Vision tracking system for UAVS formation

Jin, Ruibing

2016

Jin, R. (2016). Vision tracking system for UAVS formation. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/68877>

<https://doi.org/10.32657/10356/68877>

VISION TRACKING SYSTEM FOR UAVS FORMATION

NANYANG
TECHNOLOGICAL
UNIVERSITY

JIN RUIBING

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2016

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Wang Jianliang for his guidance, understanding, encouragement, invaluable advice for allowing me considerable freedom in doing this research. He gave me a lot of support on the project I joined, which is a very valuable experience for me.

I also would like to thank Mr. Liu Zhichao, a Ph.D student, for his selfless help and patient instructions, especially in doing experiments outdoors. Appreciation also to Mr. Tan Chunkiat, a Ph.D student, for his help on experiments and constructive criticism. A further acknowledgement is due to Dr. Lin Feng, a Research Assistant Professor in NUS Temasek Laboratories, for his technical support and guidance. I also would like to thank all the staff in Control Engineering Laboratory for their kind support in doing experiments and permission for using the various tools in the lab.

Abstract

In this thesis, a vision tracking system for a UAV formation without inter-vehicle communication is proposed. A monocular camera, as a sensor, is installed on the follower UAV in this system, which is to track the leader UAV to maintain a relative distance from the leader UAV with the help of markers on the leader. At first, a target detection algorithm is used to extract the coordinates of the specified markers and then, a pose estimation algorithm is used to calculate the relative distance and the pose of the leader. After that, a prediction part is proposed to predict the location of the leader in the next image frame of the camera based on Kalman filter. After that, the prediction result is fed back to the target detection algorithm, which decreases the possibility of detecting targets wrongly by reducing the size of detection area in the whole image. At the end of this thesis, some experiments are designed and conducted to verify the effectiveness of this system.

Contents

Acknowledgements	i
Abstract	iii
Contents	v
List of Figures	viii
List of Tables	xii
1 Introduction	1
1.1 Motivation	1
1.2 Scope and Objectives	3
1.3 Tracking System Structure	4
1.3.1 Object Detection	5
1.3.2 Pose Estimation	5
1.3.3 Kalman filter	5
1.4 Main Contribution	6
1.5 Thesis Outline	6
2 Literature Survey	7
2.1 Object Detection	8
2.2 Color Detection	9

2.3	Pose Estimation	11
3	Target Detection	13
3.1	Camera Imaging Principle	13
3.1.1	Camera Sensor Introduction	13
3.1.2	Pinhole Camera Model	16
3.2	Camera Calibration	20
3.3	Color Detection	26
3.4	Image Noise Processing	30
3.4.1	Noise Analysis	30
3.4.2	Dilation and Erosion	31
3.4.3	Gaussian Blur	34
3.4.4	Find Target Contours and Target Selection	36
4	Pose Estimation	39
4.1	Theoretical Principle	39
4.2	Simple Pose Estimation	45
4.3	Improved Pose Estimation	48
5	Target Prediction	55
5.1	Discrete White Noise Acceleration Model	55
5.2	Kalman filter Design	56
5.3	Image Segmentation	61
5.3.1	Reset System and Limitation	64
5.4	Algorithm Optimization	65
6	Development Environment and Experiments	67
6.1	Development Environment	67
6.1.1	Introduction of Ubuntu	67

<i>CONTENTS</i>	vii
6.1.2 Introduction of ROS	69
6.1.3 Introduction of OpenCV	70
6.2 Test Board Experiments	70
6.3 UAV Experiments	78
7 Conclusion and Future Work	81
7.1 Conclusion	81
7.2 Future Work	81
Author's Publications	84
Bibliography	84
Appendix	91

List of Figures

1.1	UAV Formation	3
1.2	Vision Tracking System Structure	4
3.1	Bayer Mask	15
3.2	1. Original image. 2. An image with a Bayer filter. 3. An color-coded image with Bayer filter. 4. Reconstructed image.	16
3.3	Pinhole Camera Model	18
3.4	Pinhole Camera Model	19
3.5	The geometry of the model seen from the Y_c axis	20
3.6	Calibration user interface	21
3.7	Camera Calibration Completion	22
3.8	Checkerboard away from camera	22
3.9	Checkerboard move to the left corner	23
3.10	Checkerboard move to the right corner	23
3.11	Checkerboard tilt	24
3.12	Camera Calibration Data	25
3.13	Hsv Color Space	27
3.14	HSV Sliding Bar Window	28
3.15	Original image	29
3.16	HSV target extraction result	29

3.17 Image Example	31
3.18 Filtered Image with Dilation	32
3.19 Filtered Image with Erosion	32
3.20 Original Image	33
3.21 Binary Image	33
3.22 Filtered Image	34
3.23 Gaussian Blur Effect	35
3.24 Binary Image Without Gaussianblur	36
3.25 Binary Image With Gaussianblur	36
3.26 Object Courts in Binary Image	37
3.27 Target Detection	38
4.1 Transformation from world coordinate into camera coordinate	40
4.2 X_c axis orthogonal decomposition	41
4.3 Y_c axis orthogonal decomposition	42
4.4 Transformation from camera coordinate into image plane	43
4.5 CCD Model	44
4.6 Camera Coordinate System	46
4.7 Initial Target in Pose Estimation	46
4.8 Target moves towards the camera	47
4.9 Target rotate about Z axis	47
4.10 Target rotate about Y axis	48
4.11 Target rotate about X axis	48
4.12 Test Board	49
4.13 The test board without rotation in short distance	52
4.14 The test board rotates about Z_c axis in short distance	52
4.15 The test board rotates about X_c axis in short distance	52
4.16 The test board rotates about Y_c axis in short distance	52

4.17	The test board without rotation in long distance	53
4.18	The test board rotates about Z_c axis in long distance	53
4.19	The test board rotates about X_c axis in long distance	53
4.20	The test board rotates about Y_c axis in long distance	53
5.1	Kalman Filter Work Principle	57
5.2	Before crossing the paper on Kalman Filter Experiment	59
5.3	Crossing the paper on Kalman Filter Experiment	60
5.4	After crossing the paper on Kalman Filter Experiment	60
5.5	Image Segmentation	62
5.6	Anti-noise Example	63
5.7	Image Expression in Image Processing	65
6.1	Ubuntu Desktop Environment	68
6.2	OpenCV Logo	70
6.3	Digital Laser Distance Meter	71
6.4	Marker Coordinates at 9.85m	72
6.5	Marker Coordinates at 4.84m	72
6.6	Marker Coordinates at 1.93m	73
6.7	Experiment Outdoors	78
6.8	Experiment Outdoors Plot	79
1	Ideal Image	92
2	Reflection on Target	92
3	Low Light Intensity	93
4	LEDs on The Maker	93
5	LEDs Too bright	94
6	LEDs Circuit	94
7	Red LED	95

8	Yellow LED	95
9	Camera Sensor	96
10	Thin Lens Formula	98
11	Camera Lens	99
12	Original jDrone	100
13	jDrone with markers	100
14	3DR Pixhawk	102

List of Tables

4.1	Rotation State	50
4.2	Rotation State	51
6.1	Parameters of The Digital Laser Distance Meter	71
6.2	Relative Distance Experiments	74
6.3	Influence of the Change of Marker Coordinate	75
6.4	Relative Distance Experiments	77
1	Parameters for Each Marker	96
2	Parameters of Camera Sensor	97
3	Parameters of Camera Lens	99
4	Parameters of Motor	101
5	Power output from ESC, 3S LiPo, 12×45 Propeller	101

Chapter 1

Introduction

1.1 Motivation

Hexarotors are a type of multi-rotor helicopter which can lift and propel themselves by using six rotors and are often used as UAVs in many applications. Recently, there is a boom in research for hexarotor UAVs because of their advantages compared with an equally scaled helicopter. At first, no mechanical linkage between the rotor blade pitch enable their design and maintenance simplified [1]. Secondly, the diameter of each rotor in the six rotor combination is smaller compared with only one rotor in helicopter, so less kinetic energy is created by each rotor and this combination reduces the rotor damage [2]. As a useful tool and platform, hexarotors can be used to carry out many experiments in evaluating new ideas in different fields such like navigation, real time systems, robotics and flight control theory. Besides, a great number of advantages are shown when they are used as versatile test platforms such as cheap price, simple construction and easy maintenance. Currently, they have been widely used in various areas of electrical engineering, mechanical engineering and computer science.

However, as the performance of a single hexarotor is very limited, which cannot

fulfill the requirement of the tasks like cooperative surveillance, UAV formation is developed by researchers. Currently, UAV formation flight technology has received broad attentions [3]. In the field of UAV formation flight, the Formation Holding (FH) is the basis of formation flight. It is to make UAVs fly in a certain UAV formation like a fixed geometric shape. It makes each UAV keep a stable position relative to its neighbors. Then, UAVs can fly in a formation as a whole with the shapes of a line, a triangle, etc [4,5].

The FH can be realized by two types of methods: acoustic method [6] and visual method [7]. Radar is suitable for measuring the relative distance [8], but it becomes ineffective in short distances. In comparison, visual method performs very well in short distances [9,10]. In the above cases, visual method is an effective method as various visual information can be obtained from images.

Visual method is composed of two parts: objection detection and pose estimation. In object detection, most object detection methods are based on feature and machine learning. These methods require too much computation load to implementation on micro-processor for real time detection. However, in practice, the motion of the UAVs varies with time going by, a real time detection is very important and these high cost target detection methods cannot fulfill the requirement of real time detection. Motivated by it, a low cost object detection method is proposed.

The reason why feature detection requires much computation is that features encode ad-hoc domain knowledge. This knowledge is difficult to learn using a small quantity of training data [11] and a simple program. Besides, as the size of the object varies with the variation of relative distance, the size of the related features vary at the same time. As this size of the related features are not prior knowledge, to capture this object, the size of feature detection sub-windows need to be changed in any possible size. This mechanism makes the computation of feature detection very large. Compared with feature, color is the original information which can be

obtained directly from the color space matrices like RGB or HSV. The object with some specified color can be detected easily by setting a certain range for the color space ahead of time. The computation is very small compared with feature. In pose estimation, currently, it requires four coplanar, non-collinear points at least [12]. Although the pose estimation dose not cost too much computation compared with object detection, as the aim of this thesis is to realise real time detection and decrease the computation, only four coplanar, noncollinear points are used in this part. With the analysis above, a low cost vision tracking system for UAVs formation is proposed in this thesis, which is based on color detection and pose estimation.

1.2 Scope and Objectives

The main aim of this thesis is to develop a low cost vision tracking system for UAVs formation with the help of four markers in specified pure color. In this thesis, a line formation flight shape composed of two UAVs shown as Fig.1.1 is only considered.

Figure 1.1: UAV Formation

As the aim of this thesis is to propose a low cost tracking system for UAVs formation, how to distinguish the four markers in specified color, how to estimate the pose of the leader UAV shown in Fig.1.1 and how to avoid the influence of noise

in the environment are the main part of the thesis. In this study, there are three parts in this thesis. Before discussing on each part in chapters, to understanding the relationship among three parts, the tracking system structure is presented at first.

1.3 Tracking System Structure

The vision tracking system in this thesis is divided into three parts: target detection, pose estimation and target prediction, as shown in Fig.1.2.

Figure 1.2: Vision Tracking System Structure

This vision tracking system is based on four LED markers with specified colors which are fixed on the leader in advance. At first, the first part, target detection, recognises these markers from the image captured by a camera installed on the follower UAV. Then, target detection determines the coordinates for markers and passes the result to the second part, pose estimation. After that, pose estimation uses these coordinates to calculate the relative distance between the follower UAV and the leader UAV and the relative pose of the leader UAV. Finally, target prediction is used to predict the location of the markers in the next frame based on the pose estimation and segments four parts of the whole image. The four parts are fed back to target detection, the first part, for color detection.

1.3.1 Object Detection

In this thesis, the object is the UAV leader in the line shape formation. Actually, as color detection is used to detect this object, the objects which are detected are those markers in the specified color. The main work in this part is to extract those markers by color detection. However, there are some neighbor construction in similar color and how to filter out these disturbance is very important, too. After extracting the four markers, image moment is used to calculate the area of the marker and the center coordinates of the markers.

1.3.2 Pose Estimation

In this thesis, as the main aim is to use low cost tracking system to detect the UAV leader and the object detection is the main part, the pose estimation part is just implemented with functions provided by OpenCV library directly. In this part, four-point algorithm is used and the results of the relative pose are used to test the performance of this tracking system.

