Doctoral Dissertation

Inclination Measurement for Industrial Assembly Platforms Based on Computer Vision

September 2022

Doctoral Program in Advanced Electrical,

Electronic and Computer Systems

Graduate School of Science and Engineering

Ritsumeikan University

MENG Zelin

Doctoral Dissertation Reviewed
by Ritsumeikan University

Inclination Measurement for Industrial Assembly Platforms Based on Computer Vision
（コンピュータビジョンに基づく産業用組立プラットフォームの傾斜測定手法）

September 2022

2022 年 9 月

Doctoral Program in Advanced Electrical, Electronic and Computer Systems

Graduate School of Science and Engineering

Ritsumeikan University

立命館大学大学院理工学研究科

電子システム専攻博士課程後期課程

MENG Zelin

メン ゼリン

Supervisor：Professor TOMIYAMA Hiroyuki

研究指導教員：冨山　宏之教授

Abstract

The research and application of computer vision and artificial intelligence (AI) in the direction of industrial automation is a hot topic in the current scientific and technological circles. Because it is of great theoretical and practical significance to flexibly apply this technology to further improve the intelligence of manufacturing automation. Based on the basic principles of computer vision, combined with motion estimation technologies and AI algorithms, this study implements the methods for the inclination angle measurement of the bearing assembly platforms in manufacturing industry. The main works of this study are as follows.

For the motion estimation of the assembly platform, we illustrated and determined the mathematic expression of motion, that is, the motion of the camera coordinate system relative to the world coordinate system is expressed from the two dimensions of rotation and translation. Then, we demonstrated the coordinate transformation model from the pixel coordinate system to the world coordinate system. Afterward, considering that the camera will introduce distortions and influence the performance of the vision algorithms, we calibrated the industrial camera, and then obtained the camera's intrinsic parameter matrix.

Considering that the current project has high requirements on the accuracy and processing speed of the algorithm, as well as the computing power of the computing platform, we conducted many comparative experiments and selected the algorithm combinations of image features and feature matching approaches which are suitable for the project. To further reduce the false matching rate of feature point pairs, we applied a screening strategy based on the non-maximum suppression method. By adding the screening conditions, only a certain number of best matching point pairs are retained, which saves the computing resources and shortens the overall running time of the algorithm, and at the same time improves the accuracy of the algorithm towards motion estimation. In specific, in the evaluation experiments, 6.7% of the total matched point pairs have been eliminated when applying the screening strategy. Under the circumstance that without adding the screening conditions, the maximum good match ratio could only reach 80%, while the evaluation experiments show that the ratio of the good match reaches 92% in the case of applying the non-maximum suppression strategy with the tuned parameters.

For the motion estimation-based algorithm, we compared several mainstream motion estimation algorithms including feature matching-based, and feature tracking-based methods, etc. In specific, we implemented the perspective-n-point (PnP) method to realize the inclination measurement. In addition, we have tuned the parameters of the optimization algorithm and obtained the optimal parameters through a large number of experiments, making it more suitable for the current research project. The optimized PnP algorithm minimizes the estimation error of the inclination measurements. Evaluation results show that the time efficiency of the proposed system achieves 7.3% higher than the conventional epipolar-constraints-based ones. On the other hand, the implemented system significantly reduces the measurement error by 90% compared with the conventional epipolar constraints-based methods.

Regarding the machine learning-based algorithms for inclination angle classification, we designed a shallow neural network model based on a multilayer perceptron and determined the optimal structure of the model through extensive experiments. In addition, we have also implemented and verified the performance of other machine learning algorithms and deep learning algorithms as the baseline for this project. Through many comparative experiments, we verified the effectiveness and reliability of the designed lightweight neural network, as well as its advantages in the field of industrial automation, especially in this research project. In contrast to those traditional methods, validation experiments certificate that the proposed method achieves the best performance while reducing computational complexity by 45.31%. In addition, the validation accuracy of the designed neural network with tuned parameters reaches up to 98%, which outperforms other baseline models.

Contents

Chapter 1 Introduction

1.1 Research Background and Motivation

Machine vision has a history of more than 50 years from its appearance to the present. Currently, machine vision is widely used in the electronics manufacturing industry. Such as the assembly of printed circuit board. In addition, machine vision systems have also been widely used in many aspects of quality inspection and occupy a very important position in the field of intelligent manufacturing. Machine vision is an important link to improve the level of industrial automation. The software products and hardware products of machine vision are gradually becoming the core systems in all stages of the industrial processing and manufacturing process. Both users and vision system hardware suppliers are paying more and more attention to machine vision system, and hope that machine vision products use more "standardized technology", that is, more compatibility and openness, and can be developed multiple times according to customer requirements.

At present, with the increasing progress of science and technology, industrial robots have been widely used, and are developing in the direction of high speed, high precision, high flexibility, and high reliability. In industrial production, robots have been widely used in painting, welding, disassembly, palletizing, handling, packaging, and other operations [1]. As an automation system solution with low cost and simple system structure, industrial robots can be used in common industrial fields such as assembly, sorting, packaging, and handling while replacing labor and improving production efficiency [2]. Thus, the intelligent automation systems have significant application value. Besides, relying on its simple structure, low cost, fast running speed, suitable for mass production and other advantages, it occupies many processing and manufacturing markets [3]. With the continuous improvement of the quality of automobiles and the expansion of production scale, while on the other hand the production processes require fast processing speed and relatively high production efficiency, industrial robots have almost become the most important part of such product manufacturers [4, 5]. Because industrial robots can quickly and efficiently realize automatic grasping and installation of workpieces such as bearings, long shafts, and metal plates [6, 7]. However, we found that at present, there are still many areas that need to be improved in the production process of industrial robots.

There is an automobile gearbox manufacturing plant, which is mainly responsible for the manufacture of truck gearbox assembly modules, that is, assembling various parts and completing the assembly of the gearbox. The gearbox consists of an aluminum housing, many gears and bearings. As mentioned earlier, nearly a hundred different parts are installed one by one in the metal housing. Nowadays, the factory mainly uses industrial robots to assemble truck transmissions. The use of industrial robots not only improves the efficiency of assembly process, but also reduces the labor intensity of engineers while improving the quality of products.

Although the product quality has been improved a lot, there are still some problems in the manufacturing process [8, 9]. One of the most notable problems is: A small number of products were found to be excessively worn [91, 92]. Through analysis, engineers eventually found that the inclination between the assembly platform and the ground was the cause of the problem. During the bearing assembly process, since the robot does not have the ability to detect whether the assembly platform is inclined, when the assembly platform has some inclination due to mechanical vibration, it will affect the accuracy of the robot to install the workpiece to the designated position. In the case of a more serious inclination angle, there will even be mechanical collisions, which will reduce the yield in the assembly process. Even if the inclination angle is small, it is easy to cause some damages to the products during the installation process and reduce the service life of the bearings. One of the solutions is to analyze and judge the inclination of the assembly platforms by employing the machine vision system, so as to eliminate the possible damages to the products during the bearing assembly process as much as possible, prolong the service life of the product and improve the yield rate [10]. Therefore, before adjusting the pose of the assembly platforms, the inclination angle must be measured. As mentioned above, our research group proposes a few methods to solve the problem of inclination measurement.

1.2 Main Contributions of the Study

The main contributions of this study are as follows.
(1) The Overall Design of the Inclination Angle Measurement System for the Assembly Platforms
One of the main works of this research is to design and develop an inclination angle measurement system that suitable for the assembly platforms, hereinafter referred to as the measurement system. The basic workflow of the measurement system is as follows: First, the image of the assembly platform is captured by the camera and sent to a dedicated image processing system. After that, the image processing system performs preprocessing, target recognition, feature extraction, and other operations on the collected image information to obtain the spatial pose information/inclination angle parameters of the assembly platforms. Finally, the image processing system converts the obtained inclination information into control signals and sends them to the lower

controller such as Programmable Logic Controller (PLC). The controller controls the industrial robot system to complete the corresponding adjustment according to the received control parameters.

(2) Research on the motion estimation algorithms for the assembly platforms based on simultaneous localization and mapping technologies
The working principle of machine vision and the corresponding coordinate system transformation relationships are analyzed and studied. Besides, several motion estimation algorithms based on simultaneous localization and mapping (SLAM) technologies for the assembly platforms are implemented.

(3) Research on the inclination angle classification algorithms for the assembly platforms based on machine learning
The current mainstream machine learning algorithms and neural network structures are analyzed and studied, and a neural network structure for the inclination angle classification for the assembly platforms is proposed and validated.

(4) Analysis of the experimental results of the proposed algorithms
According to the performance of different algorithms in the verification experiments utilizing the image dataset created by our group, the advantages and disadvantages of the implemented algorithms are analyzed and discussed, so as to provide a reference direction for the future research.

1.3 Organizations of the Thesis

The organization of this study is as follows. In Chapter 2, we demonstrated the overall design of the inclination measurement system. Besides, we demonstrated the dataset created for this study. In Chapter 3, we illustrated the mathematical expressions of motion as we as calibration methods to deal with the distortion problem of the industrial camera. In Chapter 4, we illustrated the proposed motion estimation-based method for the project of inclination measurement. In Chapter 5, we explained the proposed neural network for the project of inclination measurement. In Chapter 6, we made a summary of this study and discussed some future directions of our research.

1.4 Chapter Summary

In this chapter, we firstly introduced the research background and motivation of the measurement system for the assembly platform. Secondly, we briefly illustrated the design ideas of the measurement system as well as its workflow. Finally, we summarized the main works in this study and illustrated the overall organizations of the thesis.

Chapter 2 Inclination Measurement and Industrial Assembly Platforms

2.1 Introduction

The inclination angle measurement system for the industrial assembly platform developed in this research, hereinafter referred to as the measurement system, is to first transmit the image captured by the industrial camera to the industrial computer, and then use the powerful computing power of industrial computer to analyze the image. After that, the inclination angle information of the assembly platform is converted into control signals and transmitted to PLC for controlling the actuators. Finally, the actuator arrives at the designated position and accomplish the production operation according to the guidance of the control signals.

In this study, the measurement system is divided into two parts: hardware system and software system. The hardware system part is responsible for real-time image acquisition and communication with the lower controller/actuator, and the software system part is responsible for the core image processing and analysis, and then transfer the results, such as inclination angle information of the platform, to the control signals. In this chapter, the structure and functions of the measurement system will be designed and implemented in accordance with the project requirements.

2.2 Overall Design of the Measurement System

Basically, the computer vision system designed for inclination measurement of the industrial assembly platform will be provided with the functions of image acquisition, image analyzing and communication towards the lower controller. Besides, it has several main technical requirements considered in the design of general vision systems. For one thing, the design must have reliability. Because the measurement system is mainly used in various industrial scenarios, it must withstand external disturbances under various non-ideal conditions and run stably. Second, the design should consider the processing speed of the system, to ensure that the production efficiency is improved in the case of replacing labor. Third, universality. At present, most industrial vision systems on the market can only meet a single or a few scenarios, and the ability of secondary development is insufficient. Thus, it is particularly important to develop a vision system that meets the needs of a variety of industrial scenarios. In this way, according to the above requirements, the overall structure of the measurement system designed in this project is shown in Figure 2-1. The system consists of the upper computer and the lower controller. In specific, the upper computer includes the image acquisition unit, the image processing unit, and the communication unit according to the required functions. Among them, hardware of the system mainly includes an industrial camera used for image acquisition, an industrial computer for data analyzing, and peripheral interfaces that support communications between devices. The software system mainly includes the proposed algorithms for data analyzing. Furthermore, realization and running of the proposed algorithms leave without the following environments: Windows10 operating system running on the industrial computer platform, the integrated development environment (IDE) Anaconda, the open-source computer vision library (OpenCV), and the general machine learning libraries of TensorFlow, Keras, and scikit- learn. When the system is working, the industrial camera continuously captures images of the assembly platform and transmits the image data into the image processing unit (PC) through the USB interface. By employing the proposed algorithm, analyzing results such as information about inclination angle can be obtained. Then, the system will convert the analyzing results into corresponding control signals and transmit them to the lower controller. The lower controller will make controls and drive the servo motors (actuators) to adjust the pose of the assembly platform in accordance with the control signals. Finally, the assembly platform will be guided to the optimal working position.

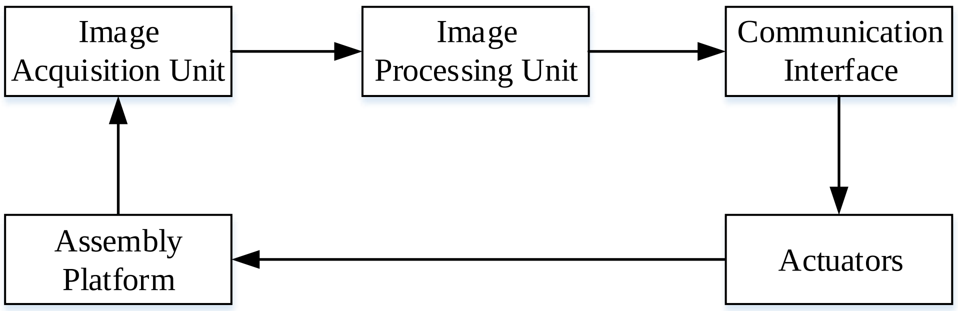


Figure 2-1. Overall design of the measurement system

The main purpose of the measurement system is to use the computer vision techniques to monitor the inclination status of the industrial assembly platform in real-time, so as to avoid slight tilting of the assembly platform due to the on-site mechanical vibration, which causes slight damage to the product during the assembly process and therefore affects the service life of the product. There are many types of visual systems. For example, the camera and the robot arm form a hand-eye system. The hand-eye system includes two main types: eye-in-hand and eye-to-hand. For the former configuration, camera is mounted at the end of the robot arm (end-effector) and moves with the robot during the robot's operations, as shown in Figure 2-2(a). As for the configuration of the latter type, camera of the eye-to-hand system is mounted in a fixed position outside the robot’s body and does not move with the robot during operations, while it able to clearly record the assembly process at the same time. In this study, considering the camera must be able to watch the overall statues of the assembly platform, the eye-to-hand configuration will make it easy to capture the images of assembly platform at any time. Thus, we consider the settings of the assembly platform, robot arm, and camera as a standard eye-to-hand configuration, refers to Figure 2-2(b) below.

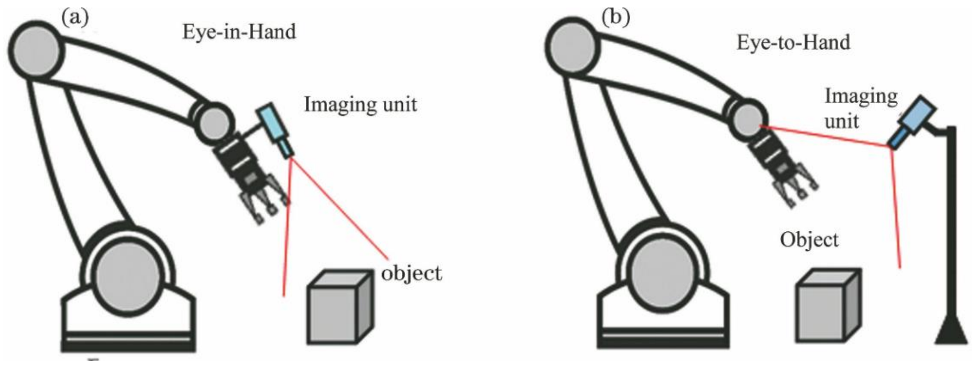


Figure 2-2. Eye-in-hand and eye-to-hand configuration [23]

2.3 Hardware System Design and Implementation

2.3.1 Design of Image Acquisition Module

The image acquisition module mainly includes camera lens, photosensitive component (Sensor) and communication interface. Its working principle is: the light reflected from the surface of the external object is transmitted to the photosensitive components through the camera lens, and the photosensitive components generate corresponding charges according to the intensity of the light, and after processing, the image information is output to the display device through the communication interface.

(1) Selection of lens

The lens of camera is mainly divided into two types. One is Fixed Focus lens; the other is Zoom Focus lens. Fixed Focus lens means that the focal length f of the lens has been fixed at the time of manufacturing. According to the size of the focal length, it can be divided into standard lens, wide-angle lens, fisheye lens and so on. Zoom Focus lens achieves zoom by actively or passively adjusting the distances among multiple optical lenses in the lens module by a mechanical device around the lens. In addition, there is a special camera lens called the focusing lens. The working principle of the focusing lens is to adjust the distance manually or automatically between the optical center of Fixed Focus lens and the photosensitive plane of the sensor, so that the optical center of the incident light accurately falls on the photosensitive plane. As a result, we could obtain the clearest imaging picture. According to the principle of maximizing the pixel gradient of images, the automatic focusing lens is adjusted in a closed-loop manner by employing an image processing chip (such as DSP, etc.) to control the VCM voice coil motor to drive the lens and achieve focusing. This kind of lens is widely used in digital cameras and smart phones.

(2) Selection of photosensitive components

A photosensitive component is a kind sensor that able to converts optical signals into electrical signals. It is mainly divided into two categories, CCD, and CMOS. The difference between the two is mainly reflected in several aspects such as sensitivity, cost performance, noise, and power consumption. First, in terms of sensitivity, since each pixel of CMOS contains an amplifier and an A/D conversion circuit, the acceptable light area per pixel per unit area is lower than that of CCD. Second, in terms of manufacturing price, CMOS adopts the MOS process, which has been improved in the semiconductor industry for many years and has become the mainstream process in the industry. The quality failure rate and manufacturing cost of CMOS are far lower than CCD. CCD transfers all the generated charges to the external amplifier circuit by setting up a separate charge transfer channel. Once the photosensitive pixel unit is damaged, a large amount of charge in the channel will not be output, making the product quality requirements of CCD higher than that of CMOS. Besides, the additional ADC and transmission channel make the cost of CCD higher than that of CMOS. Third, in terms of noise, since each CMOS pixel unit is equipped with an ADC amplifier, if there are millions, it is difficult to ensure that there is no difference between all amplifiers, which leads to a worse effect of amplification and synchronization than the CCD with a single amplifier. Therefore, the noise generated when using CMOS will be relatively high. Finally, in terms of power consumption, the charge generated by each photosensitive element of CMOS is directly amplified and output by the amplifier around the pixel, and the power consumption is low; for CCD, a higher voltage (12V) must be generated by setting the voltage circuit, so that the charge of each pixel can be transferred to the transfer channel under the guidance of potential energy. Thus, the power consumption of CCD will be higher than that of CMOS due to the additional voltage applied.

Figure 2-3. Industrial camera with USB3.0 cable

(3) Selection of communication interface

There are some commonly used communication interfaces for the industrial cameras, including Gigabit Ethernet, USB, Thunderbolt, and Camera Link. Besides, the Gigabit Ethernet and USB are widely adopted in the manufacturing industry due to its high compatibility as well as the long transmitting distance.

Table 2-1. Mainstream communication interfaces for transmitting image signals

Types of Interface	Transmission Speed (MB/s)	Transmission Distance (m)	Advantages	Disadvantages
GigE	1000	100	High Speed; Long Transmission Distance;	High CPU Usage;
USB	60-1200	50	High Speed; Support UVC; Low Price; Widely Used;	Limited Transmission Distance;
Camera Link	875	30	High Speed; High Reliability;	High Price;
FireWire	200	10	Support Multi-camera Synchronization;	Has been Eliminated by the Thunderbolt Series Interface;
Thunderbolt	5000	5	Very High Speed; Future Trend;	High Price; Not Yet Fully Ubiquitous;

According to Table 2-1, comprehensively considering with the needs of the project and the peripheral interface information of the PC, USB3.0 interface standard that supports USB Video Class (UVC) protocol is selected as the communication interface for the industrial camera and the upper computer. UVC protocol is currently supported by most operating systems. Its biggest feature is that it can be plug-and-play in a computer or an embedded device with an operating system without installing any driver, which greatly improves the versatility of the embedded device. In terms of the above analysis, as well as considering the cost and performance, we finally choose the autofocus industrial camera UC60 manufactured by MOKOSE company, which contains an 8-megapixel CMOS sensor, and supports USB3.0 communication interface, as shown in Figure 2-3.

2.3.2 Selection of PC

To implement and evaluate the proposed measurement system, the prototype work is implemented on PC, which serves as the computing and processing unit. Moreover, configurations of the hardware employed in the evaluation experiments is as follows: Mother Board with Z490 Chipset, CPU(i7-10700K), GPU(GTX1080Ti), and memory(64GB). One of the main reasons that we choose Z490 Chipset is that it provides enough USB ports, including USB2.0 and USB3.0 ports. Since the industrial camera support USB3.0, while USB ports as well as RJ45 ports are commonly used to transmit the control signals to PLC, sufficient resources of USB/RJ45 ports will benefit the overall design of the hardware system. On the other hand, considering that the data analyzing tasks require a relatively high computing power, we finally decided to choose the combination of CPU(i7-10700K) and GPU(GTX1080Ti) to fulfill the requirements while keep the system with sufficient redundant resources to a certain extent.

2.3.3 Communication System Design of the Control System

A PROFINET communication port is integrated on the S71200CPU body, which supports Ethernet and communication standards based on TCP/IP and UDP [27-28]. The communication port can realize the communication between the S7-1200CPU and the upper computer which is responsible for image processing and analysis) through a standard Ethernet cable.

Modbus is an industrial field bus standard, while ModbusTCP protocol is based on Ethernet communication. In specific, through running on TCP/IP, ModbusTCP can achieve message transmission between upper computer and PLC. Furthermore, the ModbusTCP library file is integrated in S7-1200CPU. Therefore, the library instructions "MB_CLIENT" and "MB_SERVER" of ModbusTCP could be directly called to realize the Modbus communication function in real applications. In this case, our study utilizes ModbusTCP to realize the communication among PLC, the upper computer, and actuators (servo motors).

2.4 Software Implementation of the Measurement System

All algorithms in this study are implemented by using Python. Anaconda is an open-source Python language distribution for data science, machine learning, and predictive analytics. It is dedicated to simplifying the package management system and deployment. Furthermore, Anaconda utilize the package management system named Conda. Conda is an open-source and cross-platform package/environment management system. Developers not only can use Conda to install and upgrade package dependencies, but also can easily install different versions of package environments and quickly switch between different environments. In addition, Anaconda Navigator is an user-friendly graphical interface of Anaconda, and it includes the following applications: Jupyter Notebook, Spyder, QtConsole, Glueviz, Orange, Rstudio, and Visual Studio Code. Among them, Spyder is an open-source integrated development environment (IDE) using Python language for scientific computing. Due to it integrates NumPy, SciPy, Matplotlib and IPython and is very convenient to use, this study uses Spyder as the IDE. In summary, we utilize Anaconda and its accompanying tools to configure the software development environment.

The configuration process of the software development environment is as follows. On the selected PC as mentioned before, we install the Windows10 operating system at the beginning. Then we install Anaconda and utilize Conda to create the environments for developing the dedicated algorithms of this study. In specific, we need to install the OpenCV library for image processing, as well as the dependent libraries for implementing neural networks and machine learning algorithms, including TensorFlow, Keras and the general machine learning library scikit-learn.

In this study, the overall design of the software system consists of three main partitions, including “Loading image data”, “Image anlysis”, and “Generating control signal”. In specific, after loading the input image data, it will conduct data preprocessing at the beginning, including feature extraction etc. Second, the pre-processed data will be used for analyzing and determine the inclination angle of assembly platform by applying the proposed algorithms. Finally, the analysis results will be converted as control signals and then sent to the industrial actuators. Whenever the control signals are received by PLC, it will immediately make controls for actuators such as servo motors, to adjust the pose of the assembly platform. In this way, the assembly platform could sustain an optimal working position. Furthermore, for the “Image analysis” part, it will apply different algorithms to achieve the aim of analyzing. In details, SLAM-based method and Artificial Intelligence-based method are separately implemented to accomplish the analyzing tasks. Workflow of the software system is demonstrated in Figure 2-4.

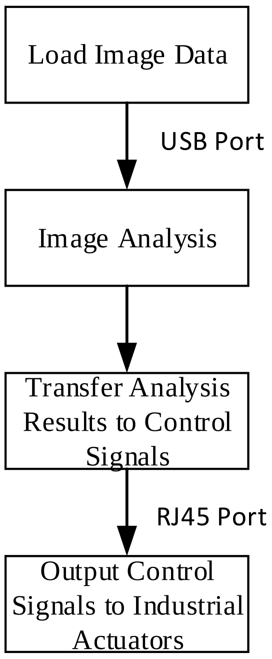

Load Image Data
USB Port
Image Analysis
Transfer Analysis Results to Control Signals
RJ45 Port
Output Control Signals to Industrial Actuators

Figure 2-4. Workflow of the software system

2.5 Establishment of the Image Dataset

In this study, the image data used for the validation experiments were all taken at the industrial site. Specifically, pictures of the assembly platform at different inclination angles are taken by an industrial camera. It is worth noting that in the Assembly Platform Dataset, only some simple scenes are currently included. Specifically, the cases when assembly platform inclines by 1°-10° along the horizontal direction (x-axis) have a total of 10 categories. In addition, we captured 100 images for each category. When assembly platform inclines by 1°-10° along the vertical direction (y-axis), there are also 10 categories in total, and 100 images are taken for each category. In the case when no inclination occurs (inclination angle is 0°), we still capture 100 images for this category. Thus, these 2100 images constitute the "Assembly Platform Dataset" named by our group members. Although the current dataset only contains some simple scenarios, its capacity is sufficient to verify the validity and reliability of the models and algorithms proposed in this study. In our future works, we will take more industrial scenarios into account, especially some complex cases.

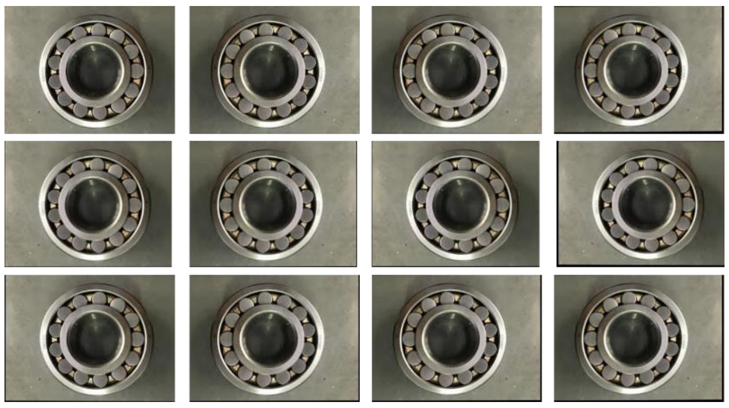


Figure 2-5. Image examples of Assembly Platform Dataset

2.6 Chapter Summary

This chapter designs the overall structure of the measurement system. First, in terms of the computing power and the convenience of experimental verification, the appropriate parameters of the hardware processing platform are selected, and through the comparison of various interfaces and photosensitive elements, an autofocus industrial camera supporting the USB3.0 interface is adopted. Second, the software framework of the measurement system is designed based on the platform, and the specific software and algorithm implementations will be deeply researched and analyzed in chapters 3-6. Finally, we created the specialized image dataset for the validation experiments of our proposed algorithms.

Chapter 3 Cameras and Coordinate Transformation

3.1 Introduction

To estimation the motion of the assembly platform, firstly, it is essential to introduce some mathematics fundamental to express the motion. In this chapter, Euclidean transformation between different coordinates systems has been illustrated. In addition, the other ways to describe rotation, such as rotation vector as well as Euler angle have also been demonstrated.

3.2 Mathematics Fundamental about Expression of Rotation

3.2.1 Euclidean Transformation between Coordinate Systems

Similar to the rotation between vectors, we can also describe the rotation and translation relationship between two coordinate systems, collectively referred to as the transformation relationship between coordinate systems [16]-[20]. In describing the motion of the industrial camera adopted in the inclination measurement system, we at first establish a world coordinate system defined by $\mathrm{x_W}, \mathrm{y_W}$ and $\mathrm{z_W}$ as shown in Figure 3-1. Second, we establish a camera coordinate system defined by $\mathrm{x_C}$, $\mathrm{y_C}$ and $\mathrm{z_C}$. Besides, assuming there is a vector $\boldsymbol{p}$ with coordinates p_c and p_w, which describing its positions in the pre-established camera and world coordinate system, respectively. To describe the transformation between p_c and p_w, we define the matrix T, as shown in Figure 3-1.

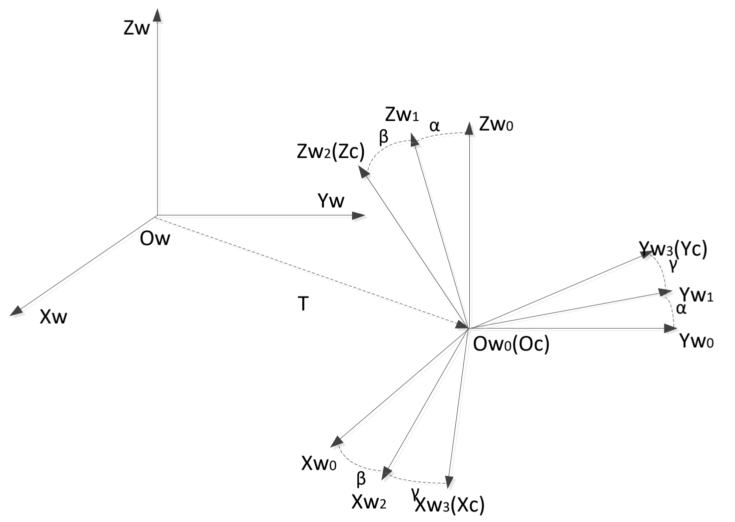


Figure 3-1. Transformation relationship between world and camera coordinate system

Camera motion is a rigid body motion that guarantees that the length and angle of the same vector will not change in each coordinate system. This transformation is called the Euclidean transformation, which consists of two parts, the rotation and translation. Firstly, the rotation is considered. Hypothesize that a unit orthogonal base (e_1, e_2, e_3) to changes to (e'_1, e'_2, e'_3) after rotation transformation. Similarly, the coordinates of a vector will be $[a_1, a_2, a_3]^T$ and $[a'_1,\ a'_2, a'_3]^T$ in the camera and world coordinate systems, respectively. According to the definition of coordinates, there are:

$$[e_1 \quad e_2 \quad e_3]\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [e'_1 \quad e'_2 \quad e'_3]\begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} \qquad (3-1)$$

Through applying some mathematic tricks, easily we can obtain the formula below:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} e_1^T e'_1 & e_1^T e'_2 & e_1^T e'_3 \\ e_2^T e'_1 & e_2^T e'_2 & e_2^T e'_3 \\ e_3^T e'_1 & e_3^T e'_2 & e_3^T e'_3 \end{bmatrix}\begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} \triangleq \boldsymbol{Ra'} \qquad (3-2)$$

We define the middle array as the rotation matrix $\boldsymbol{R}$ since it mathematically describes the rotation transformation between the two coordinated systems. Matrix $\boldsymbol{R}$ is an orthogonal matrix which determinant is of 1. Thus, the set of rotation matrices as can be given as:

$$SO(n) = \{\boldsymbol{R} \in \mathbb{R}^{n \times n}\ \boldsymbol{RR}^T = \boldsymbol{I}, det(\boldsymbol{R}) = 1\} \qquad (3-3)$$

$SO(n)$ represents the Special Orthogonal Group. It contains the rotation matrices of a n-dimensional space. In other words, the rotation matrices can describe the rotation of the camera, as well as the rotation of the industrial assembly platform, since the inverse/transpose of the orthogonal matrix $\boldsymbol{R}$ represents an inverse motion in kinematics, expressed by:

$$\boldsymbol{a'} = \boldsymbol{R}^{-1}\boldsymbol{a} = \boldsymbol{R}^T\boldsymbol{a} \qquad (3-4)$$

Where $\boldsymbol{R}^T$ portrays an inverse motion. Except for the rotation matrix $\boldsymbol{R}$, translation vector $\boldsymbol{t}$ is essential to describe the transformation. Hypothesize in the established world coordinate system, there is a vector $\boldsymbol{a}$. Thus, we could apply the above-mentioned rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$, a 3×1 vector to describe the motion from vector $\boldsymbol{a}$ to $\boldsymbol{a}'$ as the following:

$$\boldsymbol{a}' = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{t} \qquad (3-5)$$

In summary, through applying the rotation and translation matrix, we could describe the inclination motion of the industrial assembly platforms. In specific, whenever we estimated the motion of our industrial camera, the inclination motion of the assembly platforms can be obtained based on the inverse kinematics.

3.2.2 Transformation Matrix and Homogeneous Coordinates

Although Equation (3-5) can describe the motion in a mathematical way, it still have some shortcomings. In practice, no less than one time of motion needs to be described. For example, assuming that there are two transformations: R_1, t_1 and R_2, t_2, satisfying:

$$\boldsymbol{b} = \boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1, \boldsymbol{c} = \boldsymbol{R}_2\boldsymbol{b} + \boldsymbol{t}_2 \qquad (3-6)$$

But the transformation from **a** to **c** is:

$$\boldsymbol{c} = \boldsymbol{R}_2(\boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1) + \boldsymbol{t}_2 \qquad (3-7)$$

Such a form would become too complicated when making several times of transformations. In this case, homogeneous coordinates will be helpful to solve this problem and to rewrite $(3-5)$:

$$\begin{bmatrix} \boldsymbol{a}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \triangleq \boldsymbol{T} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \qquad (3-8)$$

By adding the last dimension, we describe a three-dimensional vector with four real numbers, which obviously has one more degree of freedom, but allows us to write the transformation in a linear form. In homogeneous coordinates, each component of a point *x* is multiplied by a non-zero constant k and still represents the same point. Therefore, the specific coordinate values of a point are not unique. For example, $[1, 1, 1, 1]^{\mathrm{T}}$ and $[3, 3, 3, 3]^{\mathrm{T}}$ are the same point. But when the last item is not zero, we can always divide all the coordinates by the last item, and force the last item to be 1, to get a unique coordinate representation. That is, convert it to non-homogeneous coordinates:

$$\tilde{x} = [x, y, z, w]^T = [\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1]^T \qquad (3-9)$$

At this point, the last item is ignored, and the coordinates of this point are the same as in the 3-dimentional space. Therefore, by applying the homogeneous coordinates, as well as introducing the matrix $\boldsymbol{T}$, we could describe the motion in a more intuitive way:

$$\tilde{\boldsymbol{b}} = \boldsymbol{T}_1\tilde{\boldsymbol{a}},\ \tilde{\boldsymbol{c}} = \boldsymbol{T}_2\tilde{\mathbf{b}} \Rightarrow \tilde{\boldsymbol{c}} = \boldsymbol{T}_2\boldsymbol{T}_1\tilde{\boldsymbol{a}} \qquad (3-10)$$

Hereafter, the formulations will utilize the homogeneous coordinates such as $\boldsymbol{b} = \boldsymbol{T}\boldsymbol{a}$, by default.

Regarding the introduced matrix $\boldsymbol{T}$, it is the so called Special Euclidean Group $SE(3)$.

$$SE(3) = \left\{T = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid \mathbf{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \right\} \qquad (3-11)$$

As with $SO(3)$, by solving the inverse of $\boldsymbol{T}$, we can obtain an inverse motion:

$$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \qquad (3-12)$$

Finally, in order to keep the symbol concise, without ambiguity, we will not distinguish the homogeneous coordinates from the non-homogeneous coordinates hereafter. By default, we use the one that conforms to the algorithm. For instance, in the formula $\boldsymbol{b} = \boldsymbol{Ta}$, homogeneous coordinates will be utilized, while in the case of $\boldsymbol{a}' = \boldsymbol{Ra}$, non-homogeneous coordinates will be adopted.

3.2.3 Rotation Vector and Euler Angle

3.2.3.1 Rotation Vector

Although it is clear to use the transformation matrix to describe the inclination motion of the industrial assembly platforms, it still has some disadvantages:

(1) The $SO(3)$ contains nine elements, but only three-degree-of-freedom describe the rotation. In this way, it seems somehow redundant.
(2) The rotation matrix $\boldsymbol{R}$ must be orthogonal and therefore its determinant must equal to 1. The constrained conditions will introduce additional difficulties in motion estimation and optimization.

To this end, we desire a much simpler method to solve the problem of describing motion. For the rotation of the coordinate system, the so-called rotation vector will be another choice. In this way, the rotation will be expressed by a 3×1 vector. Thus, we can express the motion by using a rotation vector combined with a translation vector.

