RANSAC (Pitch)

SiftGPU outputs 125 pairs of matches between the two images, which are used to calculate the essential matrix. After running the code for 50 times, there are 49 groups of results are correct. And the top ten sets of results from my code (5-point algorithm with RANSAC) and a group of result from the openCV implement are as follows:

Table 4-5 Result of roll

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	0.791	0.290	-0.537	-0.292	-4.186	25.267	75	2.3513	My code
2	0.910	0.379	-0.162	-1.352	-6.423	27.012	70	1.9119	My code
3	-0.998	0.031	0.035	4.279	8.828	25.552	53	1.8509	My code
4	0.730	0.313	-0.606	-0.040	-3.338	25.175	95	1.6796	My code
5	-0.998	0.058	-0.019	3.816	8.004	25.505	53	1.6932	My code
6	-0.995	0.085	-0.030	3.160	6.817	25.513	63	1.8197	My code
7	-0.980	0.116	-0.159	2.686	5.587	25.287	63	1.7493	My code
8	-0.975	0.136	-0.170	2.690	6.153	25.430	61	1.5259	My code
9	-0.994	0.094	-0.045	3.078	6.695	25.525	64	1.7111	My code
10	-0.998	0.042	0.017	4.211	8.786	25.586	54	1.7188	My code
11	-0.410	-0.137	0.901	0.032	-3.000	24.152	61	0.8215	OpenCV

The computation time is around 1.7 second, which is a shorter time. Because there are more than 0.5 of the matches are inliers. Therefore it is easy to obtain a good E that can fit 0.4 of the matches. In general, the results are good enough so far.

And if we reduce the parameter D, the rotation values can be more precise.

4.2.4 Third Rotation: Yaw

a) Yaw: The First Group

First, capture two frames when the camera do the motion pitch:

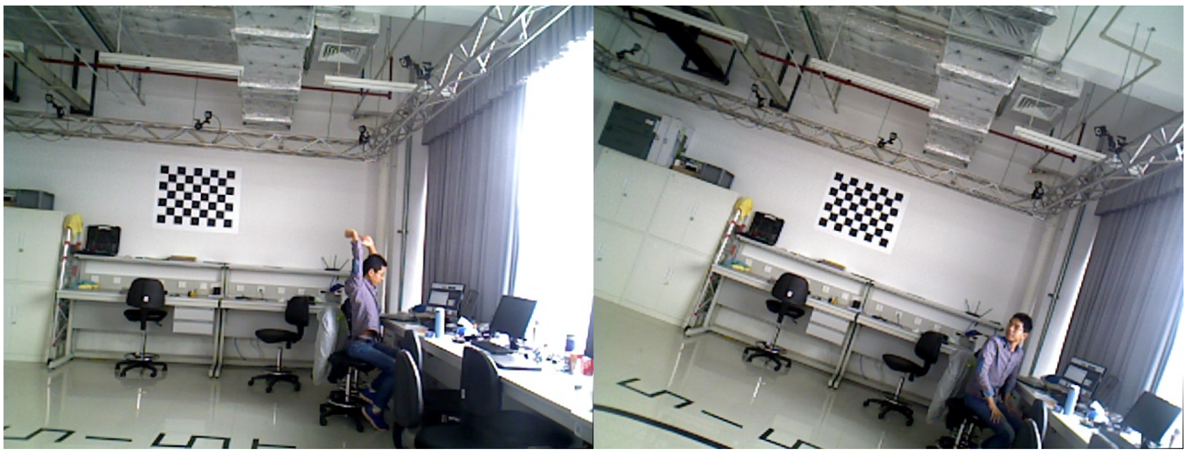


Figure 4-16 Roll

Secondly, run the code to extract and match the features between two images:

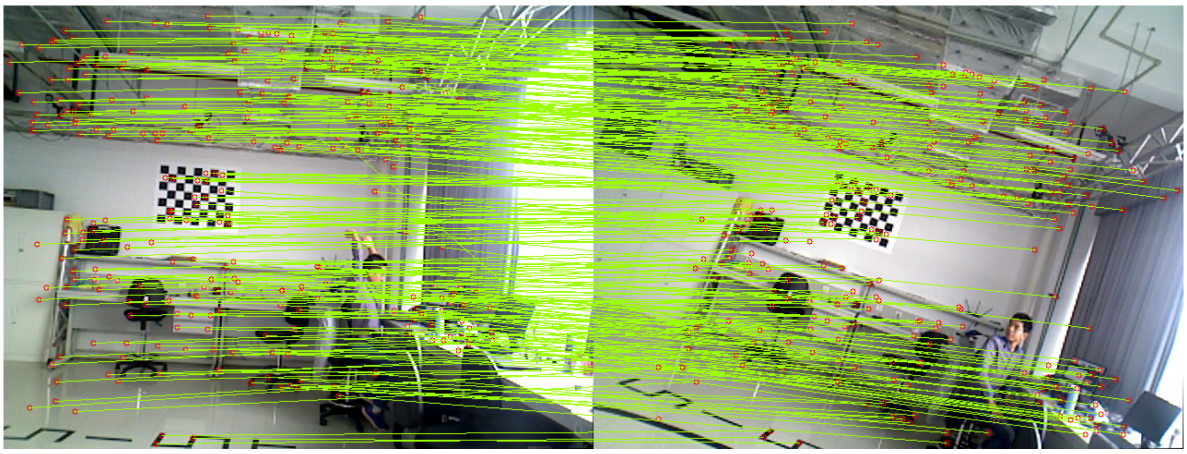


Figure 4-17 Extract and match the features between two images

From the output of the code, SIFT extracts 767 features in the first image and 818 features in the second image. And there are 296 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

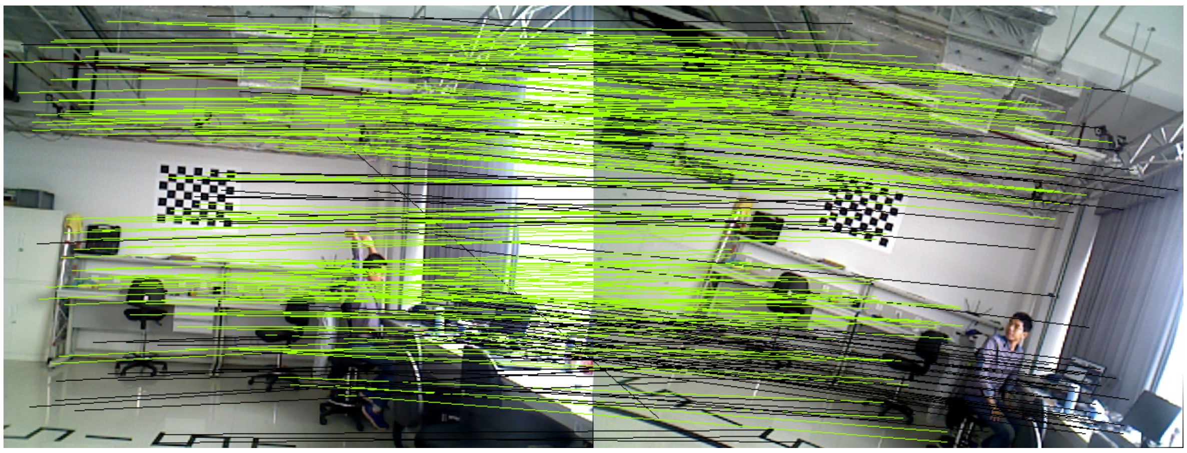


Figure 4-18 The inliers and outliers selected by RANSAC (Yaw)

SiftGPU outputs 296 pairs of matches between the two images, which are used to calculate the essential matrix. After running the code for 50 times, there are 49 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-6 Result of yaw

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-0.997	0.011	0.079	13.931	6.730	-2.348	172	1.1087	My code
2	-0.999	0.037	0.021	14.039	6.899	-2.181	157	1.0750	My code
3	-0.999	0.026	0.014	13.856	7.775	-1.970	159	1.0970	My code
4	-0.993	0.009	0.114	13.764	5.933	-2.547	171	1.1244	My code
5	-0.999	0.036	-0.003	13.965	7.735	-1.949	157	1.1077	My code
6	-0.996	0.018	0.080	13.826	5.926	-2.510	209	1.2098	My code
7	-0.995	0.013	0.092	13.853	6.463	-2.406	155	1.2474	My code
8	-0.998	0.023	0.058	13.822	6.091	-2.453	158	1.1432	My code
9	-0.997	0.021	0.063	13.811	6.084	-2.460	167	1.1928	My code
10	-0.993	-0.006	0.114	14.160	7.101	-2.346	152	1.1214	My code
11	-0.481	-0.335	0.809	14.444	8.246	-2.173	127	0.8090	OpenCV

Observe the inliers numbers selected by RANSAC in the table, we find in most time it's more than 0.5 of the matches. Therefore it's easy to obtain a good E when $D = 0.4 * Match\_Num$. Therefore, it needn't take a long time to obtain a good E.