1.3.3 Kalman filter

As color is only used as information to detect the object in this system, this system is influenced easily by other plants or construction. To solve this problem, Kalman filter is used to segment the whole image captured by the camera and make the object detector only detect the marker in the predicted region by the Kalman filter. However, as Kalman filter may loss the markers sometimes, a system which resets Kalman filter is proposed.

1.4 Main Contribution

This thesis develops a novel tracking system which uses the color as detection information directly. With this simple information, the computation is apparently very small. Kalman filter is used to limit the region of color detection, which can increase the performance of this tracking system.

1.5 Thesis Outline

- Chapter 1 presents the motivation and objective of this thesis and major contributions.
- Chapter 2 reviews the existing results on the field of this thesis. The visual detection field is mainly discussed and pose estimation is discussed together. Additionally, the state of art and limitations are summarized.
- Object detection is discussed and related hardware and theory are presented together in Chapter 3.
- In Chapter 4, pose estimation and the specific markers are addressed.
- Chapter 5 presents the mathematic model for each marker and the Kalman filter performance is tested in some experiments.
- Experimental results and error analysis are given in Chapter 6.
- Conclusion and future work are shown in Chapter 7

Chapter 2

Literature Survey

As the aim of this thesis is to develop a low cost tracking system, how to track the UAV leader in a UAV formation is the main work. From the discussion above, the object detection occupies the most of computation. To implement a low cost tracking system, how to detect an object in low cost is important. Besides, the pose estimation is reviewed briefly.

This chapter focus on the state of art in object detection and color detection. Both research fields have been researched by many researchers. For the object detection, this field has been developed for a long time. Currently, the problem for detecting and tracking some objects like pedestrian and automobile has been solved successfully. However, this kind of algorithms cannot be extended directly to aircrafts, as the motion of the aircrafts are more complex and the aircrafts become small when they fly far away from the camera.

As the color detection is applied in this tracking system, the methods in color detection are investigated. The color detection is based on color space like RGB, HSV and YCrCd. Different color space has different performance in applications. In this review, the characters of these color space are reviewed. Then, a suitable color space is determined in this tacking system.

The pose estimation has been developed for a long time. In research field, the part is classified into a direction called visual servo control. The main work in this field can be divided into position-based visual servo control and image-based visual servo control [12].

In this chapter, the state of art of object detection and color detection are reviewed. Then the disadvantage of the object detection is discussed. In the end, the principle of pose estimation is reviewed briefly.

2.1 Object Detection

The object detection for moving objects currently can be divided into three types: appearance-based methods, motion-based methods and hybrid methods [13].

- **Appearance-based methods.** Currently, most of these methods rely on machine learning and feature detection. Some powerful methods like [14], [15], and [16], have proved to be effective even in the case of complex lighting variations or cluttered background. However, the performance of these methods are good when the objects are sufficiently large and clearly visible in individual image. As the speed of the aircraft is fast, these methods are not the case in our application. Besides, as these methods are based on Deformable Part Models (DPM), Convolutional Neural Network (CNN) or Random Forests, the computation is too large to implement in real time on the micro-processor on a UAV.
- **Motion-based methods.** In this field, it can be divided into two subclasses. The first one refer to extracting the objects from the background, which highly rely on background subtraction [17,18]. The second method is a flow-based method which depends on optical flow between consecutive images [19]. However, the first one is effective only for the motion of the camera is static or

small enough to be compensated. The second one is suitable for the high quality images. In UAVs formation, the camera equipped on an UAV, which makes the camera moves in real time. For the image, as the UAV tracked flies in real time, the UAV tracked cannot be clear all the time. These features above in UAVs formation makes the motion-based methods difficult to apply.

- **Hybrid methods.** This method combines the appearance and motion of the object together. For example, histogram of flow vectors and appearance features are combined together in [20].

From the three types of methods, It is obvious that the computation is very large, though the accuracy and robustness of them are excellent. It is difficult to apply these methods on real time object detection.

2.2 Color Detection

Compared with general object detection based on features or motion, color detection is a pixel-based method which allows fast processing. So, this method is an effective method in real time detection. Although the color detection method are generally used in face skin detection like [21] and [22], the principle of the color detection is still suitable for detecting markers in a certain specified color. The color detection is based on the selected range of the certain color space. For example, in RGB color space, if the selected range for R is set (0,25), the color detection will detect all pixels, whose R value is between (0,25). If we determine a suitable range of a certain color space in our tracking system. The color detection methods are effective in this application. As the color of the markers is pure color, the color modelling of markers is only related to type of color. The color modelling of the markers is not discussed in this review. The main part of color detection reviews the different color space.

- **RGB.** The RGB(red, green and blue) is a traditional color space. It can be used to describe a certain color as a combination of three colored rays (red, green and blue). It is a widely used color space in processing and storing of digital image data. However, as some characters like high correlation between channels, significant and perceptual non-uniformity make RGB difficult to apply on color analysis and color-based recognition. This color space is used in [21] and [23].
- **HSI, HSV, HSL.** These color spaces are shorted for Hue Saturation Intensity (Value, Lightness) and are classified into hue-saturation based color space. They were developed for specifying color properties numerically. This type of methods express color with intuitive values, based on the artists idea of tint, saturation and tone. Hue denotes the main color like red, green, and yellow of an object. Saturation expresses the colorfulness of an object in proportion to the brightness of it [24]. The "intensity", "lightness" or "value" is related to the color luminance. The important features of Hue presented in [25] is that hue is invariant to highlights at white light sources. Especially in matte surfaces, it is invariant to ambient light and surface orientation relative to the light source. However, in [24], it points that there are some undesirable features in this type of color space, like hue discontinuities and the computation of "brightness" (lightness, value), which conflicts badly with the properties of color vision.
- **TSL.** It is shorted for Tint, Saturation, Lightness. This space is a transformation of the normalized RGB into more intuitive values. It is close to hue-saturation based color space.
- **YCrCb.** This color space is an encoded nonlinear RGB signal. It is often used by European television studios and for image compression work. As

the transformation from RGB is simple and can separate the luminance and chrominance components explicitly, it was used in skin color modelling [26].

In practice, as the sunlight may affect the color of the markers in camera image, a color space which can express the color stably is required. From the color space analysis above, the HSV color space is used in this tracking system because of the invariant performance.

2.3 Pose Estimation

The pose estimation belongs to the position-based visual servo control [27]. It is the task of estimating the motion parameters of an objects from the data like point coordinates. This problem is proposed based on an object with multiple feature points known. There are numerous methods for this problem [28–30]. There are three categories for this problem [31]: model-free, indirect model-use and model-based analysis-by-synthesis. The model-free and indirect model-use dose not use a priori model information. The third one is to use a template model. The model is compared of motion capabilities, joint positions and the object shape.

Pose estimation generally needs some feature point coordinates to estimate. After getting these coordinates, how to calculate the relative pose solution is a problem. Analytic solutions for this problem has been investigated. Three or four points are originated from [32–35]. For four points, there is unique solution only if the four points are coplanar, and not collinear. Six and more points also yield unique solution. Solving this unique solution generally uses iteration methods. Least-square solutions is usually used.

These methods are originated from [36–38]. However, the general least-squares solution is a nonlinear optimization problem. It has no known closed-form solution. Instead, iterative optimization methods are employed. Gauss-Newton algorithm is

used to solve this problem in [39]. In this tracking system, the LevenbergMarquardt algorithm [40] is applied.

Chapter 3

Target Detection

In this chapter, the first part, target detection, is discussed. Firstly, camera imaging principle is investigated. Then, camera calibration is discussed. After that, as this tracking system detects the leader UAV with the help of markers in specified colors, the color detection algorithm is discussed. Then, some noise reduction methods are discussed, which are used to filter out the noise in the image. At last, the coordinate calculation algorithm is investigated.

3.1 Camera Imaging Principle

3.1.1 Camera Sensor Introduction

The image sensor is the basis of camera imaging which enables cameras to capture the image from lens. Currently, image sensors in modern cameras can be divided into charge-coupled device (CCD) and complementary metal oxide semiconductor (CMOS).

As the camera used in this system is a CCD camera, camera imaging principle in this thesis focuses on CCD. CCD is a major component in digital imaging. The pix-

els in a CCD are made of p-doped MOS capacitors. When a CCD captures images from lens, a photoactive region (an epitaxial layer of silicon) is activated, and then a transmission region made out of a shift register works.

When an image projected onto the photoactive region made out of a capacitor array, each capacitor in this array accumulates electric charge which is proportional to the light intensity. In line-scan cameras, a one-dimensional array is used to capture a single slice of the image. For cameras used in video or still camera, a two-dimensional array is used to capture a two-dimensional picture. When the capacitors have accumulated electric charge, a control circuit makes each capacitor to transfer its electric charge to its neighbor, whose operation is like a shift register. The last capacitor in this array passes its charge into a charge amplifier and this amplifier converts this charge into a voltage. By repeating this operation, the control circuit converts the entire contents of the array into a sequence of voltages. However, in digital device, it is different from an analog device. These voltages are sampled and digitized first. After that, they are usually stored in memory, while in an analog device, they are processed into a continuous analog signal, which is then processed and fed out to other circuits for transmission, recording, or other processing.

However, these voltages cannot be classified into different colors directly. To solve this problem, a filter is put on the CCD. Currently, most of modern cameras are digital camera which are generally equipped with a Bayer mask as a filter over the CCD shown in the following Fig.3.1. In this mask, each square of four pixels has one filtered red, one blue, and two green because the human eye is more sensitive to green than either red or blue. As a result, luminance information is recorded at every pixel, but the color resolution is lower than the luminance resolution. After filtering the luminance information, a simple color camera is made.

Figure 3.1: Bayer Mask

Then, by adding a Bayer mask on CCD, the luminance of the filtered color (red, blue or green) on one pixel is recorded and converted into a voltage (like discussed above). However, this filter only recognises one of the three base colors and cannot display the raw image. To obtain the raw image, various demosaicking algorithms are used to interpolate a set of complete red, green, and blue values for each pixel. These algorithms estimate the values for a particular pixel by using of the surrounding pixels of the corresponding colors. Different algorithms requires different amounts of power supplying, which results in varying-quality final images such like JPEG image, TIFF image and raw image.

A simple method is to interpolate the color value of a pixel with the same color according to its neighborhood. For example, once the sensor has captured an image, each voltage value of a pixel can be obtained. For a pixel with a green filter, the exact value of the green component can be offered by itself. The red and blue components for this pixel can be obtained from its neighbors roughly. With this method, the red value of this pixel can be obtained from two pixel neighbors with red filters by interpolation and similarly, the blue value of this pixel can be obtained from two pixel neighbors with blue filters. This simple method performs well in the environment with constant color or smooth gradients, but it also causes artifacts

such like color bleeding when some abrupt changes of color or brightness happens in the area, especially for sharp edges. To solve this problem, other demosaicking algorithms are proposed to produce high-contrast edges and only interpolate along these edges, not across them. The following Fig.3.2 shows different results of the same image in different imaging stages.

Figure 3.2: 1. Original image. 2. An image with a Bayer filter. 3. An color-coded image with Bayer filter. 4. Reconstructed image.

In the Fig. 3.2, the first image is the original image. The second image is created with the Bayer filter. As this filter is composed of three types of filters which only receive red, blue or green, there are three filtered images. The Bayer filter also decrease the resolution of the original image. Then, after combining the information of the three filtered image, the third image is obtained. In the end, the fourth image is made by using demosaicking algorithms.

3.1.2 Pinhole Camera Model

The important part of this tracking system is to track and calculate the relative distance between the leader UAV and the follower UAV. In [8], the distance is measured by radars. This method, radar measurement, performs well in long distance, but it is not effective in short distance. Compared with radars, the performance of visual measurement works well in short distance measurement [9,10]. Apparently, a straightforward solution is using stereo vision which can easily provide depth measurement. However, the computation of it is too intensive to be realized easily in

airborne computers. A monocular camera was used in [41] to measure the relative distance for UAV formations with the assumption of known artificial markers on the leader. However, the problems in vision-based target detection and orientation tracking were ignored. In our vision tracking system, a monocular camera is used for marker extraction and 3D relative measurement.

Different types of cameras have different camera models. As the tracking system is required to be applied on UAV formations and for safety, each UAV should keep a certain distance from the leader, how to capture a image in high quality is very important to determine the relative distance of the leader. In this section, the Pin-hole Camera shown in Fig.3.3 is introduced and a part of related imaging principle which happens in camera is discussed. Before the camera imaging, a transformation is made to enable the plane of objects be parallel to the plane of camera (discussed in Chapter 4). Then, the camera sensor converts the 3D objects into 2D objects. As this process happens in a pinhole camera, the model is introduced at first.

