An Active Vision System for Tracking and Mosaicking on UAV

LIN, Kai Wun

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong
September 2011


THE CHINESE UNIVERSITY OF HONG KONG
23 JAN 2013

Thesis Assessment Committee

Professor Chung, Chi-kit (Chair)
Professor Liu, Yun-hui (Thesis Supervisor)
Professor Liao, Wei-Hsin (Committee Member)
Professor Wang, Zhidong (External Examiner)

Abstract

Unmanned aerial vehicles (UAVs) are regarded as excellent platforms for performing tasks to mention but not limited to wide regional surveillance, environmental monitoring, ground target tracking and aiding search and rescue operations. Apart from the flight system for the UAV itself, very often these tasks required a high performance vision processing system in order to succeed. Specifications for vision system vary with the size of UAVs being deployed and tasks want to achieve. This thesis is to address the necessity of a low power, light weight and vibration-proof vision system for use on small, hand-launch unmanned aerial vehicles as a tracking and surveillance device.

Thank to recent advance on chip technology in producing heterogeneous multi-core processor (integration of GPP, DSP and GPU on a single chip) for mobile and tablet-PC, there is a great increase on image and video processing performance on these devices as the intense maths operations are offloaded to DSP and GPU. In that sense, it makes them being potential platforms for carrying out vision tasks on small UAVs which have limited power resources and affordable payload. Here, we have developed an active vision system that based on such a high performance and low power requirement heterogeneous multi-core processor appearing in cell phone for executing vision algorithms in real-time. The low power CMOS camera module of the vision system features an electronic global shutter that completely eliminates image shearing arose under severe engine or motor vibration on aerial platforms. We have also developed a view-stabilizing mechanism with use of miniature pan-tilt actuators to solve for the rapid camera view change caused by variation in flying attitude of aircraft. Besides, the performance of the system has been verified through applications with on-board object tracking and real-time wireless image delivering for mosaicking on a ground laptop PC. A PID controller has also been implemented on the system for centering tracked target to the middle of camera's field of view using on-board visual feedback.

摘要

無人飛機系統(無人機)在執行一些特定任務上，如廣泛的區域監測，環境監測，地面目標跟踪，以及協助搜索和救援行動等，常被視為很優秀的平台。除了無人機本身的飛行系統以外，很多時候，這些任務需要一個高性能的視覺處理系統協助才能取得成功。基於所應用無人機的大小和任務要求達到的目標不同，對視覺系統規格的需求也會有不一樣。這篇論文的目標在於研發一套能够應用於小型及手發式無人機上作為跟踪和監測的視覺系統，同時能够符合低功耗，輕重量以及防震的要求。

由於近年芯片的生產技術已能夠製造出異構多核處理器(把GPP，DSP和GPU整合在一塊芯片上)，並且應用到智能電話及平板電腦上。因為可以把繁複的數學運算移到DSP和GPU上操作，使這些設備在圖像和視頻處理性能上有了很顯著的提升。同時這也使這些設備成為開展無人機視覺任務十分具潛力的平台，並能夠符合小型無人機低功耗和輕重量的要求。現在我們開發的主動視覺系統正使用了跟智能電話一樣的一個高性能，低功耗的異構多核處理器，用來實時地執行視覺算法。視覺系統上的低功耗CMOS攝像頭模塊具有全局電子快門可以完全避免圖像在飛行載具發動機或電動機振動下產生扭曲或變形。我們也透過微型驅動器開發了一個視像穩定雲台來解決飛機飛行姿態改變時導致相機視像產生快速的變化。此外，我們還通過直接在嵌入式系統上進行物體跟踪和通過無線網絡傳輸圖像到地面上的筆記本電腦進行實時圖像拼接來驗證視覺系統的性能。一個PID控制器也應用到視覺系統的目標跟踪上，通過視覺反饋把目標保持在圖像的中間。

Acknowledgement

I would like to show my gratitude to my supervisor, Prof. Liu Yun-hui, for his support and guidance on my study in pursuing a master degree in the Department of Mechanical and Automation Engineering. During the time of research in the Networked Sensors and Robotics Laboratory, I was fortunate to work with a group of people on the unmanned aerial vehicle project. Especially, Mr. Nick Lau, who always provided me suggestions towards the problems that I had met. I had also been helped by Mr. Allan Mok, the electronic officer of the department, for his assistance in soldering process of the printed circuit boards. Moreover, I would like to thank Mr. Nick Lau again and Mr. Chong in the Hong Kong Model Engineering Club for helping me to perform flight experiments with the developed vision system. Without all their help, my thesis would not have been made possible. Lastly, I really thank my family for supporting me throughout my study in the Chinese University of Hong Kong and giving me encouragement during my thesis writing period.

Contents

Abstract i

Acknowledgement iii

1 Introduction 1
- 1.1 Overview of the UAV Project . . . 1
- 1.2 Challenges on Vision System for UAV . . . 2
- 1.3 Contributions of this Work . . . 4
- 1.4 Organization of Thesis . . . 6

2 Image Sensor Selection and Evaluation 8
- 2.1 Image Sensor Overview . . . 8
  - 2.1.1 Comparing Sensor Features and Performance . . . 9
  - 2.1.2 Rolling Shutter vs. Global Shutter . . . 10
- 2.2 Sensor Evaluation through USB Peripheral . . . 11
  - 2.2.1 Interfacing Image Sensor and USB Controller . . . 12
  - 2.2.2 Image Sensor Configuration . . . 14
- 2.3 Image Data Transmitting and Processing . . . 17
  - 2.3.1 Data Transfer Mode and Buffering on USB Controller . . . 18
  - 2.3.2 Demosaicking of Bayer Image Data . . . 20
- 2.4 Splitting Images and Exposure Problem . . . 22
  - 2.4.1 Buffer Overflow on USB Controller . . . 22
  - 2.4.2 Image Luminance and Exposure Adjustment . . . 24

3 Embedded System for Vision Processing 26
- 3.1 Overview of the Embedded System . . . 26
  - 3.1.1 TI OMAP3530 Processor . . . 27
  - 3.1.2 Gumstix Overo Fire Computer-on-Module . . . 27
- 3.2 Interfacing Camera Module to the Embedded System . . . 28
  - 3.2.1 Image Signal Processing Subsystem . . . 29
  - 3.2.2 Camera Module Adapting Board . . . 30
  - 3.2.3 Image Sensor Driver and Program Development . . . 31
- 3.3 View-stabilizing Biaxial Camera Platform . . . 34
  - 3.3.1 The New Camera System . . . 35

3.3.2 View-stabilizing Pan-tilt Platform 41
3.4 Overall System Architecture and UAV Integration 46

4 Target Tracking and Geo-locating **50**
4.1 Camera Calibration 51
4.1.1 The Perspective Camera Model 51
4.1.2 Camera Lens Distortions 53
4.1.3 Calibration Toolbox and Results 54
4.2 Selection of Object Features and Trackers 56
4.2.1 Harris Corner Detection 58
4.2.2 Color Histogram 59
4.2.3 KLT and Mean-shift Tracker 59
4.3 Target Auto-centering 64
4.3.1 Formulation of the PID Controller 65
4.3.2 Control Gain Settings and Tuning 69
4.4 Geo-locating of Tracked Target 69
4.4.1 Coordinate Frame Transformation 70
4.4.2 Depth Estimation and Target Locating 74
4.5 Results and Discussion 77

5 Real-time Aerial Mosaic Building **89**
5.1 Motion Model Selection 90
5.1.1 Planar Perspective Motion Model 90
5.2 Feature-based Image Alignment 91
5.2.1 Image Preprocessing 91
5.2.2 Feature Extraction and Matching 92
5.2.3 Image Alignment using RANSAC Algorithm 94
5.3 Image Composition 95
5.3.1 Image Blending with Distance Map 96
5.3.2 Overall Stitching Process 98
5.4 Mosaic Simulation using Google Earth 99
5.5 Results and Discussion 100

6 Conclusion and Further Work **108**

A System Schematics **111**

B Image Sensor Sensitivity **118**

Bibliography **120**

List of Figures

1.1 Unmanned helicopter governing by hierarchical PD controller . . 2
1.2 Images captured by USB web camera on the gasoline helicopter 3
1.3 Gumstix Overo Computer-on-Module (CoM) 4

2.1 Rolling shutter effects of CMOS image sensor 10
2.2 CMOS camera module and image acquisition board 12
2.3 Connections between camera module and image acquisition board 13
2.4 A complete data transfer on I^2C bus lines 14
2.5 Time diagram for start condition on imager serial bus 15
2.6 State transition of the read and write operations on the serial bus interface 16
2.7 Windowing capability of the image sensor 16
2.8 Skip and bin mode of the image sensor 17
2.9 Available endpoint configurations on the USB micro-controller . 18
2.10 Synchronization signals from image sensor 19
2.11 Time diagram for slave FIFO synchronous write on the USB micro-controller 19
2.12 RGB Bayer pattern of the image sensor 20
2.13 Bilinear interpolation for Bayer image 21
2.14 Splitting image obtained after interpolation 22
2.15 Observation of signal pins on oscilloscope 22
2.16 Frame rate achieved with image acquisition board and program 23
2.17 Observation of signal pins with image sensor configured to 76fps 24
2.18 Image exposure under different ADC gains 25

3.1 System architecture of OMAP3530 processor 27
3.2 Gumstix Overo CoM and its expansion board 28
3.3 Camera interface subsystem (ISP) of OMAP3530 processor . . . 29
3.4 Camera module adapting board for Gumstix Overo CoM 30
3.5 Logo of the OpenEmbedded project 31
3.6 System setup for image capturing on CoM 32
3.7 High resolution images captured on CoM 33
3.8 Low resolution image captured on CoM 33
3.9 The new MT9V032 camera module 35

3.10 Interfacing module of new camera system 37
3.11 MCU module of new camera system 37
3.12 Stacking configurations of new camera system 38
3.13 Imaging performance of new image sensor 39
3.14 Setup and images captured for evaluating shutter performance . 40
3.15 Pan-tilt kits from Lynxmotion 41
3.16 Customized micro pan-tilt kit 42
3.17 CAD drawings of pan-tilt platform for foam airplane 43
3.18 Pan-tilt platform installed on foam airplane 43
3.19 Servo hacking for angular position feedback 44
3.20 Autopilot on pan-tilt platform for view-stabilization 45
3.21 Block diagram of the overall system 46
3.22 System boards of the embedded platform 47
3.23 Vision system on large fix-wing airplane 48
3.24 Vision system on hand-launch foam fix-wing airplane 48

4.1 Frontal perspective image projection 52
4.2 GUI of the camera calibration toolbox for MATLAB 54
4.3 Corner extraction process for camera calibration 55
4.4 Extrinsic parameters of calibrated images 55
4.5 Optical flow over Gaussian pyramid of image 62
4.6 Coordinate assignment for the pan-tilt system 65
4.7 Coordinate frames assigned to experimental setup 72
4.8 Projection of a point in camera frame to image frame 75
4.9 Vector relations for depth estimation 76
4.10 Experiment on object tracking using KLT feature tracker 77
4.11 Experiment on target tracking and auto-centering using mean-shift tracker 78
4.12 Centering error for a color blob with uniform motion 79
4.13 Experiment record on the centering error, position commanded and the control latency of the tracking system for tracking free moving color box 81
4.14 Dummy used in search and rescue challenge 82
4.15 Setup for detecting IR point source in indoor 82
4.16 Image thresholding and bounding external contour in the illuminating point detection process 83
4.17 Place for putting the IR illuminator and position measurement with GPS 84
4.18 Estimated position of the IR illuminator (Data set 1) 86
4.19 Estimated position of the IR illuminator (Data set 2) 87

5.1 Lens undistortion on perceived aerial image 92
5.2 SURF feature extraction and feature correspondence 93

5.3 Blending of aligned images by simple averaging 95
5.4 Distance map that used for image blending 96
5.5 Image blending by weighted average using distance map 97
5.6 Flow chart for aerial image stitching process 98
5.7 Capturing aerial image from Google Earth 99
5.8 Mosaic created using aerial image from Google Earth 100
5.9 Aerial mosaic of the airfield in Yuen Long 102
5.10 Aerial mosaic over United College in CUHK campus 103
5.11 Aerial mosaic near the coast of TKO industrial area 105
5.12 Aerial mosaic over flat land roads in TKO industrial area 106

A.1 Schematic of the USB image acquisition board 112
A.2 Schematic of camera module adapting board for Gumstix CoM . 113
A.3 Schematic of the new MT9V032 camera module 114
A.4 Schematic of the interfacing module for new camera system . . . 115
A.5 Schematic of the MCU module for new camera system 116
A.6 Schematic of the SPI-to-UART serial expansion board 117

B.1 Quantum efficiency of MT9T001 image sensor 119
B.2 Quantum efficiency of MT9V032 image sensor 119

List of Tables

2.1 Comparing CCD and CMOS image sensor 9

3.1 Video streaming resolution and frame rate 34
3.2 Comparing MT9T001 and MT9V032 CMOS image sensor . . . 36
3.3 Specifications and features of pan-tilt kit used 42

4.1 Calibration results using toolbox for MATLAB 56
4.2 Calibration results using functions in OpenCV library 56
4.3 Gain settings for the angular displacement PID controller 69
4.4 Target centering error on the tracking systems 79

Chapter 1

Introduction

In recent decades, unmanned aerial vehicle platforms have been widely adopted in the military for performing reconnaissance and target acquisition missions. With the introduction of the technology used on these systems to the commercial market and the reduction of production cost, UAVs start to gain popularity to be used in civilian applications such as border inspection, search and rescue, disaster and emergency management, etc.[69]. The system design, search algorithms and flight path planning implementing on UAVs for carrying out those tasks have also become hot topics in the research field [31, 42]. Among all those applications mentioned above, an image acquisition and processing system or at least a camera for video downstreaming should be available on the UAV system for performing the desired tasks. Being a key component for the success of missions, these vision systems usually impose rigorous specifications on processing speed, power consumption and overall system weight.

This thesis aims at developing an embedded vision system which meets the specifications required by the aerial platforms developed in the Networked Sensors and Robotics Laboratory (NSRL) for performing tasks like object tracking, path following, indoor navigation and reconnaissance in outdoor environment.

1.1 Overview of the UAV Project

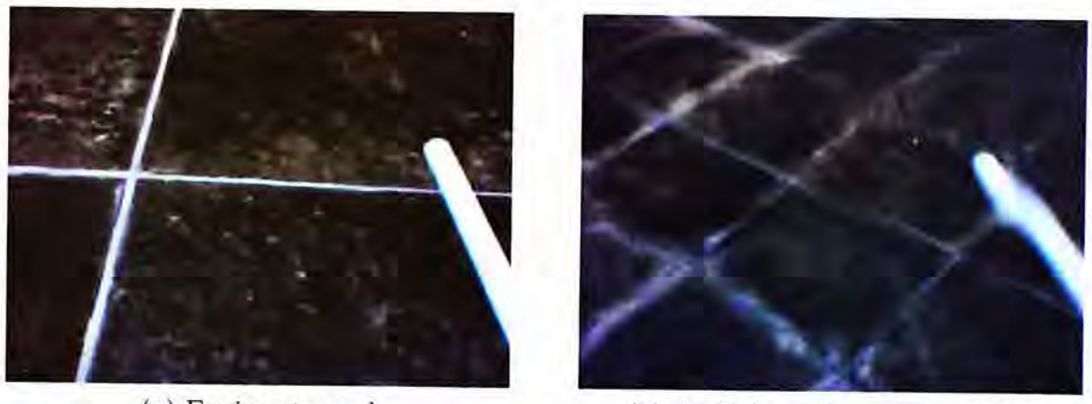
The UAV project at NSRL was started in 2007, with the motivation of developing small autonomous helicopters for vision-based control, tracking and surveillance applications. Early stage of the project mainly concentrated on

Figure 1.1: Unmanned helicopter governing by hierarchical PD controller

the flight system development for acquiring sensors data, controlling helicopter rotor through RC servos and handling power and system fail-safe issues. Later on, we managed to perform hovering of a gasoline miniature helicopter by a hierarchical PD controller implemented on the flight system with the use of sensor data from GPS and IMU (Inertial Measurement Unit) [48]. Recently, we also got succeed in controlling a coaxial mini-helicopter for path following in an indoor environment aiding by a motion caption system. Apart from Vertical Take-Off and Landing (VTOL) aerial vehicles, we also extend our aerial platforms to small fix-wing aircrafts (including both the hand-launch ones and those required a runway for take-off and landing) with specific interest on their capability of long rang flight. This property makes them being excellent platforms for carrying out reconnaissance or search and rescue missions in wide area of coverage.

1.2 Challenges on Vision System for UAV

In the mean time of setting up flight system for the UAV, we had also performed tests on obtaining images from general USB web camera mounting on the gasoline helicopter with a net-book PC. It is found that the severe vibration induced by the single-stroke engine on the helicopter leads to highly distorted images (Figure 1.2b) which would definitely considered unacceptable for use with vision algorithms that rely on clear and stable image features. Therefore, suitable vibration isolation mechanism and a capable image acquisition device are essential to the vision system of UAVs for performing tracking and



(a) Engine stopped (b) At flight with engine running

Figure 1.2: Images captured by USB web camera on the gasoline helicopter

surveillance tasks. Besides the vibration and image acquisition problems, power consumption and system weight are also important factors that influence the flight endurance and applicability of the UAV system. For small unmanned aerial platforms, it is very critical to ensure all avionics including the flight and vision system are within the allowable payload such that they would not pose significant reduction on the maneuver ability of the aerial platforms. Hence, a vision system possesses high processing performance with low power consumption and light weight is always favorable for small airframe vehicles similar to the coaxial mini-helicopter and the EPS (Expanded Polystyrene) foam fix-wing airplane in our laboratory.

The advance in nowadays mobile phone technology gives us clues on the way we could go. You may probably experienced that newer generation high-end cell phones get exclusive processing power more comparable to a laptop PC, especially in media playback and gaming performance. This is contributed by the new chip manufacturing technology that put different processing cores such as GPP, DSP and GPU together on a single die resulting in a heterogeneous multi-core architecture [47, 67]. Therefore, the computational intensive image and video processing can be offloaded to these high performance DSP or GPU, resulting in rather more smooth video playback and gaming experiences on these mobile devices. Recent researches show it is possible to achieve real-time object recognition and target tracking at interactive frame rates based on invariant image features using mobile phones [32, 81, 23, 82], though it requires modification on the feature descriptors and tracking algorithms used.

Figure 1.3: Gumstix Overo Computer-on-Module (CoM)

With some searching on the specifications of mobile phones and the Internet, we find an embedded system development platform (named as Gumstix) that uses an exact processing unit appearing in mobile phone. Some UAV projects [58, 63] have tried to use this system for indoor navigation with their mini aerial platforms and obtained quite impressive results.

When looking back to our UAV project, since we get multiple aerial platforms from small to mini VTOL vehicles and to hand-launch fix-wing foam aircrafts, it would greatly reduce the system development effort have to spend if we can build a modularized vision system that applicable on all those UAV platforms for handle tacking and surveillance tasks. With the new tiny heterogeneous multi-processor Computer-on-Module (Figure 1.3), we can meet the weight and power consumption requirements on our mini-helicopter and extend the vision system with support of view-stabilizing mechanism on platforms with larger airframe.

1.3 Contributions of this Work

Starting with the development of our own camera module to the end application of the camera module with the embedded system for object tracking and building real-time aerial mosaic, the contribution of this work can be summarized as follows:

1. We manage to develop a real-time vision system for small unmanned aerial vehicles under limited power resources, strict payload requirement and severe vibration induced by gasoline engine or electric motor on

the aerial platforms. The low power CMOS camera module we developed features an electronic global shutter that capable of perceiving shear-free image under severe vibration. Through interfacing with a high performance, low power consumption and light weight Computer-on-Module, we are able to operate high-level vision algorithms on the vision system in real-time and satisfy the payload requirement of small UAVs.

2. A view-stabilizing mechanism is developed to solve rapid view change problem arises due to variation in flying attitude of aerial platform. With the use of control output from an off-the-shelf autopilot that integrates inertial sensors such as accelerometers and rate gyros, and a two-axis pan-tilt platform actuated by RC (remote-controlled) servos, we can maintain the camera module being mounted to the platform in a downward looking orientation regardless the rolling or pitching motion of the aerial vehicles. This especially enhances the view stability for surveillance on the fix-wing aircraft and produces better results of aerial mosaic in experiments.

3. The processing capability of the vision system is evaluated through example applications with on-board object tracking and on-board image compression and transmission for real-time mosaicking on a ground laptop PC. An angular position PID controller is implemented with the pan-tilt platform for locating the tracked target to the center of camera's field of view based on the mean-shift color tracker and a self-designed IR (infra-red) point tracker. Image stitching algorithms are used to composite the received aerial images from the flying aerial vehicle to small aerial maps on the ground station in real-time through a Wi-Fi Ad-hoc network established between the embedded system and the laptop PC.