.

b) Yaw: The Second Group

First, capture two frames when the camera do the motion pitch:

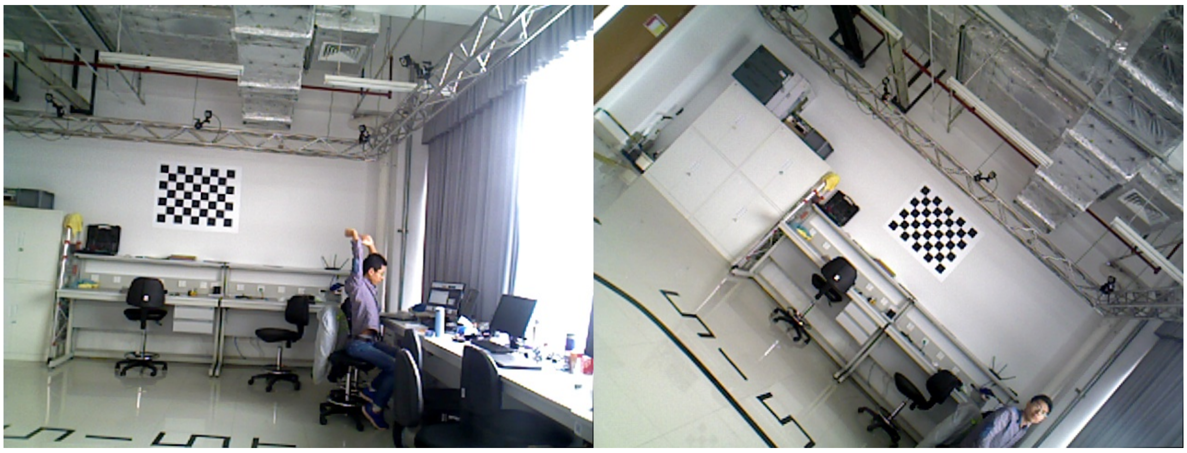


Figure 4-19 Roll

Secondly, run the code to extract and match the features between two images:

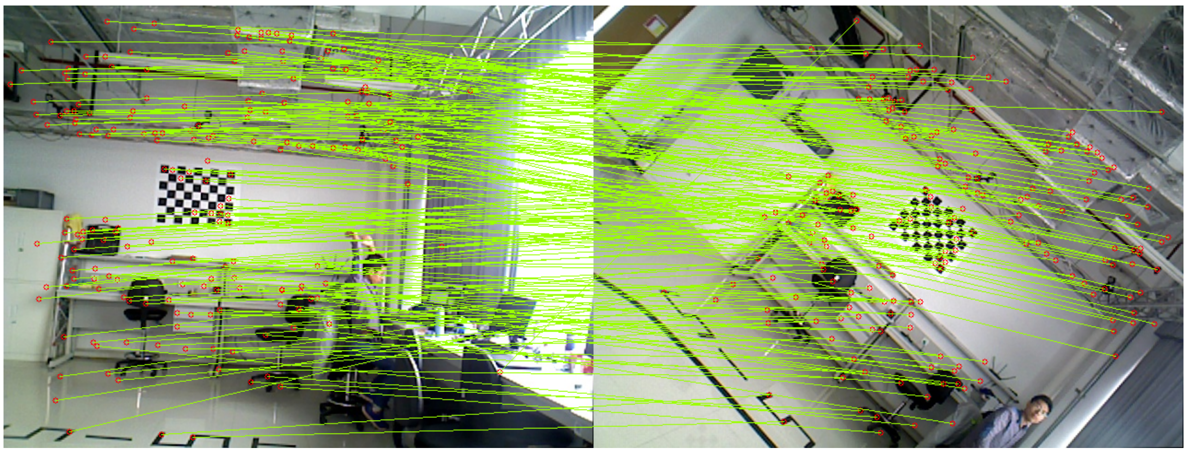


Figure 4-20 Extract and match the features between two images

From the output of the code, SIFT extracts 767 features in the first image and 773 features in the second image. And there are 205 matches between these two images.

The code with RANSAC selects the inliers to compute the essential, the code are run for one time and the inliers (green lines) and outliers (black lines) are drawn below:

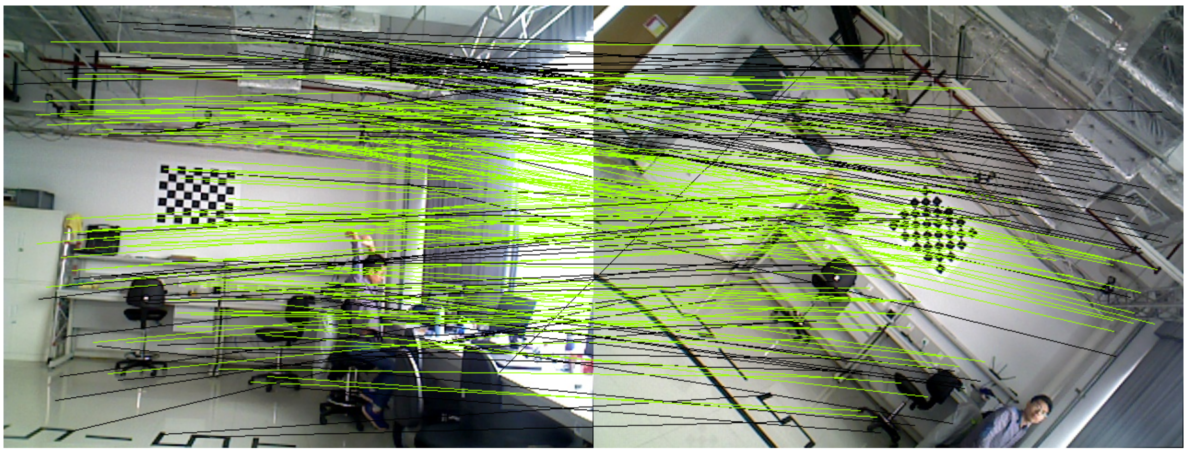


Figure 4-21 The inliers and outliers selected by RANSAC (Yaw)

SiftGPU outputs 205 pairs of matches between the two images, which are used to calculate the essential matrix. After running the code for 50 times, there are 42 groups of results are correct. And the top ten sets of results of my code (5-point algorithm with RANSAC) and a group of result of the openCV implement are as follows:

Table 4-7 Result of yaw

	X-axis	Y-axis	Z-axis	Yaw	Pitch	Roll	Inliers	Time (s)	Code
1	-0.998	0.012	0.047	31.773	16.886	-1.631	107	1.5873	My code
2	-0.998	0.051	-0.025	31.559	17.703	-0.273	103	1.0123	My code
3	-0.999	0.029	0.005	33.300	15.676	-2.555	105	1.0715	My code
4	-0.999	0.018	0.033	31.701	17.491	-1.138	136	1.0228	My code
5	-0.999	0.020	0.017	31.643	19.369	0.251	121	0.8853	My code
6	-0.003	0.999	-0.003	0.175	0.008	0.569	162	0.9507	My code
7	-0.998	0.046	-0.017	31.583	17.945	-0.209	110	0.9468	My code
8	-0.999	0.008	0.040	31.777	19.173	-0.137	109	0.9384	My code
9	-0.999	0.018	0.028	32.547	13.944	-3.605	116	0.9394	My code
10	-0.999	0.023	0.016	32.901	15.963	-2.360	122	0.9233	My code
11	-0.427	0.351	-0.833	38.949	15.755	-1.458	108	0.802	OpenCV

The computation time is around 1 second, which is not a long time. Because there are more than 0.5 of the matches are inliers. Therefore it is easy to obtain a good E that can fit 0.4 of the matches. Besides, we find that the fluctuation of the rotation values in a wide range, which means that the best E given finally is not obtained through enough comparisons. To solve this problem, reducing the D is a good choose.

4.3 Testing Experiment with Different Parameters in RANSAC

We use two groups of images to do the experiment in this chapter. In this chapter, the code will be run two hundred times to verify the correctness of the code as well as the computation time.

4.3.1 Change Parameters of the No Motion Group

a) The Significance of Changing Parameters

First of all, the group taken at the same position is important, whose matches almost all are inliers. For this reason, it's easy to obtain 5 inliers in 5 random points, which leads to a small number of iterations. We can compute a reasonable iteration number as follows:

$$K = \frac{log(P)}{log(1-W^N)} = \frac{log(0.01)}{log(1-0.8^5)} = (int)11.599 = 12$$

where P is the possibility of getting a correct model (essential matrix E) after K iterations, which is 99%, and we assume the inliers are 0.9 of the matches. Besides, I will set $K = 86$ (sufficient number of iterations) in contrast.

In addition, change the D (if the inliers fitting the model are more than D, it can be seen as a good model) and T (if the error between the point and the model is less than T, the point can be regarded as a inlier) for several times, then we can find the relationship between the operation time and the T and D.

b) Compare the Results

Record the computation time and correct rate after 200 times running code in the below table:

Table 4-8 Different parameters of the no motion group

K	D(of match number)	T	Correctness	Computation time(s)	Deviation	Inliers
12	0.8	1e-4	173/200	0.8631	0.01	544.66/575
86	0.8	1e-4	200/200	4.7681	1e-11	555.27/575
12	0.4	1e-4	179/200	0.6681	0.001	502.2/575
12	0.8	1e-2	177/200	1.2315	0.01	539.61/575
12	0.8	1e-14	146/200	0.5212	0.0001	553.2/575
86	0.8	1e-14	200/200	4.3475	1e-11	529.53/575

c) Analyze the Experiment Results

According to the table, we can get the following conclusion:

(1) When K=86, all the results obtained by running the program are correct, which says that it is enough to get very good essential matrix by 86 iterations. And form the figure, we can see that the error is as small as 1e-11. So we have reason to think that we can get nearly one hundred percent of correct results with very accurate numerical values when the input is good(having little outliers of matching features), and iteration number is enough. But, it needs more time to achieve the good property, more than 4 seconds.