Pinhole camera model is a simple camera without a lens and with a single and small aperture. It is just like a light-proof box with a small hole on one side and the opposite side of the box is exposed to the light that passes through the single point and then the opposite side of the box shows an inverted image. In this box, it is completely dark on all the other sides of the box including the side in which the hole is created. There is a thin screen which looks like a projector sheet, and is set between the dark side adjacent to the pinhole.

Figure 3.3: Pinhole Camera Model

Optimally, the size of the aperture (the small hole) should be $1/100$ or less of the distance between it and the projected image. As a pinhole camera usually requires a lengthy exposure and its shutter is manually operated which is implemented with a flap made of light-proof material to cover and uncover the pinhole, the typical exposures range from 5 seconds to several hours. To simplify the problem, this model does not include factors such like geometric distortions, finite sized apertures and blurring of unfocused objects caused. It also does not consider that most practical cameras have only discrete image coordinates which means that the pinhole camera model can only be used as an approximation of the mapping from a 3D scene to a 2D image and in this case, the 3D scene is assumed to be parallel to the 2D image plane.

It is well-known that the image is expressed by the pixels and the number of the pixel for the target is generally a criteria to judge whether a detected object is the target(discussed later). Because of it, it is necessary to calculate the number of the pixel for a certain target in a certain distance. As the pixel size is constant in the same camera, if the object size is available, the number of pixels can be calculated. For calculating the object size in camera, the mathematics of the pinhole camera

should be investigated at first. Its geometry is shown in Fig.3.4.

Figure 3.4: Pinhole Camera Model

In this figure, R denotes the point at the intersection of the optical axis and the image plane, and P denotes the point in the world coordinate system (X_c, Y_c, Z_c) . As we have assumed that the 3D scene is parallel to the 2D image plane, the camera coordinate system is not discussed in this chapter (discussed later). In practical situations, generally, $Z_c > 0$ is often assumed. In the image plane, a 2D coordinate system is defined with the origin R and axes u and v which are parallel to X_c and Y_c , respectively. The point Q is represented with (u, v) . In this figure, we assume that all projection lines which pass through the aperture of the pinhole camera are infinitely small, a point. If we look down in the negative direction of the Y_c axis in the following Fig.3.5, it is easier to understand how the coordinate (X_c, Y_c, Z_c) of point P project onto the coordinate (u, v) of point Q .

Figure 3.5: The geometry of the model seen from the Y_c axis

In this figure, two similar triangles are shown and parts of the green projection line become their hypotenuses. The side length of the left triangle are $-u$ and F and the side length of the right triangle are X_c and Z_c . As the two triangles are similar, it follows this equation:

$$\frac{-u}{F} = \frac{X_c}{Z_c} \quad \text{or} \quad u = -\frac{FX_c}{Z_c} \quad (3.1)$$

By measuring the length of X_c , Y_c and F , it is easy to calculate the length of u . And if we obtain the pixel size S , we can calculate the number of the pixel n as follows:

$$n = \frac{u}{S} \quad (3.2)$$

3.2 Camera Calibration

The camera used in this system is a pinhole camera, and the images it captures are raw images. Some radial distortions are caused on these raw images because

of the curved lens of this camera. Additionally, the position of the image center is typically not at half width and half length of the image plane on the camera. To calculate the data from the images, an undistorted replot of the image should be carried out and the real image center needs to be spotted out as well. This problem is called camera calibration.

In ROS, a package, *camera_calibration*, is provided to calibrate the camera which can be run by inputting the camera's *topic* to the *roslaunch camera_calibration* command in the command window. Then, a calibration *node* runs and a user interface is shown like the following Fig.3.6.

Figure 3.6: Calibration user interface

For camera calibration, a checkerboard of proper size is needed. In this process, this checkerboard should be moved around the camera and this board movement range is required to cover top, left, right, and bottom part of the camera. Besides, to get a good calibration result, tilting the checkerboard at the corners of the camera is important, too. Moreover, the orientation and the size of the checkerboard is also crucial in this process. When you move this board, it should be set in different orientations at various positions before the camera and moved forward and backward to get different distances from the camera. The calibration interface lists several buttons (shown in Fig.3.7) like X, Y, and Size bar at the side, which instructs user

to change the moving direction of the checkerboard from left to right or from top to bottom, as well as toward, away and tilt. When the camera calibration is finished, the three indications bar will change to green (shown in Fig.3.7). The calibration process described above is shown in Fig.3.8 to Fig.3.11.

Figure 3.7: Camera Calibration Completion

Figure 3.8: Checkerboard away from camera

Figure 3.9: Checkerboard move to the left corner

Figure 3.10: Checkerboard move to the right corner

Figure 3.11: Checkerboard tilt

When the camera calibration process is finished, the CALIBRATE button listed on the interface lights up. Then, the calibration data is processed by clicking the “CALIBRATE” button and afterwards the SAVE button lights up and the data of camera calibration is saved in a plain text file as shown in Fig.3.12.

```

[image]

width
1288

height
964

[narrow_stereo]

camera matrix
1610.936106 0.000000 645.275188
0.000000 1613.071500 462.581071
0.000000 0.000000 1.000000

distortion
-0.347202 0.152194 0.000190 0.001596 0.000000

rectification
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000

projection
1516.943237 0.000000 646.496528 0.000000
0.000000 1560.096802 460.901219 0.000000
0.000000 0.000000 1.000000 0.000000

```

Figure 3.12: Camera Calibration Data

From the above figure, this CCD camera has a resolution of 1288×964 . The pixel size is $3.75\mu m$. Using the Camera matrix data, the focal length in X and Y directions can be calculated as follows:

$$f_x = 1610.936 * 3.75\mu m = 6.0mm \quad (3.3)$$

$$f_y = 1613.072 * 3.75\mu m = 6.0mm \quad (3.4)$$

and the center of the camera is determined as:

$$u_x = 645.275 * 3.75\mu m = 2.420mm \quad (3.5)$$

$$u_y = 462.581 * 3.75\mu m = 1.735mm \quad (3.6)$$

The distortion matrix is $[-0.347202, 0.152194, 0.000190, 0.001596, 0.000000]$. After calibration, there is no bias in the data obtained from the image, which makes the data of an image more reliable.

3.3 Color Detection

In the tracking system, four markers with specified colors are used, so color detection is the basis of tracking the leader UAV. In practice, the color detection is very sensitive to light condition and the object orientation, which means different orientation of the same object may reflect different amount of light. As the discussion in Chapter 2, HSV color space (shown in the Fig.3.13) can express a kind of color with a single parameter range (hue). The parameter hue can determine which type of color and the other parameters can determine the color shades. So, it is easier for researchers to find a suitable parameter range.

Figure 3.13: Hsv Color Space

From Fig. 3.13, with the variance of hue, the type of color varies. The parameters, saturation and value affect the color shades.

In order to adjust the HSV parameters easily, the calibrated image is firstly transformed from RGB color space into HSV color space with OpenCV lib. Then, a slide bar is produced for manual adjustment of the HSV maximum and minimum thresholds of the three parameters respectively as shown in Fig.3.14.

Figure 3.14: HSV Sliding Bar Window

In Fig. 3.14, the H_MIN and H_MAX denote the minimum and maximum value of the hue parameter in the color detection range. Similarly, S_MIN and S_MAX are for saturation and V_MIN and V_MAX are for value. By moving the slide bar, the detected color can be determined.

After adjusting the parameters on the HSV sliding bar, a OpenCV function *scalar* is used to transform the original and color image shown in Fig.3.15 into a binary image Fig.3.16 with the target in white and background in black.

Figure 3.15: Original image

Figure 3.16: HSV target extraction result

However, in some cases, the binary image is not satisfying, because there are some noise and the pixels around the target is also scattered which makes it more difficult to determine whether these white pixels denote the same target (discussed later). To solve this problem, some image processing algorithms need to be used.

3.4 Image Noise Processing

As the problem mentioned above, a simple color detection is insufficient to generate a good binary image due to the noise in the original and color image. To solve this problem, some image processing algorithms provided by OpenCV library are applied on it. Before image noise processing, it is necessary to analysis the image noise.

3.4.1 Noise Analysis

In this vision tracking system, target detection is based on color detection and those objects in similar color as markers may cause disturbance on target detection. These objects in similar color are regarded as noise. As there are few objects in nature that are of pure single color and the size of such single color objects is usually small, as well, so an effective method to distinguish markers from environment is using the area of the objects. However, in some cases, there are several objects in a pure single color which may affect the color detection. To effectively solve this problem, the noise is divided into two types: strong noise and weak noise.

Strong noise denotes the areas of the objects in similar color as markers are large enough and can make the tracking system recognise them as markers. For this type of noise, the tracking system filters it with the help of the third part of our vision tracking system, target prediction, which will be discussed in Chapter 5.

Weak noise denotes the areas of the objects in similar color as markers are small. If the target detection recognises the objects with the largest area, the weak noise will not affect target detection. However, with the relative distance increasing, the size of area of markers on the image decreases and an extra pixel near a marker may affect the pose estimation result dramatically (discussed in Chapter 6), some image processing algorithms are applied to filter out the noise. In the following section, the main work focuses on filtering out weak noise.

3.4.2 Dilation and Erosion

There are some algorithms on noise filter in image processing. Morphological operations are important parts of them. They apply a *structuring element* to an input image and then, generate an output image [42]. Among morphological operations, erosion and dilation are the most basic of them. There is a wide range of applications for these two operations such like filtering noise, isolating individual elements and joining disparate elements in an image, and finding intensity bumps or holes in an image.

To explain how they work, an image shown in Fig.3.17 is used as an example.

Figure 3.17: Image Example

Dilation. This operation convolutes an image A with *structuring elements* B . For the *structuring elements* B , it has a simple and defined shape in general. When the dilation is used on the image, the *structuring elements* B is scanned over the image. Then, the maximal value of pixel overlapped by B is computed and the image pixel at the anchor point position is replaced with that maximal value. So, this operation makes bright regions within an image “grow”. After dilation, the filtered image is shown in Fig.3.18. Obviously, the background (white) dilates around the black region of the letter.

Figure 3.18: Filtered Image with Dilation

By applying this operation, some weak noise which occupies a small number of pixels will be eliminated. However, this operation also causes the size of target becoming small, which is not beneficial to target detection. In order to solve this problem, another opposite operation is applied.

Erosion. This operation is the sister of dilation. Compared with dilation, it computes a local minimum over the area of the *structuring elements*. When erosion is used on the image, the *structuring elements* B is scanned over the image and the minimal value of pixel overlapped by B is computed and the image pixel at the anchor point position is replaced with that minimal value. The result of erosion for the image example is shown in Fig.3.19. It is obvious that the white areas of the image (the background) becomes thinner, whereas the dark zones becomes bigger.

Figure 3.19: Filtered Image with Erosion

With the help of both algorithms, by using erosion at first and then dilation, the weak noise on the image can be filtered out and the area of markers does not change. The image with noise is shown in Fig.3.20, the corresponding binary image is shown in Fig.3.21 and the filtered image is shown in Fig.3.22. In Fig. 3.21, it can be seen that there are some noise in this image, like some white pints on the right

and several short lines on the left. In Fig. 3.22, it can be seen that the white points has been filtered out and the white lines become thin. The object, white circle does not change apparently.

Figure 3.20: Original Image

Figure 3.21: Binary Image

Figure 3.22: Filtered Image

3.4.3 Gaussian Blur

In target detection, it extracts the contour of the markers, then calculates the center coordinates of the markers and finally passes the results to pose estimation. However, after erosion and dilation, a problem is that the pixels on the markers are scattered and shake in time, which is not beneficial to determine the markers location. At pose estimation, one pixel change for the marker center can change the result of pose estimation and this effect increases with the relative distance increasing (discussed in Chapter 6). To solve this problem, another image processing method, Gaussian blur, is used.

Gaussian blur is also called Gaussian smoothing. It uses a Gaussian function to blur an image, which is widely used in graphics software, especially in reducing image noise and detail. This blurring technique provides a visual effect, a kind of smooth blur resembling [43]. It is just like viewing the image through a translucent screen, which is different from the bokeh effect made by an out-of-focus lens or the shadow of an object under usual illumination. This algorithm is also applied at the pre-processing stage in computer vision to enhance image structures at different scales.