In fact, the method mentioned above is the Lie algebra that we are going to use in the following chapters. Assuming we have a rotation described by the two parameters $n\ and\ \theta$, thus we can easily obtain θn. Besides, the Rodrigues's Formula can be utilized to calculate the corresponding rotation matrix of an arbitrary rotation vector, vice versa:

$$\boldsymbol{R} = \cos\theta\,\boldsymbol{I} + (1-\cos\theta)\boldsymbol{n}\boldsymbol{n}^T + \sin\theta\,\boldsymbol{n}^\wedge \qquad (3-13)$$

Where the symbol " $\wedge$ " represents the anti-symmetric conversion of a vector.

$$\begin{aligned} tr(\boldsymbol{R}) &= cos\ \theta\ tr(\boldsymbol{I}) + (1 - cos\ \theta)tr(\boldsymbol{nn}^T) + sin\ \theta\ tr(\boldsymbol{n}^\wedge) \\ &= 1 + 2cos\ \theta \end{aligned} \qquad (3-14)$$

Thus,

$$\theta = \arccos\left(\frac{tr(\boldsymbol{R}) - 1}{2}\right) \tag{3-15}$$

As for the rotation axis $\boldsymbol{n}$, because it will not change even applying the rotation transformation, there are:

$$\boldsymbol{Rn} = \boldsymbol{n} \tag{3-16}$$

That is, $\boldsymbol{n}$ is the eigenvector corresponding to case when the eigenvalue of $\boldsymbol{R}$ equal to 1. Besides, we can get $\boldsymbol{n}$ by solving the equation above.

3.2.3.2 Intuitive Description of Motion with Euler Angle

Though the rotation vector has its advantages in describing the motion, it is unintuitive for the researchers to quickly read the information about the estimated motion. To this end, we will introduce another method to describe the motion while in an intuitive manner. That is Euler Angle. Besides, this method which contains three parameters including "yaw angle", "pitch angle", and "roll angle" is frequently applied in the aerospace industry. Furthermore, one of the most common definitions of Euler Angle is to describe a motion with the parameter order of yaw-pitch-roll. In this study, we only utilize Euler Angle to intuitively verify the experiment results.

3.2.4 Quaternion

3.2.4.1 Definition of Quaternion

The above-mentioned mathematical expressions of motion all have their advantages as well as drawbacks. Except for the rotation vector, there is still a commonly used method to describe the inclination motion for this research project, namely, quaternion. The mathematical expressions of the motion using quaternion is both compact and non-singular.

Assuming we have a quaternion q described by the real part q_0 and three imaginary parts i, j, k, thus we have:

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{3-17}$$

Besides, the imaginary parts must satisfy the following constraints:

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases} \tag{3-18}$$

Because of this form of representation, it is equivalent to denote a quaternion like:

$$\boldsymbol{q} = [s, \boldsymbol{v}], s = q_0 \in \mathbb{R}, \boldsymbol{v} = [q_1, q_2, q_3]^T \in \mathbb{R}^{\mathbf{3}} \tag{3-19}$$

Where s denotes the real part, and $\boldsymbol{v}$ denotes the imaginary part of the quaternion. If a quaternary imaginary part is 0, it is called a real quaternion.

Quaternion is very similar to the complex number. Considering that three dimensions require three axes, the quaternion also has three imaginary parts. In fact, a virtual quaternion can also correspond to a spatial point. We know that a complex number with a length of 1 can represent a pure rotation on a complex plane without scaling of the length, therefore the rotation in the same three-dimensional space can also be expressed in units of quaternions.

Suppose there is a rotation vector $\theta\boldsymbol{n}$, while vector $\boldsymbol{n} = [n_x, n_y, n_z]^T$, then the quaternion form of this rotation can be expressed as:

$$\boldsymbol{q} = [\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}]^T \qquad (3-20)$$

Conversely, the corresponding rotation vector can be obtained if we know the expression using quaternion:

$$\begin{cases} \theta = 2\arccos(q_0) \\ [n_x, n_y, n_z]^T = [q_1, q_2, q_3]^T / \sin(\theta/2) \end{cases} \qquad (3-21)$$

Adding *2π* to θ of equation (2-20) represents the same rotation, but the corresponding quaternion becomes $-q$. Therefore, in a quaternion, any rotation can be represented by two quaternions that are opposite to each other. Similarly, taking θ to 0 gives a real quaternion without any rotation：

$$\boldsymbol{q}_0 = [\pm 1,0,0,0]^T \qquad (3-22)$$

3.2.4.2 Motion Description with Quaternion

Assuming we have a three-dimensional point $p = [x, y, z] \in R^3$, and a rotation specified by the axis angle $\theta\boldsymbol{n}$, while the three-dimensional point p becomes p' after being rotated. If a matrix description is used, then $p' = Rp$. In the case when the rotation is described by a quaternion:

$$\boldsymbol{P} = [0, x, y, z] = [0, \boldsymbol{v}] \qquad (3-23)$$

This is equivalent to the fact that we associate the three imaginary parts of the quaternion with the three axes in space. Then, referring to the equation (3-20), we have:

$$\boldsymbol{q} = \left[\cos\frac{\theta}{2}, \boldsymbol{n}\sin\frac{\theta}{2}\right] \qquad (3-24)$$

Thus, this pure rotation transformation will be described as the equation below:

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1} \qquad (3-25)$$

3.2.4.3 Transformation between Quaternion and Rotation Matrix

The transformation between a quaternion and the corresponding rotation matrix has been given in the equation (3-21). So now it seems that the most intuitive way to convert a quaternion into a matrix is to convert the quaternion q to the axis angle $\theta\boldsymbol{n}$, and then convert it to a rotation matrix according to the Rodrigues formula. However, it is more expensive to calculate an “arcos” function. In fact, this calculation can be bypassed by certain techniques:

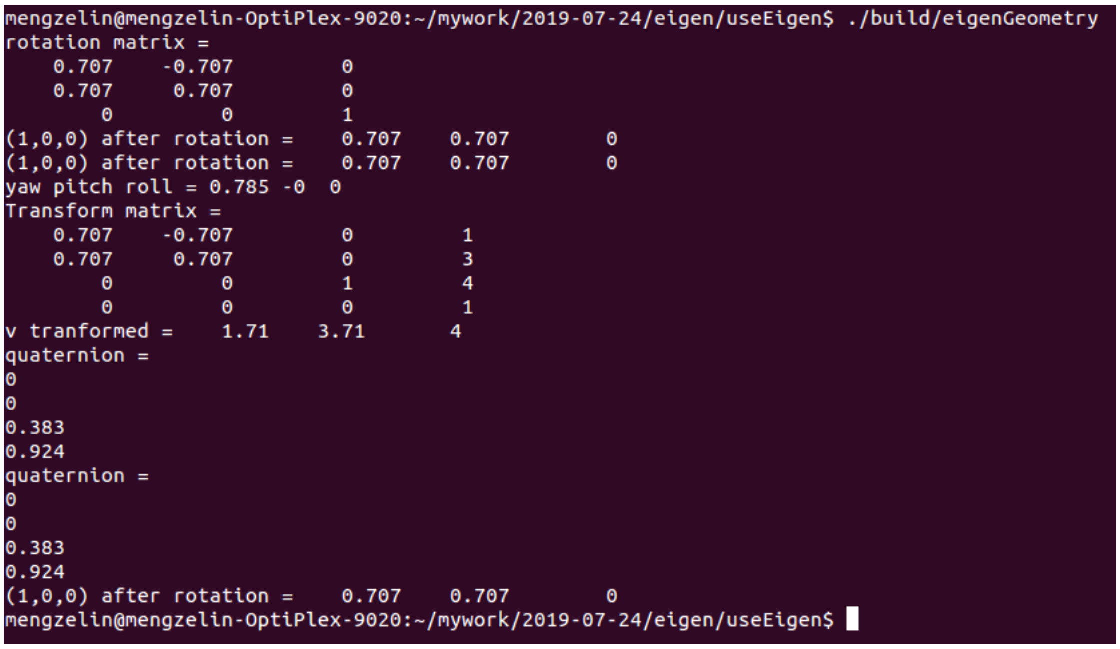
Let quaternion $q = q_0 + q_1 i + q_2 j + q_3 k$, the corresponding rotation matrix $\boldsymbol{R}$ is:

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \qquad (3-26)$$

Conversely, the conversion from the rotation matrix to the quaternion is as follows. The hypothesis that matrix $\boldsymbol{R} = \{m_{ij}\}, i, j \epsilon [1,2,3]$, and its corresponding quaternion $\boldsymbol{q}$ is given by:

$$q_0 = \frac{\sqrt{tr(R)+1}}{2}, q_1 = \frac{m_{23} - m_{32}}{4q_0}, q_2 = \frac{m_{31} - m_{13}}{4q_0}, q_3 = \frac{m_{12} - m_{21}}{4q_0} \qquad (3-27)$$

It is worth mentioning that since $\boldsymbol{q}$ and $-\boldsymbol{q}$ describe the same motion, in fact, the quaternion description about matrix $\boldsymbol{R}$ is not unique. In practice, basically the most convenient form will be applied in this study. In addition, experiment results of testing Eigen library are shown in Figure 3-2. as the following:



```
mengzelin@mengzelin-OptiPlex-9020:~/mywork/2019-07-24/eigen/useEigen$ ./build/eigenGeometry
rotation matrix =
    0.707    -0.707         0
    0.707     0.707         0
        0         0         1
(1,0,0) after rotation =    0.707    0.707        0
(1,0,0) after rotation =    0.707    0.707        0
yaw pitch roll = 0.785 -0  0
Transform matrix =
    0.707    -0.707         0         1
    0.707     0.707         0         3
        0         0         1         4
        0         0         0         1
v tranformed =    1.71    3.71       4
quaternion =
0
0
0.383
0.924
quaternion =
0
0
0.383
0.924
(1,0,0) after rotation =    0.707    0.707        0
mengzelin@mengzelin-OptiPlex-9020:~/mywork/2019-07-24/eigen/useEigen$
```

Figure 3-2. Experiment results of testing Eigen library

3.3 Coordinate Transformation

3.3.1 Camera Model and Coordinate Transformation between World and Camera Coordinate System

Binocular vision technology, also known as stereo vision technology, is to obtain two pictures in different perspectives of an object and extract the depth information through the overlapping parts of the two images to obtain the 3D spatial coordinates of the object. The typical binocular camera is designed from the human visual system. Figure 3-3. shows a typical pin-hole camera model. In Figure 3-3., by placing two cameras of the same size in the horizontal direction, two images are generated. Image processing obtains a disparity map, which in turn recovers the three-dimensional information of the object through the disparity map.

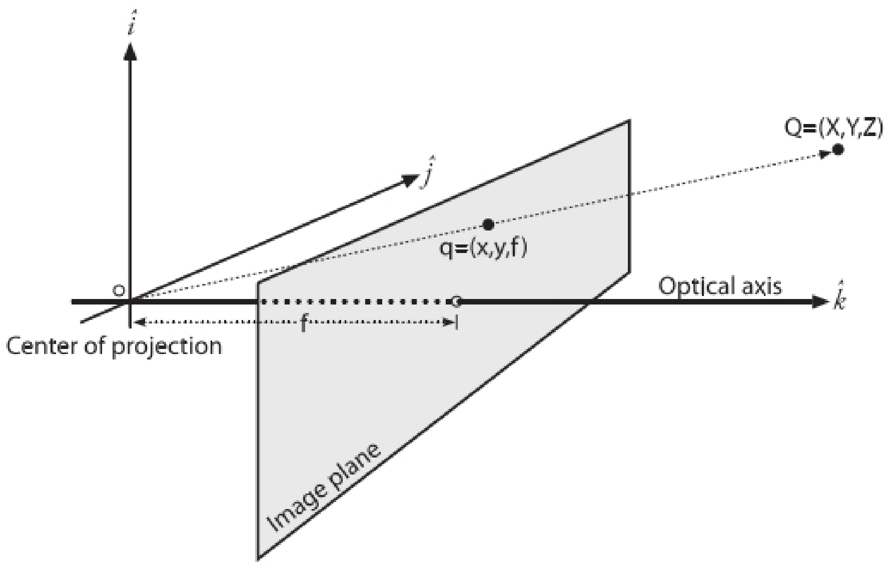


Figure 3-3. Schematic graph of camera pin-hole model

In this section, we will make a review about the transformation relationship between camera and imaging plane coordinate system. To describe the relationship, we establish three-dimensional coordinate system $OX_cY_cZ_c$ as camera coordinate system, as showed in Figure 3-4. Also, we establish the two-dimensional coordinate system O_1XY as imaging plane coordinate system. O_c is optical center of the camera lens. $O_1$is called as the principle point of image, which is the intersection point of camera optical axis's center line, that is the Z_c axis of the coordinate systems of camera and imaging plane. In addition, for one point P in space, which coordinate is (x_c, y_c, S) in the camera coordinate system. The imaging point of P in the imaging plane is $P'(x, y)$. According to the imaging mechanism of camera and properties of similar triangles, we could get the following equations:

$$\frac{x}{f} = \frac{x_c}{S} \quad (3-28)$$

$$\frac{y}{f} = \frac{y_c}{S} \quad (3-29)$$

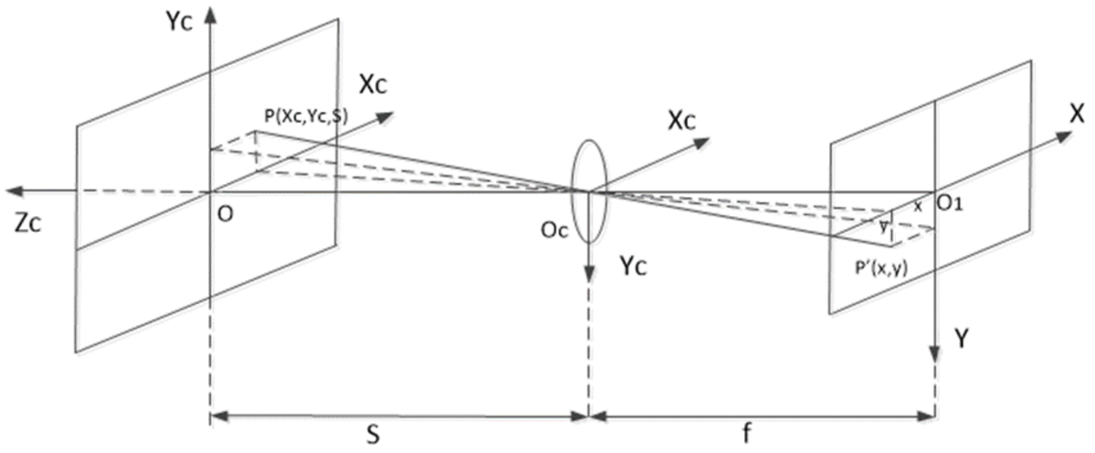


Figure 3-4. Relationship between camera coordinate system and imaging plane coordinate system

In Equation (3-28) and (3-29), S represents the distance between O_c and O, which denotes the origin of the camera coordinate system. f denotes the focal length of the lens. Write it into the matrix form, thus:

$$S\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad (3-30)$$

Equation (3-30) is the matrix expression of the relationship between camera coordinate system and imaging plane coordinate system. Through equation (3-30), we could get the matrix expression of the relationship between the world coordinate system and imaging plane coordinate system:

$$S\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix}\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (3-31)$$

In equation (3-31), (x, y, z) denotes the coordinates of point P in imaging plane, while (X_w, Y_w, Z_w) represents the coordinates of point P in the established world system.

3.3.2 Transformation between Image Plane and Pixel Coordinate System

In this section, we illustrate the mathematical relationship between imaging plane and pixel coordinate system. As Figure 3-5. shows, establish a pixel coordinate system $u-v$ with point O_0 as the origin. Pixel's abscissa u and ordinate v are the number of columns and the number of rows in the image array. Suppose that the origin O_1 of imaging plane coordinate system has a coordinate of (u_0, v_0) in pixel coordinate system. And the x, y axis are parallel to u and v axis, respectively. Assume that d_x and d_y are the physical sizes of each pixel in x axis and y axis, respectively [5]. So that we could get the relationship between imaging plane coordinate system and pixel coordinate system as follows.

$$u = \frac{x}{d_x} + u_0 \tag{3-32}$$

$$v = \frac{y}{d_y} + v_0 \tag{3-33}$$

Rewrite equation $(3-32)$ and $(3-33)$ into matrix form:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3-34}$$

In equation (3-34), assume that the unit of imaging plane coordinate system O_1xy is millimeter, so that the unit of d_x and d_y is mm/pixel. And both the unit of x/d_x and y/d_y are pixel. All in all, according to equation (3-31) and (3-34), we could get the matrix expression of the relationship between the world coordinate system and pixel coordinate system as follows:

$$S \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{d_x} & 0 & u_0 & 0 \\ 0 & \frac{f}{d_y} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \boldsymbol{M}_2 \boldsymbol{M}_1 \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3-35}$$

In equation (3-35), $\boldsymbol{M}_2$ is the intrinsic parameter matrix of 3×4.

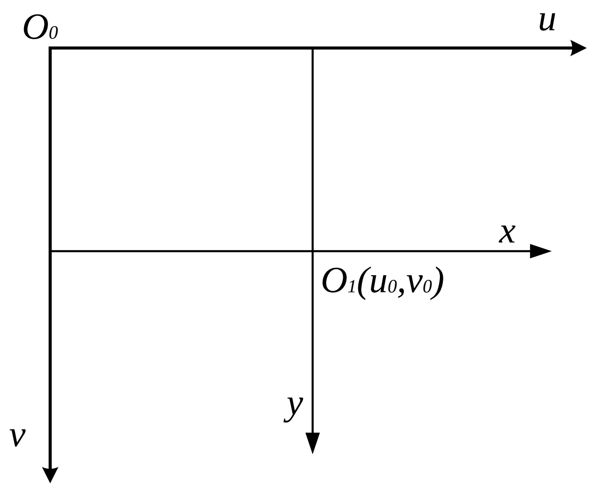


Figure 3-5. Imaging plane and pixel coordinate system

3.3 Camera Distortion Model

In the previous research, the ideal model of the camera was used for analysis and derivation. However, in the actual environment, the processing and assembly of the camera lens inevitably introduce various errors, which would cause distortion of the imaging information of the object. Seriously, the final result may be completely wrong. Therefore, the main purpose of camera calibration is to derive the error parameters between the real camera and the ideal model.

Under ideal conditions, the relationship between the world coordinate system and the imaging coordinate system is shown in equation (3-35), but in the real environment, the position of the points in the image is deviated due to the influence of the lens distortion of the camera. Therefore, an imaging model with distortion parameters is required.

Suppose that in the pixel plane, the point $p(x_p\ y_p)$ is the pixel points under the ideal model, and after the lens distortion is introduced, its real coordinate becomes $d(x_d\ y_d)$, then:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d \end{bmatrix} \qquad (3-36)$$

The correspondence between the point $p(x_p, y_p)$ and the point $d(x_d, y_d)$ can be obtained by the equation (3-36). Among them, the radial distortion parameters are $k_1\ k_2\ k_3$ and the tangential distortion coefficients p_1 and p_2.

The distortion of the camera mainly comes from radial and tangential distortion. They are all nonlinear distortions. The radial distortion is caused by the deformation of the camera lens, while the tangential distortion is due to manufacturing process limitations during the installation process.

The photosensitive center has no radial distortion, but the closer to the edge position, the greater the radial distortion. For ordinary CCD cameras, radial distortion is usually depicted by two coefficients, the first item is usually k_1 and the second item is k_2. For cameras with large radial distortion, such as fisheye cameras and panoramic cameras, a third radial distortion parameter k_3 can be introduced. Normally, the radial position of the CCD camera is adjusted according to equation (3-37):

$$\begin{cases} x_{correct} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{correct} = y(1 + k_1 r^2 + k_2 r^2 + k_3 r^6) \end{cases} \qquad (3-37)$$

Here (x, y) is the original position of the distortion point on the imager, $(x_{correct}, y_{correct})$ is the corrected new position.

As mentioned above, tangential distortion is caused by the ideal state in which the lens is in parallel with the imager due to manufacturing process limitations during the installation process. It can be described by two parameters, p_1 and p_2, as shown in equation (3-38). Figure 3-6. shows some examples of distortions occurred in the image.

$$\begin{cases} x_{correct} = x + [2p_1 y + p_2(r^2 + 2x^2)] \\ y_{correct} = y + [p_1(r^2 + 2y^2) + 2p_2 x] \end{cases} \qquad (3-38)$$

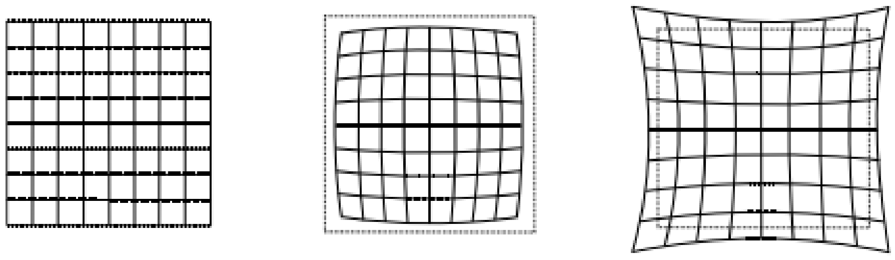


Figure 3-6. Examples of distortions in image [33]

3.4 Camera Calibration

3.4.1 Camera Calibration

For the planar calibration, the most classic is the Zhang's calibration method [29]. The main idea is that, firstly, in the ideal model without distortion, assuming that there is a point in the outside world where $Q(X, Y, Z)$ is mapped to $q(x, y)$ in the camera, the relationships between the two can be expressed by the rotation matrix ***R*** and the translation matrix ***t***, then the combination matrix of the two is

$$\mathbf{T} = [r_1, r_2, r_3, \mathbf{t}] \qquad (3-39)$$

Then the relationship between ***Q*** and ***q*** can be expressed as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s\boldsymbol{M}[r_1 \quad r_2 \quad r_3 \quad r_4]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (3-40)$$

Where s is the scale factor of the matrix and $\boldsymbol{M}$ is the intrinsic parameter matrix of the camera:

$$\boldsymbol{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (3-41)$$

For the representation space in general, it is only necessary to consider mapping relationships on a certain plane. Therefore, the characterization plane of $Z = 0$ can be chosen such that equation (3-40) is reduced to the following formula:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s\boldsymbol{M}[r_1, r_2, r_3, t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = s\boldsymbol{M}[r_1, r_2, t]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (3-42)$$

Let $\boldsymbol{H} = s\boldsymbol{M}[r_1,\ r_2,\ r_3,\ t]$, then equation (3-39) can be simplified as:

$$\boldsymbol{q} = s\boldsymbol{HQ} \qquad (3-43)$$

H is a 3×3 homography matrix. Therefore, $\boldsymbol{H}$ reflects the relationship between the image plane and the external representation plane. Let $\boldsymbol{H} = [h_1, h_2, h_3]$, thus:

$$\boldsymbol{H} = [h_1, h_2, h_3] = s\boldsymbol{M}[r_1, r_2, t] \qquad (3-44)$$

Due to $\boldsymbol{R}$ is an orthogonal matrix, $r_1^T r_2 = 0$ is established, so the following two constraints can be obtained:

$$\begin{cases} h_1^T \boldsymbol{M}^{-T}\boldsymbol{M}^{-1} h_2 = 0 \\ h_1^T \boldsymbol{M}^{-T}\boldsymbol{M}^{-1} h_2 = h_2^T \boldsymbol{M}^{-T}\boldsymbol{M}^{-1} h_2 \end{cases} \qquad (3-45)$$

For the above two constraints, the intrinsic and extrinsic parameters of the camera cannot be obtained. There must be more constraints. Let the matrix $\boldsymbol{B} = \boldsymbol{M}^{-T}\boldsymbol{M}^{-1}$, expand to get the following formula:

$$\boldsymbol{B} = \boldsymbol{M}^{-T}\boldsymbol{M}^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & \frac{-c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix} \qquad (3-46)$$

Let $h_i^T B h_j$ be used to represent the first two constraints, and let $\boldsymbol{B} = [B_{11} \quad B_{12} \quad B_{23} \quad B_{13} \quad B_{23} \quad B_{33}]$, thus have the following equation:

$$v_{ij}^T b = h_i^T \boldsymbol{B} h_j \tag{3-47}$$

Use v_{ij}^T to represent the two intrinsic constraints as:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{12})^T \end{bmatrix} b = 0 \tag{3-48}$$

According to v_{ij}^T, $\boldsymbol{V}$ is a matrix of $2N \times 6$, and only when $N \geq 2$, the equation $\boldsymbol{Vb} = 0$ has a solution. Then many intrinsic parameters of the camera can be obtained according to the solution of the equation $\boldsymbol{Vb} = 0$:

$$\begin{cases} f_x = \sqrt{\dfrac{1}{sB_{11}}} \\ f_y = \sqrt{\dfrac{B_{11}}{s(B_{11}B_{22} - B_{12}^2)}} \\ c_x = -sB_{13}f_x^2 \\ c_y = \dfrac{B_{12}B_{13} - B_{11}B_{23}}{(B_{11}B_{22} - B_{12}^2)} \\ s = \dfrac{B_{11}}{B_{33} - [B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})]} \end{cases} \tag{3-49}$$

After the intrinsic parameters are obtained, the values of extrinsic parameters of the respective rotation matrix and translation matrix can be easily obtained according to the intrinsic parameter values:

$$\begin{cases} r_1 = \dfrac{\boldsymbol{M}^{-1}h_1}{s} \\ r_2 = \dfrac{\boldsymbol{M}^{-1}h_2}{s} \\ r_3 = r_1 \times r_2 \\ t = \dfrac{\boldsymbol{M}^{-1}h_3}{s} \end{cases} \tag{3-50}$$

Where r_1 *and* r_2 are orthogonal, $s = \|\boldsymbol{M}^{-1}h_1\|$.

According to the above derivation and analysis of Zhang's calibration method, the four intrinsic parameters of the camera (f_x 、f_y 、c_x 、c_y) and the six extrinsic parameters(ψ 、φ 、ϑ、T_x 、T_y、T_z) of the rotation R and translation t matrix between the camera and the chessboard are calculated. For the lens distortion parameters, since the Zhang's calibration method only considers the radial distortion and does not take into account the tangential distortion, the classical Brownian method [30] is used here, assuming that the external three-dimensional point P is completely projected by the camera lens under the ideal camera model. On the imaging picture point $p(x_p, y_p)$, the same point is projected onto the imaging picture as the point $d(x_d, y_d)$ in the case of distortion, and the maximum is re-appeared by the equation (3-49) according to the equation (3-35). However, it is estimated that the optimal parameters are obtained in a large number of equations, which is the distortion parameter.

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|p_{ij}-\hat{p}(M,k_1,k_2,k_3,R_i,t_i,P_j)\right\|^2 \qquad (3-51)$$

According to $\boldsymbol{Vb}=0$, as long as there is a 3×3 calibration board with different viewing perspectives, the intrinsic and extrinsic parameters of the camera can be obtained. However, considering the noise and robustness, the general calibration field of view is more than 10 times, and the effective corner point of the chessboard is of 4×6.

3.4.2 Calibration Experiments of the Measurement System

In this experiment, the chessboard with the inner corner point of 4×6 is selected for calibration. It is printed on A4 paper. The chessboard is fixed on the plane, and the industrial camera is placed in front of the calibration plate. Move and rotate the camera to obtain chessboard images from different viewing angles and ensure that the corners within the image of each chessboard are included, as shown in Figure 3-7. The workflow of this process is shown in Figure 3-8., while Figure 3-9. shows the extracted corner points on the chessboard.

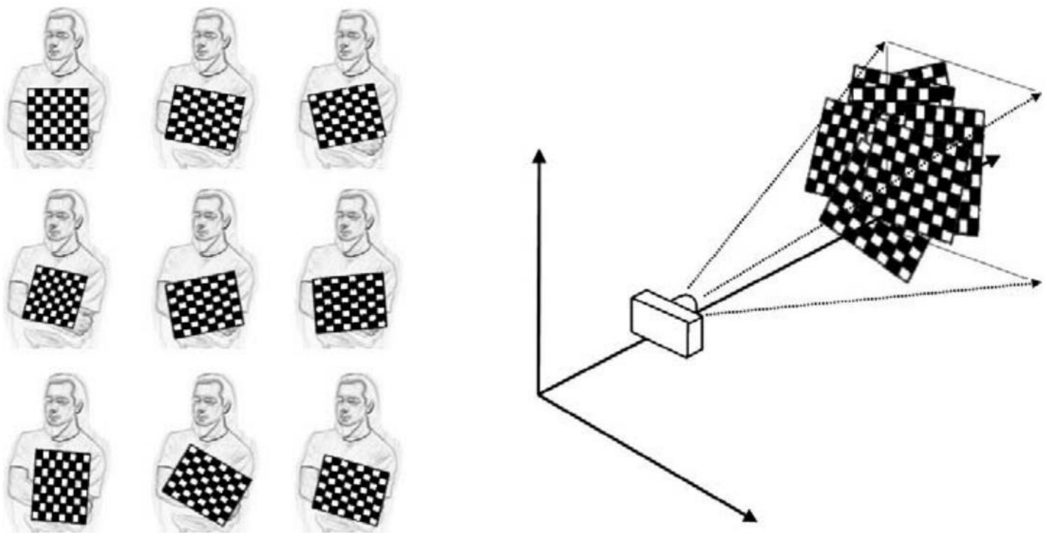
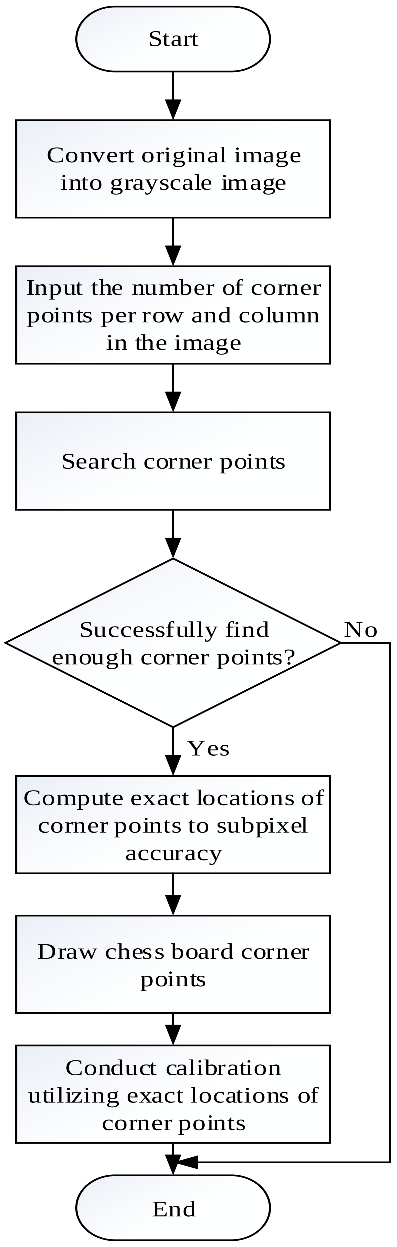


Figure 3-7. Images of a chessboard being held at various orientations (left) provide the intrinsic and extrinsic parameters of the camera calculated by utilizing OpenCV [33]

Table 3-1. Intrinsic and distortion parameters obtained from calibration

Parameters	*Camera*
f_x	*811.17*
f_y	*811.26*
u_0	*352.18*
v_0	*197.85*
K_1	*-0.25*
K_2	*3.84*
P_1	*0.00*
P_2	*0.00*

By running the calibration program which utilizes OpenCV library, the extracted corner points are calibrated. The calibration results are summarized in Table 3-1. According to the calibration results shown in Table 3-1., the calibration method based on OpenCV can obtain good results. If the calibration results are required to be more accurate, it is recommended to use the MATLAB toolbox for calibration. But the disadvantage is that it is not convenient for real-time embedded systems to calibrate the camera. Through the OpenCV-based calibration method, the calibration result can be transmitted to the subsequent correction and the presentation of the disparity map in real time, which better realizes the convenience of image processing.



Start
Convert original image into grayscale image
Input the number of corner points per row and column in the image
Search corner points
Successfully find enough corner points?
No
Yes
Compute exact locations of corner points to subpixel accuracy
Draw chess board corner points
Conduct calibration utilizing exact locations of corner points
End

Figure 3-8. Workflow of the camera calibration

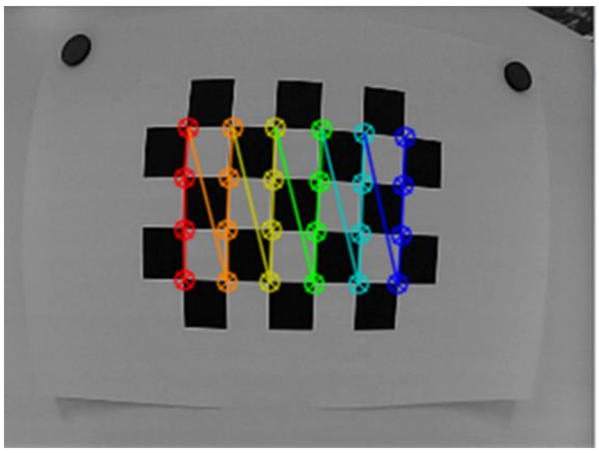


Figure 3-9. Experiment results of the corner point extraction of chessboard

3.5 Chapter Summary

In this chapter, we firstly illustrated the mathematic methods used to express the motion. Secondly, we demonstrated Euclidean transformation between different coordinates systems. Besides, we discussed the problem that the introduced camera distortion might influence the algorithm performance of the SLAM-based method for inclination measurement. Thus, we conducted the camera calibration and obtained the intrinsic parameters as well as distortion parameters of the industrial camera.

Chapter 4 Inclination Measurement Based on Motion Estimation

This chapter addresses the inclination measurement for the industrial assembly platforms by applying the motion estimation-based method*.

4.1 Introduction

4.1.1 Research Background and Motivations

In the field investigation of the assembly enterprise, it was found that the the enterprise adopted the traditional control system to realize automation of the production process while ignores problem of slight inclination of the platform, by which the lifetime of products and the yield rate will be influenced. Because the traditional controllers represented by PLC are good at the process control with excellent robustness, whereas are bad at the large-scale analysis and computing. Thus, the computer vision system will be a proper choice to serve as a monitoring point and to guarantee the producing process will not be influenced by the occurrence of inclination on the assembly platforms. In this case, to overcome the above problems, and to further improve the intelligence and automation of the assembly enterprise, in this chapter, we will demonstrate the motion estimation-based method for tackling the inclination measurement of industrial assembly platforms.

4.1.2 Related Works

As mentioned in [21], for the motion estimation-based method, there are two main categories. Namely, the feature matching-based methods like [31, 71] and the feature tracking-based methods [56, 57, 58, 59]. One of the most prevalent feature tracking-based methods is represented by the Lucas-Kanade Optical Flow. Although the feature tracking-based methods might save a portion of computing resources in the feature tracking stage, the three-dimensional coordinates of the landmarks need be known in the stage of optimization of motion estimation. In this case, there are two solutions to obtain the depth information or three-dimensional coordinates of the landmarks in general. One of the most convenient methods is to employ the RGB-D camera to get

*This chapter is a refined and reproduced version of the paper originally published in International Journal of Advanced Mechatronic Systems [91] copyrighted by Inderscience Publishers.

the depth information for the landmarks [36, 77]. Although this kind of solution works well, the cost of the vision system is relatively high, mainly because the cost of RGB-D cameras is much higher than that of general monocular cameras. The second method is to apply the stereo vision technology to estimate the depth information for the landmarks in the adjacent pictures. Although the relatively expensive RGB-D will be unnecessary by applying this method, it will introduce additional computation burden to the vision system.

Regarding the feature matching-based method [38, 39, 40], when dealing with the same number of landmarks, it will introduce a higher computing burden to find the matched point pairs than that of the optical flow-based method. However, since only a few matched point pairs will be sufficient in the real application, if we limit the number of matched point pairs, the computing burden will be decreased. Thus, through applying the non-maximal suppression strategy to screen the point pairs with the highest similarity in the two adjacent frames, the actual computing complexity will be reduced.

4.1.3 Main Works

According to the above analysis, to solve the problem of inclination measurement with a relatively low hardware cost as well as to keep a good measurement performance, the main works in this chapter are as follows: Feature matching-based method combined with the stereo vision technologies have been applied to implement the inclination measurement for the industrial assembly platforms. In specific, for the first step, an image feature algorithm is utilized to extract the feature points in adjacent frames. Secondly, the feature matching algorithm combined with the non-maximal suppression-based screening strategy are employed to obtain the best matching point pairs in the adjacent frames. Afterward, according to the coordinate information of the matching point pairs, epipolar constraints are applied to estimate the motion of the industrial camera. Thirdly, the principle of triangulation is implemented to estimate the depth information of the spatial points according to the obtained camera motion. Thus, the three-dimensional coordinate information of the spatial points is obtained. Then, based on the 3D coordinates of the landmarks and the 2D coordinates of the matched point pairs, 3D-2D (PnP) method is applied to estimate the motion of the camera. Finally, the Trust Region-based Levenberg-Marquardt optimization method is implemented to find the optimal motion.