Comparing to similar low cost embedded vision system developed in [66, 65], our vision system has higher computational capability that possible for handling high-level vision task based on invariant point features. Our system also showed better performance on response and speed for on-board color blob tracking in contrast to similar use of the system mentioned above. Consider the application of vision system for real-time aerial mosaicking, in commercial aerial system like [12], inertial information of UAV and camera orientation were used for building aerial map on a ground station through analog video

transmission. The approach used requires accurate inertial data for good image alignment and noisy images may also be obtained due to analog video transmission. While in our system, image stitching algorithms are used for building aerial mosaic using digital images perceived with the vision system and transmitted to ground through Wi-Fi connection such that clearer images are obtained and more accurate image alignment can be achieved for building good quality aerial map.

1.4 Organization of Thesis

This thesis is organized into six chapters. First half of the chapters detail the hardware and software development for the camera modules and the embedded system, while second half of the chapters discuss the vision algorithms used and how they are applied to the embedded system and a ground laptop PC for performing object tracking and real-time aerial mosaicking. Followed by this introductory chapter,

- In Chapter 2, we show how we develop and evaluate our own image capturing device with a rapid image acquisition platform. First, we compare the properties of the two general types (CCD and CMOS) of image sensors. Then, an image acquisition board based on a USB micro-controller is introduced for obtaining image data from a self-developed CMOS image sensor module and transmitting them to a desktop PC for image recovery, on screen display and sensor performance evaluation. Details on the image retrieval, recovering missing pixel colors from the Bayer output of image sensor and problems we met are discussed in later sections of this chapter.

- In Chapter 3, we deal with the system integration for our developed camera module to the embedded CoM platform and other flight control system on our UAVs. An overview on the features and specifications of the selected embedded system platform is given at beginning of the chapter. Then, we show how the camera module is interfaced to the image signal peripheral of the CoM system through selectable parallel or serial connection. The device driver structure and procedures for software development on the embedded system are mentioned next. Finally, the

design and view-stabilizing mechanism of the biaxial pan-tilt platform together with the overall vision system and UAV platform integration are discussed.

- In Chapter 4, we implement object tracking and target geo-locating algorithms to the embedded system and establish experiments to evaluate the system performance. First, we perform camera calibration and eliminate the lens distortion. Then, the mean-shift color tracker and a self-designed IR point detector are used for tracking desired target. The tracked target is also moved to the center of camera's field of view by the two-axis pan-tilt platform governing by an angular position PID controller. Finally, we can compute the geographical location of the tracked target with position and attitude information from GPS and inertial sensors attaching on the aerial platform.

- In Chapter 5, we extend our UAV system to surveillance application through real-time image transmission with the embedded vision system and produce on-the-fly aerial mosaics on a ground laptop PC. First, we choose a 2D motion model and a planar surface for image composition. Then, we pre-process the images to remove the lens distortion. A feature based image alignment approach using RANSAC is adopted. Finally, the aligned images are composited together with image feathering using a distance map. Simulation and outdoor experiments performed are shown and discussed at the end of the chapter.

- In Chapter 6, we summarize the work of this thesis that contributes to the UAV project and introduce further development on the vision system and our UAV platforms can be used for tracking and surveillance applications.

The appendixes at the end of the thesis show the schematics for developing the camera module, image acquisition board, camera interfacing boards, expansion and micro-controller modules. Details regarding the sensitivity of image sensors we used to different light bands (visible light and near infra-red) are also presented here.

□ End of chapter.

Chapter 2

Image Sensor Selection and Evaluation

In this chapter, we show how we develop our own camera modules from bare image sensors with the intention to solve image acquisition problem under severe vibration on UAVs. By studying the properties of the two types (CCD and CMOS) of image sensor, we know which sensor can meet the requirement of the vision system for our UAV platforms. We have also developed an image acquisition board for the camera module using a high speed USB micro-controller. With the USB controller, we can rapidly obtain image data from sensor and recover it on a desktop PC through process such as bilinear interpolation. That allows us to evaluate the sensor image quality and decide whether it would be applicable to our vision system.

2.1 Image Sensor Overview

CCD (Charge Coupled Device) and CMOS (Complementary Metal Oxide Semiconductor) image sensors are the two devices commonly used nowadays to convert light photons from scene to electric charges and digitally recorded it as an image. The two sensors rely on different technology to acquire light photons, convert them to digital signals and involve different fabrication processes. Though they may possess close image capturing quality, however, they still get quite different properties regarding responsivity, dynamic range, shuttering, speed, etc..

2.1.1 Comparing Sensor Features and Performance

In CCD image sensor, the light photons are detected by photo-detectors and converted to electric charges inside the sensor chip. The accumulated electric charges are shifted across the chip and produce a voltage during the read out process. The analog voltages are then converted to digital output by an off-chip circuit. While in CMOS image sensor, each pixel on the sensor chip consists a circuitry to directly convert the electric charge to a voltage. Amplifier and digital converter are also presented to produce digital output immediately at each sensor pixel.

Due to possibility for including additional circuitry to the CMOS sensor, it is less complex for developing a CMOS camera modules as most functions (e.g. timing and analog to digital conversion) are generally integrated to the chip while extra circuitries have to be added to CCD camera's print circuit board to realize these functions. Moreover, CMOS image sensors possess special functionality in arbitrary regional readout (windowing) that CCD imagers lack. The windowing function can alter the sensor to produce a much higher frame rate than a usual full frame readout and can be applied to high-temporal-precision object tracking in subregion of an image. However, at the same time, the addition circuitry reduces the number of photo-detecting elements can be filled on the sensor space. So, usually a CCD imager performs better (lower image noise) than a CMOS imager in low light environment.

In Table 2.1, we show a comparison on some general features and different

Features/Performance	CCD	CMOS
Signal out of chip	analog voltage	digital bits
Fill factor	high	moderate
System noise	low	moderate
Sensor complexity	low	high
Power consumption	high	low
Responsivity	moderate	slightly better
Dynamic range	high	moderate
Uniformity	high	low to moderate
Shuttering	uniform	non-uniform to uniform
Speed	moderate to high	higher
Windowing	limited	extensive

Table 2.1: Comparing CCD and CMOS image sensor, from [2]

performance properties among the two types of image sensor. For more details regarding the internal structure of the imagers, how they operate, and more in depth comparison can refer to [51, 60, 47].

2.1.2 Rolling Shutter vs. Global Shutter

For the features we compare in previous subsection, we find that CCD and CMOS image sensors possess different ability in exposure (i.e. shuttering). CCD imagers are inherently build to have the ability for exposing all the sensing elements uniformly. This is known as global shutter. While in CMOS imagers, in order to maintain the fill factor (ratio of pixel space for photon acquisition to functional circuitry)[47], usually exposure takes place in a line-by-line manner for the line-scan type of imagers and is therefore called rolling shutter. As the exposure does not take place at the same time, these type of CMOS imagers suffer from image skew (Figure 2.1a) and resulted in partial exposure (Figure 2.1b). They are also called the rolling shutter effects [50]. The distorted images we captured on a gasoline helicopter with a web camera in Section 1.2 are also suffered from this effect.

(a) Skew of a moving car (b) Partial exposure at lightning

Figure 2.1: Rolling shutter effects of CMOS image sensor, from [15]

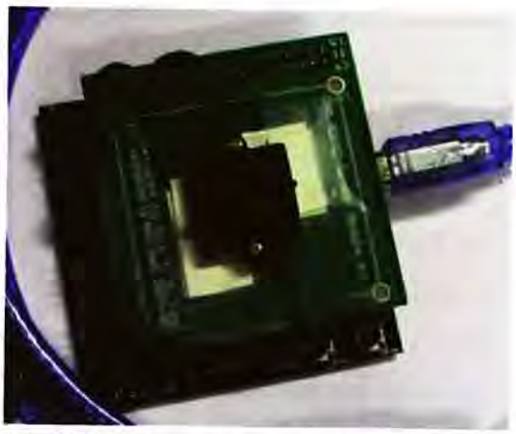
There are research work to compensate rolling shutter effect or even make use of rolling shutter distortions for object pose and velocity estimation through image processing [50, 22]. However, the algorithms are not applicable to all situations, for example, with small motions or multiple moving objects in scene. Further computational power is also used for correcting the distortions

and thus reduced the real-time capability of the vision processing system. In order to completely eliminate or reduce the rolling shutter effects, it is better resorting to the sensor itself. The image sensor should have a higher line-scan rate to increase the uniformity of exposure, or in other words, have a higher frame rate in terms of image frame output. Apart from the line-scan type imager, another type of CMOS image sensor is the area-scan imagers. They offer uniform exposure as the CCD imagers do but with the expense of reducing the fill factor, as shuttering elements have to be placed at pixel's sensing region [51] and thus resulting in smaller image resolution.

Considering the power consumption, image lag, windowing ability and camera complexity, CMOS imager offering global shutter appears to be a more suitable candidate for being the image capture device of our vision system to handle outdoor object tracking and surveillance tasks. At the time we started developing our own camera module, we were not able to find a CMOS image sensor that provides global shutter from the two leading sensor producers - OmniVision and Aptina/Micron. However, we found an Aptina/Micron CMOS imager (MT9T001P12STC) that is programmable up to 93fps with windowing control for VGA-format (640×480). The high image frame rate of the imager should greatly reduce or eliminate image distortion appears under severe engine vibration on the small gasoline helicopter. Therefore, we developed our first camera module based on this image sensor. (We are now using a CMOS image sensor featuring global shutter from Aptina/Micron for the embedded vision system we developed. More details regarding the imager will be mentioned in Chapter 3.)

2.2 Sensor Evaluation through USB Peripheral

Since we have no previous experience in developing a camera module and handling system integration for peripheral device to an embedded system, it is better for us to start evaluating the functionalities and image quality of the camera module we developed on a desktop PC first before we move to the embedded world. In order to evaluate the sensor image quality, an



(a) CMOS image sensor module attaching to USB micro-controller module

(b) Front side of USB micro-controller module

Figure 2.2: CMOS camera module and image acquisition board

image acquisition board is developed and introduced in the following section for sensor configuration and image data readout.

2.2.1 Interfacing Image Sensor and USB Controller

First of all, we have an overview on the camera module and the USB controller module we developed for image acquisition.

CMOS Image Sensor Module

The CMOS image sensor module provides additional circuitry and header pins to the image sensor for stabilizing the voltage supply, setting the serial bus logic level and connecting to external readout board. The imager we used is MT9T001P12STC from Aptina/Micron Imaging [9]. It is a 3-megapixel image sensor with the imager size of $6.55mm \times 4.92mm$. The full resolution of the image sensor is 2048×1536. The image data output from the sensor are in RGB Bayer pattern. It uses the electronic rolling shutter with a maximum master clock of 48MHz. The frame rate can be programmed through its serial interface up to 93fps for VGA-format (640×480). And the ADC resolution of the image sensor is in 10-bit while the voltage supply is 3.3V. (*The MT9T001 image sensor module is designed by Mr. T.K. Lau in NSRL*)

USB2.0 Micro-controller Module

The USB micro-controller module is used to acquire image data from the image sensor module and transmit them to the host PC through the USB interface. The USB peripheral controller we used is CY7C68013A produced by Cypress Semiconductor [4]. It consists of a USB2.0 transceiver, a serial interface engine (SIE) which handles the USB communication protocol, and an enhanced 8051 micro-controller. There are 8-bit or 16-bit programmable peripheral interface on the micro-controller chip. It also has 16 KBytes on-chip code/data RAM and 4 KBytes endpoint buffer. The USB micro-controller provides a general programmable interface (GPIF) and master/slave FIFO for connecting peripheral devices. It can operate under 48MHz/24MHz/12MHz clock. Programs can be loaded to the micro-controller from external EEPROM. The micro-controller also provides 2 USARTS, 3 counters/timers, and an integrated I^2C controller. The operating voltage of the micro-controller is also 3.3V.

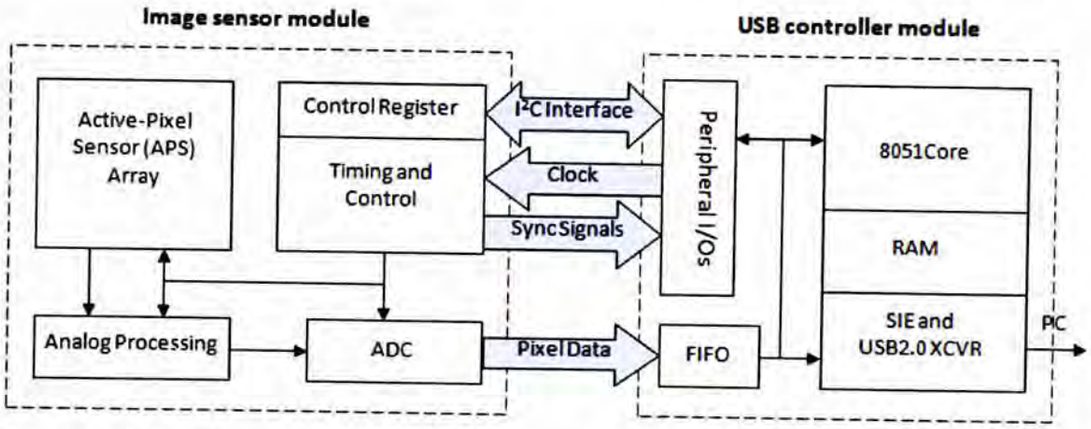


Figure 2.3: Connections between camera module and image acquisition board

Interconnections between the Modules

Since the external FIFO interface of the USB micro-controller can either be configured to 8-bit or 16-bit data width, so only the upper 8-bit image data from the digital outputs of ADC on the image sensor are used. This can optimize the buffer usage for data readout but with the expense in reducing the pixel color resolution from 10-bit to 8-bit. The 8-bit data lines of the

image sensor are connected to the FIFO buffer of the USB micro-controller. The interface clock (IFCLK) pin on the micro-controller is connected to the image sensor as the driving clock signal while a LINE_VALID signal is given by the image sensor to the micro-controller for synchronizing the image data readout. There are also I/O pins to interface the image sensor with the USB micro-controller. They provide signal paths for configuring the image sensor by the USB micro-controller. Image data can be read out from endpoint buffers on the USB micro-controller through the connections to the host PC.

2.2.2 Image Sensor Configuration

In order to configure the CMOS image sensor, we have to be able to communicate with the image sensor through its communication interface. The communication interface is very similar to the I^2C protocol. Therefore, a brief introduction to the I^2C protocol will be given in the following section before we show how the image sensor is configured through its own serial interface.

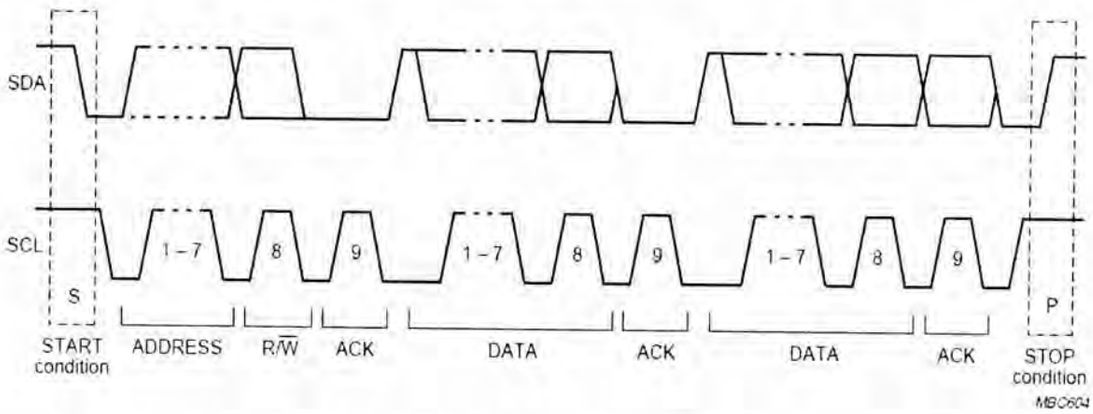


Figure 2.4: A complete data transfer on I^2C bus lines, from [17]

The I^2C Protocol

The I^2C protocol [17] requires only 2 bus lines. One is the serial data line (SDA) and the other is serial clock line (SCL). Each device connected with the data lines is addressable by their unique addresses. A master-slave relationship exists between the device which sends the data and the device which receives the data. The data transfer is limited to 8-bit data series. And the data rate

is 100 KBit/s for the standard-mode while it can reach 400 KBit/s for the fast-mode.

In Figure 2.4, data are transferred on the bus lines using the I^2C protocol. The data signal begins with a START condition (a high to low transition on the SDA line while SCL is high) followed by the ADDRESS of the target device. The R/$\bar{W}$ bit indicates the communication is a data read/write with the device. An ACK bit is received by the master when the communication is successful. The data for reading/writing will then be transferred with the ACK bit follow at the back in indicating whether the transfer succeed or not. A STOP condition (a low to high transition on the SDA line while SCL is high) is transferred when the data reading/writing ends.

Serial Bus Interface on Image Sensor

The serial bus interface on the COMS image sensor provides signal paths for which we can read register values from the image sensor or write new register values to the image sensor for configuration purpose. The serial bus interface is identical to I^2C except that it requires the SCL line to keep low for certain master clock cycles before it generates the START condition, see Figure 2.5.

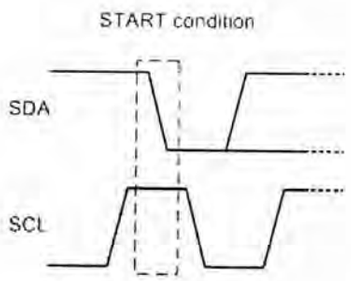


Figure 2.5: Time diagram for start condition on imager serial bus

Due to the difference in start condition, we can not use the integrated I^2C controller on the USB micro-controller for communicating with the image sensor directly. Therefore, we have to implement our own controller for handling the communication. A finite state machine (Figure 2.6) is designed and executing on the I/Os of the USB micro-controller for handling state transitions required by the read and write operation on the serial bus interface of the image sensor. And through the serial bus interface, it is feasible for us to access the registers on the CMOS image sensor for configuring, for example, its window size, frame rate, exposure time and internal gains.

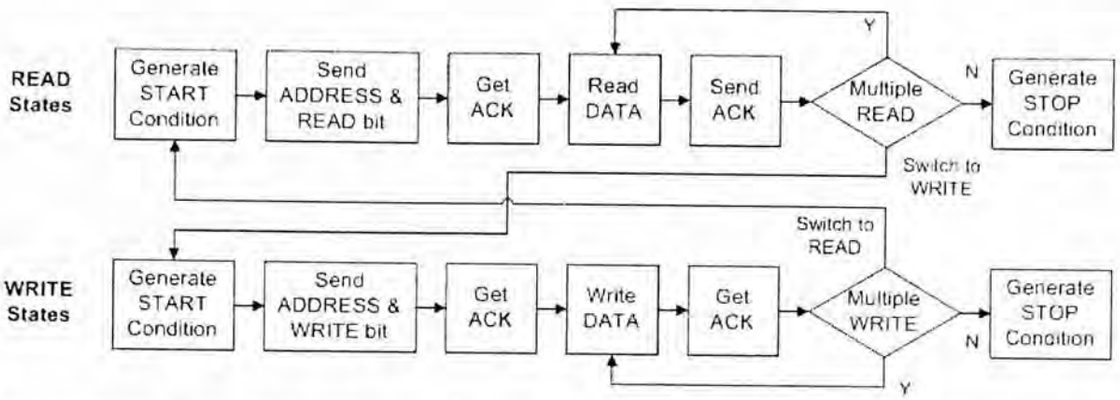
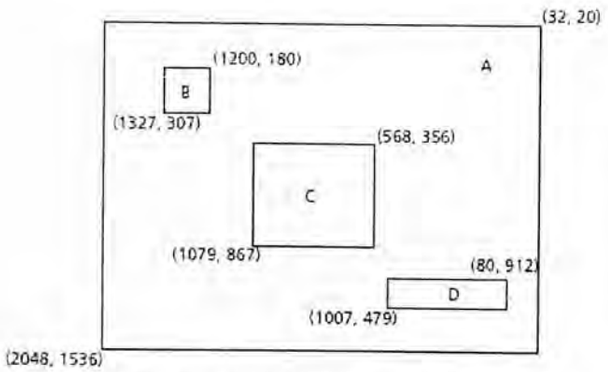


Figure 2.6: State transition of the read and write operations on the serial bus interface

Windowing Control

The MT9T001 CMOS image sensor allows us to configure its window size to achieve various resolution formats and frame rates. For example, the default window size of the image sensor is 2048×1536 (i.e. full frame size), we can program the sensor registers to achieve a VGA format with resolution of 640×480 and a maximum image frame rate of 93fps according to the frame timing formulas in [9]. Figure 2.7 shows examples for achieving different window sizes and positions with the given registers values.