(2) T=1e-14 means a very strict fitting requirements, that is the value is normal if and only if the deviation of point and model is litter that 1e-14. This kind of parameter setting has no effect on the correctness for K=86, and all the results are right. But the effect is a little big if K=12, with accuracy dropped from 173/200 to 146/200. In an iteration, it is hard to guarantee that 0.8 of matching features points are very fit to established model(essential matrix). So, if the iteration times is a little less, we cant find the correct model(essential matrix) with too high requirements.

(3) Compared to D=0.8*match_num, if D=0.4*match_num, the computation time is shorter, deviation is smaller, and correct results are more, but normal values are fewer. It is not bad that the normal values are fewer, contrarily, it can be considered to be a good model if the setting requires that only more than 0.4 of matching fit the building model. It lowers the threshold of a good model, and then some more accuracy points can have bigger opportunity to build a more accuracy model. With more opportunities to try, it's easier for us to get right results. Besides, more opportunities to try should lead to longer calculation times, but we ca save time by using fewer points in every calculation of a better E, since only fewer points are considered to be normal values.

(4) Another way to lower the threshold of a good model is to increase T, which causes more points to be considered normal(If it is big,abnormal values also will be considered to be normal). It is not to make more precise points have more chance to participate in the calculation, on the contrary, it makes less precise points have the opportunity to participate in the establishment of models. When the input includes fewer abnormal values, it does not reduce the accuracy rate,

and even improves the accuracy, but the calculation time is prolonged.

4.3.2 Change Parameters of the Second Yaw Group

a) The Significance of Changing Parameters

Let's select the second yaw group to do this testing experimental. Only about 0.5 of the matches are inliers, which means the input is not good enough (contrary to the no motion group). For this reason, it's not easy to obtain 5 inliers in 5 random points, which leads to the result that it's hard to get a good E in one iteration. Let P=99%, which means there are99% possibility to get a good essential matrix from the algorithm. We can compute a reasonable iteration number as follows:

$$K = \frac{log(P)}{log(1-W^N)} = \frac{log(0.01)}{log(1-0.5^5)} = (int)145.0507 = 146$$

we assume the inliers are 0.5 of the matches. Besides, set $K = 86$ (sufficient number of iterations) in contrast.

In addition, change the D (if the inliers fitting the model are more than D, it can be seen as a good model) and T (if the error between the point and the model is less than T, the point can be regarded as a inlier) for several times, then we can find the relationship between the operation time and the T and D.

b) Compare the Results

Record the computation time and correct rate after 200 times running code in the below table:

Table 4-9 Different parameters of the yaw group

K	D(of match number)	T	Correctness	Computation time(s)	Deviation	Inliers
146	0.5	1e-4	173/200	1.8088	0.5	114.3/205
146	0.3	1e-4	135/200	1.8567	1.7	72.5/205
246	0.5	1e-4	164/200	1.5384	1.3	114/205
146	0.5	1e-2	177/200	0.9407	1.4	116.5/205

c) Analyze the Experiment Results

According to the table, we can get the following conclusion:

(1) Compared to K=246, K=146 does not bring advantage, or even lower the correct rate, which is very surprising. In the case of the input is relatively poor (contains more outliers), more iterations may give outliers more opportunities to participate in the calculation.

(2) Another result opposite of last experiment is that when D is reduced to D = 0.3*match_num, the correct rate is down, and the rate of decline is very serious.Because the input is bad, lowering the standard of good model does not mean making a few more precise points to build a model, but means making more incorrect model be considered to be a correct model. So, while the correct rate is reduced, the calculation time is not reduced.

(3) When we increase the T to 0.01, we find that the correct rate is increased, and the computation time is reduced to a large extent.As mentioned earlier, an increase in T causes more points to be considered as a normal value, resulting in less precise points have the opportunity to participate in the establishment of the model.When the camera does a larger movement, the coordinates of points have larger change, and their corresponding coordinates will become less accurate. At this time, if we increase T to a reasonable threshold, it is more conducive to the establishment of the correct model.

4.4 Chapter Summary

In this chapter we do two experiments. In the first one we use different image pairs to verify the code. We use good input (images taken at the same position) and not good input (especially the pitch pair) to calculate their camera pose transform, and record their correctness and computation speed, and compare their results with that of the OpenCV code. In the second experiment, we use the same pair of images as input, but change the parameters of the RANSAC algorithm, to see how the correctness, computation time and the deviation change. In order to obtain rigorous results, we use a pair with good matching and a pair with bad matching to do the experiment. And finally, we analyze the results in detail after every experiment.