For calculating the transformation to apply to each pixel of the image, an equation

of a Gaussian function in one dimension is as follows:

$$G(x) = \frac{1}{\sqrt{2\pi\theta^2}} e^{-\frac{x^2}{2\theta^2}} \quad (3.7)$$

In two dimensions, it can be regarded as the product of two Gaussian functions, one in each dimension [44,45]:

$$G(x, y) = \frac{1}{\sqrt{2\pi\theta^2}} e^{-\frac{x^2+y^2}{2\theta^2}} \quad (3.8)$$

where x denotes the distance from the origin in the horizontal direction, y denotes the distance from the origin in the vertical direction and θ denotes the standard deviation of the Gaussian distribution. The effect of it is shown in Fig.3.23.

Figure 3.23: Gaussian Blur Effect

In the Fig. 3.23, the filter effectiveness increases with the increase of the Gaussian filter radius. However, with the increase of the radius, the image information lose, too. There is a trade off in this Gaussian filter. The second image is made by three pixel radius and the third image is made by ten pixel radius. In practice, the value of the Gaussian blur depends on the specific environment.

By using Gaussian blur, this problem that the pixels on markers are scattered can be solved. In Fig.3.24, the contour of the white area is not smooth and the pixel at

the edge may affect the center location. After using Gaussian blur, in Fig.3.25, it can be found that the contour of the marker has been smoothed.

Figure 3.24: Binary Image Without Gaussianblur

Figure 3.25: Binary Image With Gaussianblur

In the tracking system, the radius of Gaussian filter is selected at 9 pixels and the θ is equal to 2.

3.4.4 Find Target Contours and Target Selection

After image noise processing, the specified marker has been filtered as a white area while others are black. Then, the contour of this white area is extracted. In this process, a function provided by OpenCV library is used. This function called *findContours* is to find contours in a binary image and it is based on the algorithm in [46]. In this function, it provides three different mode: *CV_CHAIN_APPROX_NONE*, *CV_CHAIN_APPROX_SIMPLE* and *CV_CHAIN_APPROX_TC89_L1*.

- (i) *CV_CHAIN_APPROX_NONE*. This mode makes it to store absolutely all the contour points in sequence, which means that every stored pair of points

(x_1, y_1) and (x_2, y_2) on the contour are neighbors in horizontal, vertical or diagonal direction. The equation which expresses this condition is:

$$\max\{|x_1 - x_2|, |y_2 - y_1|\} = 1. \quad (3.9)$$

- (ii) *CV_CHAIN_APPROX_SIMPLE*. In this mode, the segments in horizontal, vertical, and diagonal are compressed and this mode only stores their end points. For example, a rectangular contour is compressed with 4 points.
- (iii) *CV_CHAIN_APPROX_TC89_L1*. In this mode, another algorithm, Teh-Chin chain approximation algorithm, is applied on finding contours [47].

In this system, to reduce the size of the variables, the *CV_CHAIN_APPROX_SIMPLE* mode is used. And the contour of the image in Fig.3.22 is shown in Fig.3.26.

Figure 3.26: Object Courts in Binary Image

From this image, it is obvious that several contours exist. The target detection algorithm only recognises the contour with maximum area as the marker. *Moment*, a class denoted as *m*, in OpenCV is used to calculate the area and the center of each contour, which is defined as follows:

$$m_{ji} = \sum_{x,y} (array(x,y) \cdot x^j \cdot y^i). \quad (3.10)$$

So, the area S and center coordinate $(\bar{x}, \bar{y})$ are calculated as follows:

$$S = m_{00}, \quad (3.11)$$

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}. \quad (3.12)$$

Then, a target crosshair is drawn at the center of the contour and the coordinate is written next to it as shown in Fig.3.27.

Figure 3.27: Target Detection

After getting the coordinates of the markers, target detection passes the result to pose estimation.

Chapter 4

Pose Estimation

In this chapter, the second part, pose estimation, is investigated. This chapter can be divided into three parts: the theoretical principle, simple pose estimation and improved pose estimation. At first, the theoretical principle is introduced in details. Then, a simple pose estimation is given based on multi-color detection. At last, an improved pose estimation with only two types of color is proposed, which can reduce the possibility of influence by plants and buildings with similar color in the environment.

4.1 Theoretical Principle

The pose of the hexarotor varies with the environment and the trajectory of flight, which changes the direction of the leader UAV flight. As the follower UAV needs to track the leader UAV, the direction of the follower UAV should be aligned with the direction of the leader UAV flight. To make the follower UAV track the leader UAV accurately, the pose of the leader UAV information is provided necessarily. Pose estimation is divided into two parts: relative distance measurement and relative pose measurement.

Camera imaging principle has been discussed in Chapter 2 where we assumed that

the plane of objects is parallel to the camera plane. However, in fact, the plane of objects is not parallel to the camera plane and the actual whole process from 3D objects to 2D objects is divided into two phases and involves three coordinates: 3D object points (X_w, Y_w, Z_w) in world coordinates, 3D points (X_c, Y_c, Z_c) in camera coordinates and projection onto 2D image plane coordinates (u, v) . The first phase is to convert objects from world coordinates into camera coordinates, which makes the plane of objects parallel to the camera plane. The second is to convert objects from camera coordinates into 2D image plane coordinates.

The first phase. In this phase, an object denoted by P_w in world coordinates is transformed into an object denoted by P_c in camera coordinates, which is shown as the Fig.4.1.

Figure 4.1: Transformation from world coordinate into camera coordinate

Because the origin of the world coordinate system is not always at the same location as the origin of the camera coordinate system, a translation matrix T_c is used in the transformation between the two coordinates. Besides, as the three axes of the camera coordinate system are not strictly parallel to the world coordinate system, three angles, θ_x , θ_y and θ_z are used in this transformation to make the plane of the objects parallel to the camera plane, which is expressed by a rotation matrix R_c . This transformation is expressed as the following equation:

$$P_c = R_c(P_w - T_c), \quad (4.1)$$

where

$$P_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \quad R_c = \begin{bmatrix} r_{11}, r_{12}, r_{13} \\ r_{21}, r_{22}, r_{23} \\ r_{31}, r_{32}, r_{33} \end{bmatrix}, \quad P_w = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \text{ and } T_c = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}.$$

For discussing this transformation in details, we assume first that the camera coordinate system only rotates θ_z about Z_c axis relative to the world coordinate system and a matrix R_z denotes the rotation matrix for this rotation which is shown in Fig.4.2 and Fig.4.3.

Figure 4.2: X_c axis orthogonal decomposition

Figure 4.3: Y_c axis orthogonal decomposition

As the camera coordinate system rotates about Z_c axis only, θ_x and θ_y are zero. So, we have,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R_z \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}. \quad (4.2)$$

Then, the rotation R_x and R_y which denote camera coordinate system rotates about X_c axis and Y_c axis respectively can be expressed as:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \text{ and } R_y = \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}. \quad (4.3)$$

So, the rotation matrix R_c can be expressed as $R_c = R_z * R_y * R_x$ and from (4.1), we have,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_c & -R_c * T_c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad (4.4)$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R_c \begin{bmatrix} I & -T_c \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \triangleq M_{ext} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \quad (4.5)$$

The second phase. In this phase, the object denoted by P_c in camera coordinate system is transformed into an image plane denoted by P , which is shown in Fig.4.4.

Figure 4.4: Transformation from camera coordinate into image plane

If the 2D image point is denoted by (x, y) in meters, a point in camera coordinate system is denoted by (X_c, Y_c, Z_c) in meters and focal length is denoted by F in meters, we can obtain from the principle of pinhole camera discussed in Chapter 2:

$$x = F * \frac{X_c}{Z_c} \quad \text{and} \quad y = F * \frac{Y_c}{Z_c} \quad (4.6)$$

In practice, the camera plane is made up of CCD which displays the image in pixel, so, to calculate the image accurately, an equation in pixel should be proposed. (u, v) is used to denote the coordinate in pixel and S_x, S_y denote the width and height of each pixel respectively. Usually, no negative index is used because the origin is at one of the corners such as the right bottom. Then, (u, v) can be calculated as follows:

$$u = \frac{F}{S_x} * \frac{X_c}{Z_c} + O_x, v = \frac{F}{S_y} * \frac{Y_c}{Z_c} + O_y. \quad (4.7)$$

Figure 4.5: CCD Model

Then, we have

$$\begin{bmatrix} s * u \\ s * v \\ s \end{bmatrix} = M_{int} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \quad M_{int} \triangleq \begin{bmatrix} F/S_x & 0 & O_x \\ 0 & F/S_y & O_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.8)$$

Combing M_{int} and M_{ext} to get P , we have

$$\begin{bmatrix} s * u \\ s * v \\ s \end{bmatrix} = M_{int} * M_{ext} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad (4.9)$$

$$\begin{bmatrix} s * u \\ s * v \\ s \end{bmatrix} = P_{(3 \times 4)} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad (4.10)$$

where M_{int} and M_{ext} are given in (4.5) and (4.8) respectively.

4.2 Simple Pose Estimation

With the Theoretical Principle in the subsection 4.1, the relative distance and the relative pose of the leader can be determined. However, as this principle requires much computation, it is difficult to use this principle directly. Fortunately, OpenCV lib discussed in Chapter 1 provides some effective functions based on this principle and we can use them directly.

In this tracking system, the *solvePnP* is used to estimate the pose of the target by providing the marker coordinate array. In this tracking system, four markers are used to determine the pose. For convenience, the camera coordinate system is reproduced in Fig.4.6:

Figure 4.6: Camera Coordinate System

Some experiments are conducted to verify the effectiveness of this algorithm. The interface for it is shown as Fig.4.7 and the result of pose estimation is output on the right window.

Figure 4.7: Initial Target in Pose Estimation

Among the output variables, the *RotationX*, *RotationY*, and *RotationZ* denote the θ_x , θ_y and θ_z angles in degrees respectively and *z_init* denotes the distance between the test board and the camera in horizontal direction in meters. As the task for this tracking system is to keep a certain distance and a certain angle, before doing this experiment, some specific value of relative distance and angles are set in

advance and the pose estimation algorithm calculates the difference value between the reference value and real value. *Asymptote* denotes the error in the distance in a certain direction and *angle_dif* denotes the error in angle. To verify the effectiveness of this algorithm, some experiments are conducted and the experimental data is listed in Chapter 6. This test board size is 43.4 cm x 43.4 cm and each marker size is 6 x 6 cm. For clarity, the two windows are zoomed in and are combined together in following figures.

Figure 4.8: Target moves towards the camera

In Fig.4.8, the target is moved towards the camera. Compared with Fig.4.7, it can be seen that z_{init} varies from 1.18684 to 0.695413.

Figure 4.9: Target rotate about Z axis

In Fig.4.9, the target rotates 45 degrees about Z_c axis. Compared with Fig.4.7, it can be seen that *RotationZ* varies from 0.165337 to 45.4404.

Figure 4.10: Target rotate about Y axis

In Fig.4.10, the target rotates 35 degrees about Y_c axis. Compared with Fig.4.7, it can be seen that *RotationY* varies from -3.22796 to 36.5933.

Figure 4.11: Target rotate about X axis

In Fig.4.11, the target rotates 35 degrees about X_c axis. Compared with Fig.4.7, it can be seen that *RotationX* varies from -3.96592 to -34.9351.

4.3 Improved Pose Estimation

The pose estimation discussed above uses four different marker colors. In practice, to solve the problems discussed in Chapter 2, the LEDs are chosen as markers. However, the color for LEDs is limited and to solve this problem, an improved pose

estimation algorithm is proposed by combining with geometric information. The test board is shown in Fig.4.12 and the number is added on this image.

Figure 4.12: Test Board

The improved pose estimation is divided into three steps: receiving the coordinates of markers, coordinate assignment and pose estimation.