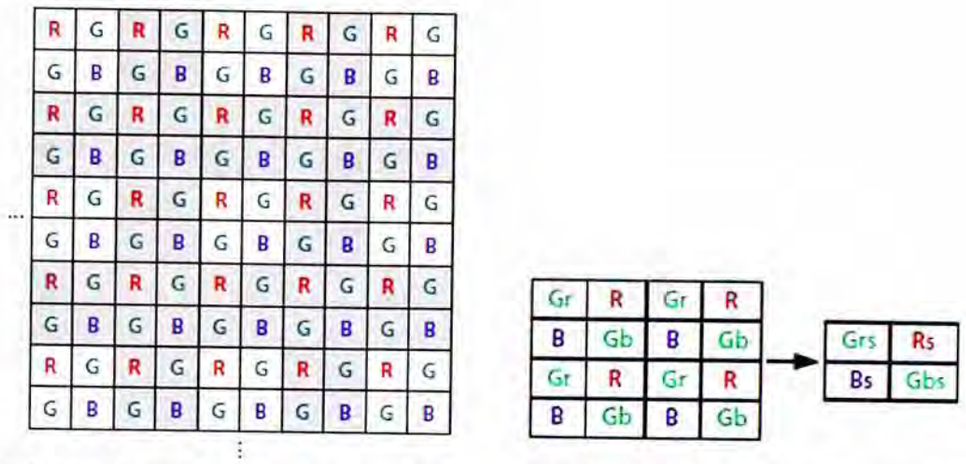


	Window Size	Reg0x01	Reg0x02	Reg0x03	Reg0x04
Window A	2,048 x 1,536	0x0014	0x0020	0x05FF	0x07FF
Window B	128 x 128	0x0084	0x04B0	0x007F	0x007F
Window C	512 x 512	0x0164	0x0238	0x01FF	0x01FF
Window D	400 x 96	0x0390	0x0050	0x018F	0x005F

Figure 2.7: Windowing capability of the image sensor, from [9]

Skip and Bin Mode

It is easy to obtain different image resolution with the use of windowing control, however, the field-of-view of the image will greatly reduced when the image resolution is small. In order to keep the field-of-view while we can still have different image resolutions, the image sensor provide a skip



(a) 2x column skip by 2x row skip mode, shaded pixels are skipped

(b) Bin 2-to-1, e.g. Grs = binning of 4 Gr[s] in a 4x4 window

Figure 2.8: Skip and bin mode of the image sensor, from [9]

mode (Figure 2.8a) for which rows and columns of the imager are sub-sampled to reduce output resolution without altering the field-of-view. But on the other hand, skip mode leads to a reduction in image quality as aliasing occurred when rows and columns are sub-sampled from the imager. That is why the image sensor comes with a bin mode (Figure 2.8b), with which the data value on the output pixels are obtained through averaging adjacent rows and columns, and this significantly reduces the aliasing effect introduced by skip mode.

The current image output resolution of the image sensor is configured to be 480×320 with a selectable frame rate of 30fps/50fps/76fps. This is achieved with a windowing control of 1920×1280 window size, a skip mode with column skip by 4x and row skip by 4x, together with a bin mode of bin 3-to-1. The frame rate is controlled by setting different shutter time through the respective registers on the image sensor.

2.3 Image Data Transmitting and Processing

After configured the image sensor, we have to retrieve the image data from the sensor to a desktop PC through the USB micro-controller module. In this section, details for the data transfer process on the USB micro-controller and image recovery carrying out on a PC will be mentioned.

2.3.1 Data Transfer Mode and Buffering on USB Controller

There are two common transfer modes that will be used for streaming data over USB interface. They are named the isochronous transfer and the bulk transfer mode. The major differences between the two transfer modes are: the isochronous transfer has guaranteed bandwidth but not guaranteed data delivery, there is no retry of transfer when data contain error or being lost; while the bulk transfer guarantees error-free data transfer, there is automatic retry for erroneous data, however the bandwidth is not guaranteed.

According to the Cypress application note [16], the average throughput can be achieved by isochronous transfer is 24MBps while it can reach 22MBps for the bulk transfer. And since the data rate for the camera settings of image resolution with 480×320 at frame rate of 76fps is only about 11.13MBps, both types of transfer mode would be applicable to the data transfer over USB. However, consider that bulk transfer can provide error-free data transfer, we therefore choose bulk transfer for data streaming over the USB.

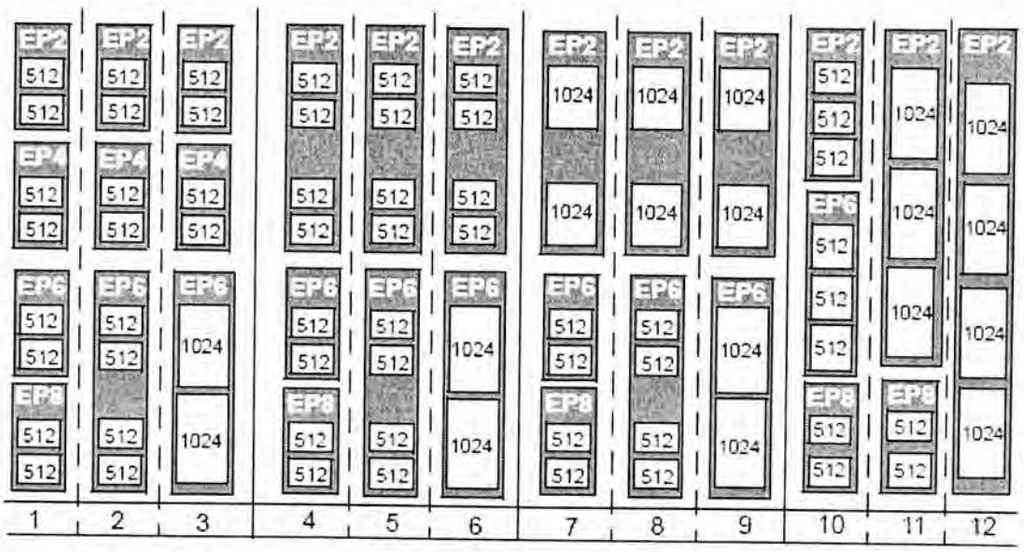


Figure 2.9: Available endpoint configurations on the USB micro-controller, from [4]

Data Transfer over Endpoint Buffers

The endpoint (FIFO) buffers are actually build-in RAM blocks on the USB micro-controller. It has a total size of 4 KBytes which can be separated into different number and size (Figure 2.9). Currently, they are configured to be quad-buffered with each consisting of 1024 Bytes.

The endpoint (FIFO) buffers of the USB micro-controller are programmed to be running in slave synchronous write mode with a setting of AUTOIN for which data on the buffers will automatically pack into packet of 1024 Bytes in size and passed to the serial interface engine (SIE) for transferring over the USB data bus when the host PC sends a request to the micro-controller.

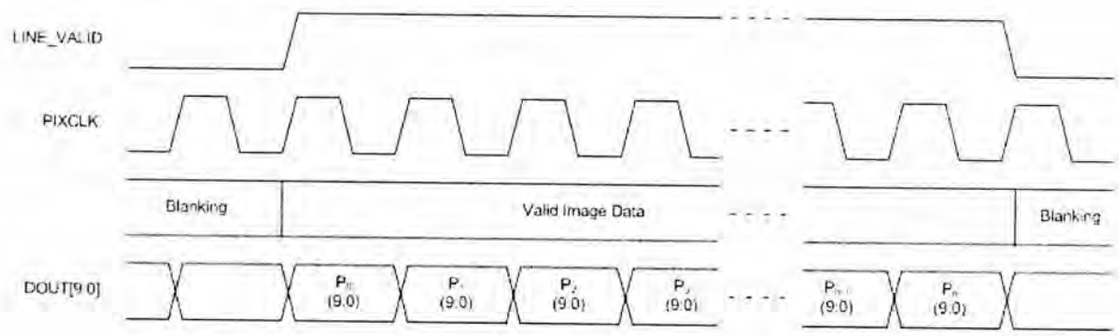


Figure 2.10: Synchronization signals from image sensor

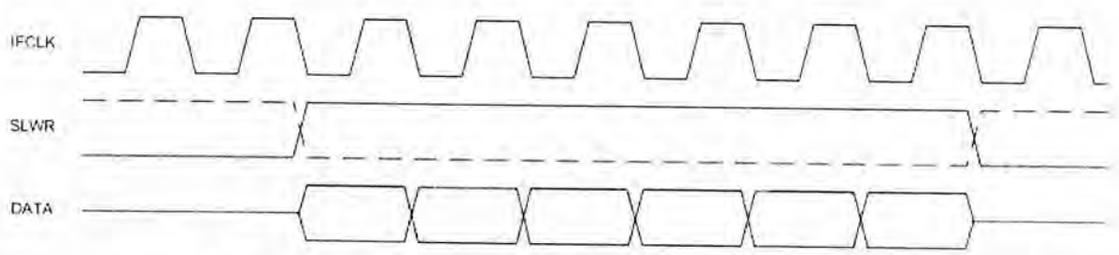


Figure 2.11: Time diagram for slave FIFO synchronous write on the USB micro-controller

In Figure 2.11, it shows the time diagram for the operation of slave FIFO synchronous write mode on the USB micro-controller. Data are written to the endpoint buffers at the rising edge of the interface clock (IFCLK) when the write strobe line (SLWR) is high (configurable to be low). We configure the

interface clock of the micro-controller to use its internal 48MHz clock source and enable its pin output for providing clock source to the image sensor as the master clock. The LINE_VALID (indicating a valid sensor row pixel readout) synchronization signal (Figure 2.10) output from the image sensor is connected to the USB micro-controller as the write strobe signal for controlling valid pixel data written to the FIFO buffer. The pixel clock (PIXCLK) signal output of the image sensor is same to its master clock and by default pixel data are available at the falling edge of PIXCLK (configurable to be at rising edge). In order to satisfy the time requirement for synchronous write on the FIFO of the USB micro-controller, the sensor register is configured to output pixel data at the rising edge of PIXCLK.

2.3.2 Demosaicking of Bayer Image Data

An image data retrieving program is developed with the use of the USB micro-controller driver and API library provided by Cypress Semiconductor for acquiring image sensor data. After attaching the CMOS image sensor module to the USB micro-controller module and connecting them with a USB cable to the host PC, the host PC program will recursively query image data with the configured resolution of 480 × 320 in size (153600 Bytes) from the USB device enumerated by the USB micro-controller. Since the data output from the image sensor are in RGB Bayer pattern (only one color of the R,G,B channels is available at each pixel, where green colors are in quincunx grid while red and blue colors are in rectangular grid, see Figure 2.12), we have to perform interpolation on nearby color pixels in order to recover the lost channels from the received Bayer image [64].

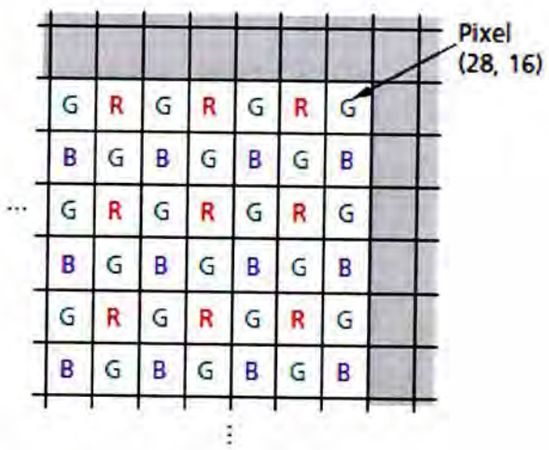
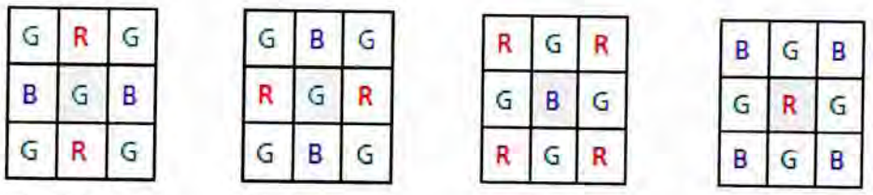


Figure 2.12: RGB Bayer pattern of the image sensor

Bilinear Interpolation

The interpolation (i.e. demosaicking) algorithm we use to recover the whole image data (R,G,B channels) from the Bayer pattern is the bilinear interpolation. We choose the algorithm for the demosaicking process because it is computationally less expensive and applicable to embedded processors but can still retain good image quality comparing to other demosaicking algorithm such as nearest neighbor or cubic convolution method [68].



(a) Odd row and column

(b) Even row and column

(c) Odd row and even column

(d) Even row and odd column

Figure 2.13: Bilinear interpolation for Bayer image

According to [68], we can recover the missed R/G/B channel at the desired image position through bilinear interpolation as follows:

- Consider both odd/even row and column, the missed R/B color channel at the middle G position is obtained by taking average on the left and right R/B color or up and bottom B/R color (see Figure 2.13a and Figure 2.13b).

- For odd row and even column or even row and odd column, the missed R/G/B color channel at the middle B/R position is obtained by taking average on the diagonal R/B color or left, right, up and bottom G color (see Figure 2.13c and Figure 2.13d).

After receiving a complete image in Bayer pattern, we can looping through the image data array; by identifying whether the image pixel is on odd/even row and odd/even column, we can perform specific bilinear interpolation process to recover the lost color channels. When the interpolation completed, we can display the demosaicked image and the whole process will begin again when new image data are received on the host PC from the USB micro-controller.

2.4 Splitting Images and Exposure Problem

When we have performed bilinear interpolation to recover and display the received image from the image sensor data, it is found that the images obtained are split into different portions (Figure 2.14). After some investigations on the sensor power supply and data transfer process, we later found that the occurring of split image is due to buffer overflow on the USB micro-controller.

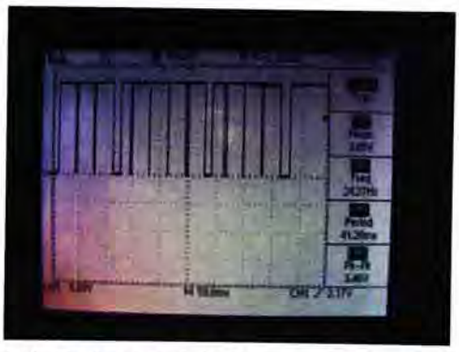
2.4.1 Buffer Overflow on USB Controller

Since the image data keeps streaming out of the image sensor, if the endpoint buffers get full, part of the image data within a frame can not be written to the endpoint buffers for transfer to the host PC. Thus, the remaining portion of the current image frame or the new portion of the next image frame will be lost.

Figure 2.14: Splitting image obtained after interpolation

Figure 2.15b shows the signal on the pin output indicating whether the endpoint buffers of the USB micro-controller get full (at where the signal is low). The buffer overflow is possibly due to latency on data retrieval on the

(a) Frame valid signal, valid image frame data at high

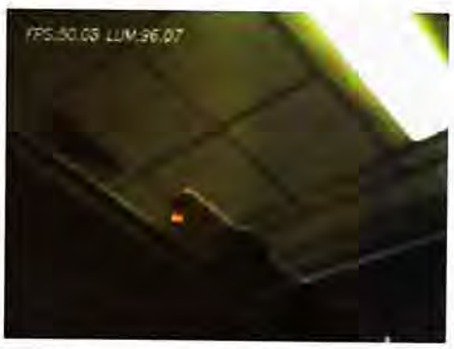


(b) Endpoint full flag, endpoint full at low

Figure 2.15: Observation of signal pins on oscilloscope

host PC. Since data transfer rate is host dependent, it depends on the processing power of the host PC and the USB host controller. Moreover, Windows is not a real time OS, it does not have a bounded interrupt/thread latency. The driver that we used for the USB micro-controller is also designed for general purpose, it may not be optimized for the use of data streaming.

(a) Shutter speed: 33.3ms, sample rate: around 30Hz



(b) Shutter speed: 20.0ms, sample rate: around 50Hz

(c) Shutter speed: 13.2ms, sample rate: around 25Hz

Figure 2.16: Frame rate achieved with image acquisition board and program

In order to cope with the above situations, we have tried to use real time software to access the real-time services layer supported by Windows OS (e.g. Ardence RTX). However, we are not able to access the Cypress USB driver with its provided API and thus we cannot retrieve image data with the use of the real time software.

Another way we have tried is to reset the image sensor and the endpoint buffers when the buffer overflows. We have successfully implemented these and the camera performs quite well where split images are seldom obtained. Figure 2.16a to Figure 2.16c show the images and number of frames can be retrieved under different shutter speed (frame rate) of the image sensor configured.

In Figure 2.17b, it shows the condition of endpoint buffers on the USB micro-controller with the image sensor configured with a shutter speed of around 76Hz (i.e. 13.2ms). The endpoint buffers get full with a frequency of around 25Hz (i.e. the image sensor

(a) Frame valid signal, image sensor frame rate @76Hz

(b) Endpoint full flag, endpoint buffers get full @25Hz

Figure 2.17: Observation of signal pins with image sensor configured to 76fps

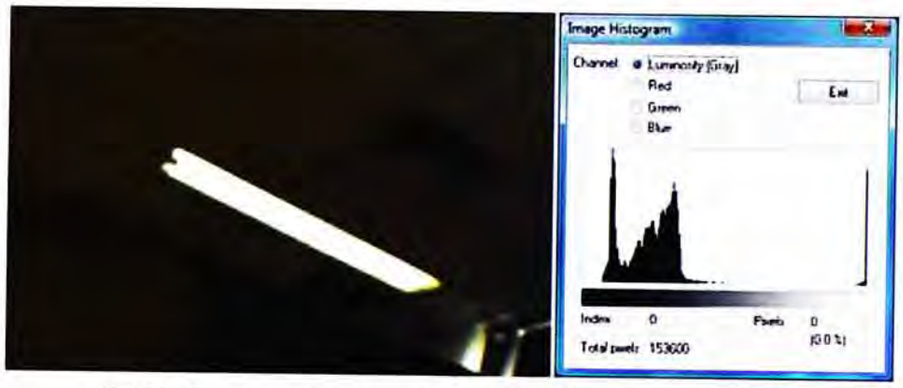
and the endpoint buffer is reset with a frequency of around 25Hz). Since a reset on the image sensor will cause the sensor to abandon the readout of current image frame and it will also treat the next frame as a bad frame and not outputting the image data, so we dropped 2 image frames for each reset on the image sensor. Therefore, we can only sample images of around 25fps on the host PC when we configure the frame rate of the image sensor to around 76Hz. In order to solve the buffer overflow problem completely, we will have to add additional memory on the USB micro-controller side for buffering image data when the endpoint buffers on the USB micro-controller get full. However, that requires further development on the image acquisition board and it is beyond our intention for evaluating the performance of image sensor on a desktop PC. Instead, we will look into other aspects regarding the image sensor itself.

2.4.2 Image Luminance and Exposure Adjustment

The image sensor we used does not offer the function of auto exposure. However, it is possible for us to adjust the exposure by controlling over the shutter time and brightness gain on the image sensor. Before doing so, we will have to measure the brightness of the image we obtained.

A luminosity histogram [21] can describe the perceived brightness distribution over an image. The luminosity is based on a weighted average of the three color channel (red-30% green-59% blue-11%) on the image pixels with the consideration that human eyes are more sensitive to green light than red

(a) Overexposed image retrieved with large ADC gains



(b) Underexposed image retrieved with small ADC gains

Figure 2.18: Image exposure under different ADC gains

or blue light.

It is found that overexposed image will have most of its image pixels locate on the right of its luminance histogram while underexpose image will have most of its image pixels locate on the left of its luminance histogram.

Through the observations on the luminance histogram on the overexposed and underexposed images, we notice that it is possible to measure the luminance level as the mean of all image pixels over the luminance histogram of an image. And by the mean luminance we calculated (also shown in the images of Figure 2.16), we can control the shutter time (will affect the image frame rate) and the ADC gains of the image sensor to achieve a referenced luminosity (e.g.128) which will probably give an image with suitable brightness level.

□ **End of chapter.**

Chapter 3

Embedded System for Vision Processing

Having understand how to configure the image sensor and acquire image data from it, we are ready to start integrating it to an embedded system as a peripheral video device and will have our images captured on the system soon, though it still involves steps to interface the camera module to the embedded system and develop driver for access and configure the image sensor.

This chapter first overviews the features and system architecture of the embedded system platorm (Gumstix Computer-on-Module) we use for image acquisition and perform image processing. Then, we will show how we interconnect the image sensor to the image signal processing interface of the embedded system and how the sensor driver and image acquisition program are developed. We will also show the new camera module (based on a CMOS image sensor featuring global shutter) and the system interfacing boards we built that allows further modularization of the vision system. Finally, a biaxial pan-tilt platform for camera view-stabilization and the overall system setup on the UAV platform will be mentioned at later sections of the chapter.