In this thesis, the main parts of monocular vision measurement is completed: one is the extraction and matching of features, another is the calculation of the transformations of camera by the corresponding image points (this thesis mainly completes the calculation of rotation part). In this thesis, for the features detection and matching, we have used the open source code, SiftGPU, while my own code is mainly on five-point algorithm with RANSAC. SIFT feature is a kind of significant and robust feature. By the pictures 2-1, we can see that most of the features matched by SiftGPU are correct. The code of using the five point algorithm with RANSAC to calculate the essential matrix has been achieved before by others, which mostly used the epipolar constraint(the distance between the corresponding points to its polar line is minimal) or Sampson distance(deduced from the epipolar line, but its results much better) to check the inliers. My code uses both of them, which make the result improve in a large degree. From the pictures from the RANSAC's outputs (figure 4-6, 4-18, 4-12, 4-9, 4-21, 4-15 and 4-3), we can see almost all the outliers are removed with good parameters of RANSAC. In this thesis, it uses OpenCV code results to compare with the results of my code, showing the calculation time of mine is not much longer than the OpenCV's. Although fluctuation range of the results of mine is relatively larger, but they are basically accurate (in rotation information aspect). The translation information results of my code is not good, and because the scale factors are not included in t vector, so it is hard to confirm the accuracy of the translation results. OpenCV results are very bad for the images at the same position (i.e., no camera movement), but my code results are incredibly accurate and with high correctness (table 4-1). In this thesis, the five-point algorithm with RANSAC are talked about a lot. As we know from the previous discription, using a single time five-point algorithm to calculate the essential matrix will have a number of wrong solutions (since solving E is actually to solve ten times polynomial), so use RANSAC for outliers exclusion is very necessary.

Conclusions

In this thesis, the main parts of monocular vision measurement is completed: one is the extraction and matching of features, another is the calculation of the transformations of camera by the corresponding image points (this thesis mainly completes the calculation of rotation part). In this thesis, for the features detection and matching, we have used the open source code, SiftGPU, while my own code is mainly on five-point algorithm with RANSAC. SIFT feature is a kind of significant and robust feature. By the pictures 2-1, we can see that most of the features matched by SiftGPU are correct. The code of using the five point algorithm with RANSAC to calculate the essential matrix has been achieved before by others, which mostly used the epipolar constraint or Sampson distance(deduced from the epipolar line, but its results much better) to check the inliers. My code uses both of them, which make the result improve in a large degree. From the pictures from the RANSAC's outputs (figure 4-6, 4-18, 4-12, 4-9, 4-21, 4-15 and 4-3), we can see almost all the outliers are removed with good parameters of RANSAC. In this thesis, it uses OpenCV code results to compare with the results of my code, showing the calculation time of mine is not much longer than the OpenCV's. Although fluctuation range of the results of mine is relatively larger, but they are basically accurate (in rotation information aspect). The translation information results of my code is not good, and because the scale factors are not included in t vector, so it is hard to confirm the accuracy of the translation results. OpenCV results are very bad for the images at the same position (i.e., no camera movement), but my code results are incredibly accurate and with high correctness (table 4-1). In this thesis, the five-point algorithm with RANSAC are talked about a lot. As we know from the previous description, using a single time five-point algorithm to calculate the essential matrix will have a number of wrong solutions (since solving E is actually to solve ten times polynomial), so use RANSAC for outliers exclusion is very necessary.

Some shortcomings of this thesis is we calculate the rotation transform of camera well, but the results of translation is actually need to be improved. As we known, the translation vector \$t\$ extracted from the essential matrix is without the scaling factor. Obviously, in this thesis the translation vector t satisfy ||t||=1 but not actually correct in direction. To converse the vector t to some units we use in usual (i.e. meter), we need to put some mark in the scene, then we can converse the distance in the photos to that in the world. But fixing the problem about the translation direction is more important.

Acknowledgments

Thanks to my two distinguished teachers --Sören Schwertfeger and Wang Xiaoning.

Professor Wang is not only strict but also kind. When I leave for Shanghai Tech University, she asked me to pay attention to safety. During my bachelor thesis outside our school, she usually sent e-mails to me and talked with me on QQ to supervise my thesis complete progress. When I had difficulties in debugging the program, she encouraged me not to give up, and told me dawn is in front.

Sören is easy to get along, who is not only my teacher but also my friend. He usually help me to debug my program, and discuss the theory problems with me. Since my English writing is really poor, he is always help me to fix the grammar mistakes in my thesis carefully. With his help, my English writing quality and speed have been improved.