- (i) Receiving the coordinates of markers: In this algorithm, there are only two colors in markers, red and yellow. As there are two markers in the same color, the target detection algorithm which only extracts the object in maximum area is not effective. In this case, for each color, red or yellow, the target detection algorithm recognises two objects whose area are larger than others and pass the coordinates to the second step.
- (ii) Coordinate assignment: As two markers are in the same color, it is difficult to know which coordinate belongs to which marker. To solve this problem, the test board states are divided into four types according to rotation angle. In Fig.4.12, markers 1 and 3 are located on the left of the test board without rotation, so their X_c coordinates are less than markers 2 and 4 in the camera

coordinate shown in Fig.4.6. For convenience, four parameters are defined to determine the test board rotation state:

$$\bar{x}_1 = \frac{x_{m1} + x_{m3}}{2}, \quad (4.11)$$

$$\bar{x}_2 = \frac{x_{m2} + x_{m4}}{2}, \quad (4.12)$$

$$\bar{y}_1 = \frac{y_{m1} + y_{m3}}{2}, \quad (4.13)$$

$$\bar{y}_2 = \frac{y_{m2} + y_{m4}}{2}, \quad (4.14)$$

$$(4.15)$$

where (x_{mn}, y_{mn}) denotes marker n coordinates. After analysis, the rotation state can be determined as follows:

Table 4.1: Rotation State

Parameter	Rotation Range (θ_z)
$\bar{x}_1 \leq \bar{x}_2$ and $\bar{y}_1 \leq \bar{y}_2$	$0^\circ \sim 90^\circ$
$\bar{x}_1 > \bar{x}_2$ and $\bar{y}_1 \leq \bar{y}_2$	$91^\circ \sim 180^\circ$
$\bar{x}_1 > \bar{x}_2$ and $\bar{y}_1 > \bar{y}_2$	$181^\circ \sim 270^\circ$
$\bar{x}_1 \leq \bar{x}_2$ and $\bar{y}_1 > \bar{y}_2$	$271^\circ \sim 360^\circ$

From the table above, it is easy to determine which state the test board is in and in each state, the relationship between the two markers in the same color is listed as follows:

Table 4.2: Rotation State

Rotation Range	Coordinates Relationship
$0^\circ \sim 90^\circ$	$y_{m1} < y_{m3}, y_{m2} < y_{m4}$
$91^\circ \sim 180^\circ$	$x_{m1} > x_{m3}, x_{m2} > x_{m4}$
$181^\circ \sim 270^\circ$	$y_{m1} > y_{m3}, y_{m2} > y_{m4}$
$271^\circ \sim 360^\circ$	$x_{m1} < x_{m3}, x_{m2} < x_{m4}$

With the help of the coordinate relationship, it is easy to know which coordinate belongs to which marker. After that, the corresponding coordinates are passed to pose estimation.

- (iii) Pose estimation. As this part is the same as the simple pose estimation, it is not discussed again and the principle of it can be referred to the simple pose estimation. The experimental results are shown in the following figures.

As the tracking system for UAVs formation is used to track UAV leader in short and long distance, the experiments are divided into two parts: short distance experiments and long distance experiments. Besides, the light and reflection also affect the color detection. Fortunately, in practice, an UAV cannot rotate at a very large angle immediately, no more than 45° in general. So, the rotation test in this experiment is only at 45° . The experiments in this chapter is to test whether this tracking system can detect and calculate the pose information in various pose and different distance. So, we just rotate the test board around 45° manually for rotation test. The experiments which is to test the accuracy are presented in Chapter 6.

In these experiments, the rotation results are listed on the right window, where the rotation X, Y, and Z denote the rotation angle in degrees in three axis directions and distance denotes the distance between camera and the center of the test board in meters.

Figure 4.13: The test board without rotation in short distance

Figure 4.14: The test board rotates about Z_c axis in short distance

Figure 4.15: The test board rotates about X_c axis in short distance

Figure 4.16: The test board rotates about Y_c axis in short distance

Figure 4.17: The test board without rotation in long distance

Figure 4.18: The test board rotates about Z_c axis in long distance

Figure 4.19: The test board rotates about X_c axis in long distance

Figure 4.20: The test board rotates about Y_c axis in long distance

From the Fig. 4.13 to Fig. 4.16, it can be seen that this pose of test board can be calculated with different pose in short distance. As the UAV flight direction is uncertain, the test board rotation test is conducted in three direction, rotating at X, Y, and Z axis.

From the Fig. 4.17 to Fig. 4.20, obviously, the size of markers decreases with the distance increasing. This tracking system still detect the test board stably. The rotation of the test board dose not affect the pose calculation.

For this improved pose estimation, the four markers are necessary to calculate the

pose the the object. If one of the four markers is lost, this improved pose estimation cannot calculate the pose estimation. This is the current limitation of this tracking system. We will solve it in the future work.

Chapter 5

Target Prediction

Although the improved pose estimation has reduced the number of colors of the markers, this tracking system is also affected by objects in the environment. To solve this problem, the target prediction is proposed. In this part, it predicts the location of these markers in the next frame and segments the original image. After that, it passes these segmented images which include the markers to the first part of the vision tracking system, target detection. With this method, most of noise can be filtered out. In this chapter, a mathematic model for each marker is proposed and then a Kalman filter is designed for this model. After that, image segmentation is proposed to reduce the detection area on the image captured by the camera. When the marker tracked by the camera does not appear in the segmented image, a reset system for image segmentation is run and makes image segmentation capture the whole image, not a part of it. This reset system is discussed at last.

5.1 Discrete White Noise Acceleration Model

In order to predict the locations of the markers in the image plane, it is necessary to construct a motion model for the tracking target in mathematics. For convenience, the two-dimensional position of the tracking target and its velocity vector are used

to model the motion in the two-dimensional plane [7]. The model for the tracking target is described as follows [48]:

$$\begin{cases} \mathbf{X}_k = \mathbf{A}_k \mathbf{X}_{k-1} + \mathbf{\Gamma}_k \mathbf{w}_{k-1}, \\ \mathbf{Z}_k = \mathbf{H} \mathbf{X}_k + \mathbf{v}_k, \end{cases} \quad (5.1)$$

where $\mathbf{X} = [x, \dot{x}, y, \dot{y}]^T$ denotes the state vector of the center of a marker in the image plane, $w(t) = [\ddot{x}, \ddot{y}]$ denotes the white noise process and T_k denotes the sampling period of this tracking system at frame k. Other parameters are defined as follows:

$$\mathbf{A}_k = \begin{bmatrix} 1 & T_k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{\Gamma}_k = \begin{bmatrix} \frac{T_k^2}{2} & 0 \\ T_k & 0 \\ 0 & \frac{T_k^2}{2} \\ 0 & T_k \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (5.2)$$

Then, a Kalman filter can be designed based on this motion model to predict the position of the target (marker) in the next frame.

5.2 Kalman filter Design

After modeling the tracking target, a Kalman filter is used to predict the location of the target in the next frame. Kalman filter is also called linear quadratic estimation. It is an algorithm that produces estimation for unknown variables by using some measurements observed over a period of time. With this method, it predicts the variables more precisely than those methods which are based on a single measurement only. Besides, the operation of the Kalman filter is in recursion on input data, so it can provide an optimal estimation of the system state. The filter is named after Rudolf (Rudy) E. Kalman who is one of the primary developers of

this theory.

Kalman filter has been developed into numerous applications in technology. Among them, a common application is used in guidance and navigation, especially in aircraft and spacecraft. Apart from it, in time series analysis, Kalman filter is often used in some fields such like signal process and econometrics. For robotics, it is also a main topic in the planning and control of robotic motion. It is also used in trajectory optimization sometimes. The work principle of this algorithm can be divided into two phases. The first phase is a prediction in which Kalman filter calculates the estimation of the current state variables with their uncertainties based on the record in the last frame. Then, in the second phase, update, it measures the tracking target state variables which necessarily include some amount of error such as random noise. Because of the nature of the algorithm, recursion, it can run in real time with only the present input measurements and the previously calculated state. The work principle of Kalman filter is shown as Fig.5.1.

Figure 5.1: Kalman Filter Work Principle

The model of Kalman filter should be figured out at first in order to predict the state variables of the tracking target. It requires us to determine some parameters of Kalman filter. In Kalman filter model, a condition is assumed, that the true state at time k is obtained from the state at $k - 1$ according to the following (5.3)

$$\mathbf{X}_k = \mathbf{A}_k \mathbf{X}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \Gamma_k \mathbf{w}_k, \quad (5.3)$$

where $\mathbf{A}_k$ denotes the state transition model, $\mathbf{X}_{k-1}$ denotes the previous system state, $\mathbf{B}_k$ denotes the control-input model, $\mathbf{u}_k$ denotes the control vector, Γ_k denotes the process noise transition matrix and $\mathbf{w}_k$ denotes the process noise. For $\mathbf{w}_k$, it is assumed to be drawn from a zero mean normal distribution with covariance $\mathbf{Q}_k$ following: $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$. At time k , the measurement $\mathbf{Z}_k$ is calculated as follows:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{X}_k + \mathbf{v}_k, \quad (5.4)$$

where $\mathbf{H}_k$ denotes a function which transforms the true state space into the observed space and $\mathbf{v}_k$ denotes the observation noise which is similar to $\mathbf{w}_k$, assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_k$ as $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$.

During the process of Kalman filter, the (5.3) and (5.4) are calculated as follows:

(i) Predict. State Estimation: $\hat{\mathbf{X}}_{k|k-1} = \mathbf{A}_k \hat{\mathbf{X}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k$

Estimation Covariance: $\mathbf{P}_{k|k-1} = \mathbf{A}_k \mathbf{P}_{k-1|k-1} \mathbf{A}_k^T + \mathbf{Q}_k$

where $\mathbf{P}_{k-1|k-1}$ denotes the a posteriori error covariance matrix.

(ii) Update. Measurement Residual: $\tilde{\mathbf{y}}_k = \mathbf{Z}_k - \mathbf{H}_k \hat{\mathbf{X}}_{k|k-1}$

Residual Covariance: $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k$

Optimal Kalman Gain: $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}$

Updated State Estimate: $\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$

Updated Estimate Covariance: $\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$

For convenience, the parameters in Kalman filter are constant in practice. By combining the Kalman filter principle with the discrete white noise acceleration model, the $\mathbf{A}_k$ matrix in (5.3) can be replaced by the matrix $\mathbf{A}$ in (5.1), $\mathbf{w}_k$ in (5.3) can be replaced by $\mathbf{w}_{k-1}$ in (5.1), $\mathbf{Q}_k$ can be replaced by $\mathbf{Q}$, $\mathbf{R}_k$ can be replaced by $\mathbf{R}$ and $\mathbf{H}_k$ in (5.4) can be replaced by $\mathbf{H}$ in (5.1). As the Kalman filter in this system is to predict the tracking location, the control vector $\mathbf{u}_k$ is zero.

However, as this algorithm requires much computation, it is difficult to realise directly. Fortunately, the OpenCV lib provides Kalman filter function to predicts the tacking target location. In this function, it requires the input of related parameters in Kalman filter. As it is discussed above, some important parameters have been specified. However, the process noise covariance matrix $\mathbf{Q}$ and measure noise covariance matrix $\mathbf{R}$ are uncertain and some experiments are conducted to get suitable value.

The experiments are designed as follows: a red ball is the target which the Kalman filter needs to predict. A sheet of paper is used as a barrier to shelter the red ball from the camera. Then, the red ball crosses this sheet of paper. A green rectangle is used to denote the real position of this red ball based on color detection and a red rectangle is used to denote the result of the Kalman filter. After the red ball crosses this sheet of paper, the size of the area of overlap part of the green and red rectangles is used to adjust Kalman filter parameters. The experiment is shown in the following figures:

Figure 5.2: Before crossing the paper on Kalman Filter Experiment

Figure 5.3: Crossing the paper on Kalman Filter Experiment

Figure 5.4: After crossing the paper on Kalman Filter Experiment

By repeating this experiment, the parameters, $\mathbf{Q}$ and $\mathbf{R}$, are determined as follows:

$$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 2.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} \quad (5.5)$$

$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad (5.6)$$

After determining the value of $\mathbf{Q}$ and $\mathbf{R}$, it is obvious that the red rectangle delays a little. After the red ball crosses the paper, the red rectangle overlaps most parts of the green rectangle in Fig.5.4. A video named *kalman_filter* for this experiment is provided in a DVD attached to this thesis. If it is not convenient to watch the video in the DVD, the video is also available in YouTube at this link:https://youtu.be/U0ZvWbpXTo4?list=PL01RcIj_lnWdXPCXVhBB-WZLezKJ2mztt.

5.3 Image Segmentation

Noise analysis is discussed in Chapter 3 and strong noise may exist in certain cases. To filter out this type of noise, image segmentation is applied. Based on the Kalman prediction result, the possible locations of markers are estimated and the target detection can only detect markers in these possible areas, not in the whole image. With this method, much noise is filtered out.

At first, image segmentation gets the possible location of the marker and then, it segments part of the image whose center is the same as the center of the marker estimated by Kalman filter and the area is four times that of the marker's, as shown in Fig.5.5.