3.1 Overview of the Embedded System

The embedded system we used for developing our vision system is the Gumstix Overo Computer-on-Module, which is a platform that featuring high performance and low power driven heterogeneous multiprocessor system-on-chip

(SoC) commercially available in the market. It is a fully functional embedded computer in module size. The Computer-on-Module (CoM) provides peripherals and interfaces to the multiprocessor System-on-Chip (SoC) it uses. Features of the SoC and the CoM are mentioned in the following parts.

3.1.1 TI OMAP3530 Processor

The heterogeneous multiprocessor SoC that used on the Gumstix Computer-on-Module is the TI OMAP3530 processor. The processor integrates three processing architectures as subsystems in a single chip set. The architectures include a general purpose processor (GPP) - ARM Cortex-A8 core, 600MHz; a digital signal processor (DSP) - TMS320C64x+ DSP Core, 430MHz and a graphical processing unit (GPU) - POWERVR SGX Graphics Accelerator. It is designed to provide best-in-class video, image, and graphics processing for mobile devices. The processor also consists a camera subsystem that supports multiple formats and interfacing options connected to a wide variety of image sensors. Multiple peripheral communication interfaces such as UART, SPI and I^2C, USB connectivity and external storage are also available on the processor [11].

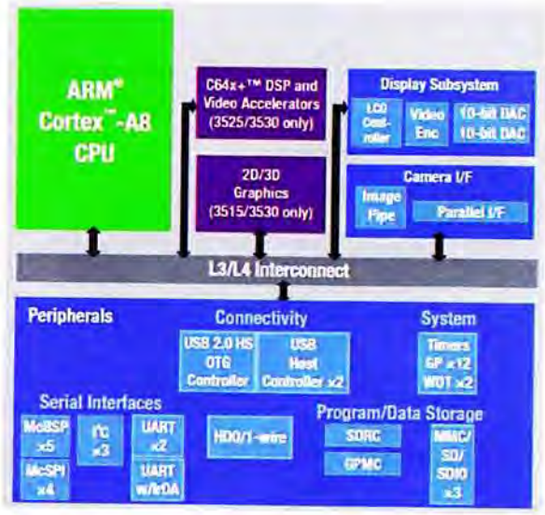


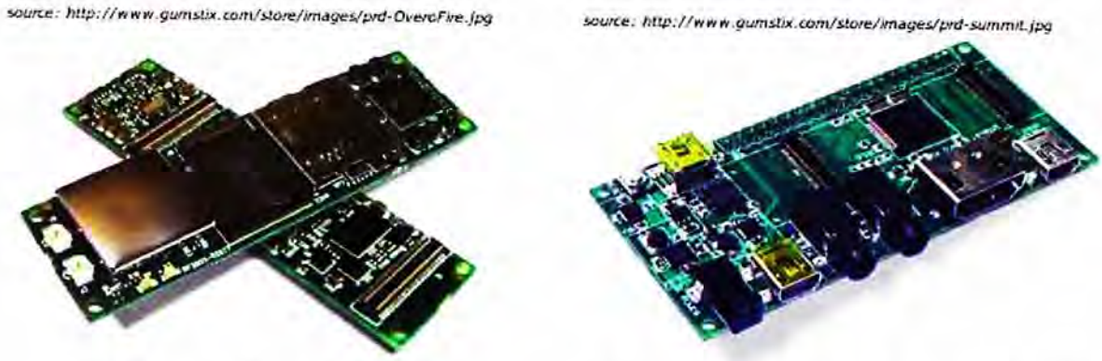
Figure 3.1: System architecture of OMAP3530 processor, from [11]

3.1.2 Gumstix Overo Fire Computer-on-Module

The Gumstix Overo Fire Computer-on-Module [8] is designed to provide peripheral interfaces to the TI OMAP3530 processor. With performance approaching that of a laptop PC, the Computer-on-Module has a size of only $80mm \times 39mm$ and features with 256MB RAM, 256MB Flash, on-board microSD card slot, USBs, HDMI, Audio In/Out, Wi-Fi 802.11g and Bluetooth wireless communication, etc. But more importantly, it provides a dedicated

connector for the camera interface to the image signal processing (ISP) subsystem of the TI OMAP 3530 processor. Thus, we can design and connect our image sensor module to the ISP interface through this connector.

Gumstix Overo Fire showing on the left (Figure 3.2a) has a size of only $58mm \times 17mm$. It is adapted to the expansion board - Summit, shown on the right (Figure 3.2b), of size $80mm \times 39mm$ to provide power and additional peripheral support such as USB, HDMI and Audio In/Out connections.



(a) Gumstix Overo Fire CoM (b) Summit - Expansion Board for CoM

Figure 3.2: Gumstix Overo CoM and its expansion board

3.2 Interfacing Camera Module to the Embedded System

The TI OMAP3530 processor consisting a camera image signal processing (ISP) subsystem that provides system interface and processing capability for connecting RAW image sensor module to the processor and it also supports imaging applications such as image previewing and image capture with digital zooming. Following parts give more details on the ISP subsystem and how we connect our camera module to the Gumstix Overo Fire CoM with an adapting board. Some initial images have also been recorded with the camera module through the camera interface, although further configurations on the system should be carried out to provide correct image format and better imaging results.

3.2.1 Image Signal Processing Subsystem

The camera image signal processing (ISP) subsystem on the OMAP3530 processor provides dedicate hardware for image signal processing on its interface that supporting various image sensors with RGB primary color or YUV complementary color outputs. Progressive or interlaced image sensors with electronic rolling shutter or global-release reset shutter are both compatible. The parallel camera interface can work with 8-, 10-, 11-, and 12-bit data in synchronous mode. Signal processing operations such as optical clamping, black-level compensation, faulty pixel correction, white balance, black adjustment, color correction, color convention, etc. are available to be performed on the RAW (in Bayer pattern) image data and converted to YUV image format. Statistic metrics and histogram can also be obtained for providing adjustment on sensor parameters to achieve auto white balance, auto exposure and auto focus. But more impressively, up to 10-bit data at 75 MHz can be processed by the image pipeline or transferred to memory [46]. That means our CMOS image sensor (operating at 48MHz with 10-bit image data output) is fully supported by the ISP interface and no further interpolation process for image recovery is required.

Figure 3.3 shows the connecting interface of the camera ISP to external

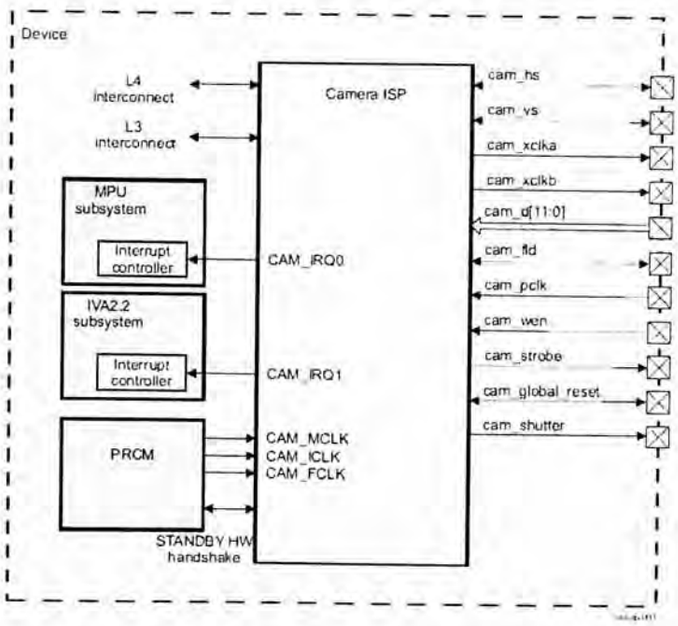
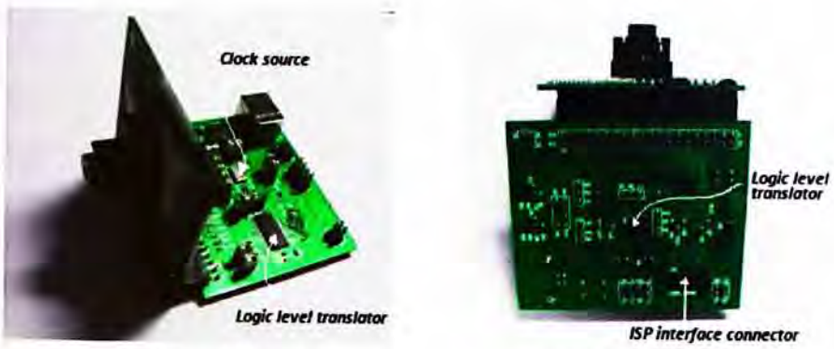


Figure 3.3: Camera interface subsystem of OMAP3530 processor, from [46]



(a) Top view

(b) Bottom view

(c) Connecting to Gumstix Overo CoM

Figure 3.4: Camera module adapting board for Gumstix Overo CoM

image sensor module. In our current application only the cam_hs(line trigger signal), cam_vs(frame trigger signal), cam_d[9:0](image data bits) and cam_pclk(pixel clock) data pins are used for image data retrieving on the ISP subsystem. They are connected to LINE_VALID, FRAME_VALID, D-OUT[9:0] and PIXCLK signal pins on the image sensor respectively (refer to Appendix A).

3.2.2 Camera Module Adapting Board

The Gumstix Overo Fire Computer-on-Module provides a camera connector which interface to the ISP subsystem on TI OMAP3530 processor. Image sensor modules can interconnect into the interface though flexible print circuits ribbon that adapts to the connector. However, since the logic level of the ISP interface on the TI OMAP 3530 processor is 1.8 volt while the logic level of our camera module is 3.3 volt. A logic level translation should be carried out in order to interface the image sensor to the ISP subsystem. Therefore,

an adapting board has been developed to meet the interfacing requirement (see Figure3.4). Besides, it also provides a bidirectional logic level shifting for the I^2C control signals (unidirectional level shifting are used for image data signals) between the image sensor and the ISP subsystem, a fixed clock signal of 48MHz to the camera module and extra power supply (through USB host connector for testing purpose) besides the 1.8 and 3.3 voltage supply on the camera connector interface.

3.2.3 Image Sensor Driver and Program Development

After successful logic level conversions between the image sensor and the ISP subsystem of OMAP processor with the use of the camera module adapting board, the hardwares for the camera system are ready. The next step is to configure the embedded Linux operating system to control the image retrieving processes. That would be a rather complicated part, as the whole process includes configurations on the Linux kernel, developing and modifying drivers for the ISP subsystem and the image sensor.

Cross-compilation Environment

Figure 3.5: Logo of the OpenEmbedded project, from [13]

In order to configure the Linux kernel, develop driver for our image sensor module and program for image acquisition, we have to set up a program development environment for cross-compiling programs to be executing on the embedded system (the OMAP3530 processor is using the ARM architecture). The OpenEmbedded [13] provides a built framework for cross-compiling the embedded Linux. It allows system developers to create different Linux distribution for wide variety of hardware architectures. Through the extensive customizable BitBake [1] recipes it provides and the BitBake packaging tool, lots of useful software packages (e.g. the OpenCV and GStreamer library that our tracking program develop upon) can be included or installed separately in the embedded Linux distribution. File patching can also be applied through the BitBake recipe for modifying board specific files and adding new device drivers to the Linux Kernel.

Linux Device Driver Module

The device driver we developed for the image sensor follows the structure of an I^2C and Video for Linux (V4L2 [70] provides unified access to video capturing device on Linux system) device driver module [36]. The driver is a loadable module that can be added to the Linux kernel at runtime. It provides function calls to configure sensor registers through the I^2C interface and I/O control methods for negotiating sensor properties (e.g. image format, resolution, frame rate, etc.) as supplied in the V4L2 user-space API.

Generally, the device driver can be grouped into following function portions:

- `reg_read`: read sensor register value through the I^2C client driver
- `reg_write`: configure sensor register through the I^2C client driver
- `device_init`: register image sensor as I^2C and V4L2 device, initialize sensor registers to start up state
- `ioctl_commands`: negotiate sensor properties and configure sensor registers
- `device_cleanup`: unregister image sensor as I^2C and V4L2 device

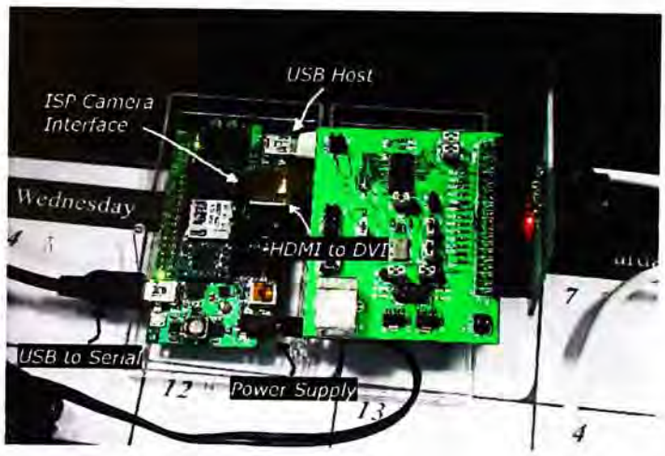


Figure 3.6: System setup for image capturing on CoM

After further board specific configuration (e.g. to include the sensor device module and specify the I^2C interface it connects) on the ISP camera interface, compilation of the Linux kernel image and user-space programs, we are able to capture image through the camera module with application running on the Gumstix Overo Fire Computer-on-Module. Figure 3.6 shows the system setup

for the MT9T001 image sensor module and the Gumstix Overo Computer-on-Module.

Image Capturing and Video Streaming on CoM

Some initial images have been captured with the system setup. The MT9T001 image sensor is configured with full resolution (2048×1536) image output, while the image is resized (down sampled) to 640×480 (see Figure 3.7) through the ISP camera interface on the Computer-on-Module. It can be noticed that the image in Figure 3.7a tends red with the default white balance coefficients being configured on the ISP camera interface, while appears normal in Figure 3.7b when the white balance coefficients are readjusted.

(a) Default white balance coefficients

(b) Readjusted white balance coefficients

Figure 3.7: High resolution images captured on CoM

Figure 3.8: Low resolution image captured on CoM

In order to increase the image frame rate for real-time video streaming on the Gumstix Computer-on-Module, we have configured the image sensor to output image with a lower resolution through the sensor skip and bin mode. With the use of MPlayer software package, we can obtain real-time video streaming from the image sensor and show it on a LCD display

through the HDMI to DVI interface of the Computer-on-Module. Table 3.1 below shows the frame rate achievable with different image resolution and the maximum frame rate the system can process:

Image Resolution	**Frame Rate** (sensor output)	**Frame Rate** (processed)
2048 × 1536 (d.s. to 640 × 480)	12fps	12fps
640 × 480	48fps	48fps
480 × 320	102fps	76fps
d.s. = down sampled		

Table 3.1: Video streaming resolution and frame rate

3.3 View-stabilizing Biaxial Camera Platform

At this development stage, we can successfully interface the high-speed CMOS image sensor (MT9T001) to the Image Signal Processing (ISP) subsystem of the heterogeneous multiprocessor System-on-Chip on the Gumstix Overo Fire Computer-on-Module platform for real-time image retrieval. To move on for achieving the aim for target tracking, geo-locating from UAV and performing real-time aerial image stitching with fix-wing aircraft, developing a pan-tilt camera platform would be necessary. As a pan-tilt platform can help to keep the tracked target within the camera's field of view and maintain a downward looking camera orientation on the fix-wing aircraft.

In the camera system mentioned in above sections, both the image sensor module and the adapting board have a relative large size comparing to that of the expansion board of the Gumstix Computer-on-Module. The movement of the camera system is also constrained by the short length of the flexible print circuit (FPC) cable. These constrains make this camera system impractical to be installed on a pan-tilt platform. Besides, the image sensor does not have build-in auto exposure and auto white balance functions, so it requires manual adjusting the exposure time and color gains on software level through the sensor driver and image acquisition application. That would possibly lead to addition processing time in between image frames. Moreover, the image sensor uses the electronic rolling shutter, even it is configured to operate at around

100 frame per second, the rolling shutter effect may not be prominent but still exist. Consider all constrains and limitations mentioned above, it is necessary for us to develop a new camera system which could eliminate all these problems.

For that we have decided to use a new CMOS image sensor (MT9V032) from Aptina/Micron Imaging which offers serial data outputs besides the parallel data outputs. The number of signaling wires can be reduced and the length of transmission can be extended. But more importantly is that the new CMOS image sensor uses global shutter instead of the generally used electronic rolling shutter. Rolling shutter effect under severe vibration can therefore be entirely eliminated. In the following parts, we will have an overview on the new camera system developed, the design of supporting modules for signal conversions, communication, servo control signal generation and the mechanism for providing view-stabilization through a biaxial platform we designed.

3.3.1 The New Camera System

The new camera system consists of three modules: a) the camera module, b) interfacing module and c) the MCU module. Each module contains specific electronic components that provide voltage regulation, logic level conversion, clock source and etc. to make the whole camera system functions. Details regarding the functionalities of each module are covered below:

(a) Front view (with lens)

(b) Front view (no lens)

(c) Back view

Figure 3.9: The new MT9V032 camera module

A. Camera Module

The camera module (Figure3.9) is one of the two main modules of the new camera system. Another main module is the interfacing module. The two modules connect together to form a complete image acquisition device of the camera system. Here, the new camera module uses a new model of CMOS

image sensor (MT9V032L12STC) from Aptina/Micron Imaging [10]. Details comparing the specifications and features of the previous image sensor (MT9T001P12STC) to the new sensor are list in Table 3.2. Although the resolution of the new sensor is far less than the previous one, that is enough for use regarding our tracking and surveillance purpose. But on the other hand, the new CMOS sensor provides global shutter, auto exposure and auto white balance that are not available in the previous image sensor. There is a voltage regulator on the camera module which provides a stable low drop-out step down 3.3V from the 5.0V input power source. The sensor system clock source is provided by an oscillator of 27MHz on board of the camera module. Moreover, the sockets on the camera module also provide data bus connection between the interfacing module and the MCU module.

	MT9T001	**MT9V032**
Imager size	1/2 inch ($6.55mm \times 4.92mm$)	1/3 inch ($4.51mm \times 2.88mm$)
Full resolution	2048×1536	752×480
Frame rate	up to 100fps @480×320	60fps @full resolution
Shutter type	electronic rolling shutter	global shutter
Auto exposure/color gain	not available	available
Data output format	raw Bayer	raw Bayer
Data output mode	parallel	parallel/serial
ADC resolution	10-bit	10-bit
Dynamic range	61dB	>55dB/>80-100dB (high dynamic mode)
Clock rate	48MHz	27MHz
Supply voltage	3.3V	3.3V
Power consumption	240mW	<320mW

Table 3.2: Comparing MT9T001 and MT9V032 CMOS image sensor

B. Interfacing Module

As mentioned above, the interfacing module (Figure 3.10) will be connected to the camera module to form a complete image acquisition device of the camera system. The major uses of the interfacing module are to provide data logic level translation to the Gumstix Overo Computer-on-Module and that of the image sensor and also two communication connections (including I^2C and UART. It includes the following voltage level transitions: 1.8V↔3.3V, 1.8V↔5.0V and

(a) Front view (b) Back view

Figure 3.10: Interfacing module of new camera system

(a) Front view (b) Back view

Figure 3.11: MCU module of new camera system

3.3V↔5.0V. Besides, the interfacing module contains electronic components for voltage step-up from 3.3V to 5.0V, data deserialization for converting serial data to parallel data, and clock source (27MHz) for the deserializer. Similar to the camera module, the interfacing module also has sockets for data bus connection but with an addition external power supply connector.

C. MCU Module

Different from the camera module and the interfacing module for providing image acquisition function on the camera system, the MCU module (Figure 3.11) is to provide servo control for the biaxial pan-tilt platform. The MCU module has a micro-controller unit (ATmega128) to generate PWM signals for controlling angular positions of RC (Remote-Controlled) servos. Besides, the analog to digital converters (ADCs) on the micro-controller unit will be used to tell the angular positions of the RC servos through measuring the voltage of its potentiometer inside. The voltage regulator on board is used to provide stable power supply to the micro-controller unit while the power supply to the RC servos is provided through external connector to power source directly. There are also sockets on the MCU module for UART communication, servo connections and micro-controller programming (JTAG interface).

Stacking Configurations

There are two stacking configurations for the new camera system namely the parallel mode and the serial mode. In parallel stacking configuration, the camera module is directly aligned and plugged onto the interfacing module with the sockets. While in serial stacking configuration, the camera module is connected to the interfacing module through an Ethernet cable and can be plugged onto the MCU module that mounting on a pan-tilt platform. Following shows the images of the two stacking configurations (Figure 3.12) and features supported by each mode.

I. Parallel Mode

Under parallel stacking mode, the image sensor is directly interfaced to the embedded Computer-on-Module through a parallel data bus. This configuration provides flexibility for initial stage of hardware diagnosis and sensor driver development on the Linux system. It also provides capability for standalone use of the camera system on mini-helicopter not requiring the lengthy Ethernet cable connection and installation of the pan-tilt platform.