Family support and care is essential for my education. My family’s pay make me could concentrate on my study happily. I am very grateful to them

Finally, thanks the teachers who attend the thesis review and defense.

References

[1] Friedrich Fraundorfer, Davide Scaramuzza. Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications[J]. Robotics & Automation Magazine IEEE, 2012, 19(2):78-90.

[2] Nistér D, Naroditsky O, Bergen J. Visual odometry[C]. Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. IEEE, 2004: I-652-I-659 Vol.1.

[3] Maimone M, Cheng Y, Matthies L, et al. Two years of Visual Odometry on the Mars Exploration Rovers[J]. Journal of Field Robotics, 2007, 24(3): 169-186.

[4] Maimone M, Cheng Y, Matthies L, et al. Two years of Visual Odometry on the Mars Exploration Rovers[J]. Journal of Field Robotics, 2007, 24(3): 169-186.

[5] Moravec H P. Obstacle avoidance and navigation in the real world by a seeing robot rover[D]. Ph.D. dissertation, Stanford University, 1980.

[6] Comport A I, Malis E, Rives P. Accurate Quadrifocal Tracking for Robust 3D Visual Odometry[C], Proceedings - IEEE International Conference on Robotics and Automation. 2007:40-45.

[7] Nister D, Naroditsky O, Bergen J R, et al. Visual odometry for ground vehicle applications[J]. Journal of Field Robotics, 2006, 23(1): 3-20.

[8] Corke P, Strelow D, Singh S. Omnidirectional visual odometry for a planetary rover[C]. Intelligent Robots and Systems, 2004 IEEE/RSJ International Conference on, 2004(4): 4007-4012.

[9] Lhuillier M. Automatic scene structure and camera motion using a catadioptric system[J]. Computer Vision and Image Understanding, 2008, 109(2): 186-203.

[10] Goecke R, Asthana A, Pettersson N, et al. Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform[J]. Intelligent Vehicles Symposium IEEE, 2007: 450-455.

[11] Tardif J, Pavlidis Y, Daniilidis K, et al. Monocular visual odometry in urban environments using an omnidirectional camera[C]. Intelligent RObots and Systems, 2008: 2531-2538.

[12] Michael J. Milford. Single camera vision-only SLAM on a suburban road network [C]. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on IEEE, 2008: 3684 – 3689.

[13] Mouragnon E, Lhuillier M, Dhome M, et al. Real Time Localization and 3D Reconstruction[C]. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2006(1): 363-370.

[14] Scaramuzza D, Fraundorfer F, Siegwart R. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC[C], IEEE International Conference on Robotics and Automation. IEEE Press, 2009:4293-4299.

[15] Pretto A, Menegatti E, Pagello E, et al. Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation[J]. International Conference on Robotics and Automation, 2011.

[16] VLFeat. SIFT++[CP]. http://vision.ucla.edu/~vedaldi/code/siftpp.html, 2016-2-17.

[17] Sinha S N, Frahm J M, Pollefeys M, et al. GPU-based Video Feature Tracking and Matching[J]. Workshop on Edge Computing Using New Commodity Architectures, 2006, 2000(Oct 21):189-196.

[18] Dirk Thomas. Wiki: Ros/introduction — wikipedia, the free encyclopedia[EB/OL]. http://wiki.ros.org/ROS/Introduction, 2015-12-17.

[19] PiyushKhandelwal. Wiki: openni_launch[EB/OL]. http://wiki.ros.org/openni_launch, 2016-1-15.

[20] Lowe D G. Object recognition from local scale-invariant features[C]. Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. IEEE, 1999, 2:1150-1157.

[21] Lowe D G. Distinctive Image Features from Scale-Invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(60):91-110.

[22] Ke Y, Sukthankar R. PCA-SIFT: a more distinctive representation for local image descriptors[J]. Computer Vision and Pattern Recognition, 2004, (2): 506-513.

[23] Mikolajczyk K, Schmid C. Performance evaluation of local descriptors.[J]. Pattern Analysis & Machine Intelligence IEEE Transactions on, 2005, 27(10):1615-1630.

[24] Matthies L, Shafer S A. Error modeling in stereo navigation[J]. IEEE Journal on Robotics and Automation, 1987, 3(3): 239-248.

[25] Lacroix S, Mallet A, Chatila R, et al. Rover Self Localization in Planetary-Like Environments[J]. Artificial Intelligence, 1999, 440:433.

[26] Olson C F, Matthies L H, Schoppers M, et al. Robust Stereo Ego-motion for Long Distance Navigation[J]. 2000, 2:453-458 vol.2.

[27] Harris C G, Pike J M. 3D positional integration from image sequences[J]. Image & Vision Computing, 2013, 6(2):87-90.