Section 4.2 illustrates the prevalent image feature algorithms as well as the feature matching methods. Besides, the advantages and disadvantages of different methods have been discussed qualitatively. On this basis, analysis and comparisons have been made to determine the most suitable combination of methods for this study. The quantitative comparisons among different algorithm combinations are presented in section 4.7. Section 4.3 demonstrates the motion estimation based on the epipolar constraints. Section 4.4 presents the triangulation method which will utilize the obtained motion information to calculate the depth information of the spatial points corresponding to the landmarks in pictures. After obtaining the three-dimensional coordinates of the landmarks in space, section 4.5 demonstrates the working schematics of the perspective-n-points (PnP) method and the trust region-based optimization, which are applied in this study to realize the motion estimation of the inclination angles of the industrial assembly platforms. Section 4.6 illustrates the feature tracking-based methods. Finally, section 4.7 presents the workflow of the inclination measurement applying the motion estimation-based methods. Besides, implementations and validation experiments have been made to verify the effectiveness and performance of the motion estimation-based method for inclination measurement.

4.2 Image Feature and Matching Approaches

The main problem to be solved in this chapter is to estimate the inclination motion of the assembly platforms by employing SLAM algorithms. In specific, we employ a method that firstly select representative points in the pictures. Because the representative points will keep invariant in the adjacent frames of images, thus it would serve as landmarks that help us to obtain the motion of the industrial camera. In sum, we will discuss the motion estimation based on this kind of points.

In vision systems, these representative points are so-called landmarks or image features points [28, 29]. Image features have been widely used in various computer vision tasks. In other words, the feature can be considered as an alternative expression of significant information in the pictures. Apparently, robust image feature will greatly guarantee the success of the computer vision tasks. Therefore, we need to select the proper feature algorithm that will benefit the inclination measurement system developed in this study.

In general, there are several types of feature points including corners, edges, and blocks. However, according to the conclusions of previous studies, it is more common to have the same corner in two images. There are fewer cases that the same edge appears in different pictures because the pictures often have high similarity along the edge. As for the cases that the identical block appears in two images, the probability is the least. In other words, the corner features have the highest probability to be identified in adjacent frames of picture.

In practice, use corner features only still dissatisfy the needs of many computer vision tasks. To this end, many researchers have made efforts in developing feature algorithms which provides more robust and stable features for the computer vision tasks. The most commonly used algorithms including SIFT, ORB, and SURF [31, 32, 93, 94]. These mentioned feature algorithms have the superior advantages compared to the corner features. In specific, ORB feature refers to the two tasks of "extracting FAST key-point" and "calculating the BRIEF descriptor" [32][33]. Key-points denotes the position information of the image features, while the descriptors are basically vectors that store the information surrounding the image features. Besides, another classic algorithm Scale-Invariant Feature Transform (SIFT) fully considers the variations in scale and illumination for the image features. However, its high robustness is followed by a relatively higher requirement of computing power.

Other features, consider appropriate reduction of accuracy and robustness, and speed up the calculation. For example, Oriented FAST and Rotated BRIEF (ORB) is one of the most representative algorithms that try to find a trade-off between the performance and cost [30, 31]. It amended the original FAST feature points and employed Binary Robust Independent Elementary Features（BRIEF) descriptors [32, 33]. So that the speed of the entire image feature algorithm is guaranteed.

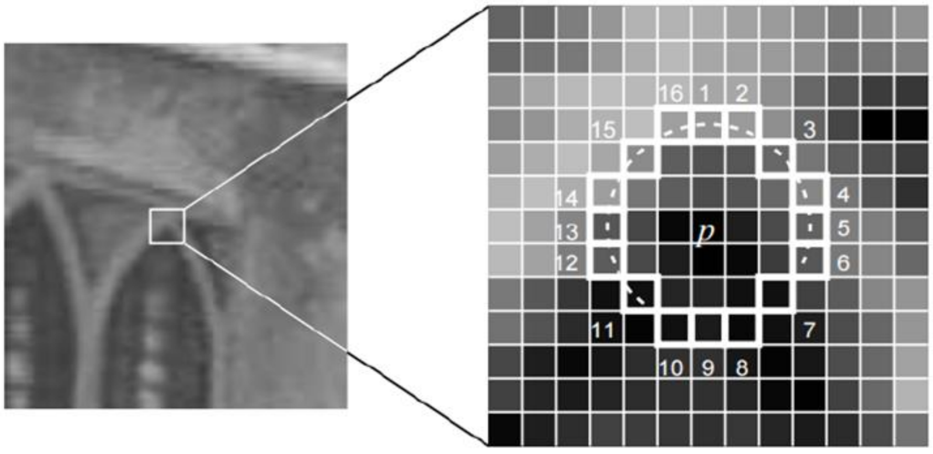


Figure 4-1. Example of extraction of FAST key-point [23]

FAST concentrate on promptly finding the obvious change of the pixel gray-scale value in a local area. Whenever the algorithm finds a pixel which differs significantly from its surrounding pixels, the pixel will be considered as a corner point. In specific, only the brightness information will be compared [34, 35]. Basic schematic of FAST is shown in Figure 4-1.

Although the time efficiency of FAST is relatively high, it indeed sacrifices the performance in return. For example, the number of FAST feature points would be large in the image, while usually researchers only need a limited number of them. To this end, an idea was proposed to tackle this problem. Namely, whenever we have many FAST feature points, a screening process will be conducted to leave the ones with the highest value of Harris response. Besides, the feature points found by FAST are lack of directionality information. When the perspective of camera drastically changes, vision system will lose the landmarks and cause the failure of the computer vision tasks. To overcome this problem, ORB introduced additional information to describe the directionality as well as the scale for the most robust feature points [31]. In specific, the scale information is provided by applying the image pyramid. And the directionality information is provided by using the intensity centroid.

The concept of centroid of mass is extended to describe the mean value of a series of pixels distributed in a local area, namely, image block [31]. Assuming we have an image block, define the moment of the block as:

$$m_{pq} = \sum_{x,y \in B} x^p y^q \, I(x,y), \qquad p,q = \{0,1\} \tag{4-1}$$

Where $I(x,y)$ indicates the grayscale of (x,y). Therefore, we have the centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{4-2}$$

Then, define the directionality of a feature as the following formula shows:

$$\theta = \arctan\left(m_{01}/m_{10}\right) \tag{4-3}$$

Afterward, we still need to calculate the descriptors for the features. In this study, we prefer to use the binary descriptors, because the computation speed of the binary descriptors is fast and thus will contribute to the real-time performance of the vision system. That is the reason why ORB has been widely employed in many computer vision systems that requires a relatively higher real-time performance. Figure 4-2. gives out the experimental results of applying ORB.

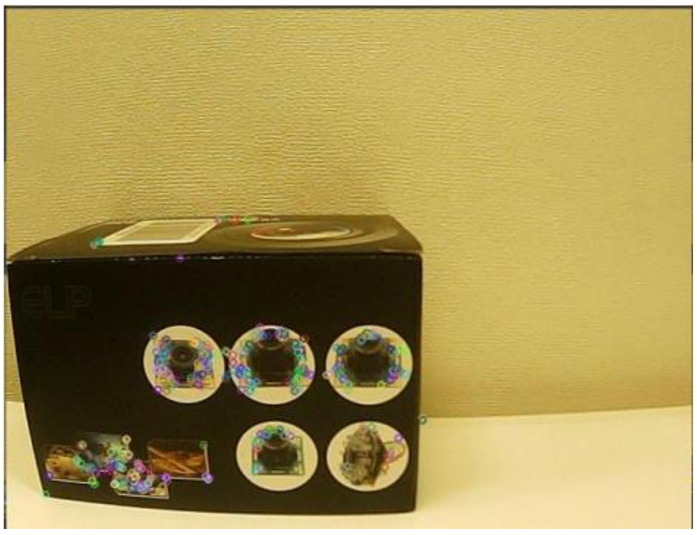


Figure 4-2. Experiment results of ORB feature extraction

Feature matching is an important step in estimation the motion of the inclined assembly platforms as well as the industrial camera. However, the problem of mismatching has not been effectively solved until now. Part of the reason is that there are often many repeated textures especially in adjacent frames of pictures, making the feature description highly similar. In this case, we first assume that all matches are correct. Then, on this basis, we will consider how to reduce the problem of mismatching.

Consider an image of two moments. If the feature point $x_{mt}, m = 1, 2, \ldots, M$ is extracted in the image I_t, the feature points $x_{nt+1}, n = 1, 2, \ldots, N$ are extracted in the image I_{t+1},

We use Brute Force Matcher, which is of high performance to look for the correspondence between these two sets of elements. In details, for one feature point x_{mt}, measure the similarity with descriptors of the other feature point x_{nt+1}. Thus, the one that has the nearest distance will be taken as the matching point.

In fact, there are many types of metrics for the similarity between features. For instance, Euclidean distance is commonly used to measure the similarity of the descriptors with decimal elements, while Hamming distance is useful for the binary descriptors. As for Hamming distance, the greater number of the different elements between two binary descriptors, the greater distance is. The following Figure 4-3. shows the experiment results of image feature extraction, descriptor calculation, and matching using the combination of ORB feature and Brute-force matching approach [31, 35], while Figure 4-4. shows the screened good matching results by applying the non-maximal suppression strategy. The images used in the experiment were taken by our camera, and it can be seen the slight changes of camera perspective exist.

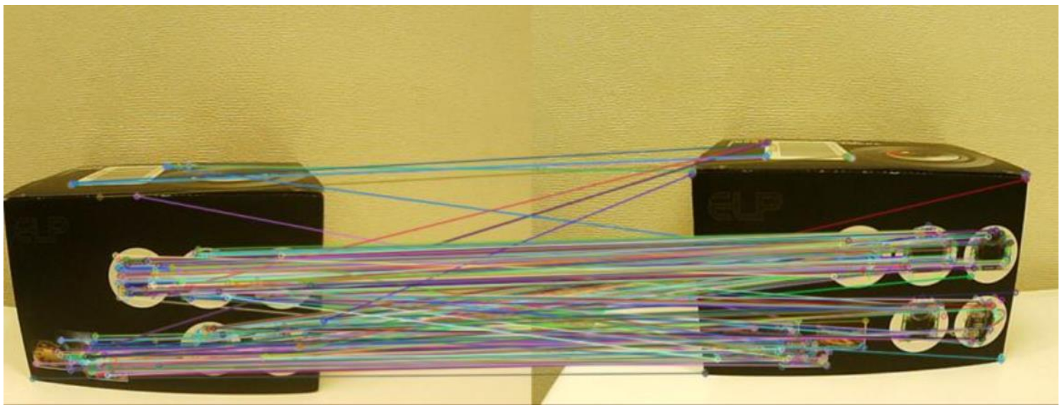


Figure 4-3. Experiment results of feature matching without screening of good match

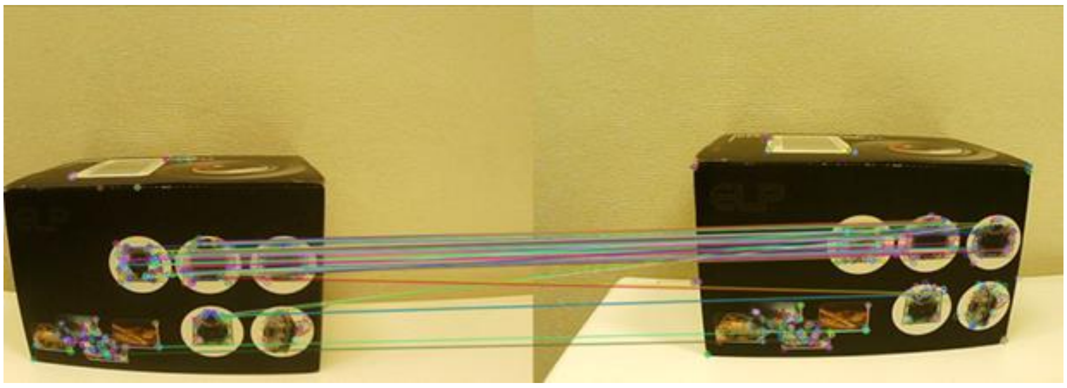


Figure 4-4. Experiment results of feature matching with screening of good match

4.3 2D-2D: Epipolar Geometry

4.3.1 Epipolar Constraint

Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model [86].

Suppose we get a pair of matching feature points p_1, p_2 from the two images I_1 *and* I_2, as shown in Figure 4-5. If there are a number of such matching points, the camera's motion between the two frames is possible to be recovered by the correspondence of these two-dimensional image points [39, 40]. The geometric relationship of the corresponding matching points in the two images is demonstrated in Figure 4-5.

Taking Figure 4-5. as an example, we want to find the motion between the two frames of images I_1 *and* I_2, let the motion from the first frame to the second frame be $\boldsymbol{R}$, $\boldsymbol{t}$. The two camera centers are O_1 and O_2 respectively. There is a feature point p_1 in image I_1, which correspond to feature point p_2 in image I_2. Both of them are obtained by feature matching. In the case of correct matching, the two are indeed the projections of the same spatial point on the two imaging planes. Habitually, people use the related terms of epipolar constraint to describe the geometric relationship of the matching pairs p_1 and p_2. First, the connection O_1p_1 and the connection O_2p_2 intersect at point P in three-dimensional space. At this time, the three points O_1, O_2, and P can determine a plane. That is the so called epipolar plane. The intersection of the O_1O_2 connection with the image planes I_1, I_2 is e_1, e_2, respectively. The two points e_1 *and* e_2 are called epipoles, O_1O_2 is called baseline. The intersection lines between the epipolar plane and two image planes I_1, I_2 are called epipolar line.

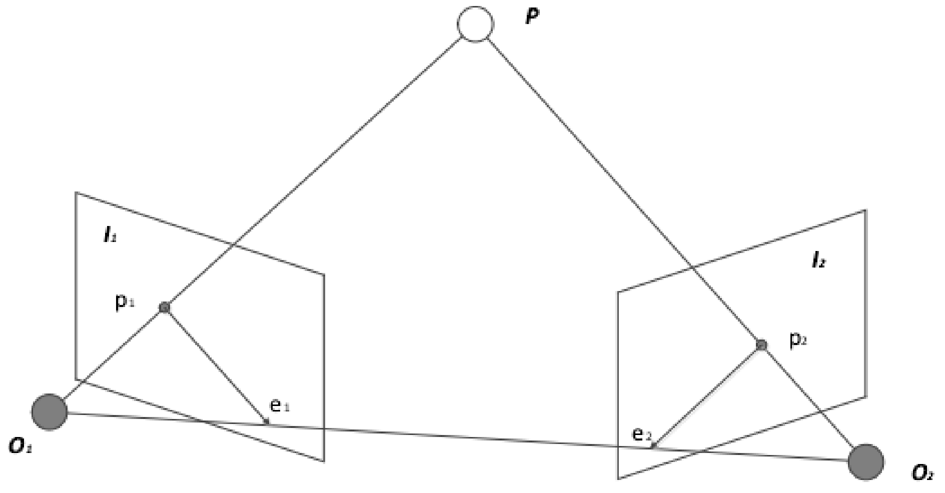


Figure 4-5. Schematic of epipolar geometry

From the first frame image, the ray O_1p_1 is the spatial position where a certain pixel may appear, because all points on the ray are projected to the same pixel. Assuming that the P position is not known, from the second frame image, the connection e_2p_2 is the position of the projection where P may appear, that is, the projection of the ray O_1p_1 in the second camera. Since the projection position of the pixel point p_2 is determined by feature point matching, the spatial position of P and the motion of camera can be inferred.

From an algebraic perspective, the above geometric relationship can be described. In the coordinate system of the first image, let the spatial position of P be:

$$\boldsymbol{P} = [X, Y, Z]^T \qquad (4-4)$$

According to the pinhole camera model, we know that the pixel positions of the two pixel points p_1, p_2 are:

$$s_1\boldsymbol{p}_1 = \boldsymbol{KP}, \qquad s_2\boldsymbol{p}_2 = \boldsymbol{K}(\boldsymbol{RP} + \boldsymbol{t}) \qquad (4-5)$$

Where matrix $\boldsymbol{K}$ consists of intrinsic parameters. Transform matrix constructed by $\boldsymbol{R}$ and $\boldsymbol{t}$ represents motion between the adjacent images. If we use homogeneous coordinates, we can also rewrite the above equation as:

$$\boldsymbol{p}_1 = \boldsymbol{KP}, \qquad \boldsymbol{p_2} = \boldsymbol{K}(\boldsymbol{RP} + \boldsymbol{t}) \qquad (4-6)$$

Let:

$$\boldsymbol{x}_1 = \boldsymbol{K}^{-1}\boldsymbol{p}_1, \boldsymbol{x}_2 = K^{-1}\boldsymbol{p}_2 \qquad (4-7)$$

Here $\boldsymbol{x_1}$, $\boldsymbol{x_2}$ denotes the two pixel points, substitute the formula (4-7) into (4-6):

$$\boldsymbol{x}_2 = \boldsymbol{Rx}_1 + \boldsymbol{t} \qquad (4-8)$$

Then both sides are left multiplied by $\boldsymbol{t}^\wedge$ at the same time, thus we have:

$$\boldsymbol{t}^\wedge\boldsymbol{x}_2 = \boldsymbol{t}^\wedge\boldsymbol{Rx}_1 \qquad (4-9)$$

After that, both sides are simultaneously multiplied by x_2^T:

$$\boldsymbol{x}_2^T\boldsymbol{t}^\wedge\boldsymbol{x}_2 = \boldsymbol{x}_2^T\boldsymbol{t}^\wedge\boldsymbol{R}\boldsymbol{x}_1 \qquad (4-10)$$

Looking at the left side of the equation,$\boldsymbol{t}^\wedge$ $\boldsymbol{x}_2$ is a vector that is perpendicular to both $\boldsymbol{t}$ and $\boldsymbol{x}_2$. When making an inner product with $\boldsymbol{x}_2$, it get 0. So we got a succinct expression:

$$\boldsymbol{x}_2^T\boldsymbol{t}^\wedge\boldsymbol{R}\boldsymbol{x}_1 = 0 \qquad (4-11)$$

Substitute with equation (4-7), therefore:

$$\boldsymbol{p}_2^T\boldsymbol{K}^{-T}\boldsymbol{t}^\wedge\boldsymbol{R}\boldsymbol{K}^{-\mathbf{1}}\boldsymbol{p}_1 = 0 \qquad (4-12)$$

These two expressions are called epipolar constraints, and their form is very simple. Their geometric meaning is that O_1, P, O_2 are coplanar. Both the translation and the rotation are included in the epipolar constraint. We denote the middle part of the left side of the epipolar constraint as two matrices: the fundamental matrix *F* and the essential matrix *E* to further simplify the epipolar constraint:

$$\boldsymbol{E} = \boldsymbol{t}^\wedge\boldsymbol{R}, \boldsymbol{F} = \boldsymbol{K}^{-T}\boldsymbol{t}^\wedge\boldsymbol{R}\boldsymbol{K}^{-\mathbf{1}} \qquad (4-13)$$

The epipolar constraint gives a succinct representation of the spatial motion for the matched point pairs. Thus, motion estimation is simplified as to firstly estimate matrix $\boldsymbol{E}$ and $\boldsymbol{F}$ using the information provided by the matched point pairs. Afterward, with the intrinsic parameters obtained through camera calibration, the transformation matrix can be calculated.

4.3.2 Essential Matrix

The definition of the essential matrix is $\boldsymbol{E} = \boldsymbol{t}^\wedge\boldsymbol{R}$, a 3×3 matrix with nine elements to be calculated [39, 40, 42, 43, 44].

Assuming we have two matched points with homogeneous coordinates: $\boldsymbol{x}_1 = [u_1, v_1, 1]^{\mathrm{T}}$, $\mathbf{x}_2 = [\mathrm{u}_2, \mathrm{v}_2, 1]^{\mathrm{T}}$. Thus, the coordinates will satisfy the epipolar constraint, and we have:

$$\begin{pmatrix} u_1 & v_1 & 1 \end{pmatrix}\begin{pmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{pmatrix}\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0 \qquad (4-14)$$

Expand the essential matrix as the following form:

$$\boldsymbol{e} = [\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}, \boldsymbol{e_4}, \boldsymbol{e_5}, \boldsymbol{e_6}, \boldsymbol{e_7}, \boldsymbol{e_8}, \boldsymbol{e_9}]^{\boldsymbol{T}} \qquad (4-15)$$

Therefore, the formula (4-14) will have the following form:

$$[u_1u_2, u_1v_2, u_1, v_1u_2, v_1v_2, v_1, u_2, v_2, 1] \cdot \boldsymbol{e} = 0 \qquad (4-16)$$

For the same reason, the same representation is given for other point pairs. If we have eight pair of matched points (u_i, v_i for the i^{th} feature point, and so on), there are:

$$\begin{bmatrix} u_1^1u_2^1 & u_1^1v_2^1 & u_1^1 & v_1^1u_2^1 & v_1^1v_2^1 & v_1^1 & u_2^1 & v_2^1 & 1 \\ u_1^2u_2^2 & u_1^2v_2^2 & u_1^2 & v_1^2u_2^2 & v_1^2v_2^2 & v_1^2 & u_2^2 & v_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_1^8u_2^8 & u_1^8v_2^8 & u_1^8 & v_1^8u_2^8 & v_1^8v_2^8 & v_1^8 & u_2^8 & v_2^8 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{bmatrix} = 0 \qquad (4-17)$$

These eight equations construct a system of linear equations. Whenever the rank of the coefficient matrix equals to 8, we can use the formula (4-17) to obtain all the nine elements of the essential matrix.

Afterward, by applying the singular value decomposition (SVD) with the essential matrix, we can estimate the motion. Hypothesize the essential matrix $\boldsymbol{E}$ will be decomposed as:

$$\boldsymbol{E} = \boldsymbol{U\Sigma V}^T \qquad (4-18)$$

Where $\boldsymbol{U}, \boldsymbol{V}$ denotes the matrix of orthogonality, while $\boldsymbol{\Sigma}$ denotes a matrix of singular values. Due to the properties of $\boldsymbol{E}$, the matrix $\boldsymbol{\Sigma}$ must satisfy the condition of $\boldsymbol{\Sigma} = diag(\sigma, \sigma, 0)$. When conducting the decomposition, for one essential matrix $\boldsymbol{E}$, there will have two possible $\boldsymbol{R}, \boldsymbol{t}$ corresponding to the essential matrix $\boldsymbol{E}$:

$$\begin{cases} \boldsymbol{t}_1^\wedge = \boldsymbol{U}\boldsymbol{R}_Z\left(\frac{\pi}{2}\right)\boldsymbol{\Sigma}\boldsymbol{U}^T, \boldsymbol{R}_1 = \boldsymbol{U}\boldsymbol{R}_Z^T\left(\frac{\pi}{2}\right)\boldsymbol{V}^T \\ \boldsymbol{t}_2^\wedge = \boldsymbol{U}\boldsymbol{R}_Z\left(-\frac{\pi}{2}\right)\boldsymbol{\Sigma}\boldsymbol{U}^T, \boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{R}_Z^T\left(-\frac{\pi}{2}\right)\boldsymbol{V}^T \end{cases} \qquad (4-19)$$

Where $R_Z(\pi/2)$ denotes the matrix $\boldsymbol{R}$ with a 90-degree rotation along the Z axis. Due to the reason that neither positive nor negative sign of matrix E and t will get the same decomposition results, thus we will usually get four possible solutions after the decomposition.

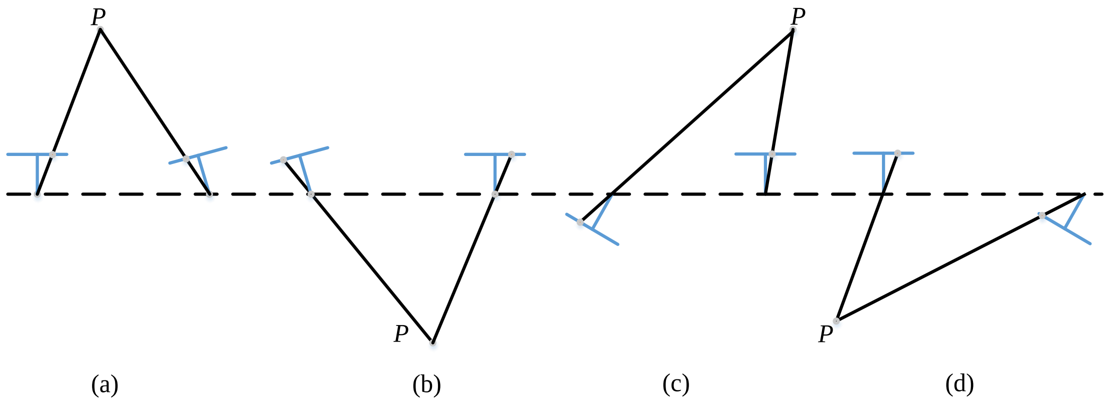


Figure 4-6. Four kinds of possible solution of matrix $\boldsymbol{R}$ and $\boldsymbol{t}$

Figure 4-6. illustrates the possible solutions mentioned above. The blue line denotes the projection position of the 3D point on the imaging plane of the camera. In Figure 4-6 (a), the depth of the two projection positions on the imaging plane are both positive, which indicates the correct solution. In this way, by examining the sign of depth, we can screen the correct solution after the decomposition.

We know the projection of the spatial point on the camera, which is noted by blue line, and want to solve the camera's motion. In the case of keeping the two projection points unchanged, four possible cases can be drawn, but fortunately, in the first solution, P has a positive depth in both cameras. Therefore, as long as substitute any coordinates of points into the four solutions and detect the depth of the point under the two cameras, the correct solution can be determined.

However, in terms of essential matrix $\boldsymbol{E}$, which is solved according to the linear equation, may not satisfy its intrinsic property, that is, the singular value may not satisfy the form like "$\sigma, \sigma, 0$". To this end, when conducting SVD, matrix $\boldsymbol{\Sigma}$ will be deliberately adjusted into the above form. Usually, we will get the singular value matrix $\boldsymbol{\Sigma} = diag(\sigma 1, \sigma 2, \sigma 3)$ after the decomposition, assume that $\sigma 1 \geq \sigma 2 \geq \sigma 3$, set:

$$\boldsymbol{E} = \boldsymbol{U} diag\left(\frac{\sigma_1 + \sigma_2}{2}, \frac{\sigma_1 + \sigma_2}{2}, 0\right) \boldsymbol{V}^T \qquad (4-20)$$

For simplicity, in practice we can directly set the singular value matrix as $diag(1, 1, 0)$, which is also feasible according to the scale equivalence property of matrix $\boldsymbol{E}$.

4.3.3 Homography Matrix

Homography matrix $\boldsymbol{H}$ is commonly used to describe the transformation relationship between two flat planes [41, 42, 43]. Assuming we have a pair of matching points p_1 and p_2 distributed in two planes. Assume that the equation of the plane is:

$$\boldsymbol{n}^T \boldsymbol{P} + d = 0 \qquad (4-21)$$

Reorganize it, then get:

$$-\frac{\boldsymbol{n}^T \boldsymbol{P}}{d} = 1 \qquad (4-22)$$

Then, according to formula (4-5), we have:

$$\boldsymbol{p}_2 = \boldsymbol{K}(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{t}) = \boldsymbol{K}\left(\boldsymbol{R}\boldsymbol{P} + \boldsymbol{t} \cdot \left(-\frac{\boldsymbol{n}^T \boldsymbol{P}}{d}\right)\right) = \boldsymbol{K}\left(\boldsymbol{R} - \frac{\boldsymbol{t}\boldsymbol{n}^T}{d}\right)\boldsymbol{K}^{-1}\boldsymbol{p}_1 \qquad (4-23)$$

Apparently, the formula (4-23) express the transformation between p_1 and p_2, namely:

$$\boldsymbol{p}_2 = \boldsymbol{H}\boldsymbol{p}_1 \qquad (4-24)$$

Similarly, we can use the information of the matched point pairs to obtain $\boldsymbol{H}$ by solving the linear equations. Afterward, singular value decomposition will be applied to get the information of motion, as the following:

$$\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \qquad (4-25)$$

Expand the above formula (4-25), there are:

$$\begin{cases} u_2 = \dfrac{h_1 u_1 + h_2 v_1 + h_3}{h_7 u_1 + h_8 v_1 + h_9} \\ v_2 = \dfrac{h_4 u_1 + h_5 v_1 + h_6}{h_7 u_1 + h_8 v_1 + h_9} \end{cases} \qquad (4-26)$$

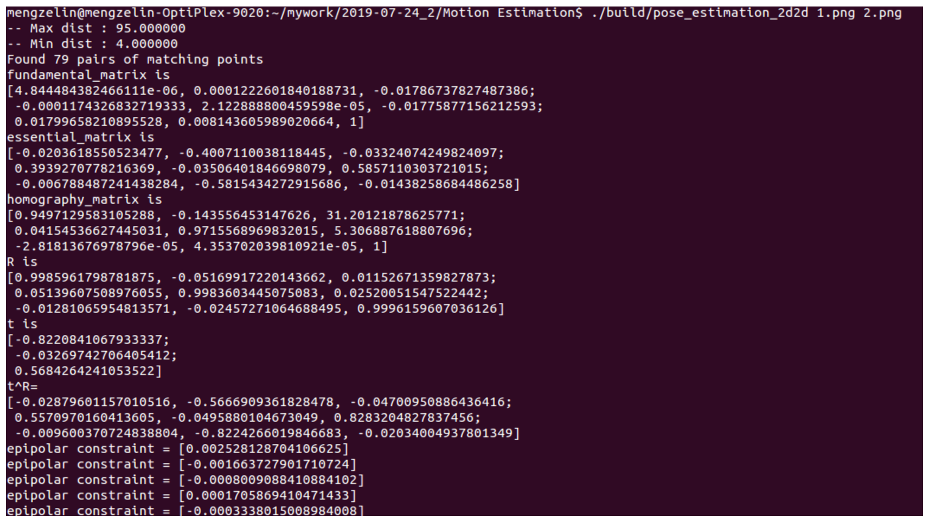
Therefore,

$$\begin{cases} h_1 u_1 + h_2 v_1 + h_3 - h_7 u_1 u_2 - h_8 v_1 u_2 = u_2 \\ h_4 u_1 + h_5 v_1 + h_6 - h_7 u_1 v_2 - h_8 v_1 v_2 = v_2 \end{cases} \qquad (4-27)$$

One pair of matching points will help to build two linear equations, thus the matrix ***H*** can be solved whenever the the coefficient matrix is of full rank.

$$\begin{bmatrix} u_1^1 & v_1^1 & 1 & 0 & 0 & 0 & -u_1^1 u_2^1 & -v_1^1 u_2^1 \\ 0 & 0 & 0 & u_1^1 & v_1^1 & 1 & -u_1^1 v_2^1 & -v_1^1 v_2^1 \\ u_1^2 & v_1^2 & 1 & 0 & 0 & 0 & -u_1^2 u_2^2 & -v_1^2 u_2^2 \\ 0 & 0 & 0 & u_1^2 & v_1^2 & 1 & -u_1^2 v_2^2 & -v_1^2 v_2^2 \\ u_1^3 & v_1^3 & 1 & 0 & 0 & 0 & -u_1^3 u_2^3 & -v_1^3 u_2^3 \\ 0 & 0 & 0 & u_1^3 & v_1^3 & 1 & -u_1^3 v_2^3 & -v_1^3 v_2^3 \\ u_1^4 & v_1^4 & 1 & 0 & 0 & 0 & -u_1^4 u_2^4 & -v_1^4 u_2^4 \\ 0 & 0 & 0 & u_1^4 & v_1^4 & 1 & -u_1^4 v_2^4 & -v_1^4 v_2^4 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} = \begin{bmatrix} u_2^1 \\ v_2^1 \\ u_2^2 \\ v_2^2 \\ u_2^3 \\ v_2^3 \\ u_2^4 \\ v_2^4 \end{bmatrix} \qquad (4-28)$$

The above-mentioned method is the so called Direct Linear Transform. After applying the similar decomposition process for the essential matrix ***E***, we will utilize additional information to screen the correct decomposition results. Experiment results of estimation of motion utilizing epipolar constraint with the matched point pairs are shown in Figure 4-7.



```
mengzelin@mengzelin-OptiPlex-9020:~/mywork/2019-07-24_2/Motion Estimation$ ./build/pose_estimation_2d2d 1.png 2.png
-- Max dist : 95.000000
-- Min dist : 4.000000
Found 79 pairs of matching points
fundamental_matrix is
[4.844484382466111e-06, 0.0001222601840188731, -0.01786737827487386;
 -0.0001174326832719333, 2.122888800459598e-05, -0.01775877156212593;
 0.01799658210895528, 0.008143605989020664, 1]
essential_matrix is
[-0.0203618550523477, -0.4007110038118445, -0.03324074249824097;
 0.3939270778216369, -0.03506401846698079, 0.5857110303721015;
 -0.006788487241438284, -0.5815434272915686, -0.01438258684486258]
homography_matrix is
[0.9497129583105288, -0.143556453147626, 31.20121878625771;
 0.04154536627445031, 0.9715568969832015, 5.306887618807696;
 -2.81813676978796e-05, 4.353702039810921e-05, 1]
R is
[0.9985961798781875, -0.05169917220143662, 0.01152671359827873;
 0.05139607508976055, 0.9983603445075083, 0.02520051547522442;
 -0.01281065954813571, -0.02457271064688495, 0.9996159607036126]
t is
[-0.8220841067933337;
 -0.03269742706405412;
 0.5684264241053522]
t^R=
[-0.02879601157010516, -0.5666909361828478, -0.04700950886436416;
 0.5570970160413605, -0.0495880104673049, 0.8283204827837456;
 -0.009600370724838804, -0.8224266019846683, -0.02034004937801349]
epipolar constraint = [0.002528128704106625]
epipolar constraint = [-0.001663727901710724]
epipolar constraint = [-0.0008009088410884102]
epipolar constraint = [0.0001705869410471433]
epipolar constraint = [-0.0003338015008984008]
```

Figure 4-7. Experiment results of motion estimation employing epipolar constraint

4.4 Triangulation

Whenever we obtained the motion, we can use it to estimate the position information of the image features. Triangulation is a commonly utilized method for estimating the depth information of landmarks [76].

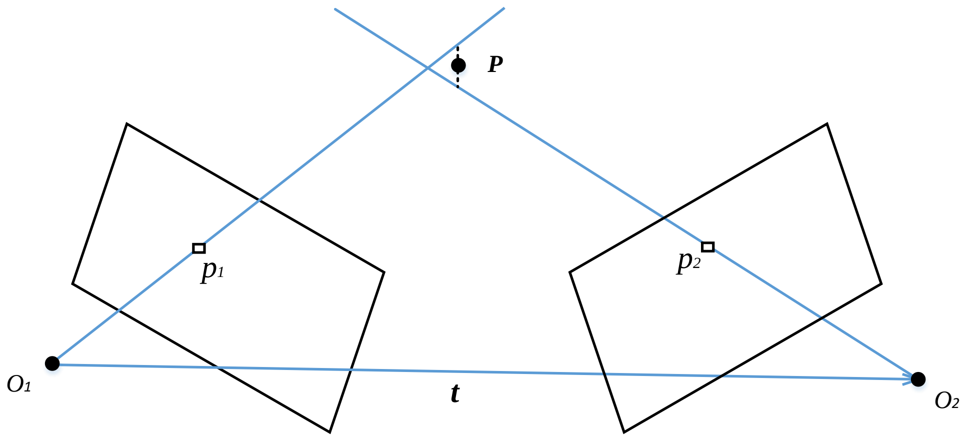


Figure 4-8. Schematic of triangulation

Similarly, consider two images I_1 and I_2, as shown in Figure 4-8., we set the image on the left as the reference frame, while the translation between I_1 and I_2 is described by $\boldsymbol{t}$. O_1 and O_2 denotes the optical centers of the camera. The matched landmarks p_1 and p_2 are distributed on I_1 and I_2, respectively. Ideally, line O_1p_1 and O_2p_2 will have an intersection at the spatial point P, as shown in the figure above. With the epipolar constraint, while assuming the normalized coordinates of the matched landmarks are x_1 and x_2, thus we have:

$$s_1\boldsymbol{x}_1 = s_2\boldsymbol{R}\boldsymbol{x}_2 + \boldsymbol{t} \qquad (4-29)$$

Now that we know $\boldsymbol{R}$, $\boldsymbol{t}$, and hope to solve the depth information s_1 and s_2 of the landmarks. In terms of the method to solve these two unknowns, first left multiply $x_1^{\wedge}$ on both sides of the above formula to get:

$$s_1\boldsymbol{x}_1^{\wedge}\boldsymbol{x}_1 = 0 = s_2\boldsymbol{x}_1^{\wedge}\boldsymbol{R}\boldsymbol{x}_2 + \boldsymbol{x}_1^{\wedge}\boldsymbol{t} \qquad (4-30)$$

It is easy to solve the equation above, then we can get the depth information. However, because the estimated rotation matrix and translation matrix may not perfectly accurate, the equation above might not perfectly equal to 0 as well. To this end, the optimization methods need to be applied to find the optimal solution.