II. Serial Mode

With the serial stacking mode, the image sensor is configured to generate serial (LVDS) data outputs and the image data is transferred through an Ethernet cable to the interfacing board. On the interfacing board, there is a deserializer to convert the serial image data back to parallel image data before interfacing

(a) Parallel mode (b) Serial mode

Figure 3.12: Stacking configurations of new camera system

to the embedded Computer-on-Module. The camera module can be plugged onto the MCU module that mounting on a pan-tilt platform. UART communication between the embedded Computer-on-Module and the micro-controller unit for RC servo control is also established through the Ethernet connection.

System Imaging Performance

Since we are using a new image sensor (MT9V032) for the camera system, new sensor driver has to be written on the embedded Linux system in order to make configuration for the new image sensor. The good news is that both the previous sensor (MT9T001) and the new sensor are developed by Aptina/Micron Imaging. Therefore, they have similar register naming and settings. Moreover, they get the same image output format, i.e. Bayer color pattern, so the previous driver framework can be reused with additional register changes. Major configurations on the new sensor driver include the color arrangement of Bayer pattern, the image resolution, serial data output mode and the auto exposure and auto color gain (white balance) functions. Figure 3.13a to Figure 3.13c show two different resolution configurations (752 × 480 and 368 × 240) of the image retrieved on the embedded system and the images retrieved with auto exposure and auto white balance being enabled.

(a) Resolution: 368×240, w/o auto exposure/color gain

(b) Resolution: 752×480, w/o auto exposure/color gain

(c) Resolution: 752 × 480, with auto exposure/color gain

Figure 3.13: Imaging performance of new image sensor

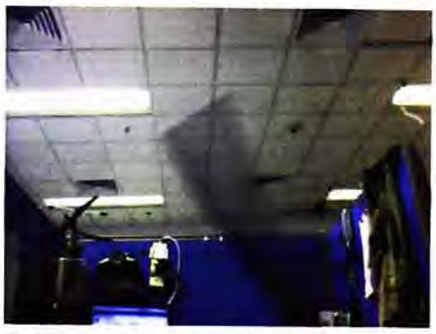
Comparing Shutter Effect

As mentioned in Section 2.1.2, line-scan type CMOS image sensor will suffer from rolling shutter effect under severe vibration and resulted in distorted image such that rigid objects will appear to have sheared or bended, while area-scan type (global shutter) CMOS image sensor can still maintain original shape of object even under severe vibration.

In order to evaluate performance of our new image sensor offering global shutter, we have performed an experiment to compare the new camera module to that of a web camera having image sensor with rolling shutter. Both cameras are connected to the embedded system for image recording. We have used a rigid carbon-fiber tube to demonstrate and evaluate the shutter effect. By swinging the carbon-fiber tube at a fix distance from the two cameras, it is found that the rigid carbon-fiber tube appears to have bended in the image captured by the web camera with rolling shutter, while it remains straight in the image captured by our new camera module offering global shutter (Figure 3.14).

(a) Cameras connected to the embedded system

(b) A rigid carbon-fiber tube



(c) Image captured by web camera

(d) Image captured by our new camera module

Figure 3.14: Setup and images captured for evaluating shutter performance

3.3.2 View-stabilizing Pan-tilt Platform

At the beginning of this section, we have discussed the need for a pan-tilt platform so as to keep tracked target within the camera's field of view wherever the UAV locates and maintain a downward-looking orientation of the camera in assisting aerial image mosaic building on fix-wing aircraft. In the following parts, we will look into details on the specifications and features of the pan-tilt platforms we selected to use and customized for installation on our UAVs. Besides, we will also show how we achieve image view-stabilization by integrating the biaxial camera platform with a tiny off-the-shelf aircraft autopilot that consisting inertial sensors.

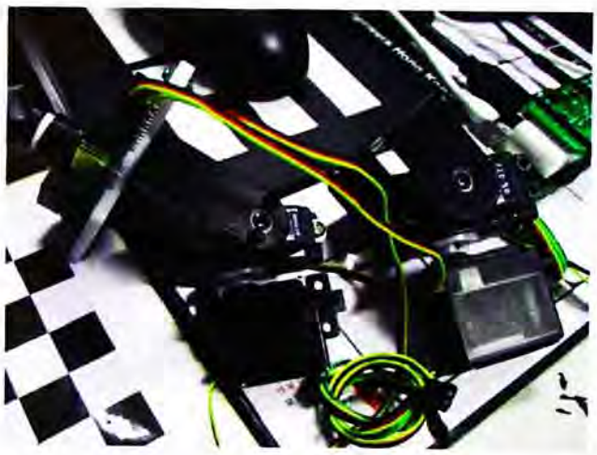


Figure 3.15: Pan-tilt kits from Lynxmotion, micro version (left) and standard version (right)

Selection and Design of Pan-tilt Platform

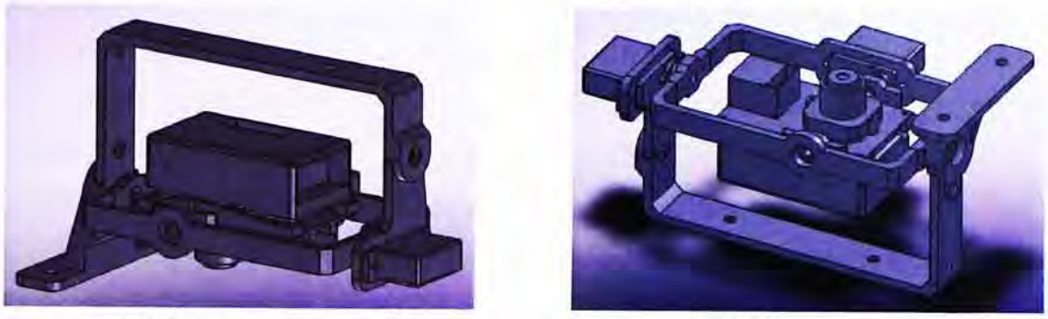
In developing the camera pan-tilt platform for our UAVs, three versions of the platform have been evolved. The first two versions of pan-tilt platforms were bought from Lynxmotion (see Figure 3.15). They are off-the-shelf pan-tilt kits that provide co-axial rotating configuration with ball-bearings at joints and are actuated by a pair of RC servos. Our camera system can be mounted on the servo brackets of the pan-tilt platform and adjust the orientation of platform through commanding the RC servos by the MCU module. Table 3.3 lists the specifications and features for the two versions of pan-tilt platform. Major differences between the two are their dimensions, weight and servos used.

Spec./Features	Standard Kit	Micro Kit
Material	aluminum	aluminum
Weight	140g (include servos)	85g (include servos)
Dimensions	58mm(L)x54mm(W) x84mm(H)	78mm(L)x44mm(w) x90mm(H)
Axis configuration	co-axial	co-axial
Servo type	standard servo	micro servo
Modulation	analog	analog
Torque	3.30kg-cm @4.8V	3.0kg-cm @4.8V
Speed	0.21sec/60° @4.8V	0.16sec/60° @4.8V
Motor type	3-pole ferrite	3-pole ferrite
Gear type	nylon	nylon
Operation range	180°	180°

Table 3.3: Specifications and features of pan-tilt kit used

Initial color blob tracking tests were performed with the standard pan-tilt kit. And later on, we moved to the micro pan-tilt kit in order to reduce the system weight and servo size. Further customization were made on the micro pan-tilt kit to replace one of its servo bracket with our RP (Rapid Prototype) parts (see Figure 3.16). The new RP parts allow us to mount an additional off-the-shelf analog CCD camera to the pan-tilt platform, such that real-time video from the UAV can stream down to ground station for system monitoring (e.g. to know whether target is being tracked) and at the same time assisting search and rescue task through manual checking. In Figure 3.16, it also shows that an IR (infrared) pass filter is being used to cover our digital camera module. It is used for filtering off (suppressing) visible light and help us to detect IR illuminating point. More details regarding the IR point detector and its usage will be mentioned in Chapter 4.

Figure 3.16: Customized micro pan-tilt kit



(a) Downward view (b) Upward view

Figure 3.17: CAD drawings of pan-tilt platform for foam airplane

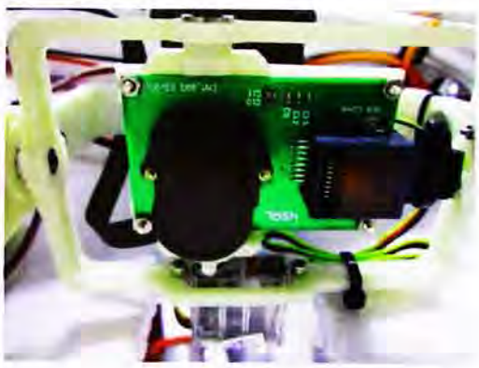


Figure 3.18: Pan-tilt platform installed on foam airplane

The third version of the pan-tilt platform (Figure 3.18) is a tailor-made design for the camera module and is being used on the small EPS foam fix-wing airplane in our laboratory for performing surveillance and aerial mosaicking task. The pan-tilt platform is assembled with RP parts that we design (see Figure 3.17) and actuated by two sub-micro RC servos. The camera module is attached to the sub-micro servo mounting on the inner frame that controls the camera's pitching while the inner frame is attached to the sub-micro servo mounting on the outer frame that controls its rolling, i.e. the camera's rolling. The new camera pan-tilt platform is also in co-axial configuration but at the same time with a faster response (servo speed: 0.14sec/60° @4.8V) and large reduction in weight (weight: 46g, including servos) when compared to previous versions of pan-tilt platform.

Servo Modification for Angular Position Feedback

In the first two version of pan-tilt platform, we have also made some hacking on the RC servos. Since typical RC servo does not have position outputs while only position inputs are allowed. It also takes time for the RC servo to move to the commanded position. If we assume the commanded position is the current

servo angular position, then a large error may be involved in calculating the target geo-location. So a mean to get an accurate angular position of the RC servo is to use the voltage level of the potentiometer inside (see Figure 3.19a). When the angular position of the RC servo changed, the voltage level of the potentiometer inside will also change. By reading the voltage level of the potentiometer inside the RC servo using analog to digital converter (ADC) on the micro-controller unit of MCU module, we are able to estimate the current angular position of the RC servo.

Consider the micro RC servo we used in the second version of pan-tilt platform, the voltage range recorded from the potentiometer is around 590mV to 1.97V for 180° rotation, therefore the internal voltage reference (2.56V) on the micro-controller unit is used. The overall resolution of the angular position of the RC servo calculated is around 0.35°, which is acceptably close to typical resolution of a RC servo.

(a) Internal control board, potentiometer and motor of RC servo

(b) Modified RC servo with wiring for position feedback

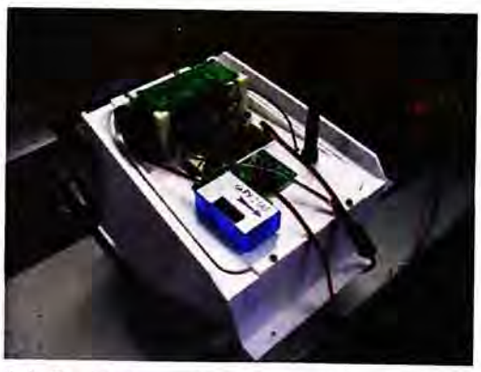
Figure 3.19: Servo hacking for angular position feedback

View-stabilization with Inertial Sensors

There are always attitude changes on fuselage of the aerial platforms when they try to make turns. Especially for the fixed-wing model planes, they do make large banking angle when performing turns. Therefore, images captured by a camera that fixed to the aerial platform will appear quite shaking and have large view changes when they make turns.

For that we try to make use of the attitude data (pitch and roll angles) output from an off-the-shelf aircraft autopilot system to adjust the camera's orientation through the RC servos on the pan-tilt platform so as to compensate the view change. The autopilot system we used is an Inertial Attitude Stabilizing System (FY-21AP) developed by FeiyuTech. It integrates 3-axis gyro and 3-axis accelerometer to provide auto balancing and piloting for fixed-wing model aircraft. However, the data rate of the attitude output from the autopilot system is only 10Hz and there is latency on the attitude output. Therefore, we can not achieve satisfactory view-stabilizing results with the above method.

But on the other hand, we notice that servo control by the autopilot system under its auto balancing mode is quite responsive. Thus, we decide to make use of the auto balancing feature of the autopilot system, which originally used for adjusting flight controls to aileron and elevator for maintaining balance when there is attitude change, for adjusting controls to the pan-tilt platform and achieve image view-stabilization. Through modifying the control gains of the system's PID-controller, direction of servo control output and the servo angular range, we can maintain the camera in a downward looking orientation and a minimum view change even there is large variation in attitude of the UAV platform. Figure 3.20 shows the two configurations for mounting the autopilot on the pan-tilt platform for view-stabilization.



(a) On 2nd version of pan-tilt platform

(b) On 3rd version of pan-tilt platform

Figure 3.20: Autopilot on pan-tilt platform for view-stabilization

3.4 Overall System Architecture and UAV Integration

In this section, we show the system architecture for the biaxial vision system we developed and how we customized the camera mounting and fuselage for best fit aerial platform integration. In Figure 3.21, we show the block diagram of the vision system that used for performing real-time aerial mosaicking task.

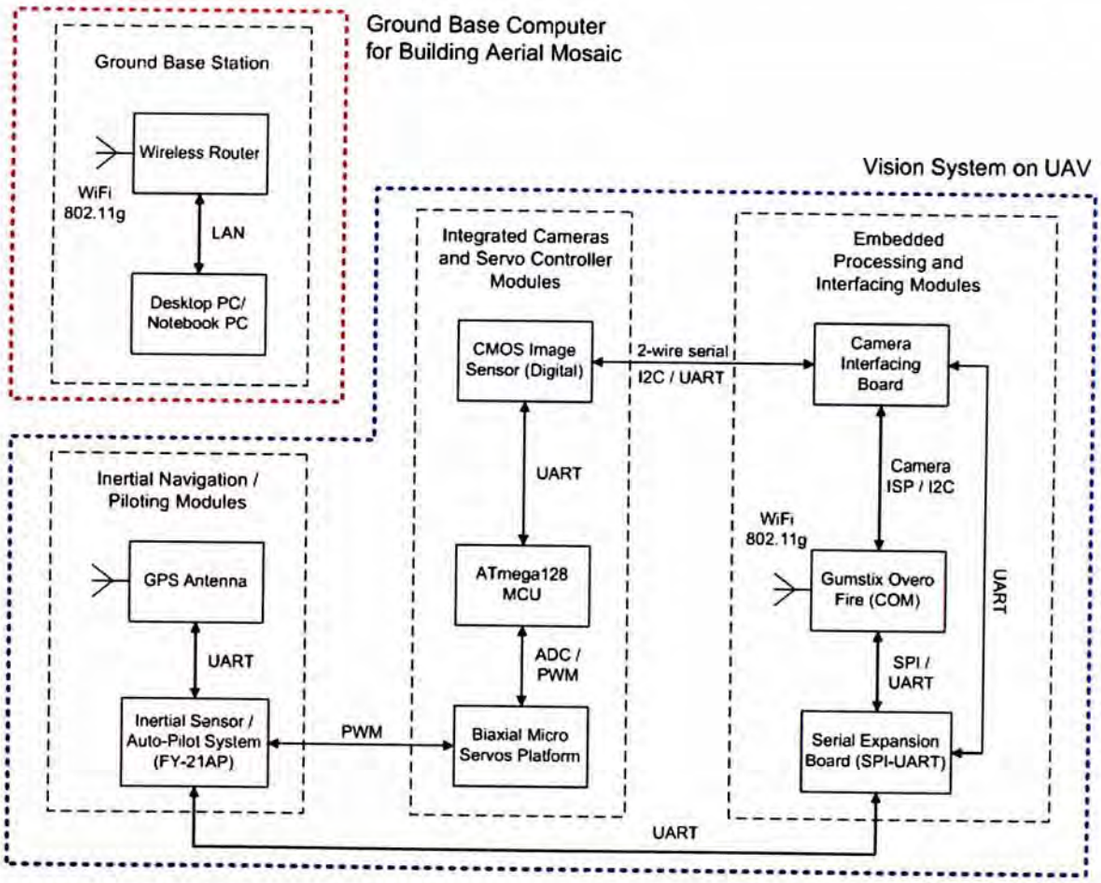


Figure 3.21: Block diagram of the overall system

There are three main parts of the vision system: 1)the embedded processing and interfacing module - it interfaces to the image acquisition device, connects to external sensors, performs vision processing on the retrieved images and provides data transmission to the ground base station; 2)the integrated camera and servo controller module - it captures images and transfers them to the embedded processing system, obtains servo angular position feedback and con-

trols the pan-tilt platform when used (e.g. for target tracking); 3)the inertial navigation/piloting module - it provides inertial and position information to the embedded processing system and controls the pan-tilt platform for view-stabilization. And, at the ground base station, we have got a wireless adapter for receiving the image data and performing aerial mosaicking on a laptop PC.

The system architecture for performing target tracking task is similar, but with the tracker implemented on the embedded processing system. Only position of the tracked target and angular change of the pan-tilt platform are transmitted to the ground base station while an additional analog CCD camera is used for real-time monitoring.

Serial Expansion Board

In order to allow the Gumstix Computer-on-Module to connect to more external sensors (typically use the UART interface for communication, e.g. the autopilot FY-21AP), a serial expansion board (see Figure 3.22a) has been developed. It is mainly used for converting the SPI communication data protocol to that of UART with specific conversion IC. Two UART communication interfaces in total have been established with the SPI data port on the Computer-on-Module. And through user-space program, we can configure the SPI-to-UART conversion IC with required data rate and data bit supported by the connected sensors. With use of some small RP parts, we have also put the serial expansion board, the embedded processing board and the camera interfacing module in stacks for easier system integration on UAV platforms (see Figure 3.22b).

(a) SPI-to-UART serial expansion board

(b) Print circuit board (PCB) stacks

Figure 3.22: System boards of the embedded platform

(a) Installation of pan-tilt platform on airplane

(b) The 1/4 scale Super Cub fix-wing airplane

Figure 3.23: Vision system on large fix-wing airplane

(a) Installation of pan-tilt platform on airplane

(b) The EPS foam fix-wing airplane

Figure 3.24: Vision system on hand-launch foam fix-wing airplane

System Integration on UAV

With considerations on the air drag, protection of the vision system and unification of the UAV platform, we have cut the plane belly and customized our camera pan-tilt platform to fit the space available. Figure 3.23 show the system integration for our second version pan-tilt platform on a 1/4 scale Super Cub fix-wing airplane. The airplane has a large wing span and propels with a 2-stroke engine that makes it suitable for long distant and long endurance flight with large payload (approximately 3kg). However, it requires a runway for taking off and landing and it is so large in size. Therefore, we have moved to a hand-launch foam fix-wing airplane instead which would be easier and safer to carry out experiments at initial development stage, though the payload is much smaller (approximately 1.2kg). And that is the reason why we have our third version of the camera pan-tilt platform. Figure 3.24 show the

integration for the vision system on the foam airplane. Apart from airplane, we have also performed experiments with the gasoline helicopter that mounted with the second version pan-tilt platform. All tests of the vision system that carried out on the aerial platforms mentioned above show that the image acquisition system is able to provide clear, shear- and jitter-free images.

□ **End of chapter.**

Chapter 4

Target Tracking and Geo-locating

One of the applications for the embedded vision system we developed would be to track an interested ground target and locate it from the image view point to the geodetic coordinates (i.e. latitude, longitude, and elevation). The geodetic location obtained after processing could be used for reconnaissance and aiding search and rescue missions. Or, when incorporating with control of the UAV, for example, to automatically landing a miniature helicopter at a marked location.

This chapter focuses on the methods we use and algorithms we have applied to achieve the objective of tracking an interested target and locating it in the geographic coordinates with the images, inertial and geodetic data obtained from the embedded vision system we developed and sensors on the UAV platform. First of all, we will have an overview on the camera calibration processes we have taken to obtain camera parameters which are required for converting the pixel data into metric information to be used on target geo-locating. Then, from our point of application, we show the image features we choose for object detection and algorithms we apply for tracking. In order to keep the the tracked target inside the camera's field of view, an PID controller is also introduced to the camera pan-tilt platform for centering target in images. Finally, we will show how we geo-locate the tracked target through a vector addition method. System setup for the experiments, the results we obtained and a general discussion will be given at the end of the chapter.