Figure 5.5: Image Segmentation

In the above image, the window located on the upper-right corner shows the segmented image for the red marker located on the lower-right corner of the whole image. For clarity, $N(k)$ is used to denote the region of the segmented area at the k^{th} frame and $\mathbf{I}$ is used to denote the whole image:

$$N(k) = \{i(x, y) \in \mathbf{I} \mid x \in [x_1, x_2], y \in [y_1, y_2]\}, \quad (5.7)$$

where

$$x_1 = \hat{x}(k|k-1) - w(k-1), \quad (5.8)$$

$$x_2 = \hat{x}(k|k-1) + w(k-1), \quad (5.9)$$

$$y_1 = \hat{y}(k|k-1) - h(k-1), \quad (5.10)$$

$$y_2 = \hat{y}(k|k-1) + h(k-1), \quad (5.11)$$

$$(5.12)$$

with w and h being the width and height of the marker detected in the previous $(k-1)^{th}$ frame. A video named *image_segment* for this experiment is provided

in a DVD attached to this thesis. If it is not convenient to watch the video in the DVD, the video is also available in YouTube at this link:https://youtu.be/MaeGkHlhtPk?list=PL01RcIj_lnWdXPCXVhBB-WZLezKJ2mztt.

With this method, the anti-noise performance of this tracking system improves and strong noise which does not overlap with markers cannot influence target detection. In Fig.5.6 , the green rectangles denote the position determined by target detection and the blue rectangles denote the prediction by Kalman filter. It is obvious that the area of the red mask is larger than each marker's, but the tracking system only detects the makers. A video named *anti_noise* for this experiment is provided in a DVD attached to this thesis. If it is not convenient to watch the video in the DVD, the video is also available in YouTube at this link:https://youtu.be/N_DLeUGE1r8?list=PL01RcIj_lnWdXPCXVhBB-WZLezKJ2mztt.

Figure 5.6: Anti-noise Example

5.3.1 Reset System and Limitation

As the Kalman filter prediction only depends on the last target detection result, the first target detection result is very important. In this thesis, we assume that the first target detection result is correct. This assumption is easily to implement like using some white board to shelter other objects in similar color.

However, the target detection only detects markers in the segmented images, and, in practice, there are some cases that the target detection cannot detect the color of markers. At first, some obstacles shelter the markers from the camera. In this case, pose estimation uses the prediction result from Kalman filter to calculate the leader UAV's pose for a period of time which is similar to the case in the above video named *kalman_filter*. However, sometimes, the motion of the UAV may changes suddenly due to the disturbance of wind or manual control. The Kalman filter cannot track the four markers, especially when the markers are sheltered.

This case belongs to the second case. In the second case, a reset system runs. The target detection will detect the markers on the whole image draw by the camera, not just on the segmented parts (provided by Kalman filter) of the whole image. This reset system is also shown in the video named *image_segment*.

But, in practice, there are other cases that the markers have moved out of the image draw by the camera and the target detection cannot detect the markers. This is the current limitation of this tracking system.

Another limitation is that the target detection only detects the markers based on color. If there is one additional marker in the same size and color, the target detection cannot distinguish the UAV leader markers from other markers. In this thesis, as we only consider that the UAV formation is a line formation flight shape composed of two UAVs, the additional marker case is not considered.

5.4 Algorithm Optimization

As four Kalman filters are used to estimate the location of the markers, four segmented images need to be used and four vector variables are used to store the four marker contours. These variables and images cost much RAM (random-access memory), which makes the tracking system run slowly on a computer. To solve this problem, this tracking system should be analyzed.

In image processing, an image is generally expressed and stored in a matrix, as shown in Fig.5.7. In Fig.1.2, the tracking system is divided into three parts. In the first part, as it requires image processing, many matrices are used to store the images. The camera resolution currently used is 1288×964 , so a color image needs three 1288×964 matrices, which costs much RAM. Besides, during the process of noise filtering, one binary image needs a 1288×964 matrix, which also cost much RAM. Therefore, the matrices which are used to store images should be reduced. In the second part, as the pose estimation only computes the relative angles and distance, the RAM is not occupied too much. In the third part, as the image segmentation is applied, each segmented image needs three matrices and four makers need 12 matrices, which adds much load on RAM and decreases the speed of the program.

Figure 5.7: Image Expression in Image Processing

From the analysis above, the matrices which are used to denote images cost the majority of RAM and to solve it, an effective method is to reduce the number of

this type of matrices. At first, several variables are created to store the temp results such like the contours of markers and a matrix is used repeatedly. For example, in noise filter, only one matrix is used in dilation, erosion and Gaussian blur by iteration and in segmentation, three matrixes are used to store the segmented color image for each marker by iteration and the coordinates for makers are stored in four temp variables. With this method, although some image operations cannot run at the same time, the cost of RAM is decreased dramatically and the speed of program increases.

Chapter 6

Development Environment and Experiments

In this chapter, the tracking system development environment is introduced at first. Then some experimental results are presented. The tracking system is used to measure the relative distance and angles between the leader UAV and the camera installed on the follower UAV and the experiments are divided into two parts. At first, some experiments are done with the test board for the relative distance and pose. Then, some experiments are done outdoors for the leader UAV and camera to test the relative distance.

6.1 Development Environment

6.1.1 Introduction of Ubuntu

Ubuntu is a Debian-based Linux operating system, which uses Unity as the default desktop environment as shown in Fig.6.1. It is a free software and named after the Southern African philosophy of ubuntu (literally, “human-ness”). It is often translated as “humanity towards other” or “the belief in a universal bond of shar-

ing that connects all humanity”. Ubuntu was developed by UK-based Canonical Ltd which is owned by South African entrepreneur Mark Shuttleworth. Canonical gets revenue by selling technical support and other services related to Ubuntu. The Ubuntu project commits to the principle of open-source software development and people are encouraged to use it, study how it works, improve it, and distribute it. Ubuntu’s default installation provides a wide range of software like Thunderbird, Firefox, Transmission, LibreOffice, and several lightweight games such as Sudoku and chess. Many additional software packages are no longer installed in default and can be obtained from Software Center or other package management softwares based on APT . For programs and applications for Microsoft Windows, they can be run in some compatibility packages like *Wine* or in a virtual machine such as *VMware*. [49]

Figure 6.1: Ubuntu Desktop Environment

The programs which run with low privileges cannot damage the files of operating system and other users’. If you want to increase security, a *sudo* command is used to obtain temporary privileges to realize some administrative tasks. This command makes the root account remain locked and prevents freshmen from making system errors accidentally or causing security holes. So, developing the tracking system for UAVs formation in this environment is safe.

6.1.2 Introduction of ROS

This tracking system is developed in ROS (Robot Operating System) environment which is installed in Ubuntu operating system. ROS is a flexible platform for developing robot software. It provides various tools, libraries, and conventions to simplify complex tasks and robust robot behaviors in a wide variety of robotic platforms. A great number of standard operating system services are available such as hardware abstraction, implementation of commonly used functionality, low-level device control, package management, and message-passing between processes. Some sets of ROS-based running processes are shown in a graph architecture which presents processes in running with *nodes* that denote receiving, posting, multiplex sensor controlling and other messages. Besides, language-independent tools and main client libraries like C++, Python, and LISP are licensed under the BSD license, so there are many open source softwares. They are free for both commercial and research use. As the ROS library is linked to a Unix-like system, the functionality and stability are optimized under Ubuntu operating system. The ROS hydro version is used in this vision-based tracking system.

Operations in ROS are mostly based on command lines in terminals and several terminals can run at the same time. Each terminal is assigned different tasks like executing a *roslaunch* or *roscpp* command, which makes the terminal deal with only one process and does not allow any other operations on this terminal. *roslaunch* is a ROS command which runs one ROS *node*, while *roscpp* could run several *nodes* at the same time. Other commands like *rostopic* denotes dealing with the current topics and *rosservice* denotes solving the current services. [50]

C++, python, and Lisp are fully supported by ROS. Some experimental libraries

are realized in Java and Lua. This tracking system is implemented with C++ programming language.

6.1.3 Introduction of OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions which is mainly developed for real-time computer vision. It was created by Intel Russia research center in Nizhny Novgorod, and currently supported by Willow Garage and Itseez. It is free for use and licensed under the BSD license. The library is a cross-platform which is mainly used in real-time image processing.

Figure 6.2: OpenCV Logo

OpenCV is programmed in C++ which is also the primary interface of OpenCV. But OpenCV still provides a less comprehensive though extensive older C interface. Now, there are full interfaces in Python, Java and MATLAB/OCTAVE and OpenCV provides an online documentation for users to search on the API for these interfaces. All of the new developments and algorithms in OpenCV are now developed in the C++ interface. So, in this tracking system, OpenCV lib is used for image processing [51].

6.2 Test Board Experiments

In this section, a Digital Laser Distance Meter shown in Fig.6.3 is used to verify the accuracy of the tracking system measurement and the parameters of it are listed

as follows:

Table 6.1: Parameters of The Digital Laser Distance Meter

Parameter	Value
Measuring range	0.2 to 50 m
Accuracy	2 mm
Dimension	$118 \times 54 \times 28$ mm

Figure 6.3: Digital Laser Distance Meter

The experiments are divided into two parts: relative distance experiment and angle experiment.

- (i) Relative Distance Experiment. In this experiment, nine different relative distances are tested and more than one hundred times are done for each relative distance. The plane of the test board is parallel to the camera roughly.

In UAV formation, the UAVs should flies in a stable state to keep a certain formation shape. This requires the pose estimation result is stable. If the pose estimation result varies greatly in a short time, the speed of UAV follower will varies greatly, which is very dangerous. As the pose estimation depends on the quality of the image itself and the coordinates of the markers, the figures for the coordinates of markers are plotted. Among them, three figures in different distance are presented as follows:

Figure 6.4: Marker Coordinates at 9.85m

Figure 6.5: Marker Coordinates at 4.84m

Figure 6.6: Marker Coordinates at 1.93m

In the Fig. 6.4, Fig. 6.5, and Fig. 6.6, the Y axis shows the coordinates of each marker and X axis shows the time frame. Different color lines denote different coordinate value. The legend has listed the meaning of them. For example, x1 denotes the x coordinates of marker 1 and y1 denotes the y coordinates of marker 1.

It can be seen that the variation of the coordinates relative to time is very small, around 1 - 2 pixel. The coordinates of each marker is very stable, which means the quality of the image is good and the disturbance from the image itself is very small after image processing. To determine the effectiveness of variation of the coordinates of markers, error analysis is presented.

For error analysis, the relative distance is measured with Digital Laser Distance Meter at first. The maximum and minimum values denote the range of the value measured by the vision tracking system. Percentage $p\%$ is calculated as follows:

$$p\% = \frac{e}{d}\% \quad (6.1)$$

The results are listed in the following table:

Table 6.2: Relative Distance Experiments

Relative Distance (d)	Min	Max	Error (e)	Percentage (p)
9.85m	10.27m	10.52m	0.42 ~ 0.67m	4.3% ~ 6.8%
7.85m	8.37m	8.44m	0.52 ~ 0.59m	6.6% ~ 7.5%
6.84m	7.33m	7.41m	0.49 ~ 0.57m	7.2% ~ 8.3%
5.93m	6.35m	6.44m	0.42 ~ 0.51m	7.1% ~ 8.6%
4.84m	5.15m	5.19m	0.31 ~ 0.35m	6.4% ~ 7.2%
3.84m	4.05m	4.08m	0.21 ~ 0.24m	5.5% ~ 6.3%
2.87m	3.02m	3.05m	0.15 ~ 0.18m	5.2% ~ 6.3%
1.93m	2.05m	2.05m	0.12m	6.2%

As the test board is $0.434\text{m} \times 0.434\text{m}$ and the marker size is $0.06\text{m} \times 0.06\text{m}$, the distance between the center of two markers along the edge of the board is 0.374m . For convenience, a line whose length is 0.374m is used for error analysis and $X_c = 0.374\text{m}$.

From the table above, the percentage error is from 4% to 8%. From $u = -\frac{FX_c}{Z_c}$ in (3.1) and $n = \frac{u}{S}$ in (3.2), we obtain:

$$Z_c = -\frac{FX_c}{Sn}, \quad (6.2)$$

It can be seen that the value Z_c is related to the value of n , because other factors are constant. Now, we analysis the factor of n . From (6.2), we have

$$\frac{dZ_c}{dn} = \frac{FX_c}{Sn^2}, \quad (6.3)$$

where n denotes the pixel number, $F = 6\text{mm}$, $X_c = 0.374\text{m}$ and $S = 3.75\mu\text{m}$.