4.4.1 Discussion about Triangulation

One thing to be mentioned is that the premise of using triangulation is that there is a translation between the perspective views of the two images. Therefore, pure rotation is impossible to use triangulation because the epipolar constraint will always be satisfied. In this thesis, the images used in the experiment satisfy the premise of using triangulation approach. However, even if in the presence of translation, we need to consider the uncertainty of triangulation.

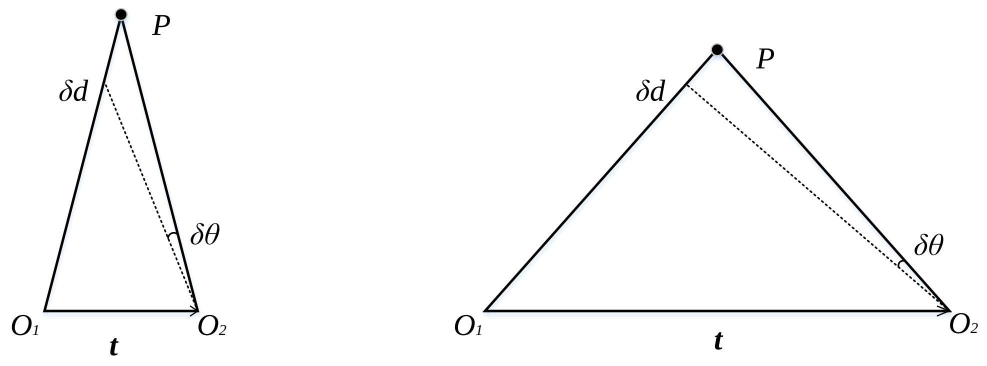
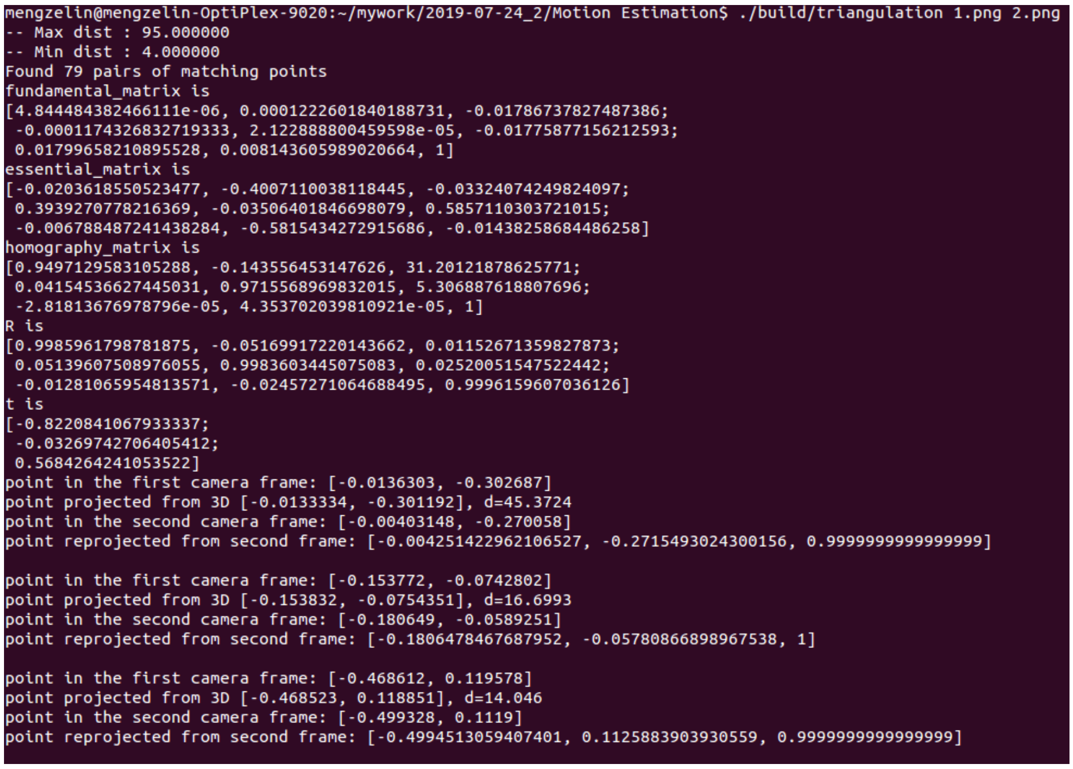


Figure 4-9. Analysis about uncertainty of triangulation

As shown in Figure 4-9., when the translation is slight, the uncertainty of the pixel will result in a large uncertainty of depth calculation. In other words, if the position of the landmark changes by δx, such that the perspective varies by $\delta\theta$, causing the calculated depth value varies by δd. As shown in the figure above, in the case of $\boldsymbol{t}$ becomes large, then δd would become very smaller, which means that the larger the translation quantity is, the more precise the triangulation is. For obtaining a better triangulation result, the method of increasing the translation quantity is proved to be thankless, since that will introduce more influence factors, such as illumination variation and mismatch problems. One of the effective ways to improve the accuracy of triangulation is to properly enhance the resolution of the pictures. Because it will reduce the noises of the landmarks, while keeps a relatively low cost of the computation power.

4.4.2 Experiment of Triangulation

The three-dimensional coordinates of the matched points are obtained by applying triangulation method. We utilize OpenCV to implement the triangulation method. Experiment results of triangulation are shown in Figure 4-10., while the obtained depth information are shown Figure 4-11. It shows that the magnitude of the error is approximately the third decimal place.

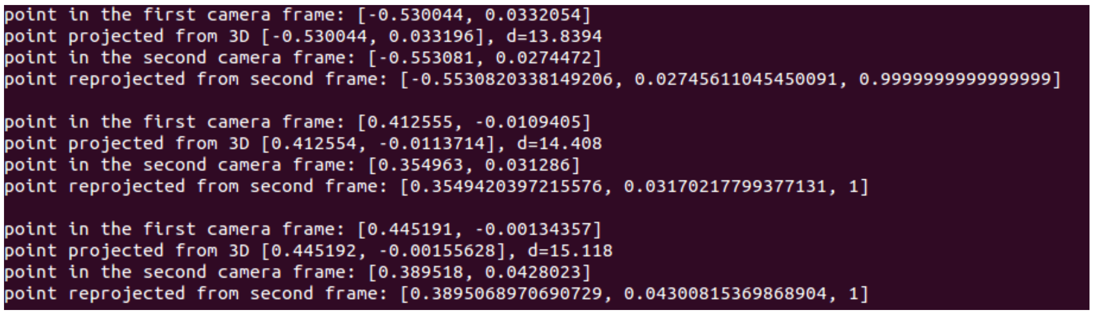


```
mengzelin@mengzelin-OptiPlex-9020:~/mywork/2019-07-24_2/Motion Estimation$ ./build/triangulation 1.png 2.png
-- Max dist : 95.000000
-- Min dist : 4.000000
Found 79 pairs of matching points
fundamental_matrix is
[4.844484382466111e-06, 0.0001222601840188731, -0.01786737827487386;
 -0.0001174326832719333, 2.122888800459598e-05, -0.01775877156212593;
 0.01799658210895528, 0.008143605989020664, 1]
essential_matrix is
[-0.0203618550523477, -0.4007110038118445, -0.03324074249824097;
 0.3939270778216369, -0.03506401846698079, 0.5857110303721015;
 -0.006788487241438284, -0.5815434272915686, -0.01438258684486258]
homography_matrix is
[0.9497129583105288, -0.143556453147626, 31.20121878625771;
 0.04154536627445031, 0.9715568969832015, 5.306887618807696;
 -2.81813676978796e-05, 4.353702039810921e-05, 1]
R is
[0.9985961798781875, -0.05169917220143662, 0.01152671359827873;
 0.05139607508976055, 0.9983603445075083, 0.02520051547522442;
 -0.01281065954813571, -0.02457271064688495, 0.9996159607036126]
t is
[-0.8220841067933337;
 -0.03269742706405412;
 0.5684264241053522]
point in the first camera frame: [-0.0136303, -0.302687]
point projected from 3D [-0.0133334, -0.301192], d=45.3724
point in the second camera frame: [-0.00403148, -0.270058]
point reprojected from second frame: [-0.004251422962106527, -0.2715493024300156, 0.9999999999999999]

point in the first camera frame: [-0.153772, -0.0742802]
point projected from 3D [-0.153832, -0.0754351], d=16.6993
point in the second camera frame: [-0.180649, -0.0589251]
point reprojected from second frame: [-0.1806478467687952, -0.05780866898967538, 1]

point in the first camera frame: [-0.468612, 0.119578]
point projected from 3D [-0.468523, 0.118851], d=14.046
point in the second camera frame: [-0.499328, 0.1119]
point reprojected from second frame: [-0.4994513059407401, 0.1125883903930559, 0.9999999999999999]
```

Figure 4-10. Experiment results of triangulation



```
point in the first camera frame: [-0.530044, 0.0332054]
point projected from 3D [-0.530044, 0.033196], d=13.8394
point in the second camera frame: [-0.553081, 0.0274472]
point reprojected from second frame: [-0.5530820338149206, 0.02745611045450091, 0.9999999999999999]

point in the first camera frame: [0.412555, -0.0109405]
point projected from 3D [0.412554, -0.0113714], d=14.408
point in the second camera frame: [0.354963, 0.031286]
point reprojected from second frame: [0.3549420397215576, 0.03170217799377131, 1]

point in the first camera frame: [0.445191, -0.00134357]
point projected from 3D [0.445192, -0.00155628], d=15.118
point in the second camera frame: [0.389518, 0.0428023]
point reprojected from second frame: [0.3895068970690729, 0.04300815369868904, 1]
```

Figure 4-11. Depth information calculated by triangulation

4.5 3D-2D: PnP

PnP (Perspective-n-Point) is a method for solving motion of 3D-2D point pairs [44, 45, 47, 48]. It is a method of estimating camera motion when n 3D spatial points as well as its 2D coordinates are known. If the 3D position of the feature points on one image is known, then we can estimate the motion only use four pairs of matched points. In specific, three pairs will be used to estimate the motion, another pair is for verification.

4.5.1 Direct Linear Transformation

Assume we have a 3D point P with coordinates $P = (X, Y, Z, 1)^T$. In image I_1, projections are made to feature points $x_1 = (u_1, v_1, 1)^T$, which is represented by homogeneous coordinates of normalized plane [47]. At this time, the camera's motion $\boldsymbol{R}, \boldsymbol{t}$ is unknown. Define that matrix $\boldsymbol{T} = [\boldsymbol{R}|\boldsymbol{t}]$ is of 3×4 containing the motion information. Its expanded form is written as follows:

$$s\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 \\ t_5 & t_6 & t_7 & t_8 \\ t_9 & t_{10} & t_{11} & t_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4-31}$$

Use the last line to eliminate s and get two constraints:

$$u_1 = \frac{t_1 X + t_2 Y + t_3 Z + t_4}{t_9 X + t_{10} Y + t_{11} Z + t_{12}} \quad v_1 = \frac{t_5 X + t_6 Y + t_7 Z + t_8}{t_9 X + t_{10} Y + t_{11} Z + t_{12}} \tag{4-32}$$

To simplify the representation, define the row vector of $\boldsymbol{T}$:

$$\boldsymbol{t}_1 = (t_1, t_2, t_3, t_4)^T, \boldsymbol{t}_2 = (t_5, t_6, t_7, t_8)^T, \boldsymbol{t}_3 = (t_9, t_{10}, t_{11}, t_{12})^T \tag{4-33}$$

Then there are:

$$\begin{cases} \boldsymbol{t}_1^T \boldsymbol{P} - \boldsymbol{t}_3^T \boldsymbol{P} u_1 = 0 \\ \boldsymbol{t}_2^T \boldsymbol{P} - \boldsymbol{t}_3^T \boldsymbol{P} v_1 = 0 \end{cases} \tag{4-34}$$

$\boldsymbol{t}$ is the variable to be sought, and it can be seen that each feature point can provide two linear constraints on $\boldsymbol{t}$. Assuming a total of *N* feature points, a linear system of equations can be listed:

$$\begin{pmatrix} P_1^T & 0 & -u_1 P_1^T \\ 0 & P_1^T & -v_1 P_1^T \\ \vdots & \vdots & \vdots \\ P_N^T & 0 & -u_N P_N^T \\ 0 & P_N^T & -u_N P_N^T \end{pmatrix} \begin{pmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \\ \boldsymbol{t}_3 \end{pmatrix} = 0 \tag{4-35}$$

Since $\boldsymbol{t}$ has a total of 12 dimensions, a solution of the matrix $\boldsymbol{T}$ can be obtained by utilizing at least six pairs of matching points. This method is also called DLT (Direct Linear Transform). In the DLT solver, we directly consider the $\boldsymbol{T}$ matrix as 12 unknowns, ignoring the connections between them. Because the rotation matrix $\boldsymbol{R} \in SO(3)$, the solution obtained by DLT does not necessarily satisfy the constraint, namely rotation matrix $\boldsymbol{R}$ is an orthogonal matrix. The translation vector is easier to handle, it belongs to the vector space, and does not have its own constraints. For the rotation matrix $\boldsymbol{R}$, it is necessary to look for a best approximation for the 3×3 upper-left matrix block of $\boldsymbol{T}$ estimated by DLT. This can be done by QR decomposition which is equivalent to re-projecting the result from the matrix space onto the *SE(3)* manifold and converting it into two parts, rotation and translation [47]. It should be explained that $\boldsymbol{x}_1$ here uses the normalized plane coordinates and removes the influence of the intrinsic matrix $\boldsymbol{K}$. Because it has been known by camera calibration.

4.5.2 Bundle Adjustment

Except for applying the linear transform to tackle the PnP, construct the PnP problem into a nonlinear least squares problem would be another option [73]. The linear method mentioned above is to first seek camera motion and then find the position of the space point. Non-linear optimization is to treat them as optimization variables and put them together for optimization. This is a general solution that can be used to optimize the results given by PnP. In PnP, the Bundle Adjustment problem is a problem of minimizing the re-projection error [73, 74, 75, 76, 77].

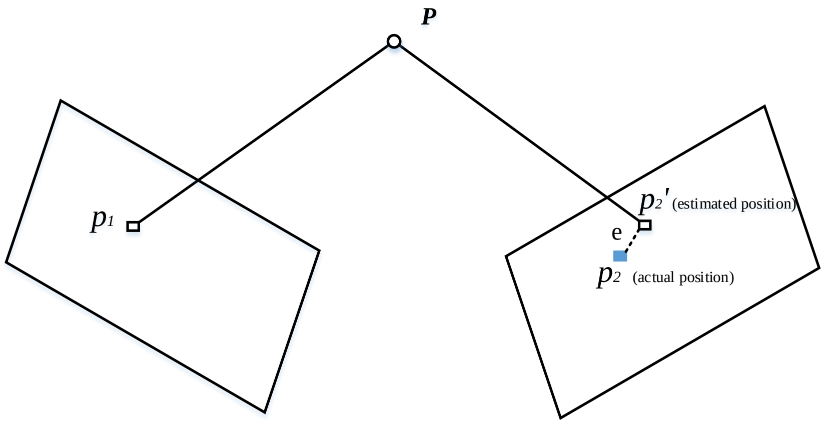


Figure 4-12. Schematic of re-projection error

Considering the n three-dimensional point P and their projection p, we want to calculate the camera's motion $\boldsymbol{R}$, $\boldsymbol{t}$, whose Lie algebra is expressed as ξ. Suppose the coordinates of a certain space point is $P_i = [X_i, Y_i, Z_i]^T$, and the pixel coordinates of the projection is $u_i = [u_i,\ v_i]^T$. The relationship between the pixel position and the spatial point position is as follows:

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \boldsymbol{K} exp(\boldsymbol{\xi}^\wedge) \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \qquad (4-36)$$

Except for camera motion represented by Lie algebra, everything is consistent with the previous definition. Written into matrix form as:

$$s_i \boldsymbol{u}_i = \boldsymbol{K} exp(\boldsymbol{\xi}^\wedge) \boldsymbol{P_i} \qquad (4-37)$$

Now, there is an error in the equation due to the unknown camera motion and the presence of noise at the observation point. Therefore, we sum the errors, build a least squares problem, and then find the best camera motion to minimize it:

$$\boldsymbol{\xi}^* = argmin \frac{1}{2} \sum_{i=1}^{n} \left\| \boldsymbol{u}_i - \frac{1}{s_i} \boldsymbol{K} \exp\,(\boldsymbol{\xi}^\wedge) \boldsymbol{P_i} \right\|_2^2 \qquad (4-38)$$

The error term of this problem is the error obtained by comparing the pixel coordinates (observed projection position) with the position where the 3D point is projected according to the currently estimated camera motion, so it is called a re-projection error. This error has 3 dimensions when using homogeneous coordinates. However, since the last dimension of u is 1, the error of this dimension is always zero, so it is simpler to use non-homogeneous coordinates, and the error is only 2 dimensions. As shown in Figure 4-12., we know that p_1 and p_2 are projections of the same spatial point P through feature matching, but we don't know the motion of camera. In the initial value, there is a certain distance between the $\widehat{p_2}$ which is the projection of P and the actual p_2. So we adjust the camera's motion to make this distance smaller. However, since this adjustment requires consideration of many points, at last the error of each point is usually not exactly zero. Using Lie algebra, it is possible to build a nonlinear optimization problem, while employing the G-N (Gauss-Newton) or L-M (Levenberg-Marquart) algorithm to get the optimal solution. Thus, before using G-N and L-M, it is important to conduct linearization:

$$\boldsymbol{e}(\boldsymbol{x}+\Delta\boldsymbol{x}) \approx \boldsymbol{e}(\boldsymbol{x}) + \boldsymbol{J}\Delta\boldsymbol{x} \tag{4-39}$$

When $\boldsymbol{e}$ is the pixel coordinate error of 2 dimensions, and $\boldsymbol{x}$ is the camera motion of 6 dimensions, $\boldsymbol{J}$ will be a 2×6 matrix. First, denote that the coordinates of the space point transformed into the camera coordinate system is P', and take out its front three-dimensional:

$$\boldsymbol{P}' = (\exp(\boldsymbol{\xi}^\wedge)\boldsymbol{P})_{1:3} = [X', Y', Z']^T \tag{4-40}$$

Then, the camera projection model is relative to P':

$$s\boldsymbol{u} = \boldsymbol{K}\boldsymbol{P}' \tag{4-41}$$

Expand it,

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \tag{4-42}$$

Using the third line to eliminate s, which is actually the distance of P', you get:

$$u = f_x \frac{X'}{Z'} + c_x, v = f_y \frac{Y'}{Z'} + c_y \tag{4-43}$$

This is consistent with previous camera models. When finding the error, we can compare the u, v here with the actual measured value and find the difference. After defining the intermediate variables, we left multiply the disturbance quantity $\delta\boldsymbol{\xi}$ for $\boldsymbol{\xi}^\wedge$ and then consider the derivative of the change in $\boldsymbol{e}$ with respect to the amount of disturbance. Using the chain rule, write the following:

$$\frac{\partial \boldsymbol{e}}{\partial \delta\boldsymbol{\xi}} = \lim_{\delta\xi \to 0} \frac{e(\delta\boldsymbol{\xi} \oplus \boldsymbol{\xi})}{\delta\boldsymbol{\xi}} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}'} \frac{\partial \boldsymbol{P}'}{\partial \delta\boldsymbol{\xi}} \tag{4-44}$$

Here " $\oplus$ " refers to the left-multiply disturbance on the Lie algebra. The first term is the derivative of the error with respect to the projection point. According to equation (4-43), it is easy to get:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}'} = -\begin{bmatrix} \frac{\partial u}{\partial X'} & \frac{\partial u}{\partial Y'} & \frac{\partial u}{\partial Z'} \\ \frac{\partial v}{\partial X'} & \frac{\partial v}{\partial Y'} & \frac{\partial v}{\partial Z'} \end{bmatrix} = -\begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{Z'^2} \end{bmatrix} \tag{4-45}$$

The second term denotes the derivative of the point after transformation,

$$\frac{\partial(\boldsymbol{TP})}{\partial \delta \boldsymbol{\xi}} = (\boldsymbol{TP})^{\odot} = \begin{bmatrix} I & -\boldsymbol{P}'^{\wedge} \\ \boldsymbol{0}^T & \boldsymbol{0}^T \end{bmatrix} \tag{4-46}$$

From equation (4-40) we have:

$$\frac{\partial P'}{\partial \delta \boldsymbol{\xi}} = [I \quad -\boldsymbol{P}'^{\wedge}] \tag{4-47}$$

Multiply these two to get a 2×6 Jacobian matrix:

$$\frac{\partial \boldsymbol{e}}{\partial \delta \boldsymbol{\xi}} = -\begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} & -\frac{f_x X' Y'}{Z'^2} & f_x + \frac{f_x X^2}{Z'^2} & -\frac{f_x Y'}{Z'} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{Z'^2} & -f_y - \frac{f_y Y'^2}{Z'^2} & \frac{f_y X' Y'}{Z'^2} & \frac{f_y X'}{Z'} \end{bmatrix} \tag{4-48}$$

The matrix $\boldsymbol{J}$ express the first-order derivative of the reprojection error regarding the camera motion's Lie algebra presentation. We retain the negative sign because the error is defined as "the observed value substract the predicted value". On the other hand, except for optimizing camera motion, we also want to optimize the spatial position of feature points. Therefore, it is necessary to derive derivative of $\boldsymbol{e}$ with respect to spatial point P. Still using the chain rule, there are:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}'} \frac{\partial \boldsymbol{P}'}{\partial \boldsymbol{P}} \tag{4-49}$$

The first item has been derived before, as for the second item, by definition

$$\boldsymbol{P}' = \exp(\boldsymbol{\xi}^{\wedge})\, \boldsymbol{P} = \boldsymbol{RP} + \boldsymbol{t} \tag{4-50}$$

We find that after deriving, the second item $\frac{\partial P'}{\partial P}$ only leaves $\boldsymbol{R}$, thus:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{P}} = -\begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{Z'^2} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{Z'^2} \end{bmatrix} \boldsymbol{R} \tag{4-51}$$

Thus, we derive the two derivative matrices of the camera observation equation for camera motion and feature points. They are significant to provide gradient directions during the optimization process and to guide the iteration of optimization.

4.5.3 Levenberg-Marquart Method

Since the second-order approximation of Taylor expansion used in the Gauss-Newton method can only have a good approximation near the expansion point [49, 50], some researchers have suggested that a "Trust Region" should be added to Δx, and it cannot be made too large to make the approximation inaccurate [85]. This sort of nonlinear optimization methods is collectively named as the Trust Region Method [18] [31]. Within the trust region, the approximation is generally considered to be valid; however, outside the region, the approximation is considered to be inaccurate. How to make sure the range depends on the difference of the approximate model and actual function. In specific, when the difference is tiny, then expand region; if the difference is large, then narrow the approximate range. So, consider using

$$\rho = \frac{f(x + \Delta x) - f(x)}{\boldsymbol{J}(x)\Delta x} \qquad (4-52)$$

to determine if the Taylor approximation is good enough. The numerator of above formula is the descent value of the actual function, and the denominator is the descent value of the approximate model. If the value of ρ is close to 1, the approximation is good. If ρ is too small, indicating that the actual descent value is much less than the approximate descent value, the approximation effect is considered to be poor, and the trust region needs to be narrowed. Conversely, if ρ is much greater, it indicates that the real descent value is greater than approximated one, therefore the trust region need to be enlarged.

Thus, the workflow of the optimization process applying the collectively called Trust Region Method is as follows:

- Set x_0 as an initial value as well as μ which denotes the initial radius of the optimization
- In the k^{th} iteration, calculating:

$$\min_{\Delta x_k} \frac{1}{2} \|f(x_k) + \boldsymbol{J}(x_k)\Delta x_k\|^2, \quad \text{s.t.} \|\boldsymbol{D}\Delta x_k\|^2 \le \mu \ (4-53)$$

Where μ is the radius of the confidence zone. ***D*** is a non-negative diagonal matrix

- Calculate ρ.
- If $\rho > 3$, then $\mu = 2\mu$.
- If $\rho < 0.25$, then $\mu = 0.5\mu$.
- If ρ is greater than a certain threshold, approximation is considered to be feasible, let $x_{k+1} = x_k + \Delta x_k$.
- Determine if the algorithm converges. If converge, end algorithm, otherwise, return to the second step.

Where the multiples and thresholds of the approximate range expansion are empirical values obtained through many experiments using the Assembly Platform Dataset. In equation (4-53), we limit the increment to a sphere with a radius of μ, which is considered valid only in this sphere. After taking ***D***, the ball can be seen as an ellipsoid.

In the optimization method proposed by Levenberg, taking $\boldsymbol{D}$ as a unit matrix $\boldsymbol{I}$ is equivalent to directly constraining Δx in a sphere.

When applying the above-mentioned L-M method for optimization, it is necessary to solve the equation (4-53) to get the gradient. This is an optimization problem with inequality constraints, we use Lagrange multipliers to turn it into an unconstrained optimization problem:

$$min_{\Delta k} = \frac{1}{2}\|f(x_k) + \boldsymbol{J}(x_k)\Delta x_k\|^2 + \frac{\lambda}{2}\|D\Delta x\|^2 \qquad (4-54)$$

Where λ is the Lagrange multiplier. Similar to the approach in Gauss-Newton, after expanding it, the core problem is to calculate the linear equation of the increment:

$$(\boldsymbol{H} + \lambda\boldsymbol{D}^T\boldsymbol{D})\Delta \mathrm{x} = \boldsymbol{g} \qquad (4-55)$$

In the incremental equation above, it introduced the $\lambda\boldsymbol{D}^T\boldsymbol{D}$ comparing with the G-N method. To simplify the equation, namely set $\boldsymbol{D} = \boldsymbol{I}$, there are:

$$(\boldsymbol{H} + \lambda\boldsymbol{I})\Delta \mathrm{x} = \boldsymbol{g} \qquad (4-56)$$

It can be seen that when the parameter λ is small, $\boldsymbol{H}$ dominates, which means that the quadratic approximation model is better in this range, and the L-M method is closer to the G-N method. On the other hand, when λ is large, $\lambda\boldsymbol{I}$ dominates, and L-M is closer to the Gradient descent method, which indicates that the nearby approximation is not good enough [78, 80]. The solution of L-M can avoid the non-singular problems of the coefficient matrix of linear equations to a certain extent, and provide a more stable and accurate incremental Δx.

In summary, the framework of nonlinear optimization problems is divided into two categories: Line Search first fixes the search direction and then looks for the step size in that direction, represented by the gradient descent method and the Gauss-Newton method [81, 82, 83, 84]. Trust Region first fixes the search area and then considers the best increment in the area [85]. Such methods are represented by Levenberg-Marquart. In practice, we choose L-M method as the gradient descent strategy. Figure 4-15. shows the experiment results of motion estimation employing PnP method with Bundle Adjustment optimization. The extracted feature points and results of feature matching are shown in Figure 4-13. and Figure 4-14, respectively.

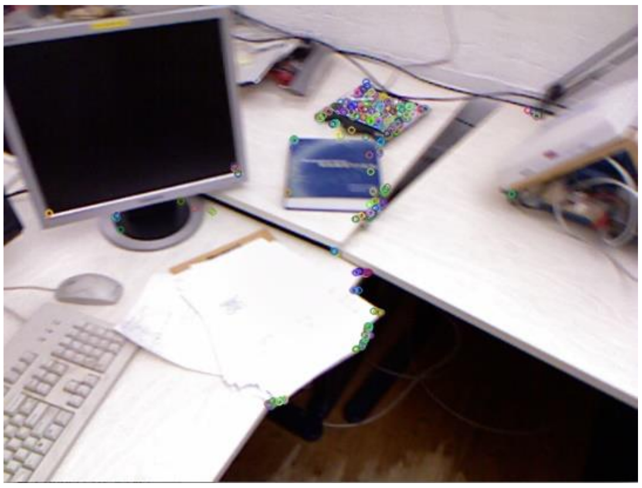


Figure 4-13. Experiment results of ORB feature extraction

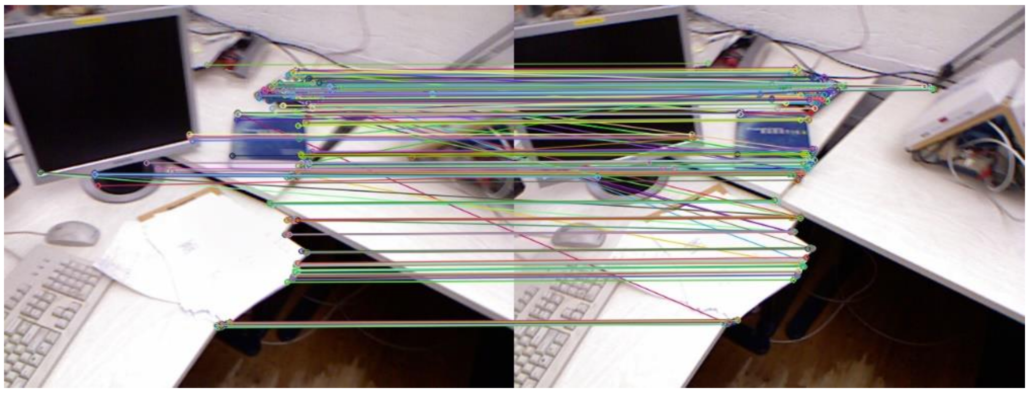
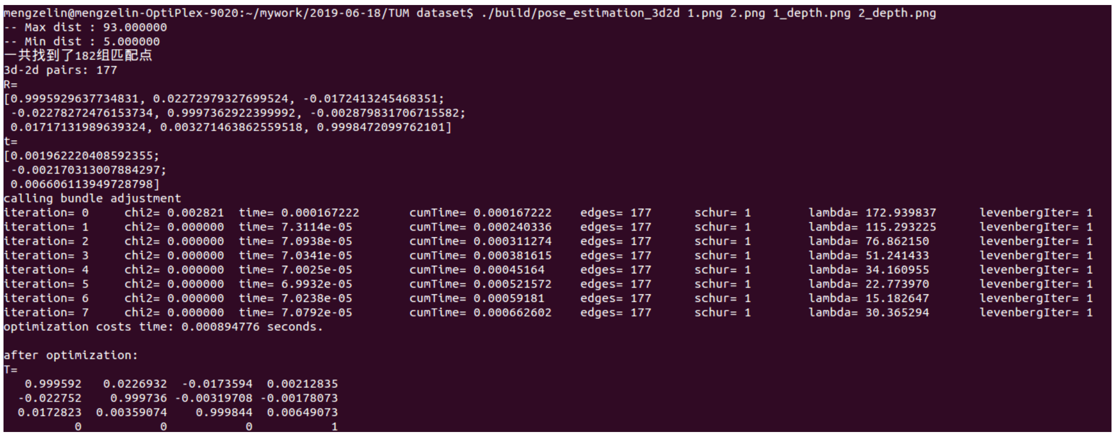


Figure 4-14. Experiment results of feature matching


```
mengzelin@mengzelin-OptiPlex-9020:~/mywork/2019-06-18/TUM dataset$ ./build/pose_estimation_3d2d 1.png 2.png 1_depth.png 2_depth.png
-- Max dist : 93.000000
-- Min dist : 5.000000
一共找到了182组匹配点
3d-2d pairs: 177
R=
[0.9995929637734831, 0.02272979327699524, -0.0172413245468351;
 -0.02278272476153734, 0.9997362922399992, -0.002879831706715582;
 0.01717131989639324, 0.003271463862559518, 0.9998472099762101]
t=
[0.001962220408592355;
 -0.002170313007884297;
 0.006606113949728798]
calling bundle adjustment
iteration= 0     chi2= 0.002821  time= 0.000167222       cumTime= 0.000167222    edges= 177      schur= 1        lambda= 172.939837      levenbergIter= 1
iteration= 1     chi2= 0.000000  time= 7.3114e-05        cumTime= 0.000240336    edges= 177      schur= 1        lambda= 115.293225      levenbergIter= 1
iteration= 2     chi2= 0.000000  time= 7.0938e-05        cumTime= 0.000311274    edges= 177      schur= 1        lambda= 76.862150       levenbergIter= 1
iteration= 3     chi2= 0.000000  time= 7.0341e-05        cumTime= 0.000381615    edges= 177      schur= 1        lambda= 51.241433       levenbergIter= 1
iteration= 4     chi2= 0.000000  time= 7.0025e-05        cumTime= 0.00045164     edges= 177      schur= 1        lambda= 34.160955       levenbergIter= 1
iteration= 5     chi2= 0.000000  time= 6.9932e-05        cumTime= 0.000521572    edges= 177      schur= 1        lambda= 22.773970       levenbergIter= 1
iteration= 6     chi2= 0.000000  time= 7.0238e-05        cumTime= 0.00059181     edges= 177      schur= 1        lambda= 15.182647       levenbergIter= 1
iteration= 7     chi2= 0.000000  time= 7.0792e-05        cumTime= 0.000662602    edges= 177      schur= 1        lambda= 30.365294       levenbergIter= 1
optimization costs time: 0.000894776 seconds.

after optimization:
T=
   0.999592   0.0226932  -0.0173594  0.00212835
  -0.022752    0.999736 -0.00319708 -0.00178073
  0.0172823  0.00359074    0.999844  0.00649073
          0           0           0           1
```
Figure 4-15. Experiment results of motion estimation using PnP with Bundle Adjustment optimization

4.6 Position Estimation Based on Direct Method

In this section, the optical flow feature point tracking method is introduced at the beginning. Moreover, a comparison between feature tracking and feature matching method has been made. On the other hand, motion estimation employing direct method based on feature tracking have been implemented [3, 86]. Experiment results prove that proposed direct method can achieve a good performance when the perspective view of camera changes slightly.

4.6.1 Introduction of the Direct Method

Except for applying the feature matching-based method to realize the motion estimation, other methods can also be helpful to implement it. For example, the optical flow-based method can be utilized to replace the process of matching the feature points [53, 54, 55, 58].

Also, some feasible ideas to overcome the disadvantages of feature matching-based methods are as follows:

(1) Retain feature points, but only calculate key-points. Then, applying the optical flow for tracking the trajectory of the extracted landmarks. In this way, it may save the time required to calculate and match the descriptors, although the optical flow method itself requires a certain amount of computation;
(2) Only key-points are calculated and no descriptors are calculated. At the same time, apply the Direct Method to track the positions of the landmarks in the adjacent frames. This also skips the computing for descriptor, and the collectively called direct method is less computationally intensive than the optical flow method.
(3) Neither calculating key-point points nor calculating descriptors, but directly calculating camera motion based on pixel grayscale differences.

The first method also depends on the extracted landmarks, and just replaces the process of matching landmarks with tracking them applying the optical flow method. On the other hand, it utilizes the epipolar constraints or 3D-2D algorithm when estimating the camera motion. In the latter two methods, they calculate the camera motion by utilizing the grayscale value of the adjacent pictures, which are called direct methods [56, 57].

When realizing the motion estimation with feature-based method, we assume that the position of the extracted landmark is fixed in the space. Then, we estimate the motion and find the optimal solution through minimizing the re-projection error based on their projected position in the adjacent pictures. Thus, the exact position of projection of the spatial point in the adjacent pictures shall be known - this is the reason to match or track the features. However, if applying the direct method, it is unnecessary to measure the similarity between different landmarks [59, 60]. Because it utilizes the luminance information of the pixels in the adjacent frames.

4.6.2 Optical Flow

The collectively called direct method has the similar assumptions as the optical flow. The optical flow method tracks the movement of the pixel in adjacent pictures, while the direct method take the model of camera motion into account.

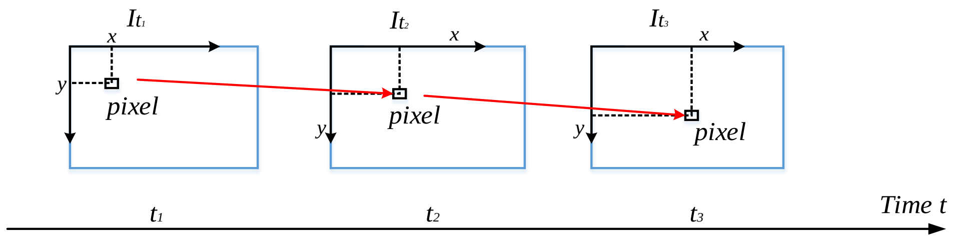


Figure 4-16. Schematic of optical flow algorithm

The optical flow tracks the movement of the pixel in adjacent pictures over time, demonstrated as Figure 4-16. above. As time passes, the same pixel moves through the image, and we want to track its motion. Tracking the motion for some of the pixels is defined as the sparse optical flow. One representative method is the Lucas-Kanade optical flow.