4.1 Camera Calibration

In computer vision, camera calibration is usually carried out to retrieve the metric information required for 3-D reconstruction with the use of 2-D images perceived by camera. There are many publications, [39, 78, 44, 55, 72, 84] to cite a few, regarding the techniques used for camera calibration, from capturing images with precise 3-D calibrating object to general static scenes. And now, camera calibration is considered a solved problem in the computer vision community. Among all the techniques available, the method suggested by [84] to calibrate a camera by observing a planar pattern (also know as chessboard) from different orientations is considered to be robust, flexible and reliable. There is a popular calibration toolbox [29] available on the Internet implemented the corresponding calibration method and with further improvements. In the embedded vision system we developed, we also make use of this toolbox for calibrating our camera.

The camera calibration process in general is to determine the metric to pixel relation, which is also known as the intrinsic parameters of camera. For estimating the geographical position of a ground target in the images captured by a camera mounting on an aerial platform, we also need to know this relation, although there are still a series of transformation required for converting from the image coordinates to the geographic coordinates. Besides the intrinsic parameters, camera lens distortions are also determined during the calibration process. Before we move to the calibration process with the use of the calibration toolbox, we will first look into the camera model and lens distortion model that involve in camera calibration.

4.1.1 The Perspective Camera Model

A camera model provides the geometric relation between the three-dimensional physical world to the two-dimensional images perceive on the image sensor. In the following, we are referring to the perspective projection camera model as described in [30], while more detailed model and further derivation of the camera model can be found in [56, 45].

Figure 4.1 shows the pinhole camera geometry for the projection of a point $Q = (X, Y, Z)$ in the 3-D camera coordinate on to a frontal image plane at

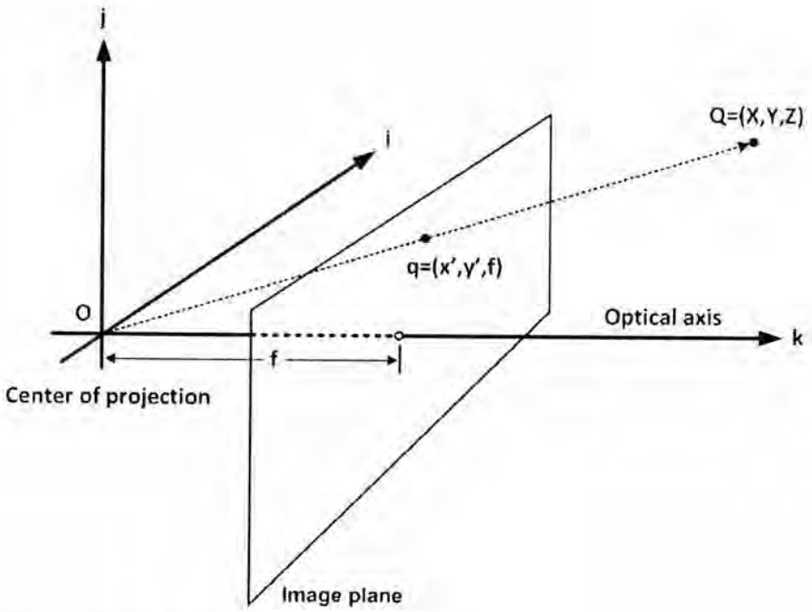


Figure 4.1: Frontal perspective image projection, from [30]

a position of $q = (x', y', f)$ through the center of projection (i.e. the optical center). The position where the optical axis passing through the image plane is referred as the principal point and f here denotes the camera focal length. We can express the relation for the projection by similar triangles as:

$$x' = f\frac{X}{Z}, \qquad y' = f\frac{Y}{Z} \tag{4.1}$$

However, in real situation for camera manufacturing, it is hardly possible for aligning the center of the image sensor to be exactly at the principal point. Therefore, additional parameters C_x and C_y are added to the projection equations in (4.1) for mapping the point in 3-D camera coordinate to the image coordinate:

$$x' = f_x\left(\frac{X}{Z}\right) + C_x, \qquad y' = f_y\left(\frac{Y}{Z}\right) + C_y \tag{4.2}$$

where f_x and f_y are camera focal length in the horizontal and vertical axis respectively, they are actually the products of lens focal length (F) and the size of pixel element (s_x, s_y) in the corresponding axises, i.e. $f_x = Fs_x$ and $f_y = Fs_y$, thus they defines the conversion from metric unit to pixel unit; while, C_x and C_y defines the center of projection on image plane.

By expressing the above equations in homogeneous coordinates, the pro-

jection of the point in 3-D camera coordinate (ijk frame) into the 2-D image coordinate can be summarized in the following matrix form:

$$q = MQ \implies \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.3}$$

Since the point q is expressed in homogeneous coordinates, it is required to divide both x' and y' by w in order to obtain the previous equations defined in (4.2). And here, the parameters in the 3-by-3 M matrix are called the intrinsic parameters and the matrix is known as the intrinsic matrix for the camera.

4.1.2 Camera Lens Distortions

With a pinhole camera, only very small amount of light can pass through the pinhole, therefore it takes very long time for exposing an image. In order to increase the frame rate and gather more light, lens is usually used in a camera while the pinhole camera imaging geometry can still be held. However, due to imperfection of lens, distortion always appears on the perceived images. There are mainly two common types of lens distortions: 1)radial distortion and 2)tangential distortion. Radial distortions arise as a result of the shape of lens, whereas tangential distortions arise from the assembly process of the camera as a whole.

The radial distortion of camera lens can be modeled and corrected by the following equations:

$$\begin{aligned} x'_c &= x' \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y'_c &= y' \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{4.4}$$

where x', y' are the original locations of distorted image point and x'_c, y'_c are the new locations after correction.

Similarly, the tangential distortion can be characterized by the following equations with another two parameters, p_1 and p_2:

$$\begin{aligned} x'_c &= x' + \left[2p_1 y' + p_2 \left(r^2 + 2x'^2\right)\right] \\ y'_c &= y' + \left[p_1 \left(r^2 + 2y'^2\right) + 2p_2 x'\right] \end{aligned} \tag{4.5}$$

here, r is the pixel distance from the principal point and $r^2 = (x' - C_x)^2 + (y' - C_y)^2$.

We can apply undistortion to the images by combining the above radial and tangential distortion model equations after performing the camera calibration as follows:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2 \left(r^2 + 2x_d^2\right) \\ p_1 \left(r^2 + 2y_d^2\right) + 2p_2 x_d y_d \end{bmatrix} \tag{4.6}$$

where $[x_d, y_d]$ and $[x_p, y_p]$ are the distorted and undistorted pixel coordinates respectively and r is the pixel distance from the principal point, i.e. $r^2 = (x_d - C_x)^2 + (y_d - C_y)^2$.

4.1.3 Calibration Toolbox and Results

There are actually many software tools available on the Internet for camera calibration. Most of them are provided by the researchers that incorporated their own techniques for camera calibration. Among all, the camera calibration toolbox for MATLAB [29] and the camera calibration functions provided in the OpenCV library [18] are known to be flexible, reliable and readily available on the Internet for calibrating a camera.

Since the camera calibration toolbox for MATLAB provides more flexibility in parameters control than calibration functions provided in the OpenCV library, and it also has an user interface (see Figure 4.2) for assisting the calibration process while users are required to write their own programs with the calibration functions in OpenCV library in order to calibrate the camera, therefore, calibration for our vision system are mostly carried out with the camera calibration toolbox for MATLAB while the calibration functions in OpenCV library have been used for further verification and comparison.

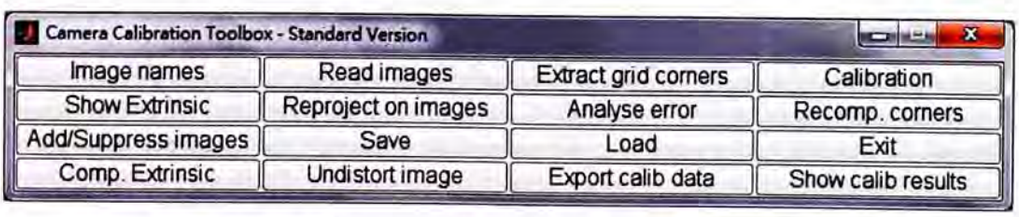
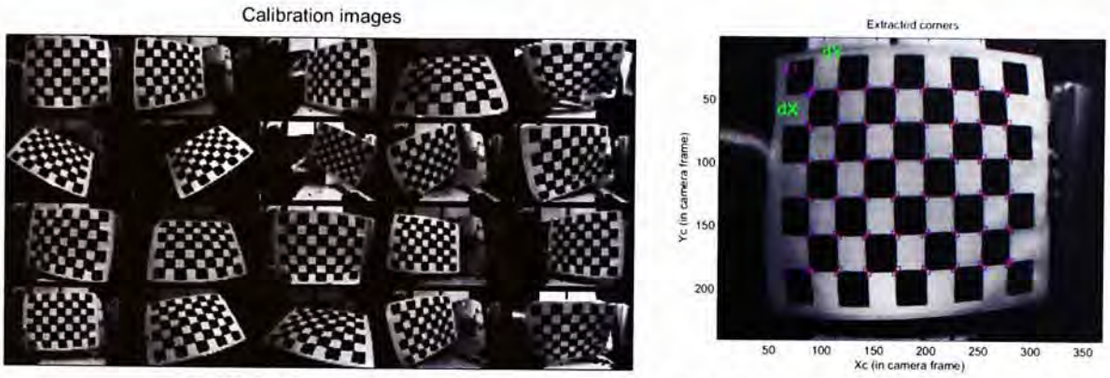


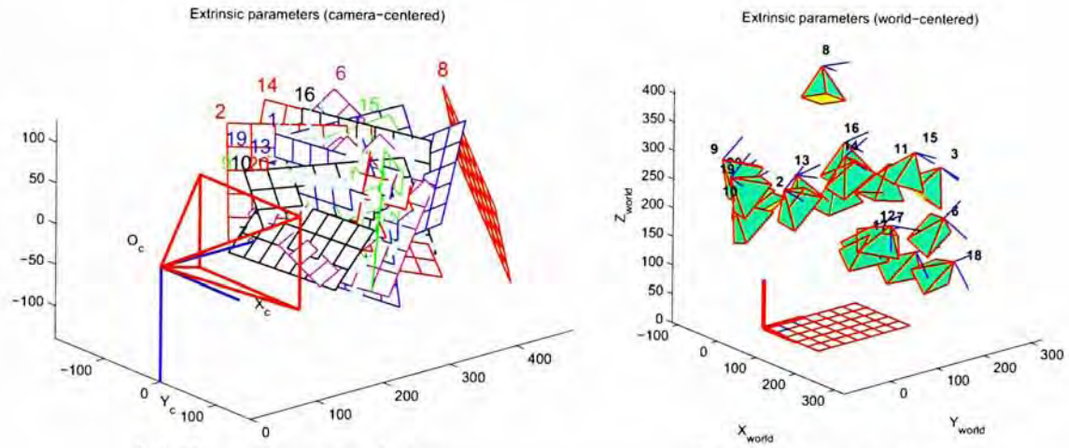
Figure 4.2: GUI of the camera calibration toolbox for MATLAB



(a) All calibration images

(b) Extracted corners on chessboard

Figure 4.3: Corner extraction process for camera calibration



(a) Camera-centered view

(b) World-centered view

Figure 4.4: Extrinsic parameters of calibrated images

The calibration process requires taking images of a planar (i.e. chessboard) pattern that have a rich coverage of the camera's field of view with different board/camera orientations using the camera to be calibrated. So, we have taken 20 images with resolution of 368×240 of a planar chessboard at different orientations using the camera of our vision system with a fixed focus (i.e. fixed focal length) at distant object. We don't have any specification of the lens we used, but we know that the lens is a wide angle lens. After taken the images, then we pass them to the MATLAB calibration toolbox for calibrations.

The calibration is based on exploiting the orthonormal constraints of the homography that maps the corner points on a planar pattern in the object

frame to that appear on the image in the camera frame. Lens distortions are ignored at the beginning, and after determined an initial set of intrinsic parameters, the full sets of parameters including the distortion parameters (with or without initial guess) are then determined by minimizing the reprojection errors using nonlinear optimization. Refer to [84, 30] for further details.

Table 4.1 shows the calibration results obtained with the camera calibration toolbox for MATLAB:

MATLAB calibration results (with uncertainties)	
Focal length: fc	= [285.16389, 294.14669] ± [1.04383, 1.09785]
Principal point: cc	= [200.30611, 112.97155] ± [1.95810, 1.46219]
Skew: alpha_c	= [0.00000] ± [0.00000]
Distortion: kc	= [−0.39674, 0.16797, 0.00019, −0.00009, 0.00000] ± [0.00724, 0.01360, 0.00079, 0.00080, 0.00000]
Pixel error: err	= [0.19588, 0.20687]

Table 4.1: Calibration results using toolbox for MATLAB

OpenCV calibration results	
Focal length: fc	= [285.30484, 294.28702]
Principal point: cc	= [200.77783, 112.46777]
Distortion: kc	= [−0.39837, 0.17061, 0.00028, −0.00023, 0.00000]

Table 4.2: Calibration results using functions in OpenCV library

We have also verified the calibrated results with that obtained using the OpenCV library and found that they are coincided to each other with very small deviations (see Table 4.2).

4.2 Selection of Object Features and Trackers

Object tracking is the process of identifying interested target in images, labeling and keep following it from frame-to-frame in the camera video stream. There are different approaches developed for handling object tracking task [83]. But all of them would required the process of selecting features to describe the target, identify/detect the target in image and, follow and generate the position/trajectory of the target in the video frames perceived by camera.

The choose of object features and method of tracking varies from application to application and the scenarios for which object tracking is carried out. The real-time capability for being operated under a desire processing system is also an important concern.

To describe an object in the image, generally color, edge and texture are used. Among them, color feature would be the simplest compared to edge and texture features. However, color feature would be more sensitive to illumination change than the other two. Until last decade, point features become so popular for object description when they demonstrate their strong invariant to rotation, scale and illumination changes. Beginning from the early Harris corner detector [43], till nowadays, SIFT (Scale-Invariant Feature Transform) [53] and SURF (Speeded Up Robust Features) [25] are the two well known, state-of-the-art robust point descriptor algorithms that widely used for object detection and matching.

Here we consider our tracking problem as being to follow an interested target provided through user input (e.g. select the target in image with a window) and, then labeling it and identifying its position over all the video frames. Since the video frames perceived by the embedded vision system on the UAV platform would mostly be the aerial image of ground landscape, constructions and transports, while the interested target could be a car in red, a building or any specific pattern appearing on ground surface, among all possible image features, color and point features would be considered to be our desired features to be used for tracking interested target on ground for their robustness (point features) and simplicity to use (color feature).

However, from [32], we know that these computational complicated point feature descriptors (SIFT, SURF) cannot be run efficiently on embedded system without being modified and optimized. Since it will take lot more effort to modify the point descriptor algorithms and optimize them for use on the embedded vision system with DSP and/or GPU, we therefore resort to the computational less expensive Harris corner descriptor for being applied to our tracking program running on the embedded system for early testing stage. In the following parts, we will describe the corner descriptor, the color histogram and the kernel based methods for corresponding inter-frame target we used for object tracking.

4.2.1 Harris Corner Detection

The Harris corner detector is to find local pixel region defined by a small window where minimum shift of the window leads to large change in image intensity. Since large intensity changes usually locate at intersection of lines (i.e. corner) or isolated point, so the detector is known as a corner detector.

According to [43] and [41], we can formulate the intensity change E in an image window for a shift of (u, v) as follows:

$$E(u,v) = \sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 \tag{4.7}$$

where $w(x,y)$ is the function specifying the image window: can be a binary or Gaussian function, and $I(x,y)$ is the image intensity at the image position (x,y).

For small value of shifts (u,v), we can approximate $I(x+u, y+v)$ by Taylor expansion up to the first order term as:

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y) \tag{4.8}$$

where I_x and I_y are the partial derivatives of I along x and y direction.

Then we substitute Equation 4.8 into Equation 4.7 to obtain:

$$\begin{aligned} E(u,v) &= \textstyle\sum_{x,y} w(x,y) \left[I(x+u, y+v) - I(x,y) \right]^2 \\ &\approx \textstyle\sum_{x,y} w(x,y) \left[I(x,y) + uI_x(x,y) + vI_y(x,y) - I(x,y) \right]^2 \\ &\approx \textstyle\sum_{x,y} w(x,y) \left[uI_x(x,y) + vI_y(x,y) \right]^2 \\ &\approx [u,v] \left(\textstyle\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) [u,v]^T \end{aligned} \tag{4.9}$$

So the intensity change E can be rewritten as follows for small shift (u,v):

$$E(u,v) \approx [u,v] \, M \, [u,v]^T \tag{4.10}$$

where $M = \begin{bmatrix} \sum_w I_x^2 & \sum_w I_x I_y \\ \sum_w I_x I_y & \sum_w I_y^2 \end{bmatrix}$ is called the Harris matrix (also known as the autocorrelation matrix) which captures the intensity structure [37] of the local neighborhood of image position (x,y) and $\sum_w$ denotes summation over the image window w.

Harris corners are places in the image where the autocorrelation matrix has two large eigenvalues. In developing a test program for tracking an interested target in the perceived images from the camera of the embedded vision system, we make use of the build in feature detection routine (`cvGoodFeaturesToTrack`) of OpenCV library that implements an improved version of Harris corner detector which defines Harris corner as good corners when the smaller eigenvalue of the autocorrelation matrix was greater than a minimum threshold [71].

4.2.2 Color Histogram

Apart from the Harris corner detector, we have also made use of color histogram for object tracking. A color histogram is usually used to represent the color distribution of an object. It is a classical tool used in computer vision for object recognition and object tracking by analyzing the color statistic of object over image frames [30]. It is a simply count on the pixel color that fall into the color range separating by different bins and spanning over the color space. Color histogram of an object is relatively invariant to translation and rotation about its viewing axis but with small variation when its viewing angle changes. And by converting the image in RGB color space to the illumination invariant RG-Chromaticity space, color histogram can also be used in situations with lighting change [3]. For our application on UAV, for example, to observe and keep tracking a red car using the embedded vision system on the UAV cruising at a high altitude. The angular change of viewing axis on the car would be small at such high altitude level while only translational and rotational motion of the car would be prominent. Therefore, color histogram would be a useful tool for us to develop a object tracking system on UAVs.

4.2.3 KLT and Mean-shift Tracker

Generally, the Harris corner and color histogram are only used to identify and describe the object in an image. To keep tracking the object between video frames, we would not repeat the process to extract the Harris corners or computing the color histogram again in every video frame, as these steps usually require hight computational cost. Instead, kernel based tracking methods (trackers) are usually used accompanying with the corner detector and color

histogram created for object tracking. The Kanade-Lucas-Tomasi (KLT) feature tracker and the mean-shift tracker are the two trackers that we have used for our application.

The KLT Feature Tracker

The Kanade-Lucas-Tomasi (KLT) feature tracker [26] is a tracking algorithm based on the Lucas-Kanade optical flow [54, 76] that iteratively running on different levels of the Gaussian pyramid of the image. Through the Lucas-Kanade optical flow, we can approximate the motion of the object that we are interested in an image.

Referring to [30], the derivation of Lucas-Kanade optical flow equation starts by the assumption on brightness consistency, that is inter-frame object intensity remains unchanged although there may be changes in position:

$$I\left[x + u\left(x, y\right), y + v\left(x, y\right), t\right] = I\left(x, y, t - 1\right) \tag{4.11}$$

where I is the image intensity and, u and v are the flow components at image position (x, y) at time frames t and $t - 1$.
Assuming that u and v are small (i.e. small motion), we can make a first-order approximation of $I[x + u(x, y), y + v(x, y), t]$ as:

$$I\left[x + u\left(x, y\right), y + v\left(x, y\right), t\right] \approx I\left[x, y, t\right] + u\left(x, y\right)\frac{\partial I}{\partial x} + v\left(x, y\right)\frac{\partial I}{\partial y} \tag{4.12}$$

where $\partial I/\partial x$ and $\partial I/\partial y$ are the components of the image gradient at (x, y) and we denote them by I_x and I_y.
By substituting Equation 4.12 into Equation 4.11, and denote $I(x, y, t) - I(x, y, t - 1)$ by I_t, we end up obtain the **optical flow equation**:

$$uI_x + vI_y + I_t = 0 \tag{4.13}$$

Since the optical flow equation is under-constrained, as there are two unknown u and v flow directions at a single pixel position (x, y), but we can still solve the equation by assuming a constant flow within the neighborhood (an image window) of the middle pixel.

Considering an image window of size $M \times N$, we can obtain $n = M \times N$ set of flow equations:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn} \end{bmatrix} \tag{4.14}$$

To solve the system of equations, we can set up a least-squares minimization of the equations that is $min\|Ad - b\|^2$, and solve the equations in the standard form of $(A^T A)d = A^T b$.
Therefore, the n set of equations can be expressed as follows:

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b} \tag{4.15}$$

And the solution for the flow is:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \left(A^T A\right)^{-1} A^T b \tag{4.16}$$

which is solvable when $(A^T A)$ is invertible and has eigenvectors with two large eigenvalues. And that happens when the image window is locating at a corner region (corner feature) of the image, which also explain why the algorithm is being used with Harris corner detector for feature tracking.