Substituting these parameters above, we have

$$\frac{dZ_c}{dn} = \frac{598.4}{n^2}. \quad (6.4)$$

And in different relative distance, the derivative of Z_c with respect to n is listed on the following table:

Table 6.3: Influence of the Change of Marker Coordinate

Relative Distance (d)	n	dZ_c/dn	Error in the number of Pixels (n_c)
9.85m	60.75	0.162m	2.6 ~ 4.1
7.85m	76.23	0.103m	5.0 ~ 5.7
6.84m	87.49	0.078m	6.3 ~ 7.3
5.93m	100.91	0.059m	7.1 ~ 8.6
4.84m	123.64	0.039m	7.9 ~ 9.0
3.84m	155.83	0.025m	8.4 ~ 9.6
2.87m	208.50	0.014m	10.7 ~ 12.9
1.93m	310.05	0.006m	20.0

where $n_c = \frac{e}{dZ_c/dn}$.

From the table above, it is obvious that one pixel change dZ_c/dn changes the relative distance measurement. The number of pixel change increases with the relative distance decreasing but the effectiveness of dZ_c/dn decreases with the relative distance decreasing. As the n denotes the number of pixel, which also denotes the distance between two marker centers on CCD. So, the marker coordinates are very important and this is the reason why the Gaussian blur is used in image processing. In this table, the pixel number between each marker is 60.75 in 9.85 m and the sum of the relative distance for four markers should be 243, in theory. However, in this experiment, as a result of light influence,

the sum of pixel is from 221 to 225. This difference between theory and practice reveals that the marker centers are not found correctly. The reason for it is that the light will change the shades of the color of the markers. Sometimes, the light even make part of the markers cannot be detected by camera. This effect makes the tracking system cannot calculate the relative distance.

From the analysis above, the marker coordinates and the relative distance are affected by light, which cause the inaccuracy of pose estimation.

- (ii) Angle Experiments. In these experiments, the angles are tested in three directions as defined in Fig.4.6 with different relative distance and the angle are 30° and 45° . As the angle is not required to estimate accurately, the real angle is measured with triangle boards. The results are listed as follows:

Table 6.4: Relative Distance Experiments

Relative Distance (d)	Angle	Direction	Min	Max	Error (e)	Percentage (p)
3.074m	30°	X_c	27.33°	28.08°	1.92° ~ 2.67°	6.4% ~ 8.9%
3.074m	45°	X_c	43.12°	44.10°	0.90° ~ 1.88°	2.0% ~ 4.2%
3.074m	30°	Y_c	31.68°	32.51°	-2.51° ~ -1.68°	-8.4% ~ -5.6%
3.074m	45°	Y_c	45.12°	46.07°	-1.07° ~ -0.12°	-2.4% ~ -0.3%
3.074m	30°	Z_c	27.81°	28.78°	1.22° ~ 2.19°	4.1% ~ 7.3%
3.074m	45°	Z_c	41.22°	43.26°	1.74° ~ 3.78°	3.9% ~ 8.4%
5.286m	30°	X_c	31.15°	33.04°	-3.04° ~ -1.15°	-10.1% ~ -3.8%
5.286m	45°	X_c	47.77°	49.87°	-4.87° ~ -2.77°	-10.8% ~ -6.2%
5.286m	30°	Y_c	34.16°	35.68°	-5.68° ~ -4.16°	-18.9% ~ -13.9%
5.286m	45°	Y_c	45.41°	46.37°	-1.37° ~ -0.41°	-3.0% ~ -0.9%
5.286m	30°	Z_c	25.91°	26.84°	3.16° ~ 4.09°	10.5% ~ 13.6%
5.286m	45°	Z_c	41.71°	43.41°	1.59° ~ 3.29°	1.3% ~ 7.3%
7.662m	30°	X_c	27.10°	29.10°	0.90° ~ 2.90°	3.0% ~ 9.7%
7.662m	45°	X_c	46.70°	47.82°	-2.82° ~ -1.70°	-6.3% ~ -3.8%
7.662m	30°	Y_c	32.36°	33.27°	-3.27° ~ -2.36°	-10.9% ~ -7.9%
7.662m	45°	Y_c	44.59°	46.70°	-1.70° ~ 0.41°	-3.8% ~ 0.9%
7.662m	30°	Z_c	27.41°	28.49°	1.51° ~ 2.59°	5.0% ~ 8.6%
7.662m	45°	Z_c	46.08°	47.57°	-2.57° ~ -1.08°	-5.7% ~ -2.4%

From the table above, the relative angle measurement is roughly accurate with the error range from -18.9% to 13.6%. Similar to the relative distance measurement, it is influenced by light condition which changes the marker coordinates when the test board rotates.

6.3 UAV Experiments

From the experiments above, the accuracy for the tracking system has been discussed. In this section, the performance for this tracking system on a moving UAV (UAV leader) is discussed. The main work in this section is to test whether the tracking system can track the UAV very well. This experiment is done on an open space, which is shown in the following Fig.6.7. In this experiment, the UAV flies towards the north direction and the direction of the camera is aligned with the direction of the UAV flight. The control algorithm is provided by Pixhawk. As the wind affects the UAV motion, the speed of the UAV is not a constant and the maximum speed is 1m/s. During the experiment, the tracking system runs at 7 Hz. A video named *exp_uav_out* for this experiment is provided in a DVD attached to this thesis. If it is not convenient to watch the video in the DVD, the video is also available in YouTube at this link:https://youtu.be/jhQnYwg3vnI?list=PL01RcIj_1nWdXPCXVhBB-WZLezKJ2mzttt.

Figure 6.7: Experiment Outdoors

The relative distance between the UAV and the camera is plotted on the following

Fig.6.8.

Figure 6.8: Experiment Outdoors Plot

As the relative distance cannot change suddenly, if the curve in this plot is not smooth, it indicates that the tracking system did not track the UAV. From this plot, it can be seen that the UAV did not fly at first and it began to take off around 38s. As the image segmentation was made by Kalman filter, the tracking system cannot track the markers when the UAV accelerated from quiescency to fly. However, when the UAV held at a constant height, the tracking system can track it for a period of time until it began to move in the sky around 52s. After a short time, when the UAV kept in steady state, the derivatives of the aircraft ground position, flight path angle and azimuth angle were constant, the tracking system can track the UAV in time from 62s to the end.

The relative distance between the UAV and camera cannot be determined limited by the laboratory equipment, because the UAV is moving in the sky. It is difficult to measure the distance in real time. GPS is used to determine the relative distance, but it fails to measure the distance as it drifts with time going by. However, we have conducted the test board experiments, which has test the accuracy of this

tracking system. The accuracy of this system has been evaluated in the test board experiments.

Apart from this experiment, other outdoor experiments are conducted. We find that the sunlight variation affects the performance of the tracking system greatly. If the sunlight is not stable and often varies in real time, the color detection may cannot detect the four markers. Besides, when the sunlight is dazzling, the power of the LEDs need to improve. Otherwise, the color detection cannot detect the markers. The reason for it is that the dazzling sunlight make the whole image darker, the information of the marker decreases and tracking system cannot detect the color.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

A vision tracking system has been presented in this thesis. The performance of this tracking system has been verified in experiments indoors and outdoors. A monocular camera is used to measure the relative distance and angles between a UAV and a camera. The targets can be tracked in time when the UAV flies in steady state. The problem to track the UAV in unstable states will be studied in future.

7.2 Future Work

- (i) In this tracking system, we note that the variation of the marker coordinates affects the measurement result dramatically, even for one pixel change for the marker position. To solve this problem, an effective method is to use more markers to decrease the influence of the fluttering of each marker coordinate. Some advanced algorithms can be proposed to extract the contour of the leader UAV and obtain more points from it.

- (ii) The performance of Kalman filter is not effective, when the leader UAV flies in unstable states, especially in acceleration. A new model for markers, extended Kalman filter (EKF) and an adaptive region of segmented image based on the motion of UAV leader need to be proposed.
- (iii) LEDs circuit in Fig.6 is currently adjusted manually. It cannot adjust the power of LEDs according to ambient lighting condition. A photo sensor will be used in this circuit, achieving automatic adjustment of the LED light intensity according to ambient light condition.

AUTHOR'S PUBLICATIONS

1. Ruibing Jin, and Jianliang Wang, "A Vision Tracking System Via Color Detection," in *IEEE Conference on Control and Automation*, Kathmandu, Nepal, June. 2016(Accepted).

Bibliography

- [1] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a quad-rotor robot,” in *Proceedings Australasian Conference on Robotics and Automation 2006*. Australian Robotics and Automation Association Inc., 2006.
- [2] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, 2007.
- [3] Q. Fan, Z. Yang, T. Fang *et al.*, “Research status of coordinated formation flight control for multi-uavs,” *Acta Aeronaut Astronaut Sin*, vol. 30, pp. 683–691, 2009.
- [4] P. Binetti, K. B. Ariyur, M. Krstic, and F. Bernelli, “Formation flight optimization using extremum seeking feedback,” *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 1, pp. 132–142, 2003.
- [5] D. Galzi and Y. Shtessel, “Uav formations control using high order sliding modes,” in *American Control Conference, 2006*. IEEE, 2006, pp. 6–pp.
- [6] H. Wu and S. Jagannathan, “Adaptive neural network control and wireless sensor network-based localization for uav formation,” in *Control and Automation, 2006. MED’06. 14th Mediterranean Conference on*. IEEE, 2006, pp. 1–6.

- [7] F. Lin, K. Peng, X. Dong, S. Zhao, and B. M. Chen, "Vision-based formation for uavs," in *Control & Automation (ICCA), 11th IEEE International Conference on*. IEEE, 2014, pp. 1375–1380.
- [8] P. E. Howland, "Target tracking using television-based bistatic radar," in *Radar, Sonar and Navigation, IEE Proceedings-*, vol. 146, no. 3. IET, 1999, pp. 166–174.
- [9] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 1, pp. 14–35, 1993.
- [10] F. Lin, X. Dong, B. M. Chen, K.-Y. Lum, and T. H. Lee, "A robust real-time embedded vision system on an unmanned rotorcraft for ground target following," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 2, pp. 1038–1049, 2012.
- [11] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [12] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 651–670, 1996.
- [13] A. Rozantsev, V. Lepetit, and P. Fua, "Flying objects detection from a single moving camera," *arXiv preprint arXiv:1411.7715*, 2014.
- [14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

- [15] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 3, pp. 411–426, 2007.
- [16] A. Bosch, A. Zisserman, and X. Munoz, “Image classification using random forests and ferns,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.
- [17] N. M. Oliver, B. Rosario, and A. P. Pentland, “A bayesian computer vision system for modeling human interactions,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 831–843, 2000.
- [18] N. SeungJong and M. Jeon, “A new framework for background subtraction using multiple cues,” in *Proc. 11th Asian Conf. Comput. Vis*, pp. 493–506.
- [19] B. D. Lucas, T. Kanade *et al.*, “An iterative image registration technique with an application to stereo vision.” in *IJCAI*, vol. 81, 1981, pp. 674–679.
- [20] S. Walk, N. Majer, K. Schindler, and B. Schiele, “New features and insights for pedestrian detection,” in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 1030–1037.
- [21] J. Brand and J. S. Mason, “A comparative assessment of three approaches to pixel-level human skin-detection,” in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 1. IEEE, 2000, pp. 1056–1059.
- [22] J.-C. Terrillon, M. N. Shirazi, H. Fukamachi, and S. Akamatsu, “Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images,” in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000, pp. 54–61.

- [23] M. J. Jones and J. M. Rehg, “Statistical color models with application to skin detection,” *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002.
- [24] C. Poynton, “Frequently asked questions about color,” *Retrieved June*, vol. 19, p. 2004, 1997.
- [25] W. Skarbek, A. Koschan, T. Bericht, Z. Veroffentlichung *et al.*, “Colour image segmentation-a survey,” 1994.
- [26] S. L. Phung, A. Bouzerdoun, and D. Chai, “A novel skin color model in ycbcr color space and its application to human face detection,” in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. I–289.
- [27] D. Kragic, H. I. Christensen *et al.*, “Survey on visual servoing for manipulation,” *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, 2002.
- [28] L. Deng, F. Janabi-Sharifi, and W. J. Wilson, “Stability and robustness of visual servoing methods,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1604–1609.
- [29] T. Heitzmann, C. Doignon, C. Albitar, and P. Graebbling, “Position-based visual servoing using a coded structured light sensor,” in *Robotic and Sensors Environments, 2008. ROSE 2008. International Workshop on*. IEEE, 2008, pp. 126–131.
- [30] J. Cid and F. Reyes, “Visual servoing controller for robot manipulators,” in *Electrical, Communications, and Computers, 2009. CONIELECOMP 2009. International Conference on*. IEEE, 2009, pp. 153–158.