4.6.2.1 Lucas-Kanade Optical Flow

In the LK optical flow, it assumes that the perspectives of the pictures vary as time passing by [79, 86]. The frames are regarded as the function of time $I(t)$. Thus, at the point-in-time t, hypothesize that one pixel located at (x, y), with grayscale value $I(x, y, t)$. Besides, the basic principle of the optical flow is as follows:

The assumption of the invariance in grayscale refers to the grayscale value of a pixel projected from a three-dimensional point will keep unchanged in the image. If there is a pixel located at position (x, y) at the point-in-time t, while at the time point $t + d_t$, it moves to $(x + dx, y + dy)$. Since the grayscale value is unchanged, there are:

$$\boldsymbol{I}(x + dx, y + dy, t + dt) = \boldsymbol{I}(x, y, t) \qquad (4-57)$$

The gray-scale invariant assumption is an ideal hypothesize, thus, probably it may not hold true in many cases. In this way, the optical flow might not reliable under the certain conditions. Under the grayscale invariant assumption, conduct Taylor expansion, while retaining the first order item, gives:

$$\boldsymbol{I}(x+dx, y+dy, t+dt) \approx \boldsymbol{I}(x,y,t) + \frac{\partial \boldsymbol{I}}{\partial x}dx + \frac{\partial \boldsymbol{I}}{\partial y}dy + \frac{\partial \boldsymbol{I}}{\partial t}dt \qquad (4-58)$$

Because we assume that the grayscale is invariant, then the grayscale at the next moment is equal to the previous grayscale, thus:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \qquad (4-59)$$

Dividing both sides by d_t gives:

$$\frac{\partial \boldsymbol{I}}{\partial x}\frac{dx}{dt} + \frac{\partial \boldsymbol{I}}{\partial y}\frac{dy}{dt} = -\frac{\partial \boldsymbol{I}}{\partial t} \qquad (4-50)$$

Where dx/dt denotes the velocity of the pixels shifting horizontally, while the dy/dt represents the velocity along the vertical direction. Besides, denote them as u, v, respectively. On the other hand, $\partial I/\partial x$ denotes the gradient of the pixel in the horizontal direction, while $\partial I/\partial y$ represents the vertical gradient, expressed by I_x, I_y. Denote the variance of grayscale with respect to time t as I_t, written into matrix form, gives:

$$\begin{bmatrix}\boldsymbol{I}_x & \boldsymbol{I}_y\end{bmatrix}\begin{bmatrix}u \\ v\end{bmatrix} = -\boldsymbol{I}_t \qquad (4-51)$$

What need to calculate are u and v, which are motion of the point (x, y). With the hypothesize of the optical flow, it assumes a block of pixel points will have the same movement trajectory. If the block is of $w \times w$ with w^2 pixel points in it, then there would be w^2 constraints for solving the motion:

$$\begin{bmatrix}I_x & I_y\end{bmatrix}_k \begin{bmatrix}u \\ v\end{bmatrix} = -\boldsymbol{I}_{tk}, k = 1, \dots, w^2 \qquad (4-52)$$

To find the least squares solution of $u, v.$, therefore:

$$\begin{bmatrix}u \\ v\end{bmatrix}^* = -(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \qquad (4-53)$$

This gives the velocity u, v of the pixel motion in the adjacent pictures. To get an optimal estimation, normally it will need to make several times of iterations.

4.6.2.2 Experiment of Lucas-Kanade Optical Flow

We apply the LK optical flow provided by OpenCV to trace the trajectories of the landmark points. Technical University of Munich (TUM) dataset is utilized to implement the experiment. It contains many RGB-D images, and it also provides an accurate trajectory measured with a motion capture system that can be regarded as a ground truth. Since the TUM images are collected from the actual environment, it is necessary to explain its data format.

(1) "rgb.txt" and "depth.txt" record the collection time and corresponding file name of each file.
(2) The "rgb/" and "depth/" directories store the captured image files which are in ".png" format. The color image is a eight-bit three-channel image and the depth map is a

16-bit single-channel image. The file name is the acquisition time.

(3) “groundtruth.txt” is the camera motion acquired by the external motion capture system in the format (time, t_x, t_y, t_z, q_x, q_y, q_z, q_w).

Due to the process of acquisition of color maps, depth maps, and standard traces are independent, and the acquisition frequency of the traces is much higher than the image. Before using the data, it will be essential to align the images according to the acquisition point-in-time to pair the RGB map with the depth map. In principle, it will be available for regarding the data with an acquisition time which close to a threshold as a pair of matched images. The accurate trajectory at a similar time point that provided by the dataset will be considered as the actual acquisition position of the image. TUM provides a python script "asso-ciate.py" to align the data.

The purpose of using LK is to track feature points. To find the corner points in the first picture and trace the trajectories of them by applying LK optical flow, then plot them in the graph. Experiment result show the situation of several frames during the running of the program. Initially we had about 1,780 feature points. Some of the feature points are lost during the tracking process, and we have about 192 feature points up to 100 frames. The perspective view of camera has also changed significantly from the original image. By analyzing the tracking process of the feature points, we find that the features at the corners of the object are more stable. The features at the edges "slide" along the edges, mainly because the content of the blocks of feature remains essentially unchanged as they move along the edges, so these regions could be easily considered to be the same place. For the feature points that are neither at the corner nor at the edge, their positions are very unstable. All in all, the corner points can edges can be recognized better.

On the other hand, as for the processing speed of the above-mentioned method, in the case of tracking 1500 landmarks, it cost 15 milliseconds around. If reducing the quantity of landmarks, the calculation time is significantly reduced. It reveals that the method avoids matching descriptors, whereas it still cost some time for computing. On our experiment platform, LK optical flow can increase the time efficiency by 20% to 25%. In addition, the tracking-base method is equivalent to get the correspondence of landmarks by applying feature matching. In practice, generally the problem of mismatch will not occur. This is the advantage of optical flow method over matching descriptors. However, the method of matching descriptors can still succeed when the camera moves significantly, while the optical flow require that the camera motion to be slight. In this respect, the robustness of the optical flow method is somewhat worse than the matching of descriptors. Finally, we can estimate the camera motion using the PnP method with the landmark pairs tracked applying the optical flow approach. In summary, above-mentioned algorithm will be helpful to accelerate the vision system under the conditions that the perspective view of the camera changes relatively slight. In Figure

4-17., it shows the experiment results of tracking feature points applying LK optical flow. Figure 4-18. shows the time consumption of tracking feature points employing LK optical flow method.

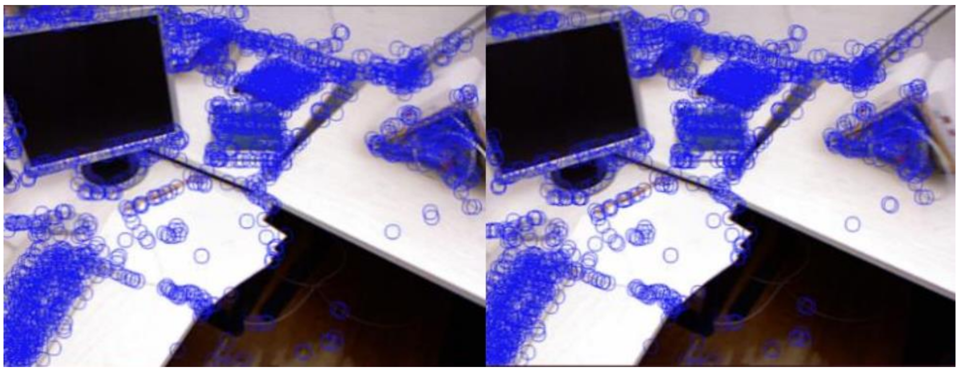
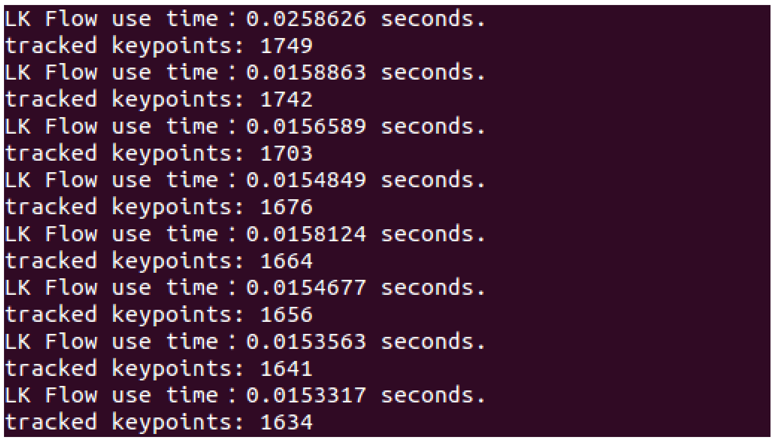


Figure 4-17. Experiment results of tracking feature points employing LK optical flow



```
LK Flow use time：0.0258626 seconds.
tracked keypoints: 1749
LK Flow use time：0.0158863 seconds.
tracked keypoints: 1742
LK Flow use time：0.0156589 seconds.
tracked keypoints: 1703
LK Flow use time：0.0154849 seconds.
tracked keypoints: 1676
LK Flow use time：0.0158124 seconds.
tracked keypoints: 1664
LK Flow use time：0.0154677 seconds.
tracked keypoints: 1656
LK Flow use time：0.0153563 seconds.
tracked keypoints: 1641
LK Flow use time：0.0153317 seconds.
tracked keypoints: 1634
```

Figure 4-18. Time consumption of LK optical flow approach

4.6.3 Direct Methods

In this section, the direct method will be introduced and implanted in the simulation experiments. The basic schematic of the direct method will be demonstrated at the beginning, afterward, the g2o optimization library will be utilized to implement the direct method.

4.6.3.1 Derivation of Direct Method

As shown in Figure 4-19., consider a point P in space and two images captured at two moments. The coordinates of P in world coordinate system is *[X, Y, Z]*, its two projection points on two images, with non-homogeneous pixel coordinates p_1 and p_2. The aim is to obtain the relative movement between the adjacent pictures. Assuming the first picture to be the reference frame, while hypothesize the movement of the current frame is expressed by $\boldsymbol{R}, \boldsymbol{t}$. Furthermore, intrinsic parameters will be denoted by matrix $\boldsymbol{K}$. The complete projection equation is as follows:

$$\begin{cases} p_1 = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_1 = \dfrac{1}{Z_1}\boldsymbol{KP} \\ p_2 = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_2 = \dfrac{1}{Z_1}\boldsymbol{K}(\boldsymbol{RP}+\boldsymbol{t}) = \dfrac{1}{Z_2}\boldsymbol{K}(\exp(\boldsymbol{\xi}^{\wedge})\mathbf{P})_{1:3} \end{cases} \quad (4-54)$$

Where Z_1 denotes the depth information of the three-dimensional point P in the reference frame, while Z_2 denotes the depth of the spatial point in the current frame.

In the case of applying the feature matching-based method, the positions of p_1 and p_2 can be obtained by matching the descriptors, we can calculate the positions of the re-projection. But in the direct method, without feature matching, there is no way to know the correspondence of p_1 and p_2. The idea of the direct method is to find the position of p_2 using the motion information. However, under the condition when the estimation results are not good enough, the position of p_2 will be significantly different from p_1. To this end, the optimization for the motion is necessary to get a more accurate positon of p_2, which could have a higher similarity to p_1. The goal of the optimization problem will be minimizing the photometric error, which is the brightness difference of point P in the two images:

$$e = \boldsymbol{I}_1(p_1) - \boldsymbol{I}_2(p_2) \quad (4-55)$$

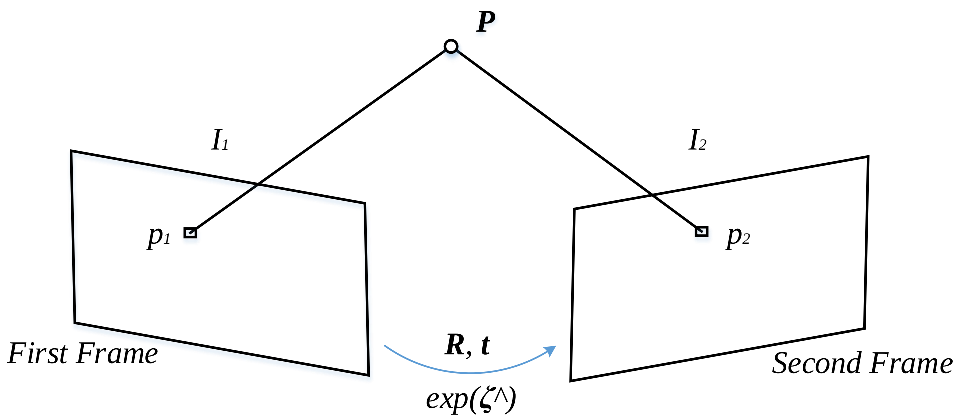


Figure 4-19. Schematic of direct method

Similarly, the optimization goal is the 2-Norm of e, gives:

$$min_{\xi}\, J(\xi) = \|e\|^2 \quad (4-56)$$

When applying direct method, it also under the assumption of the invariance of the grayscale value in the adjacent frames, as mentioned before. Therefore, it is available to assume there are N spatial points P_i that satisfy the assumption, and the optimization problem can be described as follows:

$$min_{\xi} J(\boldsymbol{\xi}) = \sum_{i=1}^{N} e_i^T e_i \,, e_i = \boldsymbol{I}_1(p_1, i) - \boldsymbol{I}_2(p_2, i) \tag{4 − 57}$$

Where $\boldsymbol{\xi}$ denotes the motion, which is the optimization variable. To make the optimization, the derivative relationships between the movement and the error need to be known. Therefore, the perturbation model on the Lie algebra is used. We left multiply $exp(\boldsymbol{\xi})$ by a small perturbation $exp(\delta\boldsymbol{\xi})$ to get:

$$e(\boldsymbol{\xi} \oplus \delta\boldsymbol{\xi}) = \boldsymbol{I}_1\left(\frac{1}{Z_1}\boldsymbol{KP}\right) - \boldsymbol{I}_2\left(\frac{1}{Z_2}\boldsymbol{K}\, exp(\boldsymbol{\xi}^{\wedge})\boldsymbol{P} + \frac{1}{Z_2}\boldsymbol{K}\delta\boldsymbol{\xi}^{\wedge}\exp\,(\boldsymbol{\xi}^{\wedge})\boldsymbol{P}\right) \tag{4 − 58}$$

Denote that,

$$\begin{cases} \boldsymbol{q} = \delta\xi^{\wedge} exp(\boldsymbol{\xi}^{\wedge})\boldsymbol{P} \\ \boldsymbol{u} = \dfrac{1}{Z_2}\boldsymbol{K}q \end{cases}$$

Where q denotes the coordinate of P in the current frame with disturbance, and u denotes the pixel coordinate of the spatial point P. Retaining the first-order Taylor expansion, there are:

$$\begin{aligned} e(\boldsymbol{\xi} \oplus \delta\boldsymbol{\xi}) &= I_1\left(\frac{1}{Z_1}\boldsymbol{KP}\right) - I_2\left(\frac{1}{Z_2}Kexp(\xi^{\wedge})\boldsymbol{P} + \boldsymbol{u}\right) \\ &= e(\boldsymbol{\xi}) - \frac{\partial \boldsymbol{I}_2}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}}\frac{\partial q}{\partial \delta\xi}\delta\boldsymbol{\xi} \end{aligned} \tag{4 − 59}$$

It can be seen that the first derivative contains three terms by applying the chain rule:

(1) $\partial I_2/\partial u$ denotes the pixel gradient of $\boldsymbol{u}$;

(2) $\partial u/\partial q$ is the derivative of the projection equation for $\boldsymbol{q}$. Denote that $\boldsymbol{q} = [X, Y, Z]^T$, in terms of the derivation shown in the previous section, the derivative is:

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial u}{\partial X} & \frac{\partial u}{\partial Y} & \frac{\partial u}{\partial Z} \\ \frac{\partial v}{\partial X} & \frac{\partial v}{\partial Y} & \frac{\partial v}{\partial Z} \end{bmatrix} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} \end{bmatrix} \tag{4 − 60}$$

(3) $\partial q/\partial \delta\xi$ is the derivative of the $\boldsymbol{q}$ regarding motion.

$$\frac{\partial \boldsymbol{q}}{\partial \delta\boldsymbol{\xi}} = [\boldsymbol{I}, -\boldsymbol{q}^{\wedge}] \tag{4 − 61}$$

Combine the latter two items them together, there are:

$$\frac{\partial u}{\partial \delta\boldsymbol{\xi}} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix} \tag{4 − 62}$$

This 2×6 matrix has been derived before, thus, derive the Jacobian matrix of the error relative to the camera motion transformation:

$$\boldsymbol{J} = -\frac{\partial \boldsymbol{I}_2}{\partial \boldsymbol{u}}\frac{\delta \boldsymbol{u}}{\partial \delta\boldsymbol{\xi}} \tag{4 − 63}$$

Assume there are *N* point pairs, it is possible to apply the above-mentioned method for obtaining the Jacobian matrix and apply G-N or L-M method to calculate the increment and conduct iteration to find optimized solution.

4.6.3.2 Discussion of Direct Method

As mentioned above, the three-dimensional position in space of *P* known. This is because using the RGB-D dataset, we can back-project any pixel into the three-dimensional space to project it into the adjacent image. In addition, the sparse direct method eliminates the process of calculating the descriptors, while just utilizes a few hundred points in image. Therefore, it will keep a pretty good time-efficiency. In particular, for the platforms with constrained computing resources, it will get a good real-time performance,

4.6.3.3 Experiment of Camera Motion Estimation Using the Direct Method

To verify the effectiveness of the above-mentioned approach, TUM dataset is employed so that the depth recovery part utilizing triangulation can be omitted. We utilize g2o optimization library to implement direct method for camera motion estimation. After defining the edge of g2o, we combine the nodes and edges into a graph. After that g2o is called for optimization. In this experiment, first read the image sequence of the dataset, and the first image is used as a reference frame, and then the direct method will be applied to solve the movement of the subsequent frames. In reference frame, the FAST key-points are extracted, then the positions of these key-points as well as the movement between the adjacent frames will be estimated by applying the direct method. Finally, we draw the positions of the tracked key-points on the adjacent frames. The experiment result is depicted in Figure 4-20. It presents that when the perspective between the adjacent frames changes slightly, the direct method adjusts the camera's motion so that most of the pixels are correctly tracked.

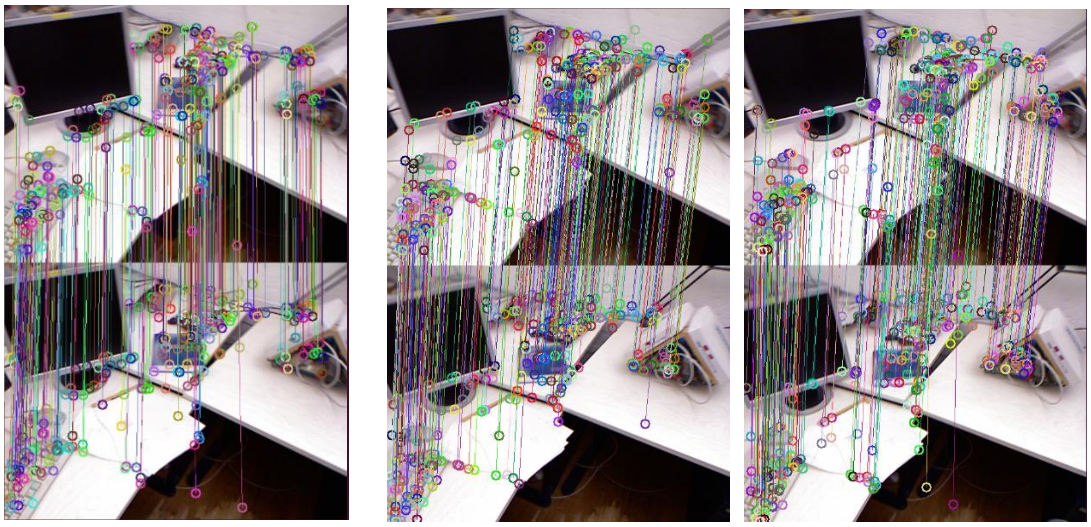
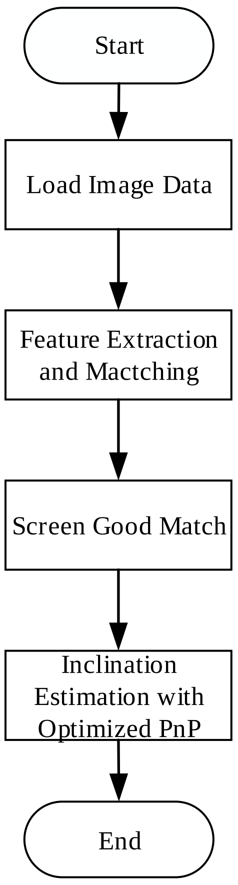


Figure 4-20. Experiment results of direct method

4.7 Implementation and Evaluation Experiments

This section gives out the outline of our proposed techniques, which solves the estimation of inclination angle of the assembly platforms in the industrial plant. First of all, the proposed method will extract the feature points from the input images captured by the industrial camera. One image is taken at the calibrated status of the assembly platform, while the other images are taken at the inclined status of the assembly platform. Then, the proposed method will associate the pictures taken at different time steps through feature matching and output the correlated feature points with the counterpart coordinates. After that, by employing the PnP method optimized by the trust region-based method, it realized the motion estimation between the pair of images. In our case, the estimated motion represents the desired inclination angle of the assembly platform. In the verification experiment, first of all, our work compared different combinations of the prevalent feature extraction and matching algorithms, including feature extraction algorithms like scale-invariant feature transform [29], speeded up robust features [30], oriented fast and rotated brief [31, 32, 33] and feature matching algorithms fast library for approximate nearest neighbors [35] and brute-force matching [30, 34]. The comparison results in total six cases indicates that the combination of ORB and BF achieves the best performance under the consideration of several factors such as matching accuracy, the amount of well-matched point pairs, and computation time which reflects the computation complexity of the algorithm. The reason of considering these factors is that, in our application scenery, the mounting plant of bearing factory, it requires the system having quick response ability as well as high measurement accuracy. On the other hand, for the inclination measurement part of our proposal, the optimized PnP algorithm outperforms the traditional PnP algorithm and epipolar-constraint-based method. Specifically, the optimized method greatly reduces the re-projection error and saves computation time compared to the traditional epipolar-constraint-based mechanisms. Figure 4-21. demonstrates the process flow of the proposed method which solves the problem of inclination measurement.



Start
Load Image Data
Feature Extraction and Mactching
Screen Good Match
Inclination Estimation with Optimized PnP
End

Figure 4-21. Workflow of the motion estimation-based inclination measurement

The rest of this section investigates the performance of the proposed mechanism for inclination measurement and compared it with other mechanisms. A prototype of the mechanism was implemented using IDE that illustrated in Chapter 2. For the validation experiments, we conducted several experiments utilizing images of Assembly Platform Dataset. To find the most suitable feature extraction and feature matching algorithm, we compared and evaluated the performance of multiple algorithm combinations. The comparison results are shown in Figure 4-22.

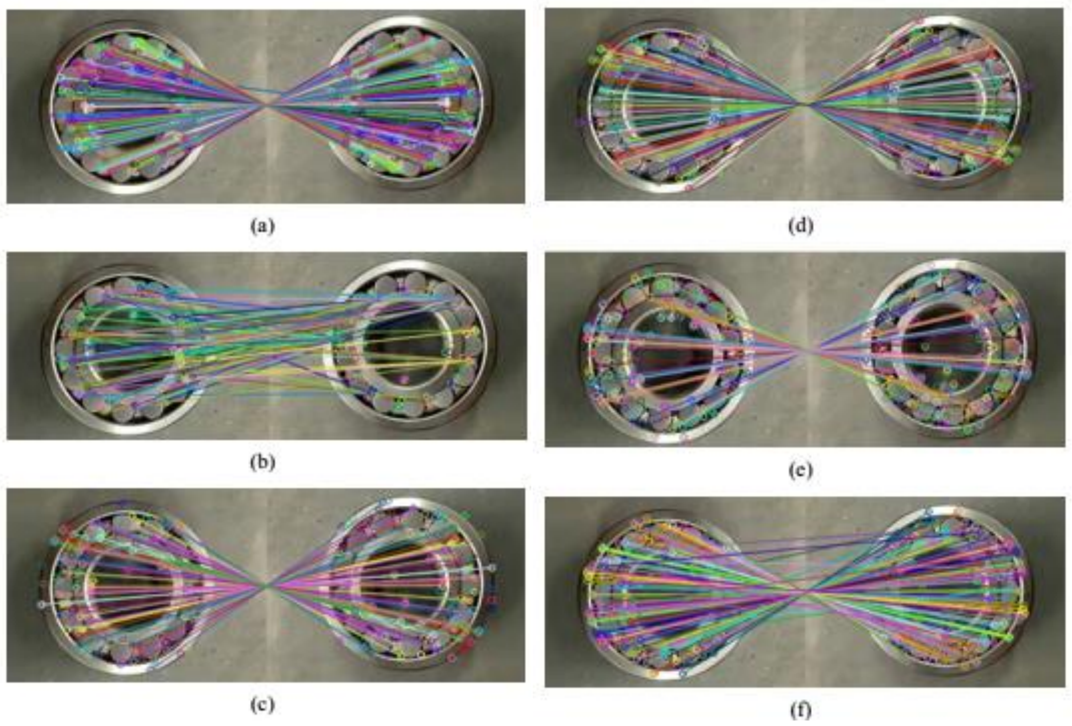


Figure 4-22. Comparison results of a few feature matching algorithms, (a) ORB with BF (b) ORB with FLANN (c) SURF with BF (d) SURF with FLANN (e) SIFT with BF (f) SIFT with FLANN

In the comparative experiment, we compared the combination of several mainstream feature extraction algorithms and matching algorithms. There are three kinds of feature extraction algorithms, namely SURF, SIFT, and ORB. The feature matching methods include the BF and FLANN. The comparison starts with several factors such as time consumption of the algorithm, number of matching points, excellent matching points, and good matching rate. Figure 4-24. shows the comparison results in regard of the computation time of the algorithms. In this case, the combination of ORB and BF matching outperforms other methods. In addition, the number of good-match and total number of matching points are summarized in Figure 4-25. It indicates that the 'ORB + BF' got the biggest number of good matching points. Therefore, as for the comparison of the good-match rate, experiment results are summarized in Figure 4-26. It exactly certificates that the ORB method combined with BF matching achieves the highest good-match rate.

Based on the comprehensive consideration about the performance of the different algorithms, we draw a conclusion that the combination of ORB and Bruce-force matching algorithm will be the most suitable one for the inclination measurement system, not only for its low computation complexity, but also for its stability and high performance. Besides, by setting the screening conditions, only a certain number of best matching point pairs are retained, which saves computing resources and shortens the overall running time of the algorithm, and at the same time improves the accuracy of the algorithm towards motion estimation. The evaluation experiments show that the ratio of the good match reaches over 90% in the case of applying the non-maximal suppression strategy with the tuned parameters.

Since the inclination of the assembly platform is required not to exceed 10 degrees in the actual production process, this study estimates the inclination angle between 1 degree and 10 degrees. The image of the scenery without inclination and the image of inclined platform were utilized to conduct the comparison and the evaluation experiments. The obtained matched point pairs among the images of standard and inclined platform are marked using connection lines, as shown in Figure 4-23. The results show part of the experiment results of feature matching in regard to the standard assembly platform and the inclined platform. The experiments prove that the combination of ORB and BF algorithm accurately matched over 92% of the corresponding feature points in the two images taken at different perspective. After obtained the accurate corresponding feature points, the coordinates of these feature points will be utilized in the process of the inclination angle estimation based on the optimized PnP method, which is one of the main contributions in this work. The performances of the proposed method and the traditional techniques in regard of the inclination angle estimation are summarized in Figure 4-27. The evaluation experiments proves that the optimized PnP method outperforms the traditional epipolar constraint-based methods, not only in time efficiency but also in the low re-projection error which indicates the high estimation accuracy.

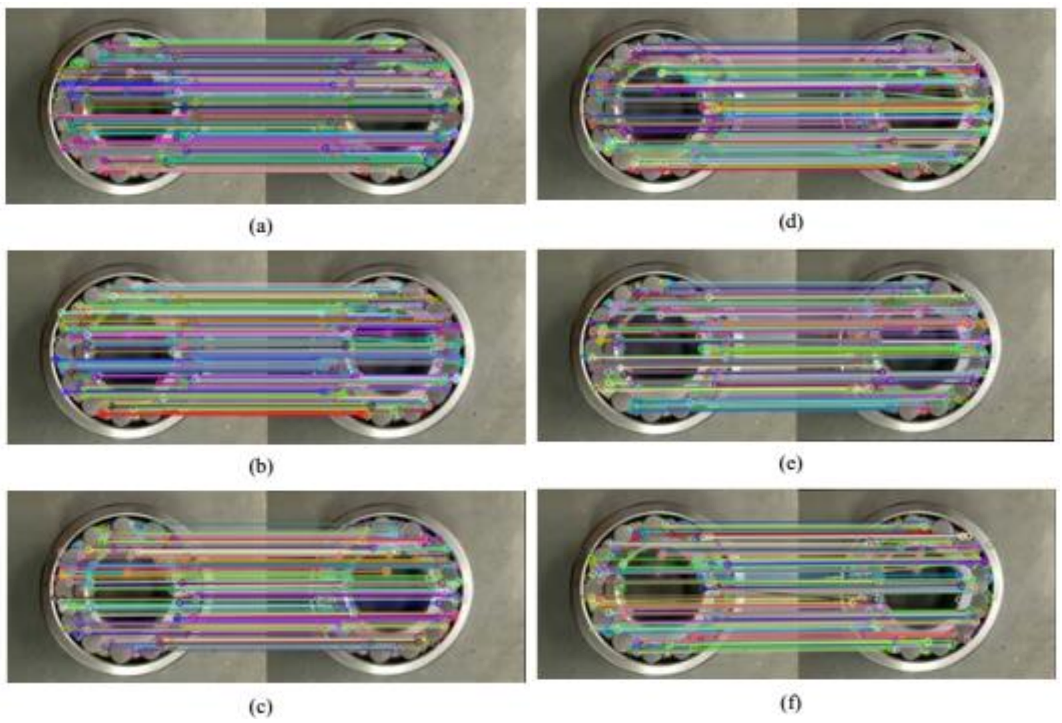


Figure 4-23. Performance of feature matching in cases of different inclination angles, (a) 1 degree of inclination (b) 2 degrees of inclination (c) 3 degrees of inclination (d) 5 degrees of inclination (e) 7 degrees of inclination (f) 9 degrees of inclination

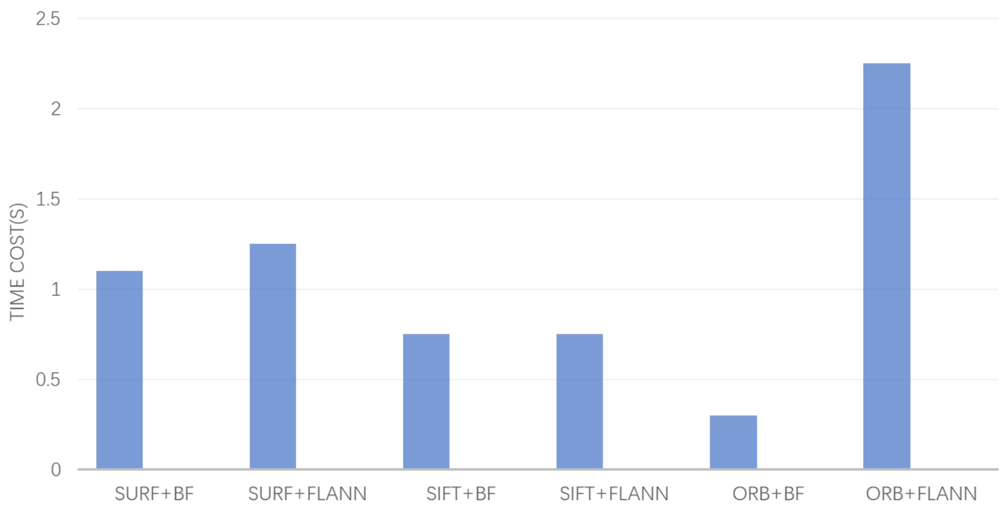


Figure 4-24. Time cost of different algorithm combinations

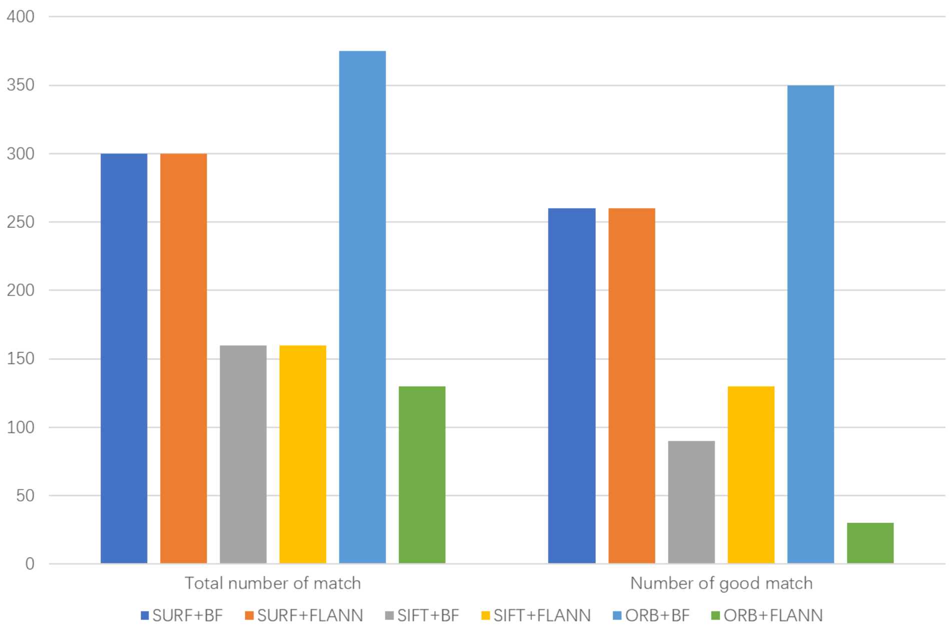


Figure 4-25. Total number of matching points and good match of different algorithms

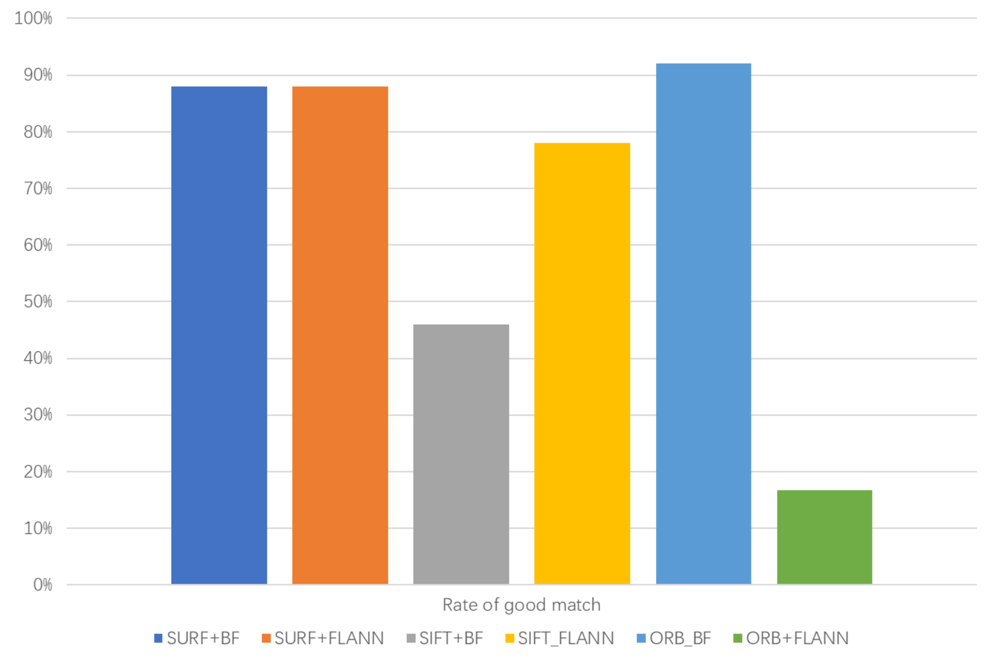


Figure 4-26. Good-match rates among different algorithm combinations

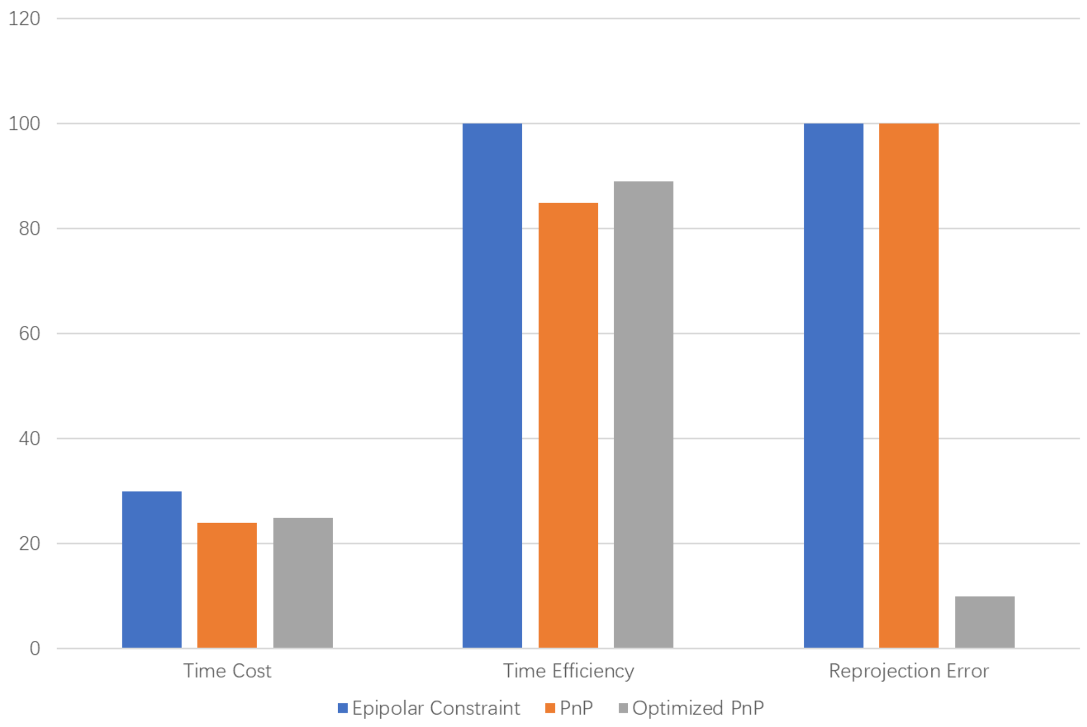


Figure 4-27. Performance comparisons about the traditional method and our proposal

4.8 Chapter Summary

Due to the requirements of high measurement accuracy, well real-time performance and limitations of computation complexity, inclination measurement for the bearing assembly platforms in an industrial environment remains a challenge. In this work, a series of algorithms have been applied to tackle the problem. First, the feature matching techniques have been employed to correlate the feature points of the current frame and reference frame. A binary descriptor-based matching algorithm with tuned parameters of the non-maximal suppression method have been found, in the interest of correlating the matched feature points as well as improving the good match rate. The good match rate reaches over 92% when applying the non-maximal suppression strategy. Besides, the optimization based on the trust region optimization of the conventional PnP algorithm has been realized. Afterwards, the optimized PnP is employed to estimate the inclination angle of the assembly platforms, to find the optimal solutions of the PnP problem. In other words, the optimized PnP algorithm minimizes the estimation error of the inclination measurements. Numerical results show that, time efficiency of the method for inclination measurement achieves 7.3% higher than the conventional epipolar constraints-based ones. On the other hand, the optimized PnP significantly reduces the measurement error by 90% compared with the epipolar constraints-based mechanisms. As for the future work, we will consider replacing the feature matching-based approaches with the feature tracking-based ones, which may further reduce computation cost as well as improve the estimation accuracy.