The Lucas-Kanade optical flow algorithm is developed upon the assumption of small and continuous motion. However, with general camera operating at only 30fps, it is not possible to satisfy such assumption as large and abrupt motion usually happens. In order to deal with the problem, the KLT feature tracking algorithm suggests applying Lucas-Kanade optical flow over different layers of the Gaussian image pyramid. It solves for optical flow beginning from the coarsest image layer (i.e. the top most layer) and use the estimated result as the initial position for next iteration for optical flow calculation at next finer image layer and keep on the iteration process until the origin image

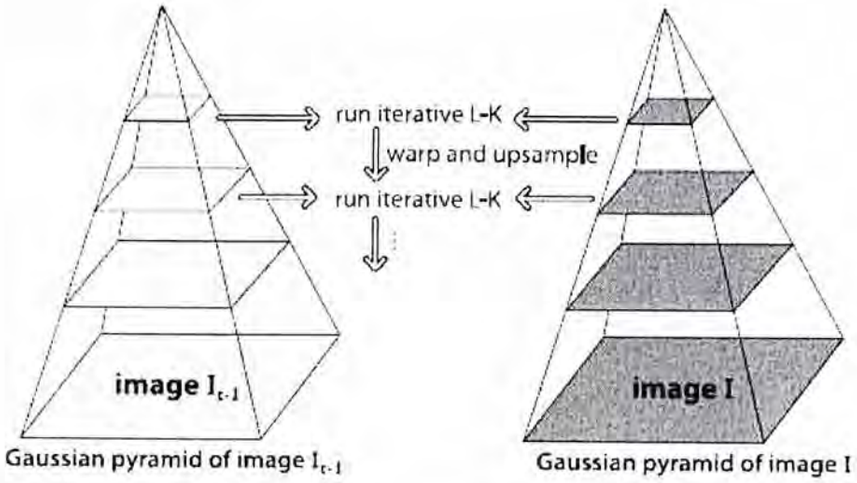


Figure 4.5: Optical flow over Gaussian pyramid of image, from [30]

(i.e. the finest and bottom most layer) is reached (illustrated in Figure 4.5). This approach can minimize situations that violate the motion assumption and applicable to long sequence of motion in video frames. In our project, the sparse optical flow routine (`cvCalcOpticalFlowPyrLK`) in the OpenCV library based on [28] has been used to estimate the positions for the given features in next video frame by computing the Lucas-Kanade optical flow over the image pyramid created. We can specify the region for the object we want to track in the perceived image and extract the corner features in this region, and then perform the pyramid Lucas-Kanade optical flow routine to track the interested object through the estimated feature locations reported. In order to get better estimation for center of the tracked object, a template matching technique using sum of square difference approach is used to match a template image (created by region specifying the object) with the neighborhood region that the features locate.

The Mean-shift Tracker

The mean-shift algorithm is a data analysis technique for finding local maximum of data distribution through computing the gradient of an estimated probability density function.

According to [75], we can use the kernel density estimation approach to estimate the density function $f(x)$ of a data set through convolving the data

with a kernel K of width h, i.e.:

$$f(x) = \sum_i K(x - x_i) = \sum_i k\left(\frac{\|x - x_i\|^2}{h^2}\right) \tag{4.17}$$

where x_i are the data samples and $k(r)$ is the kernel function.
After obtaining the density function, gradient ascent technique can be used for finding local maxima. However, it would be inefficient for computing $f(x)$ over the full searching space. So, the mean-shift algorithm uses a variant of the multiple restart gradient descent approach to estimate the gradient of data distribution instead (a hill-climbing step is used in the computed gradient direction).

By randomly picking sample data point x_i as a local maximum y_k, mean-shift computes the gradient of the density estimate $f(x)$ at y_k by:

$$\nabla f(x) = \sum_i (x_i - x) G(x - x_i) = \sum_i (x_i - x) g\left(\frac{\|x - x_i\|^2}{h^2}\right) \tag{4.18}$$

where $g(r) = -k'(r)$, and $k'(r)$ is the first derivative of $k(r)$ that can be obtained by convolving the inputs with the derivative kernel $G(x)$.
The gradient of the density function can be rewritten as

$$\nabla f(x) = \left[\sum_i G(x - x_i)\right] m(x) \tag{4.19}$$

where

$$m(x) = \left[\frac{\sum_i x_i G(x - x_i)}{\sum_i G(x - x_i)} - x\right] \tag{4.20}$$

is called the mean-shift vector that denotes the difference between the weighted mean of neighbors x_i around x and the current value of x.

By iteratively replacing the current estimated maximum y_k with use of the mean-shift vector obtained in mean-shift procedure, i.e. at iteration $k + 1$:

$$y_{k+1} = y_k + m(y_k) = \frac{\sum_i x_i G(y_k - x_i)}{\sum_i G(y_k - x_i)} \tag{4.21}$$

the algorithm is proved to converge to a local maximum of $f(x)$ when monotonically decreasing kernel function $k(r)$, such as the Epanechnikov kernel

$k_E(r) = max(0.1 - r)$, or the Gaussian kernel $k_N(r) = exp\left(-\frac{1}{2}r\right)$ has been used [33].

The operation of mean-shift tracker based on the mean-shift algorithm can be summarized as follows:

1. Define a search window size
2. Select the initial location of the search window (through user input by selecting the tracking target)
3. Compute the local maximum (centroid) of data distribution in the search window (by mean-shift algorithm)
4. Move the search window to the local maximum (centroid) of data computed in step 3
5. Repeat step 3 and 4 until convergence or the mean-shift magnitude is below a threshold

When performing object tracking in our project, the OpenCV routine function (`cvMeanShift`) that implemented the mean-shift algorithm is used for locating the center of the interested target in image. But instead of using arbitrary set of data points, the color histogram is used as a measure on the feature (color) distribution of the target object. And starting with the searching window being placed over the target object in an image frame, the `cvMeanShift()` function in OpenCV finds the new peak of feature distribution over the object that produces the closest measure to the target in next image frame. By this, moving color object can be tracked over image sequence. Further details regarding the mean-shift algorithm and how it is being used for tracking non-rigid object can refer to [33, 34].

4.3 Target Auto-centering

When performing surveillance or target tracking with a fast flying UAV, it is common that interested target will move out the camera's field of view so quickly if the camera can not keep an orientation directing to the target. It is therefore necessary that we have a controller for the pan-tilt platform which can maintain the interested target in the middle of images perceived by the

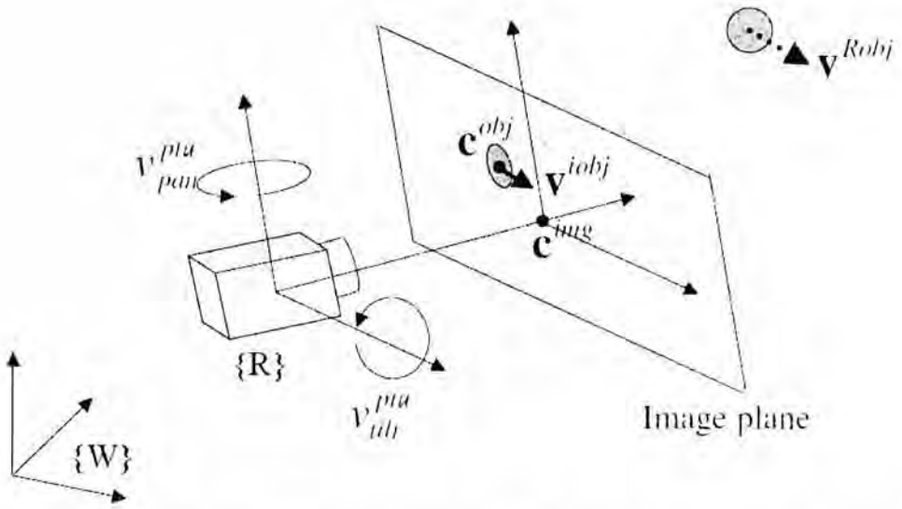


Figure 4.6: Coordinate assignment for the pan-tilt system, from [61]

attached camera such that the flight system can have sufficient time to adjust the flight path for which the target can be kept in track.

To achieve this, we can make use of the retrieved image as a visual feedback to estimate the control inputs required to orientate the pan-tilt platform for the camera to direct to the tracked target. For that, we can simply use pixel error between centers of the tracking window to that of the retrieved image and multiplied by a proportional gain as the inputs. However, on board experiments showed that the performance of this kind of controller is not responsive (if small gain) and stable (if large gain) enough. Therefore, a full PID controller would be required to provide a fast and stable control over the camera pan-tilt platform for target tracking and centering.

4.3.1 Formulation of the PID Controller

An angular velocity PID controller for controlling pan-tilt platform has been suggested by [62], which is also mentioned by [61] in English. The PID controller they designed is based on the estimation that under a fixed-viewpoint camera configuration [80], the angular velocity of the target in the world coordinate can be computed by the sum of the angular velocity of the pan-tilt platform itself and the angular velocity of the target in the robot/body frame originating at the rotating center of pan-tilt platform (see Figure 4.6).

They have formulated their PID controller as follows [61]:

$$v^u(k) = K_p \cdot \frac{1}{\alpha}\sum_{i=0}^{\alpha} v^{obj}(k-i) + K_i \cdot v^{dt}(k) + K_d \cdot \frac{1}{\beta}\sum_{i=0}^{\beta} a(k-i)\,\Delta t \quad (4.22)$$

where K_p, K_i and K_d are constant controller gains. And here in the equation, $v^u(k) = [v^u_{pan}(k), v^u_{tilt}(k)]$ is the output from the PID controller, at time instant k. $v^{obj}(k)$ denotes the angular velocity of the target object in the world coordinate $\{W\}$. $v^{dt}(k)$ denotes the tracking error in the image plane. And $a(k)$ is the linear acceleration of the target object in the world coordinate.

As mentioned above, the angular velocity of target object $v^{obj}(k)$ in the world coordinate $\{W\}$ is estimated by the sum of the target angular velocity $v^{Robj}(k)$ in the robot/body frame $\{R\}$ and the angular velocity $v^{ptu}(k)$ of the pan-tilt unit itself. So, $v^{obj}(k)$ can be represented as follows:

$$v^{obj}(k) = v^{Robj}(k) + v^{ptu}(k) \quad (4.23)$$

And, $v^{Robj}(k)$ can be estimated based on the target velocity $v^{iobj}(k)$ in the image plane that computed by the difference of target centroid in successive image frames:

$$\begin{aligned} v^{Robj}(k) &= f \cdot v^{iobj}(k) \\ &= f \cdot \left(c^{obj}(k) - c^{obj}(k-1)\right)/\Delta t \end{aligned} \quad (4.24)$$

where $c^{obj}(k)$ and Δt denote the coordinate of the target centroid in the image plane at time k and the sampling time respectively, and f is the focal length of the camera.

The tracking error $v^{dt}(k)$ in the image plane at time instant k is obtained using the coordinate of the target centroid in the image $c^{obj}(k)$ and the coordinate of the image center c_0^{img}:

$$v^{dt}(k) = \frac{f \cdot \left(c^{obj}(k) - c_0^{img}\right)}{\Delta t} \quad (4.25)$$

And, by assuming that the target object moves at a constant acceleration

during one sample time, the linear acceleration $a(k)$ is then represented as:

$$a\left(k\right)=\frac{v^{obj}\left(k\right)-v^{obj}\left(k-1\right)}{\Delta t} \tag{4.26}$$

In the above angular velocity PID controller, instead of using the angular velocity and linear acceleration at a time instant, the mean velocity and the mean acceleration in the past α and β frames are used respectively to help stabilize the pan-tilt system.

Convert Velocity Control to Displacement Control

As the orientation of our pan-tilt platform is controlled by two RC servos, only angular displacement of the servos can be controlled through varying their PWM signal inputs. Therefore, the angular velocity PID controller cannot directly applied to our system. However, it is possible to convert their controller from velocity control to displacement control for our implementation.

Here we rewrite the above angular velocity PID controller as follows:

$$\omega_{pt}\left(k+1\right)=K_p\cdot\omega_{obj}\left(k\right)+K_i\cdot\theta_{err}\left(k\right)+K_d\cdot\alpha_{obj}\left(k\right)\Delta t \tag{4.27}$$

where K_p, K_i and K_d are constant controller gains. Similar to that of the original angular velocity PID controller, in our equation, $\omega_{pt}(k+1)$ is the output from the PID controller. $\omega_{obj}(k)$ denotes the angular velocity of the target object in the world coordinate $\{W\}$, at time instant k. $\theta_{err}(k)$ denotes the tracking error in the image plane. And $\alpha_{obj}(k)$ is the angular acceleration of the target object in the world coordinate. But here, we do not use the mean velocity and the mean acceleration, as we are converting the angular velocity PID controller to angular displacement PID controller. Using a mean angular displacement between several image frames does not help stabilize the control of the pan-tilt system.

Same to the original velocity PID controller, we can rewrite $\omega_{obj}(k)$, $\theta_{err}(k)$ and $\alpha_{obj}(k)$ as follows:

$$\begin{aligned}\omega_{obj}\left(k\right)&=\omega_{pt}\left(k\right)+\omega_{obj}^{pt}\left(k\right)\\&=\omega_{pt}\left(k\right)+v_{obj}^{img}\left(k\right)/f\end{aligned} \tag{4.28}$$

$$\theta_{err}(k) = \frac{c_{obj}^{img}(k) - c_0^{img}(k)}{f} \tag{4.29}$$

$$\alpha_{obj}(k) = \frac{\omega_{obj}(k) - \omega_{obj}(k-1)}{\Delta t} \tag{4.30}$$

where $\omega_{obj}^{pt}(k)$ is the angular velocity of the object with respect to the rotating center of pan-tilt (i.e. the robot/body frame $\{R\}$), $v_{obj}^{img}(k)$ is the linear velocity of the object with respect to the image coordinate and c_0^{img} is the center of the retrieved image.

To convert the velocity control to displacement control, we multiply Δt to both sides of the modified PID controller in Equation 4.27, and we obtain:

$$\begin{aligned}
\omega_{pt}(k+1)\cdot\Delta t &= \left(K_p\cdot\omega_{obj}(k) + K_i\cdot\theta_{err}(k) + K_d\cdot\alpha_{obj}(k)\,\Delta t\right)\cdot\Delta t \\
\\
\theta_{pt}(k+1) &= K_p\cdot\left(\omega_{pt}(k) + v_{obj}^{img}(k)/f\right)\cdot\Delta t \\
&\quad + K_i\cdot\left(c_{obj}^{img}(k) - c_0^{img}\right)/f\cdot\Delta t \\
&\quad + K_d\cdot\left(\omega_{obj}(k) - \omega_{obj}(k-1)\right)\cdot\Delta t \\
\\
\theta_{pt}(k+1) &= K_p\cdot\left(\theta_{pt}(k) + \left(c_{obj}^{img}(k) - c_{obj}^{img}(k-1)\right)/f\right) \\
&\quad + K_i\cdot\left(c_{obj}^{img}(k) - c_0^{img}\right)/f\cdot\Delta t \\
&\quad + K_d\cdot\left(\theta_{pt}(k) + v_{obj}^{img}(k)/f\cdot\Delta t - \theta_{pt}(k-1) - v_{obj}^{img}(k-1)/f\cdot\Delta t\right) \\
\\
\theta_{pt}(k+1) &= K_p\cdot\left(\theta_{pt}(k) + \left(c_{obj}^{img}(k) - c_{obj}^{img}(k-1)\right)/f\right) \\
&\quad + K_i\cdot\left(c_{obj}^{img}(k) - c_0^{img}\right)/f\cdot\Delta t \\
&\quad + K_d\cdot\left(\theta_{pt}(k) - \theta_{pt}(k-1) + \left(c_{obj}^{img}(k) - 2c_{obj}^{img}(k-1) + c_{obj}^{img}(k-2)\right)/f\right)
\end{aligned} \tag{4.31}$$

where $\theta_{pt}(k)$ and $\theta_{pt}(k-1)$ is the current and last angular position of the RC servos on the pan-tilt platform respectively, while $\theta_{pt}(k+1)$ is the new angular position required and to be commanded to the servo controller, and $c_{obj}^{img}(i)$ is the center of target object in the image coordinate at time instant i, where $i = k, k-1, k-2...$

In order to maintain the stability of the pan-tilt system, i.e. $\theta_{pt}(k+1)$ should be equal to $\theta_{pt}(k)$, whenever there is no change on center of target object in between retrieved image frames, i.e. $c_{obj}^{img}(k) - c_{obj}^{img}(k-1) = 0$, the proportional gain K_p in the controller is deliberately set to 1.0.

4.3.2 Control Gain Settings and Tuning

After implementing the angular displacement PID controller on the pan-tilt camera system utilizing visual feedback, we have performed on board tests to tune the controller gains on-line with the system. As mentioned earlier, the proportional gain K_p of the controller is set to 1.0, therefore we have two other controller gains required for tuning. The tuning strategy is to tune the integral controller gain K_i first, which controls the response of the pan-tilt system to keep the tracked target at the image center. Then, we have to set the derivative gain K_d to reduces the system overshoot. And, if required, re-adjust the K_i gain to maintain the system stability.

Finally, we have tuned three sets of controller gains (see Table 4.3). Two sets of them contain all non-zero values for the controller gains, while one set of them having the derivative gain set to zero, i.e. only PI control involved.

	Proportional Gain (K_p)	**Integral Gain** (K_i)	**Derivative Gain** (K_d)
Set 1	1.0	0.0095	0.0395
Set 2	1.0	0.012	0.2
Set 3	1.0	0.02	0.0

Table 4.3: Gain settings for the angular displacement PID controller

4.4 Geo-locating of Tracked Target

When we are able to track an interested target on the image perceived from the embedded vision system and maintain the target within the camera's field of view (preferably at the image center), it would be great that if we could also locate the target in the geodetic coordinate, such that the position information could be used, for example, to plan the flight path of UAV in order to keep the target in track or to deliver emergency package for a person who get lost in outback region in search and rescue mission, like the competition appearing in [20].

4.4.1 Coordinate Frame Transformation

As mentioned in Section 4.1, a series of coordinate frame transformations are required before we can relate the location of object capture in image frame to the location expressed in geodetic coordinates even we have got the pixel to metric relation from camera calibration. The transformations include the conversion from the geodetic coordinates to local navigation frame and from the local navigation frame to the camera coordinates.

From Geodetic Coordinates to Local Navigation Frame

In the following part, we show the equations for interconversion between the geodetic coordinates and the local navigation frame (i.e. the local east, north, and up, ENU coordinates). They are directly obtained from [5], while more in depth discussion on the equations can be found in [85].