- [31] T. B. Moeslund, A. Hilton, and V. Krüger, “A survey of advances in vision-based human motion capture and analysis,” *Computer vision and image understanding*, vol. 104, no. 2, pp. 90–126, 2006.
- [32] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [33] R. M. Haralick, C.-n. Lee, K. Ottenburg, and M. Nölle, “Analysis and solutions of the three point perspective pose estimation problem,” in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR’91., IEEE Computer Society Conference on.* IEEE, 1991, pp. 592–598.
- [34] R. Horaud, B. Conio, O. Le Boulleux, and L. B. Lacolle, “An analytic solution for the perspective 4-point problem,” in *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR’89., IEEE Computer Society Conference on.* IEEE, 1989, pp. 500–507.
- [35] D. DeMenthon and L. S. Davis, “Exact and approximate solutions of the perspective-three-point problem,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1100–1105, 1992.
- [36] D. G. Lowe, “Fitting parameterized three-dimensional models to images,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 441–450, 1991.
- [37] S. Ganapathy, “Decomposition of transformation matrices for robot vision,” in *Robotics and Automation. Proceedings. 1984 IEEE International Conference on*, vol. 1. IEEE, 1984, pp. 130–139.

- [38] Y. Liu, T. S. Huang, and O. D. Faugeras, “Determination of camera location from 2-d to 3-d line and point correspondences,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 1, pp. 28–37, 1990.
- [39] D. Grest, “Marker-free human motion capture in dynamic cluttered environments from a single view-point,” Ph.D. dissertation, Christian-Albrechts Universität Kiel, 2007.
- [40] S. Roweis, “Levenberg-marquardt optimization,” *Notes, University Of Toronto*, 1996.
- [41] Z. Mahboubi, Z. Kolter, T. Wang, G. Bower, and A. Y. Ng, “Camera based localization for autonomous uav formation flight,” in *Proceedings of the AIAA@Infotech Conference*, 2011.
- [42] “[http : //docs.opencv.org/2.4/](http://docs.opencv.org/2.4/).”
- [43] “[https : //en.wikipedia.org/wiki/gaussian_blur](https://en.wikipedia.org/wiki/gaussian_blur).”
- [44] L. Shapiro and G. Stockman, “Computer vision,” *New Jersey: Prentice Hall*, pp. 137,150, 2001.
- [45] M. S. Nixon and A. S. Aguado, *Feature extraction & image processing*. Academic Press, 2008.
- [46] S. Suzuki *et al.*, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.
- [47] C.-H. Teh and R. T. Chin, “On the detection of dominant points on digital curves,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 8, pp. 859–872, 1989.

- [48] X. R. Li and V. P. Jilkov, “Survey of maneuvering target tracking. part i. dynamic models,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 39, no. 4, pp. 1333–1364, 2003.
- [49] “[https : //en.wikipedia.org/wiki/ubuntu_\(operating_system\)](https://en.wikipedia.org/wiki/ubuntu_(operating_system)).”
- [50] “[https : //en.wikipedia.org/wiki/robot_operating_system](https://en.wikipedia.org/wiki/robot_operating_system).”
- [51] “[https : //en.wikipedia.org/wiki/opencv](https://en.wikipedia.org/wiki/opencv).”
- [52] “[https : //en.wikipedia.org/wiki/thin_lens](https://en.wikipedia.org/wiki/thin_lens).”

Appendix: Hardware System

In this tracking system, the hardware system can be divided into three parts: markers, visual sensor, and hexarotors. A monocular camera is used as a visual sensor to capture the image in front of the follower UAV, which provides the information to determine and track the leader. The tracking system is based on four markers set on the leader and markers are very important to ensure good performance in tracking. Apart from it, the parameters for the hexarotors also have a influence on the quality of image captured by the camera, so it is necessary to investigate them. In this appendix, marker parameters are discussed at first, and then, the monocular camera is introduced. At last, the parameters for hexarotors are given.

Marker Parameters

Four markers are used in this tracking system to enable the camera installed on the follower UAV to recognise the leader UAV. So these markers are very important for the tracking system. In this thesis, markers are chosen from three aspects.

- (i) Marker size. An image is recorded by pixels in a camera, and the number of pixels for each marker has a great influence on marker detection. The number of pixels decreases with the relative distance increasing according to the camera imaging principle. So the size for each marker should be selected carefully. In a UAV formation, the distance between the UAVs should be long

enough to make sure that they can fly safely in high speed and the relative distance is determined around 10 m in this thesis. As a result of it, marker size should be large enough to make the number of pixels enough to be detected on the image.

- (ii) Marker structure. As the vision tracking system is to be used outdoors, the environment factors affect the marker detection largely such as reflection and light intensity. In fact, with the environment changing, the detectable area for each marker changes. In some severe environment, the strong contrast of light and shade even decreases the quality of image and changes the original color of the markers. In Fig.1, a camera is used to detect red targets and the white area in the right window denotes the area which is considered as red by the camera. It is obvious that the performance is pretty good in this ideal environment.

Figure 1: Ideal Image

Figure 2: Reflection on Target

However, reflection cannot be avoided in practice. In Fig.2, there is reflection on the marker and part of white area changed into black in the right window. In this short distance, as the number of pixels is very large, the camera can detect the red target. But when this target moves far away from the camera, the number of pixels decreases dramatically and may be drowned in noise. To solve this problem, the marker size should be determined carefully.

Apart from it, marker detection is very difficult under low light intensity case as well. In Fig.3, as the light intensity decreases, the camera cannot detect the red target.

Figure 3: Low Light Intensity

To solve this problem, an effective method is to add several LEDs on this marker which is shown in Fig.4. With LEDs turned on, the red marker can be detected.

Figure 4: LEDs on The Marker

But the contrast of light and shade is another factor which may damage the target detection performance in Fig.5. So, a simple marker with LEDs cannot fulfill the requirement of the vision tracking system.

Figure 5: LEDs Too bright

To solve this problem, an adjustable LEDs circuit is proposed in Fig.6. In this circuit, four electric capacities are used to filter current noise and R_{SRC} is used to limit the current. Besides, R1 is a rheostat which can adjust the power of LEDs by changing the circuit current and the schottky barrier rectifier is used to limit the current direction and protect LEDs from reverse-direction current damage.

Figure 6: LEDs Circuit

With this method, the power of LEDs can be adjusted according to environ-

ment. A photo sensor can be used to detect the ambient lighting condition and it can be used to control R1, here, achieving automatic adjustment of the LED light intensity according to ambient light condition.

- (iii) Marker weight. The four markers are installed on the leader UAV and UAV motor force and battery capacity are limited, so the weight of markers should be small, which dose not affect the balance and flight performance of the leader UAV.

Considering the three points above, the markers are determined as follows:

Figure 7: Red LED

Figure 8: Yellow LED

The markers parameters are listed as follows:

Table 1: Parameters for Each Marker

Parameter	Value
Weight	8.4g
Diameter	60mm
Thickness	9.5mm
Voltage	12V

Monocular Camera Introduction

A monocular camera is used in this system as a visual sensor. It is well-known that the image is expressed by pixels and generally, the quality of the image improves with the increasing number of the pixels in an image. Besides, in target detection, the number of the pixels plays an important role in distinguishing makers from other noise, so more pixels in a camera is preferred in target detection. However, with the increase of the pixel number, the FPS decreases dramatically and it is necessary to select a suitable resolution for the camera. From research experience and some experiments, a point grey CCD camera named *BFLY-U3-13S2C-CS* shown in Fig.9 is chosen as the visual sensor in this tracking system and the parameters are listed in Table 2:

Figure 9: Camera Sensor

Table 2: Parameters of Camera Sensor

Parameter	Value
Resolution	1288 * 964
Frame Rate	30FPS
Megapixels	1.3MP
Chroma	Color
Sensor Type	CCD
Sensor Format	1/3"
Pixel Size	3.75 μ m
Mount	CS
Mass (g)	36
Dimensions	29mm $\times$ 29mm $\times$ 30mm

Apart from the pixel number, the focal length also has a great influence on camera imaging. From the thin lens formula (7.1) below, it is clear that the focal length f affects the distance from the object to the lens denoted by S_1 [52].

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}, \quad (7.1)$$

where S_1 denotes the distance from the object to the lens, S_2 denotes the distance from the lens to the image, and f denotes the camera focal length, as depicted in Fig.10.

Figure 10: Thin Lens Formula

In this tracking system, the maximum value of S_1 is 10m. To make the camera track UAV at 10m. the focal length for the camera should be selected carefully. In this system, the camera lens is YV2.42.5A-2, as shown in Fig.11 and the parameters are listed in Table 3. If $S_2 \approx 6\text{mm}$ and the diameter of marker is 60mm, the number of pixel for each marker on CCD is 72 in theory.

As the color is used to track the UAV leader, the color of neighbor plants and constructions affect the performance of the color detection. In this tracking system, the area of the objects in the specified color is used to distinguish markers from others. Apparently, if there are some other objects with a larger area in these specified color, the color detection may cannot track the markers. So, to distinguish markers from other objects, the number of the pixel for the markers depends on the application environment. To solve this problem, the Kalman filter is used to predict the location of the markers, which can limit the region of the color detection. This part is discussed in details in Chapter 5. If these other objects with a larger area in these specified color have been filtered out by using Kalman filter, generally, the pixel number for the specified color on the segmented image is around 2 to 10. Only if the number of the marker pixel on CCD is larger than 10, the markers can be detected correctly. However, as the reflection of the sunlight or the shelter affects the number of the marker pixel on CCD, some pixel number redundancy should be kept in advance. The experiments we done in indoors and forest shows that 72

pixels on CCD is enough to distinguish markers from environment in general.

Figure 11: Camera Lens

Table 3: Parameters of Camera Lens

Parameter	Value
Focal Length(mm)	2.5 ~ 6
Iris Range	F1.2 ~ T360 · C
Mount	CS
Mass (g)	30

Hexarotor Parameters

The hexarotor used in this tracking system is a multi-rotor helicopter which is radio controlled and built by the company jDrones&Co. It is provided by *jDrones* whose products are open source and easily developed by researchers. This hexarotor has shown many features such as its good stability and control performance. The application of this hexarotor is not limited to entertainment and navigation, and it is widely used by researchers. This company provides many kinds of hardware and development tools which are beneficial for researchers to add extra hardware or functions on UAVs.

The picture of the hexarotor is shown in Fig.12.

Figure 12: Original jDrone

As this hexarotor is used in this tracking system with the help of markers, several markers and some related hardware are installed on it, as shown in Fig.13.

Figure 13: jDrone with markers

As mentioned above, this hexarotor has lots of features and some of them are very important for our tracking system. The relevant specifications and features that are relevant to our tracking system are listed as follows:

- (i) Weight and Dimensions. As it is shown in Fig.12, many kinds of hardware

packages are available and can be used depending on the research requirement. The diameter and height of it are 657 mm and 302 mm respectively. The total weight for it with GPS module, four markers and a battery is 1.828 kg.

- (ii) Motors. jDrone is equipped with 6 ArduCopter specific motors which drive the 6 propellers respectively, for heavier payloads. The parameters of them are listed in Table 4.

Table 4: Parameters of Motor

Parameter	Value
Size	$28 \times 36\text{mm}$
Shaft	4mm
RPM/V	880Kv
Weight	72g
Cable	35cm/18 AWG
Batt	2-4 S LiPo

Table 5: Power output from ESC, 3S LiPo, 12×45 Propeller

Parameter	25%	50%	75%	100%
Amp	1.5A	6.1A	14.2A	20A
Wattage	17.5W	70W	160W	210W
Thrust	230g	650g	1290g	1380g

- (iii) Battery. The battery is made of three elements LiPo rechargeable battery with a total of 5000mAh. The weight of each battery is 380g.

- (iv) Control Module. Hexarotor is equipped with a control module called 3DR Pixhawk shown in Fig.14. Pixhawk is an advanced autopilot system developed by the PX4 open-hardware project and manufactured by 3D Robotics. The

advanced processor and sensor technology in it is made by ST Microelectronics. A NuttX real-time operating system is installed in it to realise incredible performance, flexibility, and reliability for autonomous vehicle.

Figure 14: 3DR Pixhawk

In this module, it has a microprocessor, 32-bit STM32F427 Cortex M4 core with a 256KB FPU in 168MHz, 2MB RAM Flash and a 32 bit STM32F103 failsafe co-processor. For sensors, the ST Micro L3GD20 3-axis 16-bit gyroscope, ST Micro LSM303D 3-axis 14-bit accelerometer, Invensense MPU 6000 3-axis accelerometer and MEAS MS5611 barometer are equipped on it, which make it easier for researchers to obtain the data of motion state for UAVs.