Chapter 5 Inclination Measurement Based on Artificial Intelligence

5.1 Introduction

Machine learning (including deep learning) is an engineering science that is largely directed by the engineering practice, not the theory. In this case, the improvements in algorithms are available only if there are sufficient computational power as well as dataset for the engineers to make their ideas into practice [93, 94]. In the two decades from 1990 to 2010, the Internet developed rapidly, while the graphics boards with high computational power were designed to meet the demands of the computer games. During this period, the ascend of the Internet also made it available to accumulate, store and distribute tremendous datasets for try new ideas in the machine learning field. Except for the computing resources and validation dataset, twenty years ago, the researchers still had no appropriate way to train the neural networks with over 10 layers [90]. This changed until 2009, when several very important algorithm improvements emerged, including better activation functions and better optimization schemes. These improvements allow researchers to train models with more than 10 layers [107]. So far, as a branch of machine learning, deep learning has begun to shine. In the following years, some other excellent methods that benefit with the gradient propagation have been developed, including the batch normalization, residual structures, as week as the depthwise separable convolutions. Deep learning has revolutionized progress in practice, yielding impressive results on perceptual problems, such as very high-performance classification for computer vision and the recognition with high accuracy for speech [108]. Although these problems mentioned are very easy to handle for the humans, but for several decades, they are regarded very hard for machines to solve*.

In recent years, people have gradually begun to apply deep learning to many important problems, from medical diagnosis to digital assistants, where deep learning has played a transformative role. AI research has been advancing at an astonishing speed over the past decade, in large part due to unprecedented funding in AI research field, but so far, few of these advances have been able to convert into the real products or the practical processes that change the various industries. Besides, most of the achievements in deep learning has not been applied to solve the problems in many parts of people’s daily life.

*This chapter is a refined and reproduced version of the paper originally published in Wireless Communications and Mobile Computing [92].

Especially in the production line of the manufacturing industry, the control systems still widely employ the relatively traditional but highly reliable controller represented by Programmable Logic Controller. After investigation and research, as described in previous chapters, our group manage to design an inclination measurement system suitable for the bearing assembly platforms. In addition, a series of algorithms based on the machine learning as well as the neural networks have been applied on the engineering project of the inclination measurement [91, 92]. However, for the simplex machine learning methods, although they could obtain a relatively good performance regarding the classification of inclination angles, the classification accuracy is not good enough that can satisfy the high requirements of the assembly plant. On the other hand, for the deep learning models, the complex structure and tremendous of parameters make it hard to address the classification tasks with limited training samples.

To overcome these problems, as well as to fulfill the constraints about computing resources from the assembly enterprise, the main works of this study are as follows: First of all, a shallow network structure is designed that is efficient and robust for tackling the classification of inclination angles. Secondly, considering the characteristics of the Assembly Platform Dataset, feature engineering based on K-Means clustering have been implemented to reduce the computational burden and to improve the classification accuracy. Besides, through extensive experiments, the optimal value of K has been found. Finally, some baseline machine learning algorithms have been implemented and utilized in the validation experiments to validate the effectiveness and performance of the image classification-based inclination measurement.

The rest of the chapter is organized as follows: Section 5.2 illustrates the basic principles of the Multilayer Perceptron, activation functions, and training process. Section 5.3 demonstrates the shallow network structure designed in this study. Section 5.4 illustrates the baseline methods employed in this study, while section 5.5 presents the validation experiments conducted to examine the effectiveness and performance of the designed neural networks.

5.2 Neural Networks

Multilayer Perceptron (MLP) is an artificial neural network with a forward structure that maps a set of input vectors to a set of output vectors [107]. MLP can be viewed as a directed graph consisting of multiple layers of nodes, each of which is fully connected to the next layer. Except for the input nodes, each node is a neuron or processing unit with a nonlinear activation function. A supervised learning method called the backpropagation algorithm is often used to train MLPs. Multilayer perceptron follows the principles of the human nervous system to learn and make data predictions. It first learns, then uses weights to store data, and uses algorithms to adjust weights and bias during training, to reduce the errors between actual and predicted values. The basic structure of multi-layer perception consists of three layers: the first input layer, the middle-hidden layer, and the last output layer. The product of input elements and weights is fed to a summation node with neuron biases. The main advantage is its ability to quickly solve complex problems. MLP is a generalization of the perceptron, which overcomes the weakness that the perceptron cannot identify linearly indivisible data.

5.2.1 Activation Function

If the activation function of each neuron is a linear function, then an MLP with any number of layers can be reduced to an equivalent single-layer perceptron [107]. In fact, the MLP itself can use any form of activation function, such as the step function, logic sigmoid function, etc. However, to use the backpropagation algorithm for effective learning, the activation function must be restricted to differentiable functions. Due to their good differentiability, many S-functions, especially the hyperbolic tangent and logistic functions, are used as activation functions.

In recent developments in deep learning, linear rectification is more frequently used to overcome numerical problems associated with sigmoid functions.

The two historically common activation functions are hyperbolic tangent function (Tanh) and Sigmoid function, and are described by

$$y(v_i) = \tanh(v_i) \qquad (5-1)$$

$$y(v_i) = (1 + e^{-v_i})^{-1} \qquad (5-2)$$

The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function, which is similar in shape but ranges from 0 to 1. Here y_i is the output of the i^{th} node (neuron) and v_i is the weighted sum of the input connections. Alternative activation functions have been proposed, including the rectifier and softplus functions. More specialized activation functions include radial basis functions (used in radial basis networks, another class of supervised neural network models).

5.2.1.1 Rectified Linear Unit

Rectified Linear Unit (ReLU), is an activation function commonly used in artificial neural networks, usually referring to nonlinear functions represented by ramp functions and their variants.

$$f(x) = max(0, x) \qquad (5-3)$$

The more commonly used linear rectification functions are the ramp function, and the leaky rectification function (Leaky ReLU), where $\boldsymbol{x}$ is the input to the neuron. Linear rectification is considered to have a certain biological principle and is widely used by today's deep neural networks for applications such as image recognition because it usually has better performance than other commonly used activation functions (such as logistic functions) in practice.

5.2.1.2 Softmax Function

In mathematics, especially in probability theory and related fields, the Softmax function, or normalized exponential function, is a generalization of the logistic function. It can generalize a K-dimensional vector containing any real number into another K-dimensional real vector, so that the range of each element is between $(0, 1)$, and the sum of all elements equal to 1. The form of Softmax function is usually given by the following formula:

$$\sigma(z)_j = \frac{e^{z^j}}{\sum_{k=1}^{K} e^{z_k}} \quad for\ j = 1, \dots, K. \qquad (5-4)$$

The Softmax function is actually a logarithmic normalization of the gradient of a finite-term discrete probability distribution. Therefore, Softmax function is widely used in various probability-based multi-classification problem methods including multinomial logistic regression, multinomial linear discriminant analysis, naive Bayes classifier and artificial neural network. Especially, in artificial neural network, the input to the function is the result obtained from K different linear functions, and the probability that the sample vector $\boldsymbol{x}$ belongs to the j^{th} class is:

$$P(y = j|\boldsymbol{x}) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \qquad (5-5)$$

5.2.2 Learning/Training

Learning means the process of updating the weights after a portion of data has been processed, while the updating depends on the difference between the output value and the expected value. This gives a sample process of the collectively called supervised learning. The algorithm applied to update the weights is backpropagation, which is commonly regarded as a milestone in the field neural networks [101].

Denote the error of an output node j for the n^{th} data point (training samples) as

$e_j(n) = d_j(n) - y_j(n)$,in which d represents the target value while y denotes the value produced by the perceptron. Therefore, the weights of the nodes will be adjusted based on corrections that minimize the error in the entire output, given by:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \qquad (5-6)$$

Applying gradient descent, the change in each weight will be:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \qquad (5-7)$$

where y_i denotes the output of the previous neuron, while η represents the learning rate, which is determined to guarantee the convergence of the weights.

The derivative to be calculated depends on the induced local field v_j, which itself varies. It is available to certificate that for one output node, the derivative could be simplified as:

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\phi'\left(v_j(n)\right) \qquad (5-8)$$

Where ϕ' represents the derivative of the activation function mentioned above, which itself does not vary. The analysis of updating the weights for a hidden node will be much difficult, therefore here directly gives out the derivative as the following:

$$-\frac{\partial \varepsilon(n)}{\partial v_j(n)} = \phi'\left(v_j(n)\right)\sum_k -\frac{\partial \varepsilon(n)}{\partial v_k(n)} w_{kj}(n) \qquad (5-9)$$

It relies on the update of weights of the k^{th} nodes, which denotes the output layer. Thus, to update the weights of the hidden layers, the weights of the output layer must update in terms of the derivative with respect to the activation function. In sum, the above-mentioned illustrates the backpropagation process of the activation function [101].

Backpropagation method is commonly employed by MLPs for training, while it represents one of the most classic training approaches in the field of machine learning [107]. Besides, MLPs is still a prevalent research topic in the machine learning field, because it is not only feasible to approximate the complex functions, but also performs well in many classification tasks. In recent years, MLP has regained the attention because of the advances in deep neural networks.

5.3 Design of the Neural Networks

5.3.1 Feature Engineering

Feature engineering refers to hard-coded transformations (not learned by the model) on the data using our own knowledge of the data and machine learning algorithms to improve the performance of the model before feeding it into the model. In most cases, a machine learning model cannot learn from completely arbitrary data. The data presented to the model should be easy for the model to learn.

Let's look at an intuitive example. Suppose we want to develop a model, input a clock image, the model can output the corresponding time, as shown in Figure 5-1.

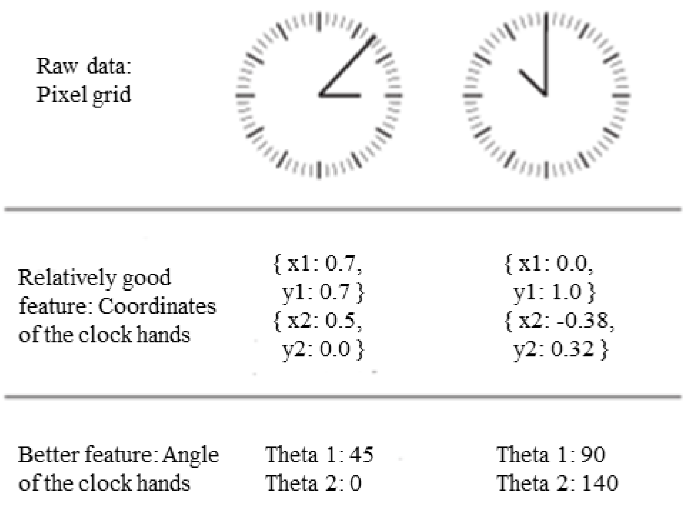

Raw data: Pixel grid
Relatively good feature: Coordinates of the clock hands
{ x1: 0.7, y1: 0.7 } { x2: 0.5, y2: 0.0 }
{ x1: 0.0, y1: 1.0 } { x2: -0.38, y2: 0.32 }
Better feature: Angle of the clock hands
Theta 1: 45 Theta 2: 0
Theta 1: 90 Theta 2: 140

Figure 5-1. Feature engineering to read time from the clock

If we choose to use the raw pixels of the image as input data, then this machine learning problem will be very difficult. Not only does it take a convolutional neural network to solve this problem, but it also takes a lot of computational resources to train the network.

But if we understand the problem from other perspectives, we can find alternative features to be utilized by the classifiers. Thus, it is available to write a simple Python script to find the black pixel corresponding to the clock hands and output the (x, y) of each pointer tip coordinate. A simple machine learning algorithm can then learn how these coordinates correspond to time.

We can also think a little further: make a coordinate transformation to convert the (x, y) coordinates to polar coordinates relative to the center of the image. In this way, the input becomes the angle theta of each clock hand. The features now make the problem so simple that no machine learning is needed at all, as applying the dictionary lookups are sufficient to complete the task.

This example reveals the essence of feature engineering: formulating the problem in a simpler way makes the problem easier. It usually requires a deep understanding of the problem.

Before the advent of deep learning, feature engineering used to be very important because classical shallow algorithms did not have a large enough hypothesis space to learn useful representations on their own. The way data is presented to an algorithm is critical to problem solving. For example, prior to the success of the convolution manipulation of deep learning on the image classification task for the digit utilizing MNIST dataset, the mainstream methods at that time usually employing the hard-coded features, including the circles numbers in a picture, or the height of the digits, the histogram with respect to the grayscale values, etc.

Luckily, with the advances of the deep learning, in more cases, the researchers need not to conduct feature engineering since it is capable for the neural networks to extract beneficial features from the original data. However, it does not indicates that the researchers do not need to worry about feature engineering if applying the convolution neural networks, due to the following cause.

Excellent features make it feasible to use less computing resources to achieve higher performance of the algorithm. For example, using deep learning to read time on the clock face is thankless.

Good features allow us to solve problems with less data. The ability of deep learning models to learn features autonomously relies on a large amount of feasible dataset. If only a limited number of training data is available, the informative meaning of the feature will become very important.

In this study, the proposed inclination measurement method which based on image classification largely depend on the image features. After obtaining hundreds of image feature points from the input samples, we will get many binary feature descriptors with high dimensionality which are redundant and will cost more computing resources for training. Therefore, instead of directly utilizing feature vectors with high dimensionality to train the neural networks, we employ K-means algorithm to conduct feature engineering, namely, to reduce the dimensionality of the feature vectors. In this way, not only can we save the computation resources, but also will improve the classification performance of the proposed neural networks.

By making feature engineering, we finally got the feature vectors suitable for this project. Based on this, we designed the following neural network structure, as shown in Figure 5-3. The input layer of the neural network contains 70 nodes. This is because the clustering algorithm obtained the optimal number of feature centers is 70. The hidden layer contains 3 layers. Besides, the three hidden layers contain 140 nodes, 280 nodes and 540 nodes, respectively. The last layer is the output layer with softmax activation function, which outputs the probability information of each classification category. The reason of doubling the nodes in the hidden layer is to increase the information capacity of the network and improve the generalization ability of the network (this will be introduced later on the neural network design method). Furthermore, we utilize Keras library to realize the specific implementation of proposed networks.

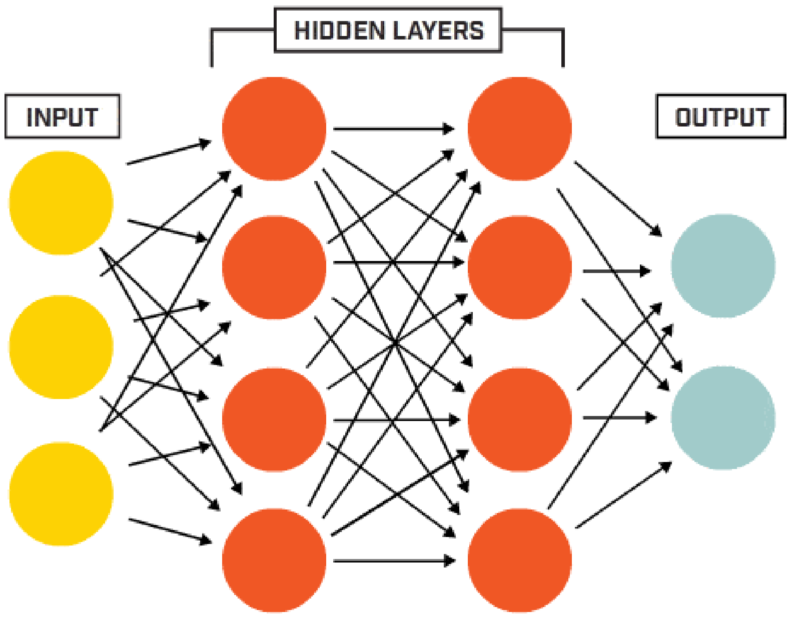


Figure 5-3. Examples of the neural networks [108]

5.3.2 Underfitting and Overfitting

During training and validation of the initially designed neural network, we found that as the training process lasts for a few epochs, the performance of the network utilizing the validation dataset will usually reach a peak value, afterward, it started to decline. That is, the overfit phenomenon of the network starts to occur during the training process. Overfitting is a common problem when training the neural networks. Therefore, knowing the methods to deal with the overfitting will be critical. Finding a balanced position between the optimization and generalization is necessary. Optimization indicates the process of tuning a model to get the best performance on training data, while generalization considers that whether a tuned network performs well on unseen samples. Definitely, the goal of neural networks is to obtain an excellent performance in generalization.

For the early epochs of training, the optimization and generalization are related: the smaller the loss on the training samples, the smaller the loss on the testing samples. The model at this time will be underfit, namely, there is still room for improvement, and the network has not modeled all relevant patterns in the training data. However, after a certain number of iterations on the training data, the generalization is no longer improved, and the verification index is unchanged at first, and then begins to deteriorate, that is, the model begins to overfit. This is when the model starts to learn patterns that are only relevant to the training data, but that are completely irrelevant to the unknown samples.

To avoid the model from learning wrong patterns from the training samples, one optimal method is to obtain a large number of training samples. The more training samples the model has, the better its generalization ability. If larger dataset is not available, the second method is to adjust the memory capacity for the model to store information, or to place limitation conditions on the information that the model stores. In the case that a network could only remember a few patterns, the optimization process forces the model to concentrate on learning the most important patterns, that is more likely to generalize well. This method of reducing overfitting is called regularization.

The easiest method to avoid overfitting is to eliminate the model capacity, that is, reduce the number of layers/nodes in the network. In deep learning, the quantity of parameters in a network is often regarded as the capacity of the network. Basically, a model with a large number of learnable parameters will have a larger capacity for memorization, thus it will be available to learn an excellent dictionary mapping between training data and targets without any generalization ability. Assume a network with 500,000 binary parameters will attain the ability of learning the classes corresponding to all the digits in the MNIST training set - we just need to assign 50,000 digits to 10 binary parameters each. However, such models will fail to classify any other unknown samples.

Deep learning models are usually pretty good at fitting the training data, but the real challenge is generalization, not fitting. In contrast, if the network has limited memory resources, this mapping cannot be easily learned. Therefore, to minimize the loss, the model must fit one compressed representation which is highly predictive of the target, which is the data representation we are interested in. We also need to notice that the model we use should have enough parameters to prevent underfitting, i.e. the model should avoid running out of memory resources. Thus, we need to find a tradeoff between overcapacity and under-capacity.

However, there is no magic formula that can calculate the optimal layer numbers and the optimal capacity of different layers. Thus it must evaluate a range of different network structures on the dataset to obtain the best structure regarding the dataset. To find an appropriate model size, the general workflow is to start with a relatively small number of layers and parameters, and then gradually increase the size of the layers or add new layers until the effect of this increase on the validation loss becomes small.

5.3.2.1 Add weight regularization

The principle of Occam's razor is widely used in the natural sciences. Occam's Razor: If there are two explanations for an event, the most likely correct explanation is the simplest one, the one with fewer assumptions. This principle also applies to models learned by neural networks: given some training data and a network architecture, many sets of weight values (that is, many models) can explain the data. Simple models are less prone to overfitting than complex models.

A simple model here refers to a model with less entropy for the distribution of parameter values or a model with fewer parameters. Therefore, a common method to reduce overfitting is to force the model weights to only take small values, thereby limiting the complexity of the model, which makes the distribution of weight values more regular. This approach, called weight regularization, is accomplished by adding a cost associated with larger weight values to the network loss function. It mainly includes two forms: L1 and L2 regularizations.

Figure 5-4. below shows the effect of L2 regularization utilizing Keras. It shows that the network adding the L2 regularization (dots) is less likely to overfit compared with the reference network (crosses), even if the two models have the same capacity.

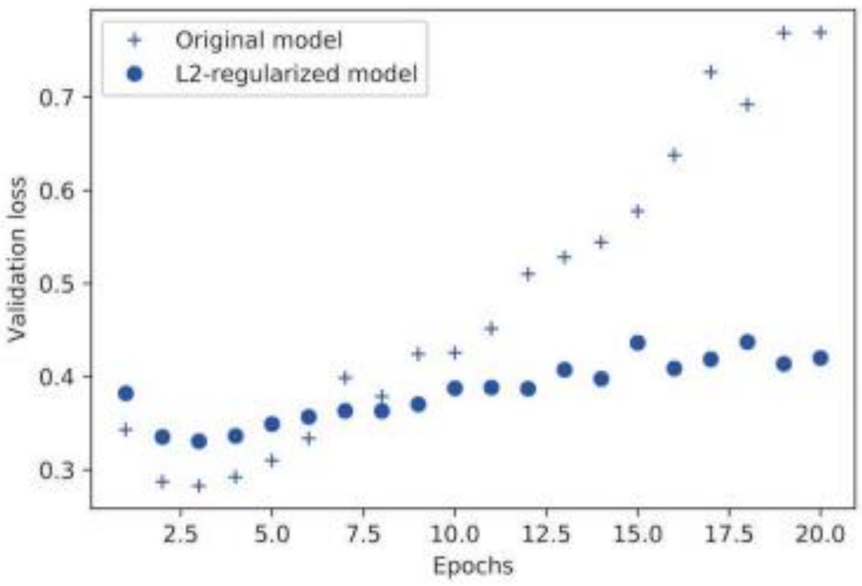


Figure 5-4. Experiment of adding L2-regularization [108]

5.3.2.2 Add Dropout Regularization

One of the most effective and commonly used regularization methods for neural networks is dropout, which was developed by Geoffrey Hinton and his students at the University of Toronto. Using dropout for a layer is to randomly drop some output features of the layer (set to 0) during the training process. Suppose that during training, the return value of a layer for a given input sample should be a vector [0.2, 0.5, 1.3, 0.8, 0.7]. After using dropout, a few random elements of this vector will become 0, such as [0, 0.5, 1.3, 0, 0.7]. The dropout rate is the proportion of features that are set to 0, usually in the range of 0.2~0.5. No units are dropped at test time, and the output value of this layer needs to be scaled down by the dropout ratio because there are more units activated than at training time, which needs to be balanced. The main idea is to introduce disturbance in the output values of a layer, breaking insignificant accidental patterns. If there is no noise, the network will remember these occasional patterns.

Figure 5-5. below shows an illustration of the result of adding dropout regularization. Again, we see that the performance of this method is significantly improved compared to the reference network.

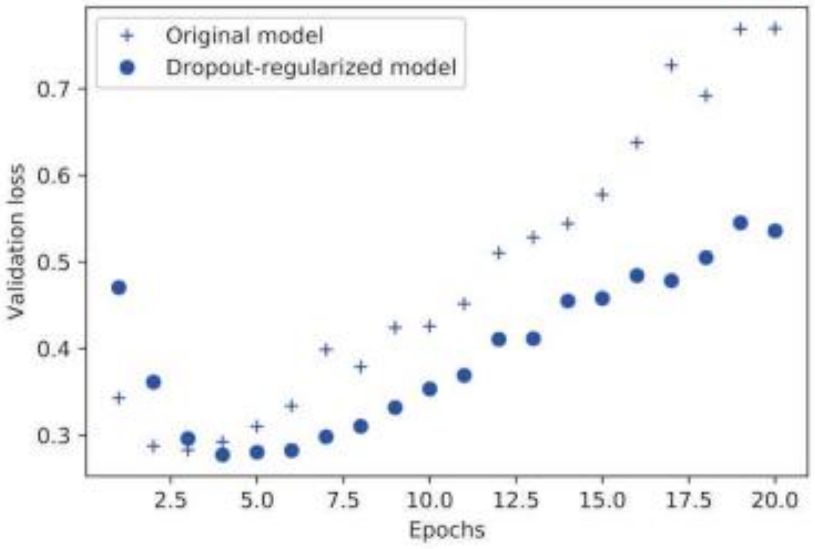


Figure 5-5. Experiment of adding Dropout regularization [108]

5.3.3 Choose Metrics to Measure Success

For balanced classification problems (equal likelihood for each class), accuracy and area under the receiver operating characteristic curve (ROC AUC) will be the commonly used metrics. As of the imbalanced class, precision and recall could be utilized. About the ranking problems or multi-label classification, mean average precision could be adopted. Custom metrics to measure success are also common. In addition to this, browse the data science competitions on the Kaggle website, which showcase a variety of questions and evaluation metrics. Since this study belongs to a balanced classification problem, accuracy is used as the evaluation metric. Therefore, in this study, due to the limited number of samples in the dataset, we adopted the "repeated K-fold validation" method to evaluate the designed neural network model.

5.3.4 Model Regularization and Tuning Hyperparameters

The step of model regularization and tuning hyperparameters is the most time-consuming: because it requires constant tuning of the model, training, evaluation on validation data, tuning the model again, and repeating the process until the model achieves optimal performance. We have done the following work.

- Add dropout.
- Try different architectures: increase or decrease the number of layers.
- Add L2 regularization.
- Try different hyperparameters (like the number of units per layer) to find the best configuration.

It is important to note that every time the model is tuned using feedback from the validation process, knowledge about validation process is leaked into the network. In the case when it is only repeated a few times, it doesn't matter; but if it is systematically iterated many times, we can end up overfitting the model to the validation process. it reduces the reliability of the verification process. To avoid this situation, this study uses repeated K-fold validation to evaluate the model.

After obtaining a satisfactory model configuration, three hidden layers, doubling the number of nodes in hidden layers, L2 regularization for hidden layers, ReLu activation function for input and hidden layers, Softmax activation function for output layer, we train the fine-tuned model on all available data $(training\ data\ +\ validation\ data)$, and then finally evaluate once on the test set. The experimental results show that the performance of the model on the test set is the same as that on the validation set.

5.3.5 Selection of the Activation Function

Without activation functions such as ReLU, the fully connection layer can only make linear operations.

$$output\ =\ dot(W, input) +\ b \qquad (5-10)$$

In this way, the Dense layer can approximate the linear transformations of the input data only: the hypothesis space of this layer is the set of all possible linear transformations from the input samples to the output value. This hypothesis space is very limited and cannot take advantage of multiple representation layers, because the stacking of multiple linear layers still implements linear operations, and adding layers does not expand the hypothesis space. To obtain a richer hypothesis space and take full advantage of the multi-layer representation, we employ the commonly used ReLu.

5.3.6 Selection of the Cost Function and Optimizer

Finally, we need to choose a loss function and optimizer. Since this study is faced with a multi-classification problem, and the network output is a probability value, (the last layer of the network uses a softmax activation function), cross-entropy is the best choice. Cross-entropy is a concept from the field of information theory that measures the distance between probability distributions. In the experimental verification, we use Categorical_crossentropy in the Keras library as the loss function. The optimizer selects the Adam optimizer and selects the default parameter configuration.

5.4 Machine Learning Algorithms

5.4.1 KNN

The k-nearest neighbor (KNN) algorithm is a basic classification and regression method. We only discuss the k-nearest neighbor algorithm in classification problems here [97].

The input of the k-nearest neighbor algorithm is the feature vector of the instance, corresponding to the point in the feature space; the output is the class of the instance, which can take multiple classes. The k-nearest neighbor algorithm assumes that given a training dataset, the class of instances in it is given. When classifying, the new instance is predicted by majority voting according to the class of its k nearest neighbors of the training instance. Therefore, the k-nearest neighbor algorithm does not have an explicit learning process.

In practice, the k-nearest neighbor algorithm uses the training dataset to partition the feature vector space and act as a "model" for its classification. The choice of k value, the distance measure and the classification decision rule are the three basic elements of the k-nearest neighbor algorithm. The workflow of KNN is as follows:

1. Suppose there is a labeled sample data set (training sample set), which contains the correspondence between each piece of data and the category to which it belongs.
2. After entering new data without labels, compare each feature of the new data with the features corresponding to the data in the sample set.

i. Calculate the distance between the new data and each data in the sample data set.
ii. Sort all the obtained distances (from small to large, smaller means more similar).
iii. Take the classification labels corresponding to the first k (k is generally less than or equal to 20) sample data.
3. Find the most frequent classification label among the k data as the classification label of the new data.

5.4.1.1 Application Scenarios of KNN

Movies can be classified by subject matter, so how can we differentiate between action movies and romance movies?
1. Action Movies: More fights
2. Romance Movies: More kisses
Based on the number of kisses and fights in the movie, the k-nearest neighbor algorithm is used, and the genre of the movie can be automatically divided.

Table 5-1 Dataset for the classification of movie types

Title of the Movie	*Number of fight scenes*	*Numbers of kiss scenes*	*Type of the Movie*
California Man	*3*	*104*	*Romance movie*
He's Not Really into Dudes	*2*	*100*	*Romance movie*
Beautiful Woman	*1*	*81*	*Romance movie*
Kevin Longblade	*101*	*10*	*Action Movie*
Robo Slayer 3000	*99*	*5*	*Action Movie*
Amped II	*98*	*2*	*Action Movie*
?	*18*	*90*	*unknown*

Table 5-2. Distance between known movies and the unknown movie

Title of the Movie	*Distance with the Unknown Movie*
California Man	*20.5*
He's Not Really into Dudes	*18.7*
Beautiful Woman	*19.2*
Kevin Longblade	*115.3*
Robo Slayer 3000	*117.4*
Amped II	*118.9*

According to the distances between all the movies in the sample set we obtained above and the unknown movies, we can find k movies with the closest distances in ascending order of distances. Assuming $k = 3$, the three closest movies are, in order, He's Not Really into Dudes, Beautiful Woman, and California Man. The k-nearest neighbor algorithm determines the type of the unknown movie according to the types of the three closest movies, and these three movies are all romance movies, so we determine that the unknown movie is a romance movie. In the project of inclination measurement system, we mainly employ KNN as a baseline in the validation experiments.

5.4.2 K-Means Algorithm

K-Means is a clustering algorithm that finds K clusters of a given dataset [95]. It is called K-means because it can find K different clusters, and the center of each cluster takes the mean value of the value contained in the cluster. The number of clusters K is specified by the user, and each cluster is described by its centroid, that is, the center of all points in the cluster. The biggest difference between clustering and classification algorithms is that the target category of classification is known, and the target category of the clustering is unknown. In this study, we mainly employ K-Means to make feature engineering instead of directly use the image features to train the proposed neural network model. Advantages and shortcomings of K-Means are summarized as follows:
Advantage:

- Belongs to unsupervised learning, no need to prepare training set.
- Easy to implement.
- Good interpretability of results.

Shortcomings:

- The k value needs to be set manually. Before the algorithm starts to predict, we need to manually set the k value, that is, to estimate the approximate number of categories of the data. An unreasonable k value will make the results less explanatory.

- May converge to local minima.
- Time-consuming on large datasets, and sensitive to outliers.

5.4.3 Classification Method Based on Bayesian Decision Theory

5.4.3.1 Bayesian Theory

Bayesian classification is a general term for a class of classification algorithms, all of which are based on Bayesian theorem, so they are collectively referred to as Bayesian classification [98]. We first introduce the basis of the Bayesian classification algorithm - Bayesian theorem, and then discuss the simplest one of Bayesian classification: Naive Bayesian classification through an example.

Assume that we now have a dataset that consists of two classes of data, with the distribution of the data as shown in Figure 5-6.

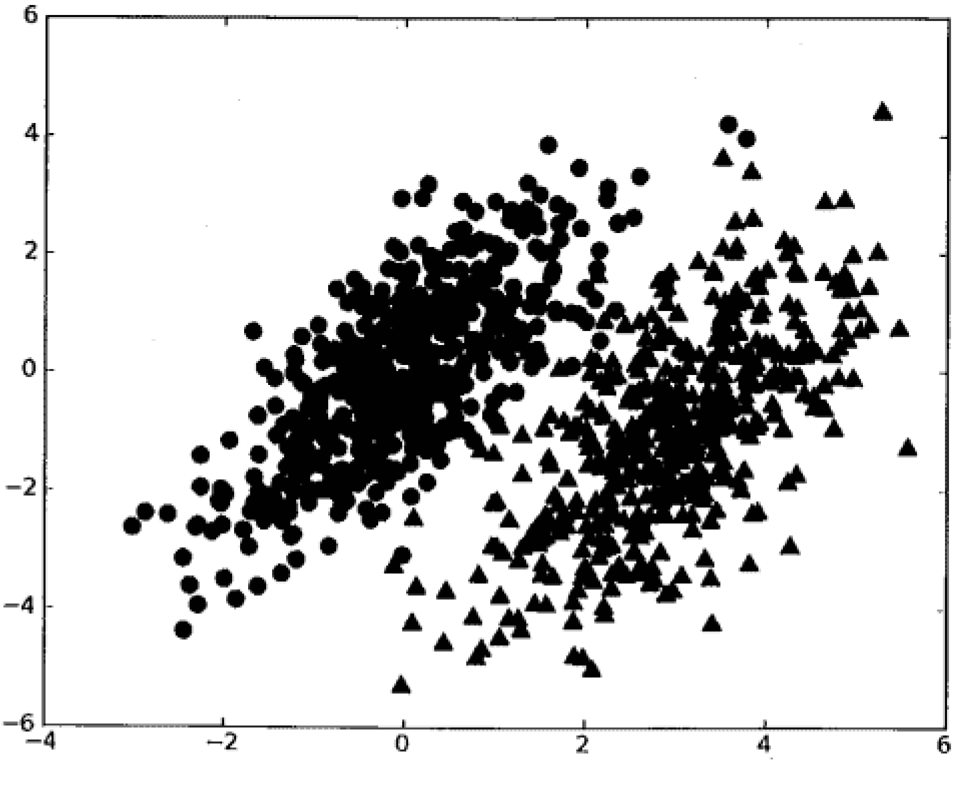


Figure 5-6. Two probability distribution with known parameters, and the parameters determining the shape of the distribution [107]

We now use $p_1(x, y)$ to denote the probability that the data point (x, y) belongs to class 1 (the class represented by the dots in the figure), and $p_2(x, y)$ to denote that the data point (x, y) belongs to the class 2 (the category represented by the triangle in the figure), then for a new data point (x, y), the following rules can be used to determine its category:

- If $p_1(x, y) > p_2(x, y)$, then it belongs to class 1;
- If $p_2(x, y) > p_1(x, y)$, then it belongs to class 2.