[28] Shi, J, Tomasi, C. Good features to track[C]. Computer Vision and Pattern Recognition, 1994. CVPR 1994. Proceedings of the 1994 IEEE Computer Society Conference on, 1994:593-600.

[29] Rosten E, Drummond T. Machine Learning for High-Speed Corner Detection[C], European Conference on Computer Vision. Springer-Verlag, 2006:430-443。

[30] Bay H, Ess A, Tuytelaars T, et al. Speeded-Up Robust Features (SURF)[J]. Computer Vision & Image Understanding, 2008, 110(3):346-359.

[31] Agrawal M, Konolige K, Blas M R, et al. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching[J]. European Conference on Computer Vision, 2008: 102-115.

[32] Siegwart R, Nourbakhsh I R, Scaramuzza D, et al. Introduction to Autonomous Mobile Robots, second edition[J]. Mit Press, 2011, 2(4):645 - 649.

[33] Wikipedia contributors. Graphics processing unit [EB/OL]. https://en.wikipedia.org/w/index.php?title=Graphics_processing_unit&oldid=721153681, 2016-3-25.

[34] Absurdwho. GPU 概念[EB/OL]. http://www.docin.com/p-516164711.html, 2016-3-25.

[35] Changchang Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT) [EB/OL]. http://cs.unc.edu/~ccwu/siftgpu, 2016-2-1.

[36] G. Ziegler and et al. GPU point list generation through histogram pyramids[R]. Saarbrücken: Max-Planck-Institut für Informatik, 2006.

[37] Wikipedia contributors. Epipolar geometry — wikipedia, the free encyclopedia [EB/OL].

[38] Hartley R, Zisserman A. Multiple View Geometry in Computer Vision[M]: Epipolar Geometry and the Fundamental Matrix pp. 239-261. Second edition. Cambridge University Press, 2004.

[39] Jianxiong Xiao. Multi-view 3d reconstruction for dummies [R]. Princeton University, 2016.

[40] Wikipedia contributors. Skew-symmetric matrix — wikipedia, the free encyclopedia[EB/OL].https://en.wikipedia.org/w/index.php?title=Skew-symmetric_matrix&oldid=709299705, 2016-3-22.

[41] Wikipedia contributors. Camera matrix — wikipedia, the free encyclopedia[EB/OL]. https://en.wikipedia.org/w/index.php?title=Camera_matrix&oldid=693428164, 2015-12-27.

[42] Ramani Duraiswami. Camera Calibration - UMIACS [R]. University of Maryland

Institute for Advanced Computer Studies, 2000.

[43] Bradknox. Wiki: camera_calibration/Tutorials/MonocularCalibration[EB/OL]. http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration , 2015-3-7.

[44] Longuet-Higgins H C. A computer algorithm for reconstructing a scene from two projections[J]. Nature, 1981, 293(5828):133-135.

[45] Q T Luong. Matrice fondamentale et auto-calibration en vision par ordinateur[D]. PhD thesis, Universite de Paris-Sud, Orsay, 1992.

[46] Nister D. An efficient solution to the five-point relative pose problem[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(6): 756-777.

[47] Li H, Hartley R. Five-Point Motion Estimation Made Easy[J]. International Conference on Pattern Recognition Hong Kong, 2006, 1:630-633.

[48] Nister D. An efficient solution to the five-point relative pose problem[C]. IEEE Computer Vision and Pattern Recognition, 2003(2): 195–202.

[49] David A Cox, John Little, O'Shea D. Using algebraic geometry[M]. 2nd Edition. Springer Verlag, 1998.

[50] Strutz T. Data Fitting and Uncertainty[M]. Vieweg+Teubner, 2016.

[51] Wikipedia contributors. Ransac — wikipedia, the free encyclopedia[EB/OL]. https://en.wikipedia.org/w/index.php?title=RANSAC&oldid=709504871, 2016-3-11.

[52] Martin A Fischler, Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography[M]. Morgan Kaufmann Publishers Inc. 1987:726-740.

[53] P. D. Kovesi. MATLAB and Octave Functions for Computer Vision and Image Processing [EB/OL]. http://www.peterkovesi.com/matlabfns/, 2016-2-23.

[54] Jose Luis Blanco. RANSAC C++ examples[CP]. http://www.mrpt.org/tutorials/programming/maths-and-geometry/ransac-c-examples/, 2016-4-3.

[55] Z. Yaniv. Random sample consensus (ransac) algorithm, a generic implementation[J]. Insight Journal, 2010.

[56] 王文斌，刘桂华，刘先勇,等．本质矩阵五点算法伪解的两种剔除策略[J]．光電工程, 2010, 37(8):46-52.