That is, we will choose the category with high probability. This is the core idea of Bayesian decision theory, which is to choose the decision with the highest probability.

5.4.3.2 Conditional Probability

There is a jar with 7 stones, of which 3 are white and 4 are black. If a stone is randomly taken from the jar, what is the probability that it is a white stone? Since there are 7 possibilities to pick a stone, 3 of which are white, the probability of picking a white stone is $3/7$. So what is the probability of getting a black stone? Obviously, it's $4/7$. We use $p(white)$ to represent the probability of getting a white stone, which can be obtained by dividing the number of white stones by the total number of stones.

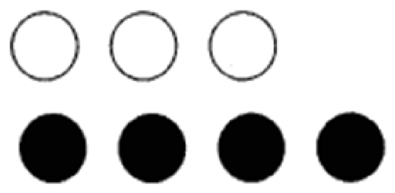


Figure 5-7. A collection of seven stones in either white or black color

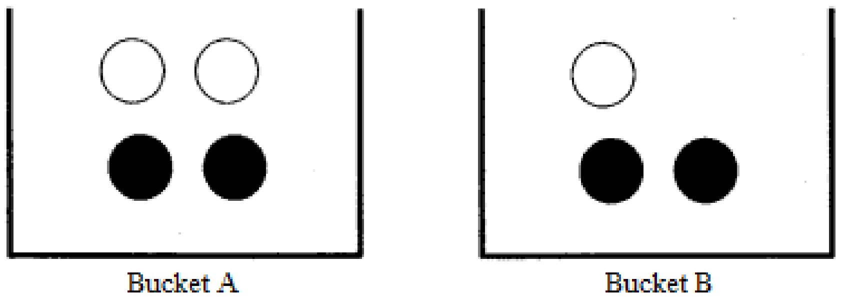


Figure 5-8. Seven stones in two buckets

If the 7 stones are placed in two buckets as shown in Figure 5-8., how should the above probability be calculated?

Calculate $p(white)$ or $p(black)$, if we know in advance the information of the bucket where the stone is, it will change the result. This is called conditional probability. Assuming that the calculation is the probability of taking a white stone from bucket B, this probability can be recorded as $p(white|bucketB)$, we call it "probability of taking a white stone under the condition that the stone is known to come from bucket B". It is easy to get that the value of $p(white|bucketA)$ is $2/4$ and the value of $p(white|bucketB)$ is $1/3$.

The formula for calculating conditional probability is as follows:

$$p(white|bucketB) = p(white\ and\ bucketB) \,/\, p(bucketB) \qquad (5-11)$$

First, we divide the number of white stones in bucket B by the total number of stones in the two buckets to get $p(white\ and\ bucketB) = 1/7$. Second, since there are 3 stones in bucket B, and the total number of stones is 7, so $p(bucketB)$ is equal to $3/7$. So again $p(white|bucketB) = p(white\ and\ bucketB) \,/\, p(bucketB) = 1/3$.

Another way to efficiently compute conditional probabilities is called the Bayesian criterion. The Bayesian criterion tells us how to exchange conditions and outcomes in conditional probability, that is, if $p(x|c)$ is known and $p(c|x)$ is required, then the following calculation method can be used:

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)} \qquad (5-12)$$

5.4.3.3 Making Classification with Conditional Probability

Above we mentioned that Bayesian decision theory requires the computation of two probabilities $p_1(x,y)$ and $p_2(x,y)$:

- if $p_1(x,y) > p_2(x,y)$, then it belongs to class 1;
- If $p_2(x,y) > p_1(x,y)$, then it belongs to category 2.

This is not all about Bayesian decision theory. The use of p_1 and p_2 is just to simplify the description as much as possible, but what really needs to be calculated and compared is $p(c_1|x,y)$ and $p(c_2|x,y)$. The specific meanings represented by these symbols are: Given a data point represented by (x,y), what is the probability that the data point is from class c_1? What is the probability that the data point is from class c_2? Note that these probabilities are not the same as the probabilities $p(x,y|c_1)$, but Bayesian criterion can be used to exchange conditions and outcomes in probabilities. Specifically, applying the Bayesian criterion to get:

$$p(c_1|x,y) = \frac{p(x,y|c_1)p(c_1)}{p(x,y)} \qquad (5-13)$$

Using these definitions above, the Bayesian classification criterion can be defined as:

• if $p(c_1|x,y) > p(c_2|x,y)$, then it belongs to class c_1;
• If $p(c_2|x,y) > p(c_1|x,y)$, then it belongs to class c_2.

In document classification, the entire document (such as an email) is the instance, and certain elements in the email constitute features. We can observe the words that appear in the document, and use each word as a feature, and the occurrence or non-occurrence of each word as the value of the feature, so that the number of features will be as many as the number of words in the vocabulary.

We assume that features are independent of each other. The so-called independence refers to independence in the statistical sense, that is, the probability of a feature or word occurrence is not related to its neighbors with other words, for example, the occurrence of "ice-cream" and "delicious". Probability has nothing to do with these two words being adjacent. This assumption is exactly what the term naive means in Naive Bayesian classifiers. Another assumption in Naive Bayesian classifiers is that each feature is equally important.

5.4.4 Decision Tree Algorithm

5.4.4.1 Basic Principle of Decision Tree

Decision tree is a common machine learning method, which makes decisions on samples based on a tree structure, just like a natural processing mechanism of the human brain when faced with decision-making problems [100]. Generally speaking, one decision tree contains a root node, several internal nodes and several leaf nodes, where:

- Leaf nodes correspond to decision results, each other node corresponds to an attribute test.
- The sample set contained in each node is divided into sub-nodes according to the result of the attribute test.
- The root node contains the full set of samples.
- The path from the root node to each leaf node corresponds to a decision test sequence.

When constructing a decision tree, the first problem we need to solve is, which feature on the current dataset is determined when dividing the data classification. Features play a decisive role in classifying data. To find the decisive features, and to divide the best results, we must evaluate each feature. After testing, the original dataset was divided into several subsets of data. These subsets of data are distributed across all branches of the first decision point. If the data under a branch is of the same type, the data classification has been correctly classified and no further data set segmentation is required. If the data within the data subsets do not belong to the same type, the process of dividing the data subsets needs to be repeated. The algorithm for how to divide the subset data is the same as the method for dividing the original data set, until all data of the same type are in one data subset. There are various ways to divide the dataset, ID3, C4.5 and CART, etc.

5.4.4.2 Information Gain

In this study, we use the Iterative Dichotomiser 3 (ID3) algorithm to divide the dataset. Assuming that dividing the data according to a certain attribute will produce 2 possible values, we will divide the data into 2 pieces and create 2 different branches. We only select one feature attribute each time we divide the dataset. If there are 20 features in the training set, which feature do we choose as the reference attribute for the division the first time?

The general principle of dividing dataset is to make unordered data more orderly. We can divide the dataset in a few ways, but each method has its own advantages and disadvantages. One way to organize disorganized data is to measure information using information theory, a branch of science that deals with information quantitatively. We can use information theory to quantify the content of measure information before or after dividing the data. The change of information before and after dividing the dataset is called information gain. Knowing how to calculate the information gain, we can calculate the information gain obtained by dividing the dataset for each feature value and obtaining the feature with the highest information gain is the best choice. Before we can evaluate which partition of data is the best, we must learn how to calculate information gain. The measure of aggregate information is called Shannon entropy or simply entropy. Entropy is defined as the expected value of information, indicating the degree of confusion of the information, that is: the more orderly the information, the lower the information entropy. If the transaction to be classified may be divided into multiple classifications, the information of the symbol x_i is defined as

$$l(x_i) = -log_2 p(x_i) \qquad (5-14)$$

where $p_{(xi)}$ is the probability of choosing that class.

To calculate entropy, we need to calculate the expected value of information contained in all possible values of all categories, which is obtained by the following formula:

$$H = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \qquad (5-15)$$

where n is the number of categories. The base of the logarithm can be 2 or 10, but the units of the calculated information entropy are different, the former is nat, and the latter is Hart.

The higher the entropy, the more data is mixed. After obtaining the entropy, we can divide the dataset according to the method of obtaining the maximum information gain. In addition to measuring the information entropy, the decision tree classification algorithm also needs to divide the dataset and measure the entropy of the divided dataset, to judge whether the dataset is correctly divided. We will calculate the information entropy of the result of dividing the dataset for each feature, and then determine which feature is the best way to divide the dataset.

5.4.4.3 Build a Decision Tree Recursively

It works as follows: get the original dataset, and then divide the dataset based on the best feature value, since there may be more than two feature values, there may be more than two branches of the dataset division. After the first division, the data will be passed down to the next node of the tree branch where we can divide the data again. Therefore, we can use the recursive principle to process the dataset. The condition for the end of the recursion is that the program has traversed all attributes that divide the dataset, or all instances under each branch have the same classification. If all instances have the same classification, a leaf node or termination block is obtained. Any data that reaches a leaf node must belong to the classification of the leaf node. For the example of decision tree, see the dataset described in Table 5-3. and decision tree constructed in Figure 5-8. as follows:

Table 5-3. Dataset of marine creatures

No.	*Whether can live without surfacing?*	*Whether have flippers?*	*Whether belongs to fish?*
1	*Yes*	*Yes*	*Yes*
2	*Yes*	*Yes*	*Yes*
3	*Yes*	*No*	*No*
4	*No*	*Yes*	*No*
5	*No*	*Yes*	*No*

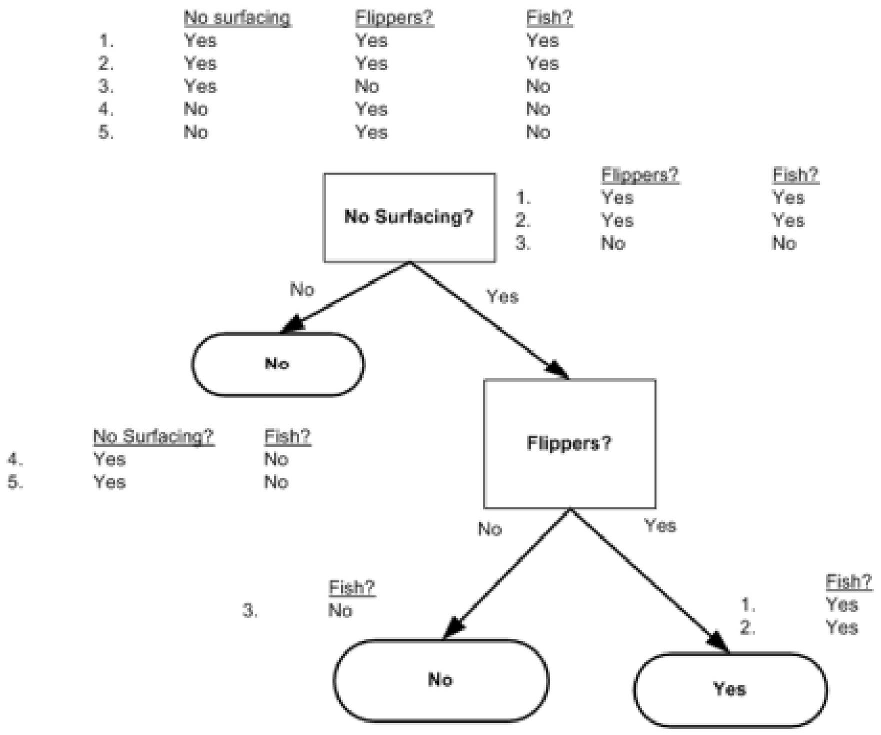


Figure 5-8. Application of Decision Tree on the dataset of marine creatures [107]

5.4.5 Support Vector Machine

5.4.5.1 Basic Idea of Support Vector Machine

In machine learning, support vector machine, often abbreviated as SVM, is a supervised learning model and related to learning algorithms that analyze data in classification and regression analysis [96].

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Although the classifier is a hyperplane in the transformed feature space, it can be nonlinear in the original input space. It is worth noting that the higher dimensional feature space increases the generalization error of the SVM, but the algorithm can still perform well when given enough training examples. Some other common kernel-tricks include Polynomial kernel, Radial basis function kernel and Sigmoid kernel.

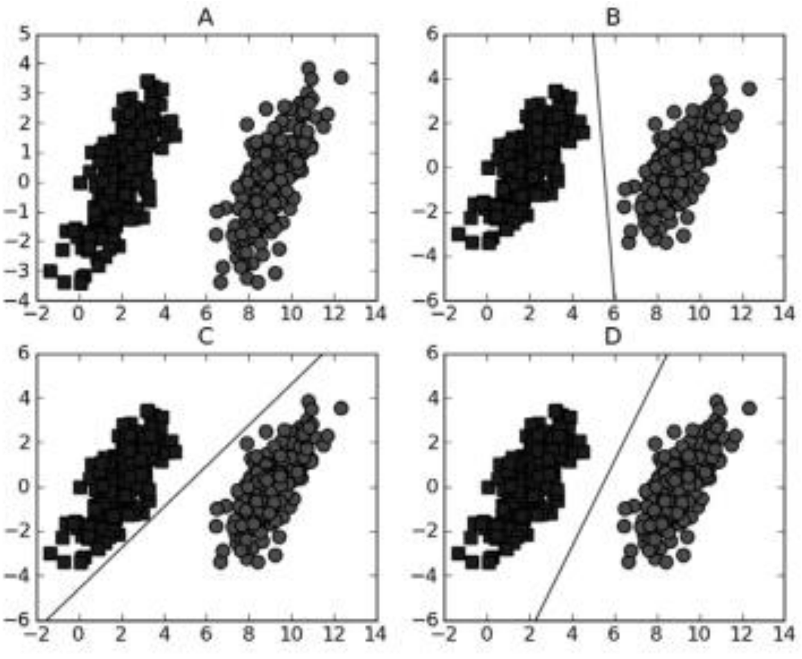


Figure 5-9. A linearly separable data set is given in box A, and separating hyperplanes that can separate the two types of data are given in boxes B, C, and D [107]

Consider the two sets of data in Box A of Figure 5-9. Intuitively, it is easy to draw a straight line on the graph to separate the two sets of data points. In this case, this set of data is called linearly separable data.

The above line separating the datasets is called the separating hyperplane. In the example given above, since the data points are all on the two-dimensional plane, the separating hyperplane is just a straight line at this time. However, if the given dataset is three-dimensional, then what is used to separate the data is a plane. Obviously, the higher dimensional case can be deduced and so on. If the dataset is 1024-dimensional, then a 1023-dimensional XYZ object is needed to separate the data. This object is called the hyperplane, which is the decision boundary for the classification. All data distributed on one side of the hyperplane belongs to one class, and all data distributed on the other side belongs to another class. We want to be able to build a classifier in such a way that the farther a data point is from the decision boundary, the more credible its final prediction will be. Consider the three lines in boxes B, C, and D of Figure 5-9. All of them separate the data, but which one is the best? Should the average distance of the data points to the separating hyperplane be minimized to find the best straight line? If we do so, is the line in boxes B and C in Figure 5-9. really better than the line in box D? The answer is negative. In fact, we want to find the points closest to the separating hyperplane, making sure they are as far away from the separating plane as possible. The distance from the point to the dividing surface is called the margin. We want the margin to be as large as possible because we want the classifier to be as robust as possible if we train the classifier with limited data. The support vectors are those points closest to the separating hyperplane. To this end, trying to maximize the distance from the support vectors to the separating plane, we need to find an optimal solution to this problem.

5.4.5.2 Find the Largest Margin

How to find the best separating line for a dataset? Refer to as Figure 5-10. The form of the separating hyperplane can be written as $w^T\boldsymbol{x} + b$. To calculate the distance from point A to the separating hyperplane, the length of the normal or perpendicular from the point to the separating hyperplane must be given as $|w^T\boldsymbol{A} + b|/||\boldsymbol{w}||$. The constant b here is similar to the intercept w_0 in logistic regression. Here the vector $\boldsymbol{w}$ and the constant b together describe the dividing line or hyperplane of the given data.

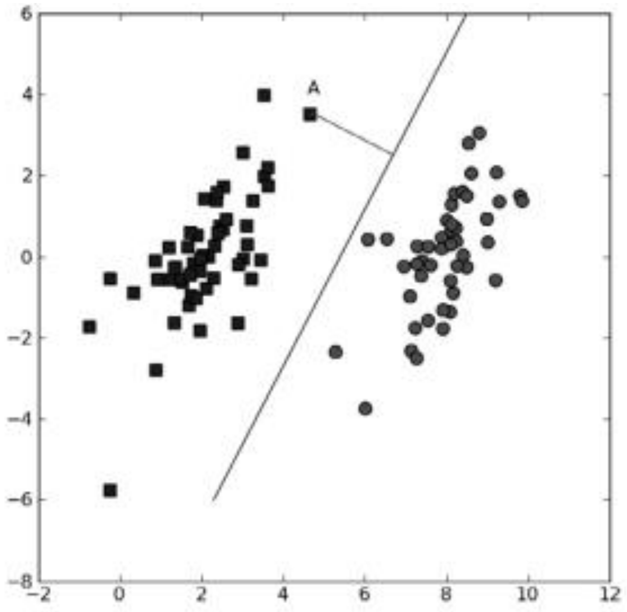


Figure 5-10. The distance from point A to the separating hyperplane is the normal length from the point to the separating hyperplane [107]

5.4.5.3 The Optimization Problem Solved by the Classification Machine

The classifier has been mentioned before, but not how it works. Understanding how it works will help to understand the process of solving a classifier based on an optimization problem. Input data to the classifier will output a class label, which is equivalent to a function like Sigmoid. The following will use a function similar to the unit step function to act on $w^T\boldsymbol{x} + b$ to obtain $f(w^T\boldsymbol{x} + b)$, where $f(u)$ outputs -1 when $u < 0$, and $+1$ otherwise.

Why do the category labels here use -1 and $+1$ instead of 0 and 1? This is because the difference between -1 and $+1$ is only one sign, which is convenient for mathematical processing. We can express the margin or the distance of a data point to the separating hyperplane by a unified formula, without worrying about whether the data belongs to the -1 or $+1$ class.

When calculating the distance from the data point to the separating hyperplane and determining the placement position of the hyperplane, the margin is calculated by $label * (w^T\boldsymbol{x} + b)$(function interval from point to separating hyperplane), then the benefits of -1 and $+1$ categories can be reflected. If the data points are in the positive direction (ie $+1$ class) and are far away from the separating hyperplane, $w^T\boldsymbol{x} + b$ will be a large positive number. In addition, $label * (w^T\boldsymbol{x} + b)$ will also be a large positive number if the data points are in the negative direction (-1 class) and are far away from the separating hyperplane. Since when the class label is -1, $label * (w^T\boldsymbol{x} + b)$ will still be a large positive number.

The goal now is to find w and b in the classifier definition. To do this, we have to find the data points with the smallest margin, which are the aforementioned support vectors. Once we find the data points with the smallest margin, we need to maximize that margin. This can be written as:

$$argmax_{w,b}\left\{min_n\,(label * (\boldsymbol{w}^T\boldsymbol{x} + b)) \cdot \frac{1}{||\boldsymbol{w}||}\right\} \qquad (5-16)$$

Solving the above problem directly is rather difficult, so we convert it into another form that is easier to solve. Let's first examine the part in the curly brackets above. Since optimizing for products is a nasty thing to do, what we do is fix one of the factors and maximize the others. If the $label * (w^T\boldsymbol{x} + b)$ of all support vectors is set to 1, then the final solution can be obtained by finding the maximum value of $||w||^{-1}$. However, not all data points have $label * (w^T\boldsymbol{x} + b)$ equal to 1, only those points closest to the separating hyperplane get a value of 1. The farther the data points are from the hyperplane, the larger the value of $label * (w^T\boldsymbol{x} + b)$.

In the above optimization problem, some constraints are given and then the optimal value is calculated, so the problem is an optimization problem with constraints. The constraint here is that $label * (w^T\boldsymbol{x} + b) \geq 1.0$. For this type of optimization problem, there is a very well-known solution method, the Lagrange multiplier method. By introducing Lagrange multipliers, we can formulate the original problem based on constraints. Since the constraints here are all based on data points, we can write the hyperplane in the form of data points. Therefore, the optimization objective function can finally be written as:

$$max_a\left[\sum_{i=1}^{m}\alpha - \frac{1}{2}\sum_{i,j=1}^{m} label^{(i)} \cdot label^{(j)} \cdot a_i \cdot a_j\langle x^{(i)}, x^{(j)}\rangle\right] \qquad (5-17)$$

While the constraints are:

$$\alpha \geq 0, and \sum_{i-1}^{m} \alpha_i \cdot label^{(i)} = 0 \qquad (5-18)$$

So far, the optimization problem is formulated, but here is an assumption: the data must be 100% linearly separable. However, we know that almost all data is not 100% linearly separable. At this point, we can allow some data points to be separated on the wrong side of the separation plane by introducing the so-called slack variable. In this way, our optimization objective remains the same, but the new constraints now become:

$$C \geq \alpha \geq 0, and \sum_{i-1}^{m} \alpha_i \cdot label^{(i)} = 0 \qquad (5-19)$$

The constant C here is used to control the weights of the objectives of "maximize the margin" and "guarantee that the function interval for most points is less than 1.0". In the implementation code of the optimization algorithm, the constant C is a parameter, so we can get different results by adjusting this parameter. Once all the alphas are found, the separating hyperplane can be expressed by these alphas. This conclusion is straightforward, and the main job in SVM is to solve for these alphas.

5.4.5.4 SMO Efficient Optimization Algorithm

In 1996, John Platt developed a powerful algorithm called SMO for training SVMs. SMO stands for Sequential Minimal Optimization. Platt's SMO algorithm is solved by decomposing a large optimization problem into multiple small optimization problems. These small optimization problems tend to be easy to solve, while solving them sequentially gives exactly the same results as solving them as a whole. On the other hand, the solution time of SMO algorithm is much shorter as well.

The goal of the SMO algorithm is to find a series of alphas and b. Once these alphas are found, it is easy to compute the weight vector w and get the separating hyperplane.

The working principle of the SMO algorithm is that two alphas are selected for optimization in each cycle. Once a suitable pair of alphas is found, increase one while decreasing the other. The so-called "suitable" here means that the two alphas must meet certain conditions. One of the conditions is that the two alphas must be outside the boundary of the interval, and the second condition is that the two alphas have not been performed interval processing or not on the boundaries. The reason for changing both alpha values at the same time is that we have a constraint:

$$\sum \alpha_i \cdot label^{(i)} = 0 \qquad (5-20)$$

The SMO pseudocode is roughly as follows:

Table 5-4. Pseudocode of Sequential Minimal Optimization

```
Create an alpha vector and initialize it to a 0 vector
When the number of iterations is less than the maximum number of iterations (outer loop)
    For each data vector in the dataset (inner loop):
        If the data vector can be optimized
            Randomly choose another data vector
            Simultaneously optimize both vectors
            If neither vector can be optimized, exit the inner loop
    If all vectors are not optimized, increase the number of iterations, and continue with the next loop
```

5.4.6 Ensemble Methods

Several different classification algorithms have been described earlier, each with their own advantages and disadvantages. We can naturally combine different classifiers, and the result of this combination is called an ensemble method or a meta-algorithm. There are many forms when using ensemble methods: it can be an ensemble of different algorithms, or it can be an ensemble of the same algorithm under different settings. It can also be an ensemble after assigning different parts of the dataset to different classifiers. Next, we will introduce two computational methods based on multiple different instances of the same classifier. In these methods, the dataset is also constantly changing and then applied to different classifier instances. Advantages: low generalization error rate, easy to code, can be applied to most classifiers, no parameter adjustment. Disadvantage: Sensitive to outliers.

5.4.6.1 Bagging

Bootstrap aggregating, also known as bagging, is a technique for obtaining S new datasets after selecting S times from the original dataset. The new dataset is the same size as the original dataset. Each dataset is obtained by randomly selecting a sample from the original dataset for replacement. Replacement here means that the same sample can be selected multiple times. This property allows duplicate values in the new dataset, while some values from the original dataset are no longer present in the new set. After the S datasets are established, a learning algorithm is applied to each dataset to obtain S classifiers. When we want to classify new data, we can apply these S classifiers for classification. At the same time, the category with the most votes from the classifier is selected as the final classification result. Of course, there are some more advanced bagging methods, such as random forest.

5.4.6.2 Random Forest Algorithm

Random forest refers to a classifier that uses multiple trees to train and predict samples. The construction of random forest has two aspects: randomization of data and randomization of features to be selected. These two aspects allow the decision trees in random forests to be different from each other, thereby increasing the diversity of the system and thus the classification performance [99].

5.4.6.3 Workflow of Random Forest Algorithm

In general, Random Forest algorithm consists of two main tasks, including randomization of data and randomization of candidate features, as demonstrated below.

Randomization of data: Make decision trees in random forests more generalized and suitable for more scenarios.
1. Adopt a sampling method with replacement to construct subsets to ensure that the order of magnitude between different subsets is the same, whereas elements between different subsets/same subset can be repeated.
2. Use the subsets to build several sub-trees, put these data into each subtree, and each subtree outputs a result.
3. Then count the voting results of the subtrees and get the final classification results of random forest.
4. As shown in Figure 5-11., assuming that there are 3 subtrees in the random forest. If the classification results of 2 subtrees are class A, while the result of 1 subtree is class B, then the final classification result of the random forest is class A.

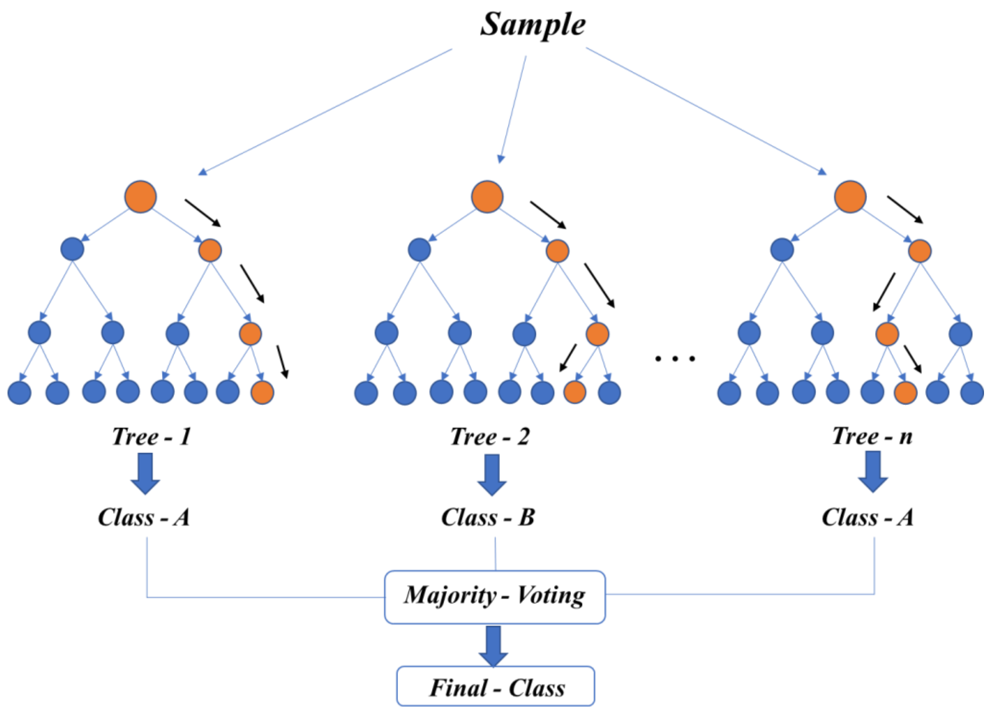


Figure 5-11. Majority voting of Random Forest

Randomization of candidate features:

1. The sub-tree randomly selects a certain feature from all the candidate features.
2. Select the best feature among the selected features.

In Figure 5-12., the blue squares represent all the features that can be selected, that is, the current candidate features; the yellow squares are the splitting features. On the left is the feature selection process of a decision tree. By selecting the optimal splitting feature from the features to be selected. As for the methods of selecting the optimal feature, please refer to ID3 algorithm as mentioned when illustrating Decision Tree. On the right is the feature selection process for a subtree in a random forest.

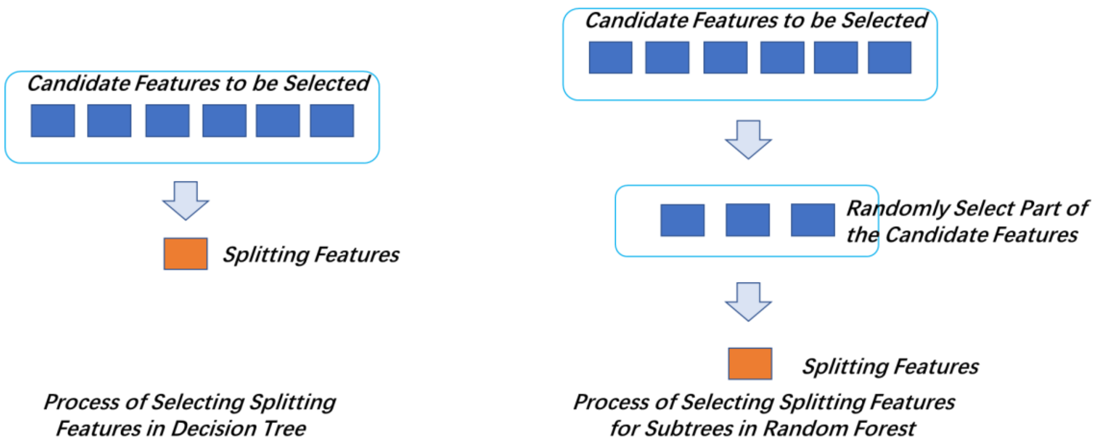

Candidate Features to be Selected
Splitting Features
Process of Selecting Splitting Features in Decision Tree
Candidate Features to be Selected
Randomly Select Part of the Candidate Features
Splitting Features
Process of Selecting Splitting Features for Subtrees in Random Forest

Figure 5-12. Process of selecting splitting features in Random Forest

5.4.6.4 Boosting Method

Boosting is a technique very similar to bagging. Whether in boosting or bagging, the types of multiple classifiers used are the same. But in the former, different classifiers are obtained by serial training, and each new classifier is trained according to the performance of the trained classifier. Boosting is to obtain new classifiers by focusing on those data that have been misclassified by existing classifiers. Since the result of boosting classification is based on the weighted sum of all classifiers, boosting is not the same as bagging. The weights of the classifiers in bagging are equal, while the weights of the classifiers in boosting are not equal, and each weight represents the success of its corresponding classifier in the previous iteration. There are several versions of the boosting method, one of the most popular algorithms is adaptive boosting (AdaBoost).

The operation process of AdaBoost is as follows: each sample in the training data is given a weight, and these weights form a vector $\boldsymbol{D}$. Initially, these weights are initialized to equal values. First train a weak classifier on the training data and calculate the error rate of the classifier, and then train the weak classifier again on the same dataset. In the second training of the classifier, the weight of each sample will be re-adjusted, in which the weight of the first-time paired sample will be reduced, and the weight of the first wrong-classified sample will be increased. In order to get the final classification result from all the weak classifiers, AdaBoost assigns a weight value alpha to each classifier, and these alpha values are calculated based on the error rate of each weak classifier. Among them, the error rate ε is defined as:

$$\varepsilon = \frac{Number\ of\ samples\ that\ were\ not\ correctly\ classified}{Total\ number\ of\ samples} \qquad (5-21)$$

The formula for calculating alpha is as follows:

$$\alpha = \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\epsilon}\right) \qquad (5-22)$$

The example of AdaBoost classifier is shown in Figure 5-13., as follows:

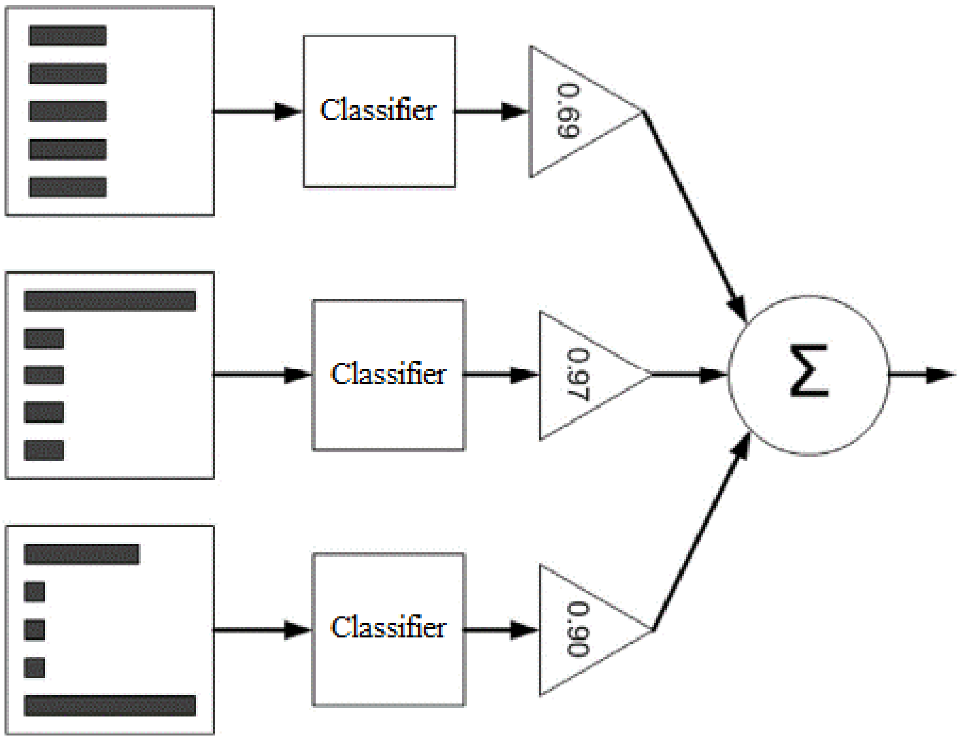


Figure 5-13. Basic structure of AdaBoost classifier

On the left of Figure 5-13. is the dataset, where the different widths of the histogram represent different weights on each example. After going through a classifier, the weighted predictions are weighted by the alpha value in the triangle. The weighted output of each triangle will be summed in the circle to get the final output.

After the alpha value is calculated, the weight vector $\boldsymbol{D}$ can be updated so that the weights of correctly classified samples are reduced, and the weights of misclassified samples are increased. $\boldsymbol{D}$ is calculated as follows.

If a sample is correctly classified, the weight of the sample is changed to:

$$\boldsymbol{D}_i^{((t+1))} = \frac{\boldsymbol{D}_i^{(t)} e^{-\alpha}}{Sum(\boldsymbol{D})} \qquad (5-23)$$

And if a sample is misclassified, the weight of the sample is changed to:

$$\boldsymbol{D}_i^{((t+1))} = \frac{\boldsymbol{D}_i^{(t)} e^{\alpha}}{Sum(\boldsymbol{D})} \qquad (5-24)$$

After calculating $\boldsymbol{D}$, AdaBoost starts to enter the next iteration. The AdaBoost algorithm will continuously repeat the process of training and adjusting the weights until the training error rate is 0 or less than a certain threshold. In this study, we mainly adopt the boosting method to serve as a baseline in the validation experiments of the proposed inclination measurement system.

5.5 Implementation and Evaluation Experiments

In this section, we would like to demonstrate the realization of the proposed method. Furthermore, evaluation experiments and comparison works will also be illustrated in this section. Specifically, the prototype work/proposed neural network is implemented by using the IDE mentioned in Chapter 2 with Python programming language and some necessary libraries including Tensorflow and Keras, etc. And the implementations of machine learning algorithms are based on scikit-learn library. Moreover, configurations of the hardware employed in evaluation experiments is as the following: CPU(i7-10700K), GPU(GTX1080Ti), and memory(64GB). As for the image dataset, we mainly utilize the Assembly Platform Dataset mentioned in Chapter 2.

When making the implementation, considering the robustness of image feature, our group finally decided to adopt a relatively heavy feature - SIFT to extract image features. This is mainly because the strict requirements of algorithms applied in the project of inclination measurement. To balance the performance and computation power requirements of the inclination system, we employ K-Means to reduce the redundant features as well as to make feature engineering. Experiments show that the proposed classification-based method can effectively reduce the computational burden of the system, while maintain good measurement accuracy and stability at the same time. The workflow of our solution is depicted in Figure 5-14.

As for the feature clustering process, we adjusted the parameters of K-Means to make it obtain the best performance when dealing with the task of inclination measurement. About the termination condition to stop clustering, whenever mean square errors (MSE) is becoming minimum, the algorithm will stop. With a tremendous time of fine tuning, our group achieved the best clustering performance under the condition that the clustering center K equals to 70.

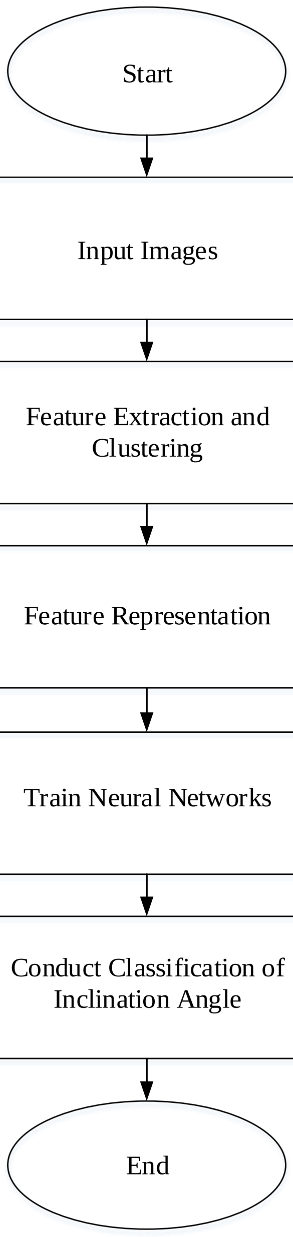

Start
Input Images
Feature Extraction and Clustering
Feature Representation
Train Neural Networks
Conduct Classification of Inclination Angle
End

Figure 5-14. Workflow of the classification-based inclination measurement

With the help of clustering algorithm, the computation burden of feature representation regarding our dataset decreases greatly, which benefits for our systems with constrained computing resources. About representation of image features, our group utilize feature vectors obtained from clustering to represent the original image data. In specific, through generating frequency histograms of image features, our group realized feature representation of image data. The workflow of this process is shown in Figure 5-15. After got the feature vectors of the images from Assembly Platform Dataset, we use the feature vectors to train the proposed neural network.

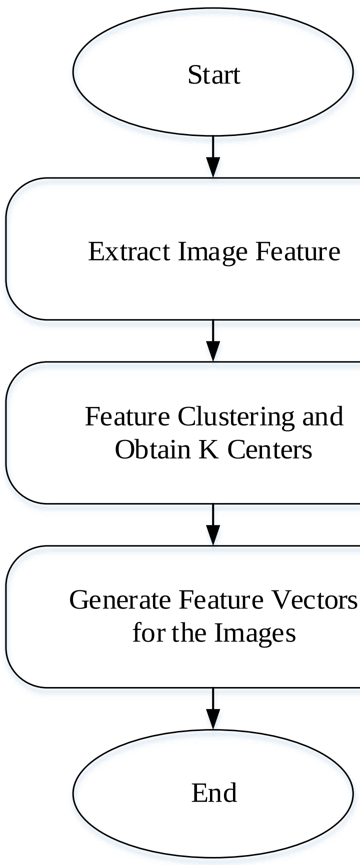

Start
Extract Image Feature
Feature Clustering and Obtain K Centers
Generate Feature Vectors for the Images
End

Figure 5-15. Workflow of the feature representation of image data

The evaluation experiments take several aspects into accounts, including time consumption, computational complexity, time cost of the prediction process, and classification accuracy. To verify the ability of reducing computation complexity, a comparative experiment between the traditional mechanism [87][88][89] and our proposal has been carried out. The comparison results about computational complexity are summarized in Figure 5-16.

As for the classification performance, our proposed mechanism is superior to other algorithms without introducing additional computational burden. Comparison results about the classification accuracy are shown in Figure 5-17. Besides, a summary of the prediction time cost among different algorithms is depicted in Figure 5-18.

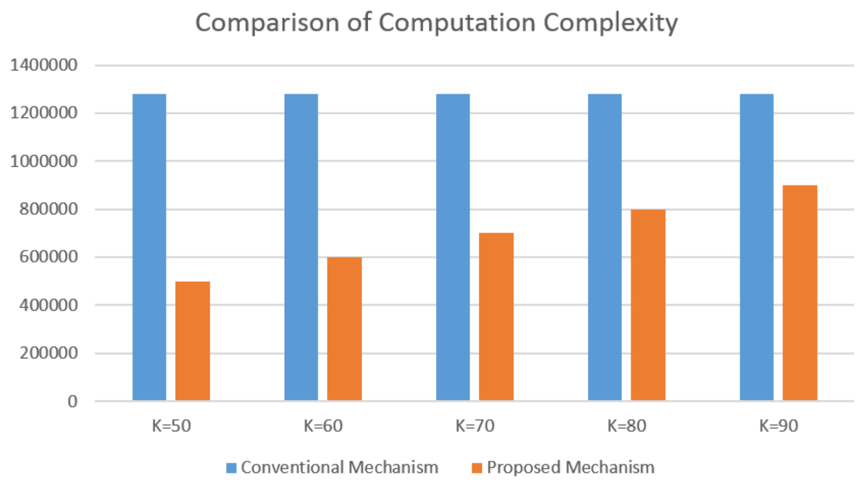


Figure 5-16. Comparison results of the computation complexity

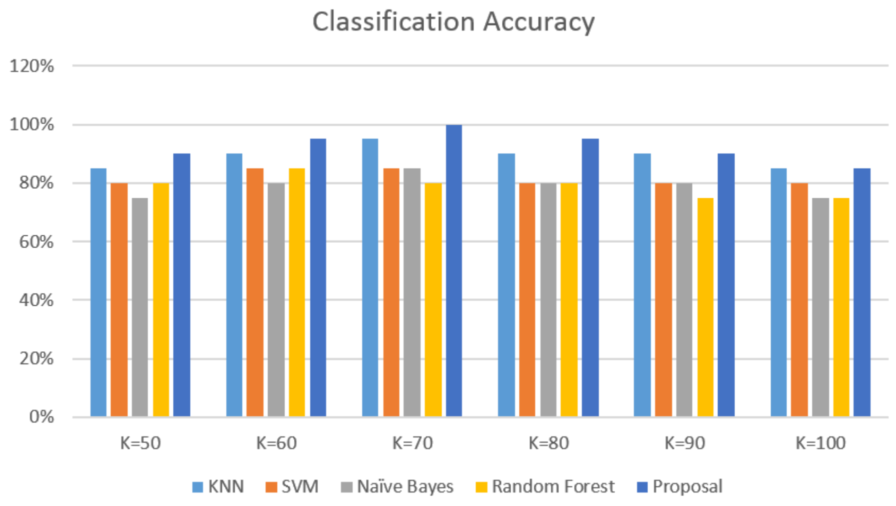


Figure 5-17. Evaluation results about classification accuracy

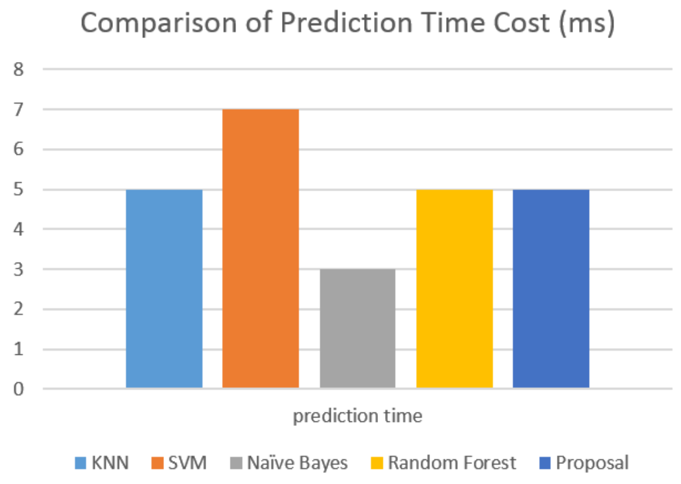


Figure 5-18. Evaluation results about time cost of the algorithms

With a comprehensive consideration about the algorithm performance, we could draw a conclusion that, our proposed algorithm which based on feature clustering combined with the proposed neural network achieves the best overall performance when dealing with inclination measurement task. The advantages involve but not limited to low computational complexity and good performance.

Experimental result shown in Figure 5-17. certificate the effectiveness of our proposed neural working structure, which is superior to the other algorithms and achieves 98% prediction accuracy under the condition of $K = 70$.

Before proposing our own algorithms, some prevalent networks have been implemented to deal with the inclination classification task. In the evaluation process, we implemented MobileNetV2 [102], Xception [103], ResNet [104, 106] and LeNet [90] to classify the inclination angles. These popular convolutional neural networks are implemented base on TensorFlow and Keras libraries. Furthermore, Assembly Platform Dataset was employed to train and test the model. Three-fold cross-validation is also applied to improve the reliability of evaluation results. The classification performance among different networks is summarized from Figure 5-19. to Figure 5-24.

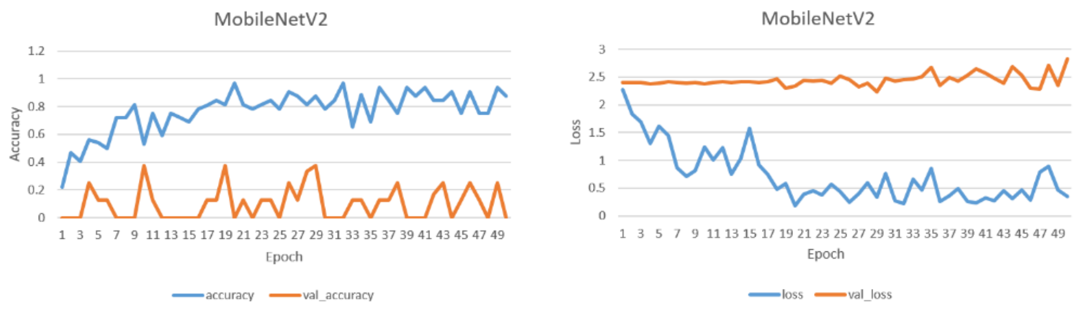


Figure 5-19. Training and validation performance of MobileNetV2

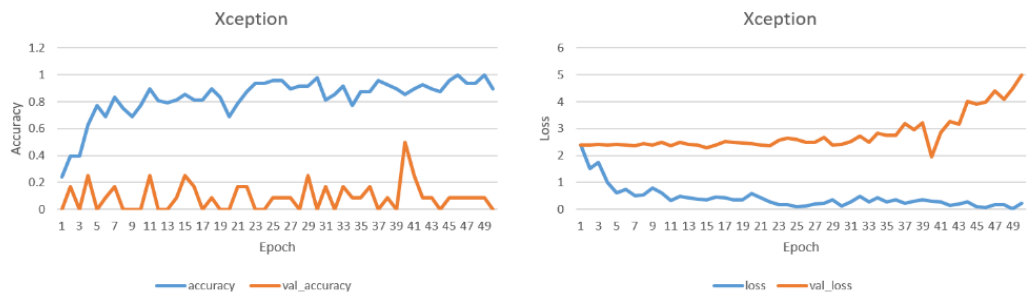


Figure 5-20. Training and validation performance of Xception

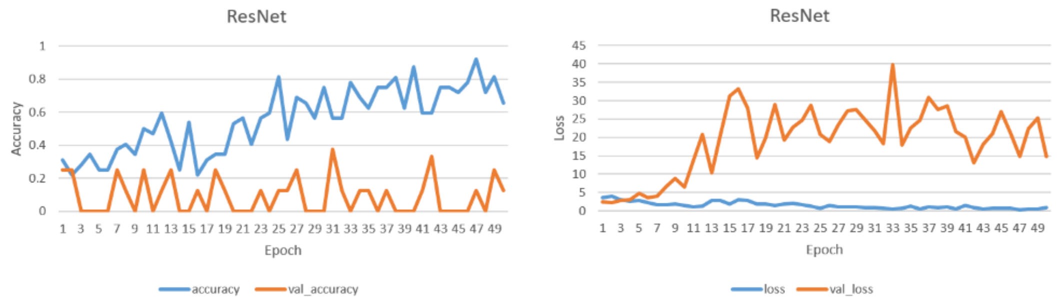


Figure 5-21. Training and validation performance of ResNet

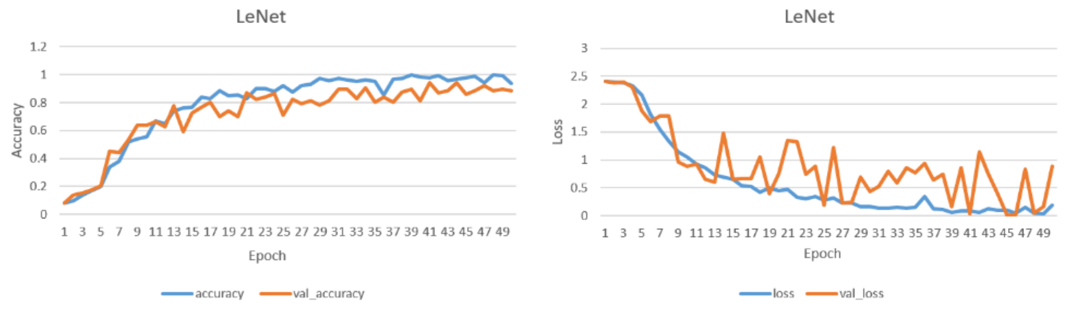


Figure 5-22. Training and validation performance of LeNet

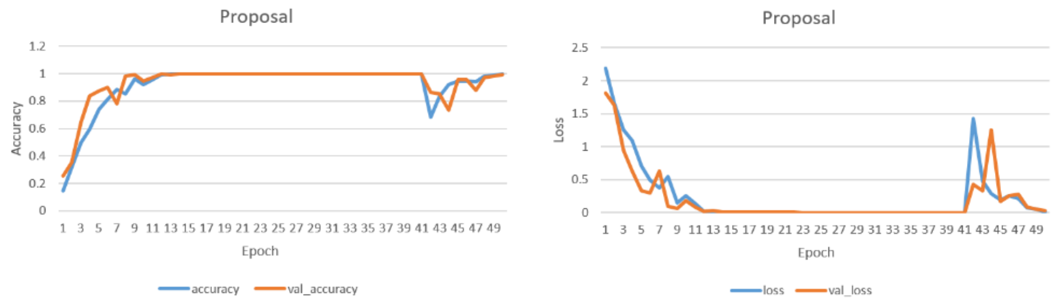


Figure 5-23. Training and validation performance of the proposal (50 epochs)

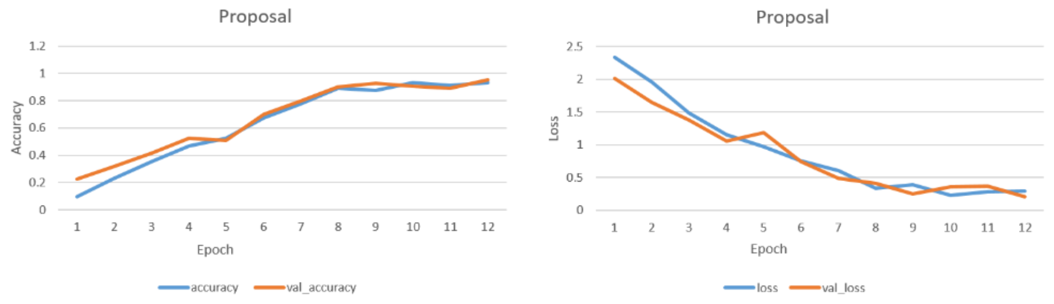


Figure 5-24. Training and validation performance of proposal (12 epochs)

From the classification performance of MobileNetV2 as shown in Figure 5-19., we found that MobileNetV2 failed to converge well within 50 training epochs, since the validation loss gradually increased as the training process continued.

Performance of Xception in regard of inclination classification is summarized in Figure 5-20. Thus, we can conclude that Xception failed to perform well in the inclination classification task. After training for 39 epochs, the validation loss increased which proved the overfitting of the networks. On the other hand, the overall validation accuracy remained under 25% through the training process, which indicates the disability of dealing with inclination classification.

The classification performance of ResNet is portrayed in Figure 5-21. It seems that ResNet failed to converge in the training process, and the overall performance keeps at a poor state, which proved the dissatisfaction towards dealing with the industrial task in this work.

Experimental performance regarding LeNet is depicted in Figure 5-22. It is easy to find that LeNet reached 94.2% about validation accuracy. However, to a certain extent, the validation loss of LeNet increases sharply, which is a disadvantage of LeNet to solve the image classification problem.

Regarding the proposed neural network in this research, Figure 5-23. summarizes its performance. Obviously, the proposed neural network converges well in the training process. In specific, the proposed method converges within 12 training epochs. For this reason, we depicted the graph of training process from 1 to 12 epochs in Figure 5-24., which showed more details of the first 12 training epochs of the proposed neural network. In epoch 12, not only the validation accuracy rate reached 100%, but the validation loss also got the minimum value. In sum, the proposed neural network is superior to other neural networks. The experimental results prove that the proposed neural network is suitable for solving the problem of inclination angle classification.

As for the dissatisfaction of MobileNetV2, Xception and ResNet when tackling the classification task of this project, it is mainly because the high complexity of the network structures. On the other hand, the input of the convolutional neural networks are images of our dataset. In other words, the features of these images in the dataset are not clustered in advance. Inputs of our proposed neural network are feature vectors obtained after the feature clustering process. In addition, proposed network structure is associated with the tuned K value, which will also affect the classification performance of image dataset established in this work. To sum up, the above-mentioned reasons lead to a good performance of the proposed neural network in the classification of inclination angles.

5.6 Chapter Summary

To some degree, the inclination measurement of the industrial assembly platforms remains a challenge. This is mainly because of the relatively high computation cost of the algorithms. Therefore, in our research, we proposed a method to tackle it. In specific, the idea of image classification is adopted to deal with inclination measurement by classifying the images of the assembly platforms. First of all, we extract the image features from the input images. Then, clustering-based method is applied to construct feature vectors for the images of the assembly platforms. Secondly, those feature vectors will be utilized to train the neural networks. In this way, inclination measurement is transformed to an image classification task. The output of the neural network gives the exact category to which an input image belongs. An improved clustering algorithm is realized, and the tuned parameter K was finally found after many iterations of testing experiment. In contrast to those traditional methods, the proposed method achieves the best performance while reducing computational complexity by 45.31%. In addition, validation accuracy of the presented neural networks with tuned parameters reaches over 95%, which outperforms other baseline models. Evaluation results certificate the effectiveness of our method for inclination measurement. As for future works, we will make effort to improve the performance of the proposed algorithm. On the other hand, boosting methods such as AdaBoost will also be considered to help enhance the performance of the inclination measurement.

Chapter 6 Conclusions and Discussions

This study focuses on the problem of the inclination angle measurement for the bearing assembly platforms in the manufacturing industry. The main research contents are as follows: (1) Through the reading of the literatures, we summarized the current research status and future development trend of computer vision and machine learning technologies. (2) The actual demand of the measurement system for the bearing assembly enterprises was investigated. (3) Considering the requirements of the research project and the existing experimental equipment, we designed a computer vision-based inclination angle measurement system. (4) In terms of the algorithms to measure the inclination angles, we proposed two algorithms which are based on SLAM technology and artificial intelligence, respectively. (5) Finally, we evaluated the effectiveness and reliability of the proposed system, as well as its advantages over other algorithms through a large number of validation experiments.

Due to the limited time and experiment conditions, there are still some shortcomings in the current research: (1) In this study, the testing scenarios have good lighting, whereas the performance of the algorithms might be influenced by the variations of the lighting conditions. Thus, in the future research, we will conduct in-depth studies under different lighting conditions. (2) The current experimental scenarios are relatively simple. Considering the real scenarios of the manufacturing field might be much more complex, we will conduct additional experiments in the scenarios with relatively complex backgrounds, and further adjust and improve the performance of the proposed algorithms for inclination angle measurement.

Acknowledgment

As time passed by inadvertently, it would also provide people with growth, physically and mentally. Five years have also shaped me a relative more mature character. Recalling that when I first arrived at Ritsumeikan University, I was mentally immature, so I felt a lot of strangeness and anxiety when facing a new environment. For now, I am getting more interested in my profession and becoming more independent than before. In the past three years, not only the vision of the professional field has been broadened, but also the vision of other fields has been improved. Every growth in life, either through many hardships, or through the guidance of the teachers. In either case, there are people who accompany and guide me. On the occasion of graduation, I would like to express my sincere gratitude to those who have guided me, helped me, and encouraged me.

First of all, I would like to express my special thanks to my supervisor, Professor Tomiyama Hiroyuki. In the past five years, Tomiyama sensei has given me a lot of guidance in my studies. In my life, his generous and kind temperament, attitude towards life, and wisdom of tackling problems have deeply influenced me. Treating people with generosity and being conscientious in works have all guided me to be a down-to-earth and rigorous person who pays attention to reasonability in life and work. Thanks for his gracious guidance, consistent encouragement, and constant support in my career plan. It would be my great honor to be supervised by him. Here, I would like to express my most sincere thanks to him.

I would also like to thank Professor Lin Meng and other professors in the Department of Electronic Systems. In the five years of study and life, some of the teachers' academic philosophy and selfless work spirit have left a deep impression on me. Not only for the development of the laboratory, but also for providing a good learning environment for the students in the laboratory, but also for the students' own development, providing a lot of constructive suggestions, which I admire very much. Thus, I would like to thank the professors for the efforts made for the students.

I would like to thank my classmates in Tomiyama laboratory. The days of living and studying with you are enjoyable, unforgettable and would be my best memories! I really appreciate their help in study and life, especially when the time that I just became a newcomer to Japan. I felt very lucky to get along with you all the way. Definitely I will always remember the five happy years of studying and growing with you.

I would like to thank my parents and family for their constant companionship, encouragement, concern, and support. They are always my strongest backing.

I would like to thank my alma mater Ritsumeikan University for providing us with an excellent learning environment and learning resources. In the minds of us, Ritsumeikan University has always been the best university.

Finally, thanks to the experts and scholars who reviewed the thesis, participated in the defense of the thesis during their busy schedules and provided valuable suggestions.

References

[1] L. Haomin, Z. Guofeng, and B. Hujun, “A survey of monocular simultaneous localization and mapping,” *Journal of Computer-Aided Design and Compute Graphics*, vol. 28, no. 6, pp. 855–868, 2016.

[2] A. Davison, I. Reid, N. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[4] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[6] T. Barfoot, “State estimation for robotics: A matrix lie group approach,” 2016.

[7] A. Pretto, E. Menegatti, and E. Pagello, “Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3289–96, IEEE, 2011.

[8] B. Rueckauer, and T. Delbruck, “Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor,” *Frontiers in Neuroscience*, vol. 10, 2016.

[9] C. Cesar, L. Carlone, et al. “Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309-1332, 2016.

[10] P. Newman and K. Ho, “SLAM-loop closing with visually salient features,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 635–642, IEEE, 2005.

[11] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” *Autonomous Robot Vehicles*, pp. 167–193, Springer, 1990.

[12] P. Beeson, J. Modayil, and B. Kuipers, “Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy,” *International Journal of Robotics Research*, vol. 29, no. 4, pp. 428–459, 2010.

[13] H. Strasdat, J. M. Montiel, and A. J. Davison, “Visual SLAM: Why filter?,” *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[14] M. Liang, H. Min, and R. Luo, “Graph-based SLAM: A survey,” *ROBOT*, vol. 35, no. 4, pp. 500–512, 2013.

[15] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[16] J. Boal, Á. Sánchez-Miralles, and Á. Arranz, “Topological simultaneous

localization and mapping: a survey," *Robotica*, vol. 32, pp. 803–821, 2014.

[17] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[18] Z. Chen, J. Samarabandu, and R. Rodrigo, "Recent advances in simultaneous localization and map-building using computer vision," *Advanced Robotics*, vol. 21, no. 3-4, pp. 233–265, 2007.

[19] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM Review*, vol. 6, no. 4, pp. 422–430, 1964.

[20] V. S. Varadarajan, *Lie groups, Lie algebras, and Their Representations*, vol. 102. Springer Science & Business Media, 2013.

[21] H. Strasdat, *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College London, 2012.

[22] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," *International Conference on Computer Vision (ICCV)*, pp. 72–79, IEEE, 2009.

[23] J. Nocedal, and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[24] M. I. Lourakis, and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[25] G. Sibley, "Relative bundle adjustment," *Department of Engineering Science, Oxford University, Tech. Rep*, vol. 2307, no. 09, 2009.

[26] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment: a modern synthesis," *Vision Algorithms: Theory and Practice*, pp. 298–372, Springer, 2000.

[27] Ceres solver. http://ceres-solver.org

[28] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: a general framework for graph optimization," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3607–3613, IEEE, 2011.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[30] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *European Conference on Computer Vision*, pp. 404–417, Springer, 2006.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to sift or surf," *International Conference on Computer Vision (ICCV)*, pp. 2564–2571, IEEE, 2011.

[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *European Conference on Computer Vision*, pp. 430–443, Springer, 2006.

[33] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *European Conference on Computer Vision*, pp. 778–792, Springer, 2010.

[34] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image*

Understanding, vol. 73, no. 2, pp. 291–307, 1999.

[35] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *VISAPP*, vol. 1, no. 2, pp. 331–340, 2009.

[36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d SLAM systems," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 573–580, IEEE, 2012.

[37] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[38] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp.33-135, 1981.

[39] H. Li and R. Hartley, "Five-point motion estimation made easy," *18th International Conference on Pattern Recognition (ICPR)*, vol. 1, pp. 630–633, IEEE, 2006.

[40] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[41] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 03, pp. 485–508, 1988.

[42] Z. Zhang, and A. R. Hanson, "3D reconstruction based on homography mapping," *ARPA Image Understanding Workshop*, pp. 1007–1012, 1996.

[43] E. Malis, and M. Vargas, *Deeper Understanding of the Homography Decomposition for Vision -based Control*. PhD thesis, INRIA, 2007.

[44] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 930–943, Aug 2003.

[45] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate O(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2008.

[46] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer, "Exhaustive linearization for robust camera pose and focal length estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2387–2400, 2013.

[47] L. Chen, C. W. Armstrong, and D. D. Raftopoulos, "An investigation on the accuracy of three-dimensional space reconstruction using the direct linear transformation technique," *Journal of Biomechanics*, vol. 27, no. 4, pp. 493–500, 1994.

[48] P3p. http://iplimage.com/blog/p3p-perspective-point-overview/

[49] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 698–700, 1987.

[50] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration

algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[51] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3D objects," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 27–52, 1986.

[52] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSAA*, vol. 4, no. 4, pp. 629–642, 1987.

[53] G. C. Sharp, S. W. Lee, and D. K. Wehe, "ICP registration using invariant features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 90–102, 2002.

[54] G. Silveira, E. Malis, and P. Rives, "An efficient direct approach to visual SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 969–979, 2008.

[55] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15–22, 2014.

[56] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[57] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *arXiv preprint arXiv:1607.02565*, 2016.

[58] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," *IEEE International Conference on Computer Vision*, pp. 1449–1456, 2013.

[59] V. Usenko, J. Engel, J. Stueckler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2016.

[60] V. Sujan and S. Dubowsky, "Efficient information-based visual robotic mapping in unstructured environments," *International Journal of Robotics Research*, vol. 24, no. 4, pp. 275–293, 2005.

[61] F. Janabi-Sharifi and M. Marey, "A kalman-filter-based method for pose estimation in visual servoing," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 939–947, 2010.

[62] S. Li and P. Ni, "Square-root unscented kalman filter based simultaneous localization and mapping," *IEEE International Conference on Information and Automation (ICIA)*, pp. 2384–2388, 2010.

[63] R. Sim, P. Elinas, and J. Little, "A study of the rao-blackwellised particle filter for efficient and accurate vision-based slam," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 303–318, 2007.

[64] J. S. Lee, S. Y. Nam, and W. K. Chung, "Robust RBPF-SLAM for indoor mobile robots using sonar sensors in non-static environments," *Advanced Robotics*, vol. 25, no. 9-10, pp. 1227–1248, 2011.

[65] A. Gil, O. Reinoso, M. Ballesta, and M. Julia, "Multi-robot visual SLAM using a Rao-Blackwellized Particle Filter," *Robotics and Autonomous Systems*, vol. 58,

no. 1, pp. 68–80, 2010.

[66] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, 2010.

[67] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: Ekf slam in O(n)," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1107–1120, 2008.

[68] O. G. Grasa, J. Civera, and J. Montiel, "EKF monocular SLAM with relocalization for laparoscopic sequences," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4816–4821, IEEE, 2011.

[69] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[70] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, "Incremental block cholesky factorization for nonlinear least squares in robotics," *Robotics: Science and Systems*, 2013.

[71] R. Mur-Artal, J. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular slam system," *arXiv preprint arXiv:1502.00956*, 2015.

[72] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[73] Bundle adjustment in the large. http://grail.cs.washington.edu/projects/bal/

[74] J. Sherman, and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

[75] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimization for constant time visual SLAM," *IEEE International Conference on Computer Vision (ICCV)*, pp. 2352–2359, 2011.

[76] B. Steinman, G. Philip, *Foundations of Binocular Vision: A Clinical Perspective*. McGraw-Hill Medical, 2000.

[77] D. Lee, and H. Myung, "Solution to the slam problem in low dynamic environments using a pose graph and an RGB-D sensor," *Sensors*, vol. 14, no. 7, pp. 12467–12496, 2014.

[78] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph slam," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[79] B. Lucas, and T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*. vol. 81, pp. 674-679, 1981.

[80] D. Koller, and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[81] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[82] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes Tree," *The International Journal of Robotics Research*, pp. 216-235, 2011.

[83] L. Basaca-Preciado, “Optical 3D laser measurement system for navigation of autonomous mobile robot,” *Optics and Lasers in Engineering*, vol. 54, pp. 159–169, 2014.

[84] C. Pham, and J. Jeon, “Domain transformation-based efficient cost aggregation for local stereo matching,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 7, pp. 1119-1130, 2012.

[85] D. M. Rosen, M. Kaess, and J. J. Leonard, “Rise: An incremental trust-region method for robust online sparse least-squares estimation,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1091–1108, 2014.

[86] D. Fleet, and Y. Weiss, *Optical Flow Estimation*. Handbook of Mathematical Models in Computer Vision, pp. 237-257. Boston, MA, Springer, 2006.

[87] C. Szegedy, and W. Liu, “Going deeper with convolutions,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015.

[88] K. Simonyan, and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[89] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[90] Y. LeCun, L. Bottou, and Y. Bengio, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[91] Z. Meng, L. Meng, and H. Tomiyama, “Motion estimation-based inclination measurement of the assembly platform,” *International Journal of Advanced Mechatronic Systems*, vol. 8, no. 4, pp. 155-165, 2021.

[92] Z. Meng, L. Meng, and H. Tomiyama, “Feature classification-based inclination measurement for industrial assembly platform,” *Wireless Communications and Mobile Computing*, vol. 2022, article 5911031, 2022.

[93] C. M. Bishop, and N. M. Nasrabadi, *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

[94] K. P. Murphy, *Machine learning: A Probabilistic Perspective*. MIT Press, 2012.

[95] K. Krishna, M. N. Murty, “Genetic k-means algorithm,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9. no.3, pp. 33-439, 1999.

[96] C. J. Burges, “A tutorial on support vector machines,” *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[97] T. Cover, and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, 1967.

[98] I. Rish, “An empirical study of the naive Bayes classifier,” *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, pp. 41-46, 2001.

[99] A. Liaw, and M. Wiener, “Classification and regression by randomForest,” *R News*, vol. 2, no. 3, pp. 18-22, 2002.

[100] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.

[101] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” *Neural Networks for Perception*, pp. 65-93, Academic Press, 1992.

[102] A. G. Howard, M. Zhu, and B. Chen, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv: 1704.04861*, 2017.

[103] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251-1258, 2017.

[104] K. He, X. Zhang, and S. Ren, “Deep residual learning for image recognition,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[105] C. Szegedy, and S. Ioffe, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *AAAI Conference on Artificial Intelligence*, 2017.

[106] G. Huang, and Z. Liu, “Densely connected convolutional networks,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700-4708, 2017.

[107] S. Haykin, *Neural Networks and Learning Machines*. Pearson Education India, 2009.

[108] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

Publications

Journals

[1] Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "Feature Classification-Based Inclination Measurement for Industrial Assembly Platform," Wireless Communications and Mobile Computing, Hindawi, vol. 2022, article 5911031, 2022.

[2] Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "Motion Estimation Based Inclination Measurement of the Assembly Platform," International Journal of Advanced Mechatronic Systems, Inderscience Publishers, vol. 8, no. 4, pp. 155-165, 2021.

[3] Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "Pneumonia Diagnosis on Chest X-Rays with Machine Learning," Procedia Computer Science, Elsevier, vol. 187, pp. 42-51, 2021.

[4] Zhichen Wang, Zelin Meng, Kenshi Saho, Kazuki Uemura, Naoto Nojiri, and Lin Meng, "Deep Learning-based Elderly Gender Classification using Doppler Radar," Personal and Ubiquitous Computing, 2021.

[5] Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "Camera Motion Estimation Algorithm for IoT Devices based on Optimized Feature Tracking Method," Procedia Computer Science, Elsevier, vol. 174, pp. 22-26, 2020.

[6] Naoto Nojiri, Zelin Meng, Kenshi Saho, and Lin Meng, "Apathy Classification based on Doppler Radar Image for the Elderly Person," Frontiers in Bioengineering and Biotechnology, vol. 8, 2020.

International Conferences

[1] Zelin Meng, Lin Meng, Kenshi Saho, Xiangbo Kong, and Hiroyuki Tomiyama, "Frailty Classification Based on Artificial Intelligence," In Proceedings of International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT), pp. 99-107, Online, August 2021.

[2] Zelin Meng, Zhiyu Zhang, Zhihong Man, Lin Meng, and Hiroyuki Tomiyama, "Rubbing Character Recognition with Machine Learning," In Proceedings of International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 290-295, Hanoi, Vietnam, December 2020.

[3] Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "A Case Study on Rubbing Character Recognition Based on Deep Learning," In Proceedings of International SoC Design Conference (ISOCC), pp. 318-319, Yeosu, Korea, October 2020.

[4] Zelin Meng, Xiangbo Kong, Lin Meng, and Hiroyuki Tomiyama, "Lucas-Kanade Optical Flow Based Camera Motion Estimation Approach," In Proceedings of International SoC Design Conference (ISOCC), pp. 77-78, Jeju, Korea, October 2019.

[5] Zelin Meng, Xiangbo Kong, Lin Meng, and Hiroyuki Tomiyama, "Camera Motion Estimation and Optimization Approach," In Proceedings of International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 212-217, Shiga, August 2019.

[6] Xiangbo Kong, Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "A Neck-floor Distance Analysis Based Fall Detection System Using Deep Camera," In Proceedings of International Conference on Artificial Intelligence and Data Engineering (AIDE), Springer AISC 1133, pp. 1113-1120, Nitte, Karnataka, India, May 2019.

[7] Zelin Meng, Xiangbo Kong, Lin Meng, and Hiroyuki Tomiyama, "Stereo Vision based Depth Estimation," In Proceedings of International Conference on Artificial Intelligence and Data Engineering (AIDE), Springer AISC 1133, pp. 1209-1216, Nitte, Karnataka, India, May 2019.

[8] Xiangbo Kong, Zelin Meng, Naoto Nojiri, Yuji Iwahori, Lin Meng, and Hiroyuki Tomiyama, "A HOG-SVM Based Fall Detection IoT System for Elderly Persons Using Deep Sensor," International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, China, October 2018.

[9] Zelin Meng, Xiangbo Kong, Lin Meng, and Hiroyuki Tomiyama, "Distance Measurement and Camera Calibration based on Binocular Vision Technology," In Proceedings of International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 349-354, Zhengzhou, China, August 2018.

[10] Xiangbo Kong, Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, "A Privacy Protected Fall Detection IoT System for Elderly Persons using Depth Camera," In Proceedings of International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 31-35, Zhengzhou, China, August 2018.

[11] Xiangbo Kong, Zelin Meng, Lin Meng, and Hiroyuki Tomiyama, “A Kinect-based Stand-to-fall Analysis for Elderly People,” International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT), Osaka, April 2018.

[12] Zelin Meng, Xiangbo Kong, Lin Meng, and Hiroyuki Tomiyama, "Coordinate Transformations for Binocular Vision Systems," International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT), Osaka, April 2018.