A. Conversion from geodetic coordinates to local ENU coordinates involves two steps:

1. Convert geodetic coordinates to Earth-Center-Earth-Fix (ECEF) coordinates

$$\begin{aligned} X &= \left(\frac{a}{\chi} + h\right) cos\phi\ cos\lambda \\ Y &= \left(\frac{a}{\chi} + h\right) cos\phi\ sin\lambda \\ Z &= \left(\frac{a\left(1-e^2\right)}{\chi} + h\right) sin\phi \end{aligned} \tag{4.32}$$

   where $[X, Y, Z]$ are the ECEF coordinates, $\phi = latitude$, $\lambda = longitude$, $h = height$, $\chi = \sqrt{1 - e^2 sin^2\phi}$, a and e^2 are the semi-major axis and the square of the first numerical eccentricity of the ellipsoid respectively

2. Convert ECEF coordinates to local ENU coordinates

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -sin\lambda & cos\lambda & 0 \\ -sin\phi\ cos\lambda & -sin\phi\ sin\lambda & cos\phi \\ cos\phi\ cos\lambda & cos\phi\ sin\lambda & sin\phi \end{bmatrix} \begin{bmatrix} X_p - X_r \\ Y_p - Y_r \\ Z_p - Z_r \end{bmatrix} \tag{4.33}$$

   where $[x, y, z]$ are the ENU coordinates, $[X_p, Y_p, Z_p]$ are the object ECEF coordinates and $[X_r, Y_r, Z_r]$ are the reference ECEF coordinates

B. Conversion from local ENU coordinates back to geodetic coordinates also involves two steps:

1. Convert local ENU coordinates to ECEF coordinates

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -sin\lambda & -sin\phi\ cos\lambda & cos\phi\ cos\lambda \\ cos\lambda & -sin\phi\ sin\lambda & cos\phi\ sin\lambda \\ 0 & cos\phi & sin\phi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} \tag{4.34}$$

2. Convert ECEF coordinates to geodetic coordinates

$$\begin{aligned} r &= \sqrt{X^2 + Y^2} \\ E^2 &= a^2 - b^2 \\ F &= 54b^2Z^2 \\ G &= r^2 + (1-e^2)\,Z^2 - e^2E^2 \\ C &= \frac{e^4Fr^2}{G^3} \\ S &= \sqrt[3]{1 + C + \sqrt{C^2 + 2C}} \\ P &= \frac{F}{3\left(S + \frac{1}{S} + 1\right)^2 G^2} \\ Q &= \sqrt{1 + 2e^4P} \\ r_0 &= \frac{-\left(Pe^2r\right)}{1+Q} + \sqrt{\frac{1}{2}a^2\left(1 + 1/Q\right) - \frac{P(1-e^2)Z^2}{Q(1+Q)}} - \frac{1}{2}Pr^2 \\ U &= \sqrt{\left(r - e^2r_0\right)^2 + Z^2} \\ V &= \sqrt{\left(r - e^2r_0\right)^2 + (1-e^2)\,Z^2} \\ Z_0 &= \frac{b^2Z}{aV} \end{aligned}$$

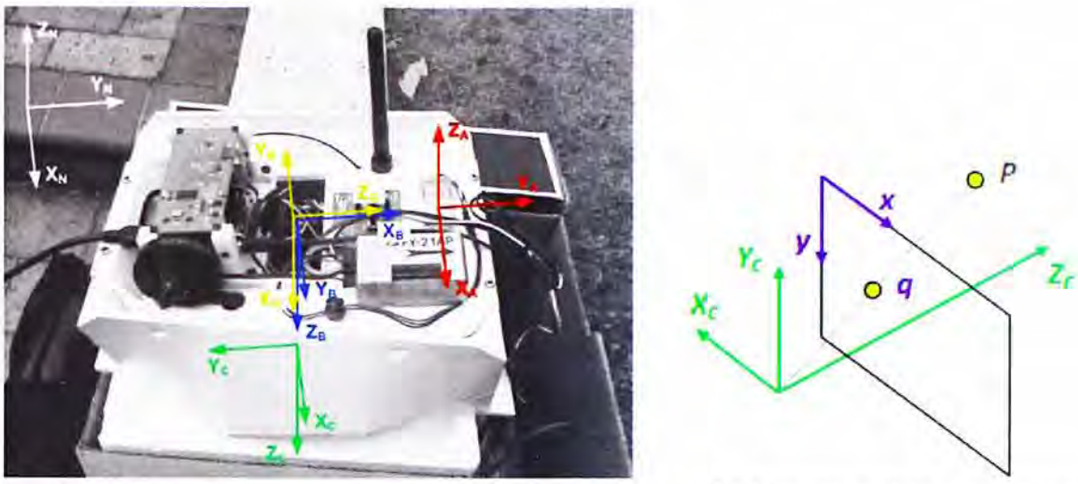
such that,

$$\begin{aligned} h &= U\left(1 - \frac{b^2}{aV}\right) \\ \phi &= arctan\left[\frac{Z + e'^2Z_0}{r}\right] \\ \lambda &= arctan2\left[Y, X\right] \end{aligned} \tag{4.35}$$

where b and e' are the semi-minor axis and second eccentricity of the ellipsoid respectively

From Local Navigation Frame to Camera Coordinates

Figure 4.7 defines the coordinate frames assigned to our experimental setup (the 2nd version pan-tilt platform) which is configured for testing with the



(a) Coordinate frames on pan-tilt platform (b) Camera and image frames

Figure 4.7: Coordinate frames assigned to experimental setup

method that used for geo-locating tracked target in the perceived images from the vision system. In the figures, $[X_N, Y_N, Z_N]$ is the local navigation frame, $[X_A, Y_A, Z_A]$ denotes the GPS antenna frame (has exact orientation as the local navigation frame but originated at the GPS antenna), $[X_B, Y_B, Z_B]$ is the body frame of the system mounting platform, $[X_G, Y_G, Z_G]$ is the gimbal frame (originated at the rotating center of the pan-tilt platform), and $[X_C, Y_C, Z_C]$ denotes the camera frame.

The transformation from the local navigation frame to the camera frame involves a number of homogeneous transformation matrices. They are obtained based on the assigned coordinate frames and listed as follows:

I. Transformation from navigation frame to antenna frame (^AT_N)

$$^AT_N = \begin{bmatrix} I & -^NP_A \\ 0 & 1 \end{bmatrix} \tag{4.36}$$

where ^NP_A denotes the location of GPS antenna in the navigation frame

II. Transformation from antenna frame to body frame (^BT_A)

$$^BT_A = \begin{bmatrix} ^AR_B{}^T & -^AR_B{}^T\ ^AP_B \\ 0 & 1 \end{bmatrix} \tag{4.37}$$

where ${}^{A}R_{B} = {}^{A}R_{BF}\ {}^{BF}R_{B}$ denotes the rotation from body frame to antenna frame and BF denotes the body fixed frame

$$ {}^{A}R_{BF} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad {}^{BF}R_{B} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} $$

$$ so,\ {}^{A}R_{B} = {}^{A}R_{BF}\ {}^{BF}R_{B} = \begin{bmatrix} c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{bmatrix} $$

such that $\phi = roll\ angle$, $\theta = pitch\ angle$, $\psi = yaw\ angle$; $s() = sin()$, $c() = cos()$; and ${}^{A}P_{B} = [0.005, -0.07, -0.03]^{T}$ is the vector from center of body frame to center of antenna frame

III. Transformation from body frame to gimbal frame ($^{G}T_{B}$)

$$ {}^{G}T_{B} = \begin{bmatrix} {}^{B}R_{G}{}^{T} & 0 \\ 0 & 1 \end{bmatrix} \tag{4.38} $$

where ${}^{B}R_{G} = {}^{B}R_{GF}\ {}^{GF}R_{G}$ denotes the rotation from gimbal frame to body frame and GF denotes the gimbal fixed frame

$$ {}^{B}R_{GF} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad {}^{GF}R_{G} = \begin{bmatrix} c\theta c\psi & -s\psi & s\theta c\psi \\ c\theta s\psi & c\psi & s\theta s\psi \\ -s\theta & 0 & c\theta \end{bmatrix} $$

$$ so,\ {}^{B}R_{G} = {}^{B}R_{GF}\ {}^{GF}R_{G} = \begin{bmatrix} -s\theta & 0 & c\theta \\ -c\theta s\psi & -c\psi & -s\theta s\psi \\ c\theta c\psi & -s\psi & s\theta c\psi \end{bmatrix} $$

such that $\theta = tilt\ angle$, $\psi = pan\ angle$; $s() = sin()$, $c() = cos()$

IV. Transformation from gimbal frame to camera frame ($^{C}T_{G}$)

$$ {}^{C}T_{G} = \begin{bmatrix} {}^{G}R_{C}{}^{T} & -{}^{G}R_{C}{}^{T}\ {}^{G}P_{C} \\ 0 & 1 \end{bmatrix} \tag{4.39} $$

where ${}^{G}R_{C}$ denotes the rotation from camera frame to gimbal frame

$$ {}^{G}R_{C} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} $$

and ${}^{G}P_{C} = [0.05, 0.0, 0.0]^{T}$ is the vector from camera center to the gimbal center

Therefore, the overall **transformation from local navigation frame to the camera coordinates** will be:

$$ {}^{C}T_{N} = {}^{C}T_{G}\ {}^{G}T_{B}\ {}^{B}T_{A}\ {}^{A}T_{N} \tag{4.40} $$

4.4.2 Depth Estimation and Target Locating

As we know, image captured through camera only retains 2D information of a 3D scene. The depth of objects in the scene is lost during the process. In order to geo-locate object captured in 2D image inside a 3D world, we have to find ways that estimate the depth information. There is an approach suggested by [24] that try to estimate the image depth through vector addition while it only requires to know the image pixel location of the tracked target, the position of the UAV, the camera orientation and the relative altitude between the UAV and the tracked target in order to localize the target in the world coordinate.

Actually, we would have to know the terrain elevation in order to obtain the relative altitude between the ground target and the UAV. However, as we are performing target geo-locating on UAV that flies with high altitude while the height change on ground surface would be small in comparison, so we can assume that the relative altitude to the ground target be the flying altitude of the UAV. Therefore, we have adopted the approach suggested in [24] for estimating the image depth and geo-locate the tracked target with use of our vision system. The approach has also been applied in UAV project appearing in [35], and obtain satisfactory results.

Estimating Image Depth

In Section 4.1, we have obtained the camera intrinsic matrix with the perspective camera model that relates pixel unit in the image to the metric unit in the 3D camera coordinates. Here similarly, we relate an image point q to it's 3D position ${}^{C}P$ in the camera coordinates (see Figure 4.8) in homogeneous coordinates with the intrinsic matrix as follows:

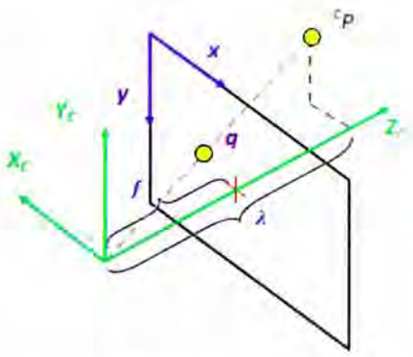


Figure 4.8: Projection of a point in camera frame to image frame

$$\lambda q = K\ {}^{C}P \implies \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_C \tag{4.41}$$

where λ is the scaling factor and K is the camera intrinsic matrix.

Therefore, from the transformation matrices obtained in previous part, we can relate the object position in an image to that in the local navigation frame by:

$$\lambda q = K\ {}^{C}T_N\ {}^{N}P \tag{4.42}$$

where ${}^{N}P$ is the object position in the local navigation frame and ${}^{C}T_N$ is the transformation matrix that transform vector in the local navigation frame to the camera frame.

According to [24], the origin of camera coordinates (i.e. the camera's optical center) and the object position in the camera frame have to be resolved in the local navigation frame first for deriving the equations to estimate the scale factor. So, we denote ${}^{C}P_{CC} = [0, 0, 0, 1]^T$ be the camera's optical center in the camera coordinates and $q_{obj} = [x_q, y_q, 1, 1]^T$ be the object position in the camera coordinates. Here, the position of image plane is regarded to be locating at one unit length of the camera focal length (f). And we obtain:

$${}^{N}P_{CC} = {}^{C}T_N{}^{-1}\ {}^{C}P_{CC} \tag{4.43}$$

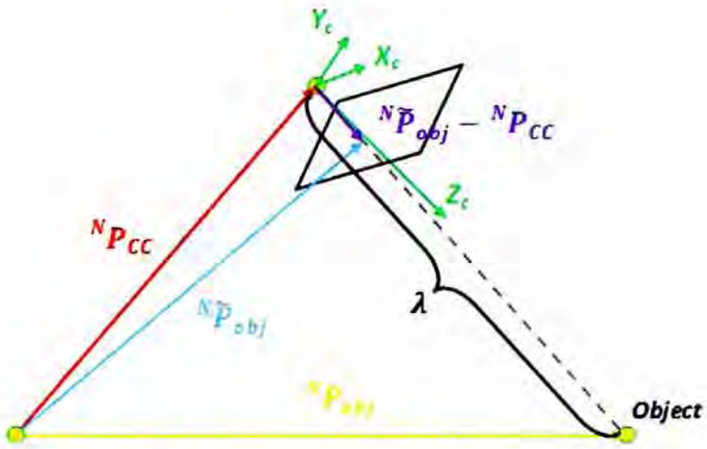


Figure 4.9: Vector relations for depth estimation

$$^{N}\tilde{P}_{obj} = \left(K\ ^{C}T_{N}\right)^{-1} q_{obj} \tag{4.44}$$

Figure 4.9 describes the vector relations between the location of the camera center in navigation frame ($^{N}P_{CC}$), the image depth (λ), and the location of the object in navigation frame ($^{N}\tilde{P}_{obj}$).

If the tracked target object is close to the image center, by means of vector addition, we can get following relation:

$$^{N}P_{obj} = {}^{N}P_{CC} + \lambda\left(^{N}\tilde{P}_{obj} - ^{N}P_{CC}\right) \tag{4.45}$$

And with a flat terrain model assumption (i.e. the altitude of the target object being zero), we can obtain the scale factor by the following relationship:

$$0 = {}^{N}z_{obj} = {}^{N}z_{CC} + \lambda\left(^{N}\tilde{z}_{obj} - ^{N}z_{CC}\right)$$

$$\lambda = \frac{^{N}z_{CC}}{^{N}z_{CC} - ^{N}\tilde{z}_{obj}} \tag{4.46}$$

By the estimated scaling value (λ), we are able to obtain the geo-location of the tracked object from the perceived image with Equation 4.45 and the transformation equations listed in Section 4.4.1 after performing image undistortion.

Moreover, as geometric uncertainty and sensor errors get involved in the geo-locating process, therefore a running averaging filter is applied to the instance geo-location estimates we obtained with the above approach for enhancing the accuracy of estimation.

4.5 Results and Discussion

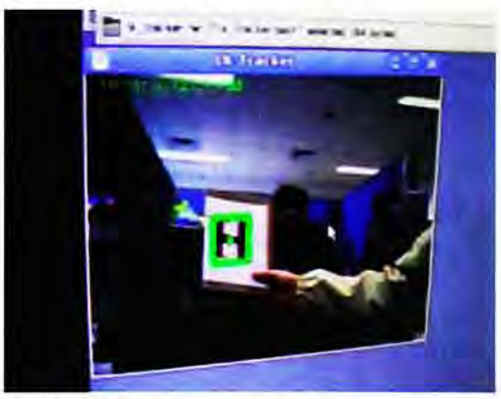
In this section, we present the experimental results we get using the approaches discussed in former parts of the chapter for tracking target with the KLT feature tracker and the mean-shift tracker, evaluate the performance of target auto-centering on the camera pan-tilt platform and the geo-locating accuracy for estimating a ground infra-red (IR) illuminator in the geographical coordinates.

Tracking with KLT Feature Tracker

As mentioned in Section 4.2.3, we have developed a program on the embedded vision system with the OpenCV library that implements the KLT feature tracker. The experiment we performed was based on the early version of the camera module (with MT9T001 image sensor) and the system interfacing board that we have developed (refer to Section 3.2). The image resolution of the perceived image was configured to 480×320 and the maximum number of corners to extract with the program was set to 50 with a 3-level Gaussian of image pyramid being used for tracking. Figure 4.10a and Figure 4.10b show the experimental setup and the KLT feature tracker that ran on the embedded system for tracking a letter 'H' printed on a cardboard.

The tracker could keep track with the selected letter under a gentle move-

(a) Experimental setup for object tracking

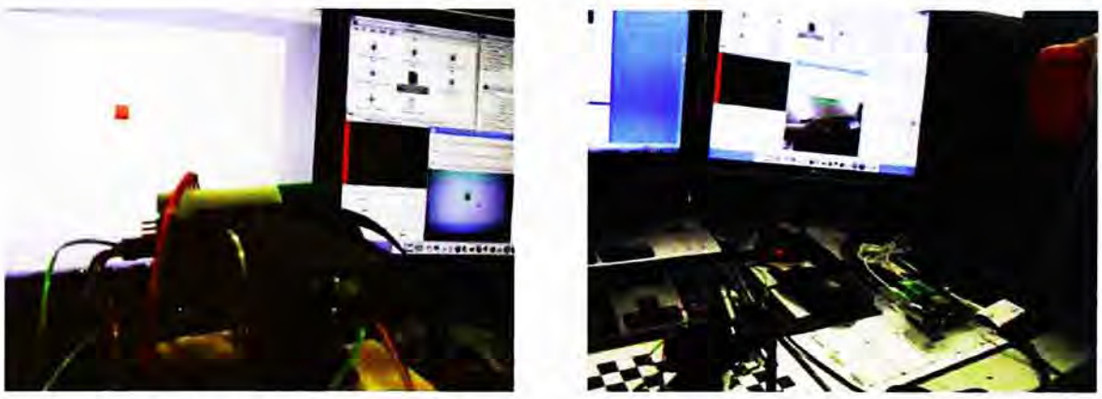


(b) KLT feature tracker running on the embedded system

Figure 4.10: Experiment on object tracking using KLT feature tracker

ment and rotation of the cardboard. However, there was a latency of around a second for the images being processed and shown on the system display. Although we had already reduced the maximum number of corners to extract and level of Gaussian image pyramid being used, the performance was not that good as expected. The major reason for the long latency would possibly due to the high computation cost required to build the Gaussian image pyramid. At the current stage, it would be hard to further reduce the latency without offloading the expensive image pyramid creation process to the DSP and/or GPU on the embedded system. And it would possibly not able to obtain satisfactory performance if the tracker is used with the PID controller for target auto-centering. Therefore, we had decided not to use the KLT feature tracker for object tracking before we managed to use DSP and/or GPU for co-processing with the ARM CPU core on the embedded system.

Tracking and Auto-centering with Mean-shift Tracker



(a) Tracking of color blob under uniform motion

(b) Tracking of color box with fast free hand movement

Figure 4.11: Experiment on target tracking and auto-centering using mean-shift tracker

To determine the performance of the mean-shift color tracker and the angular displacement PID controller on the pan-tilt platform for target auto-centering, two sets of experiments were done to evaluate the system response and centering error when tracking fast moving targets. The first experiment was to track a red color blob shown on LCD display which undergoing uniform motion in an '∞' shape that animated by a MATLAB program. The second

experiment was to track a red color box that moved freely holding by hand. The experimental setup we used was the new camera system with the standard pan-tilt platform (see Section 3.3). We have also hacked the RC servos to obtain the voltage values across their potentiometer for having an immediate measurement of the servo angular positions (by interpolating the voltage at current angular position to the voltages at minimum and maximum servo angles). Figure 4.11a and Figure 4.11b show the setup and the experiments we had performed.

In both of the experiments, we had used the first set of control gains appearing in Table 4.3 for the PID controller as they had shown to be more responsive and stable. The program for tracking the color box was developed with the OpenCV library that implements the mean-shift tracker and made use of the tracked target center as visual feedback for the angular displacement PID controller to maintain the target within the camera's field of view and keep it close to the middle of perceived image.

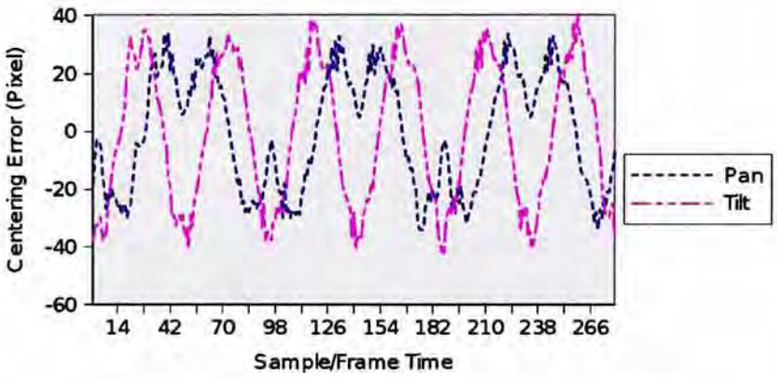


Figure 4.12: Centering error for a color blob with uniform motion

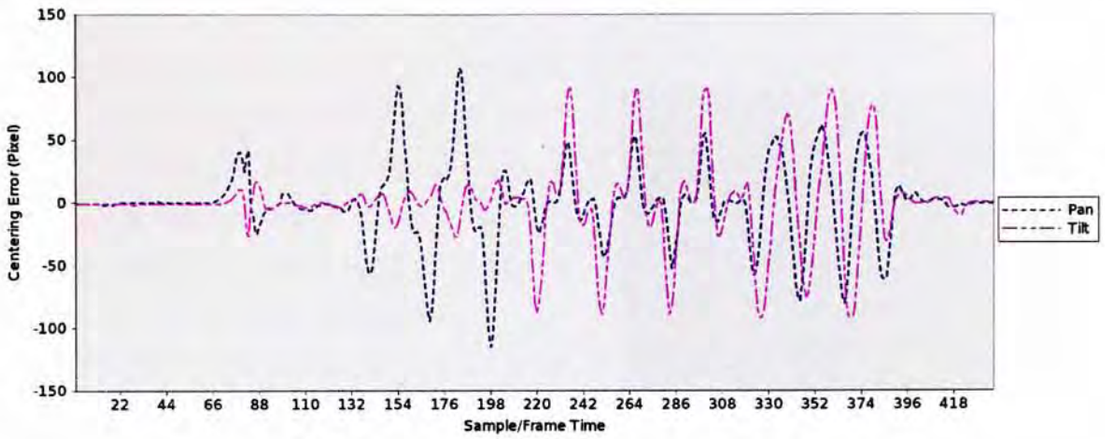
unit = pixel	Pan		Tilt	
	Error	**S.D.**	**Error**	**S.D.**
Our tracking system (image size: 368 × 240)	33.92 (max.)	20.26	42.57 (max.)	24.20
System used in [62] (image size: 640 × 480)	44.53 (avg.)	81.93	25.78 (avg.)	25.80

Table 4.4: Target centering error on the tracking systems

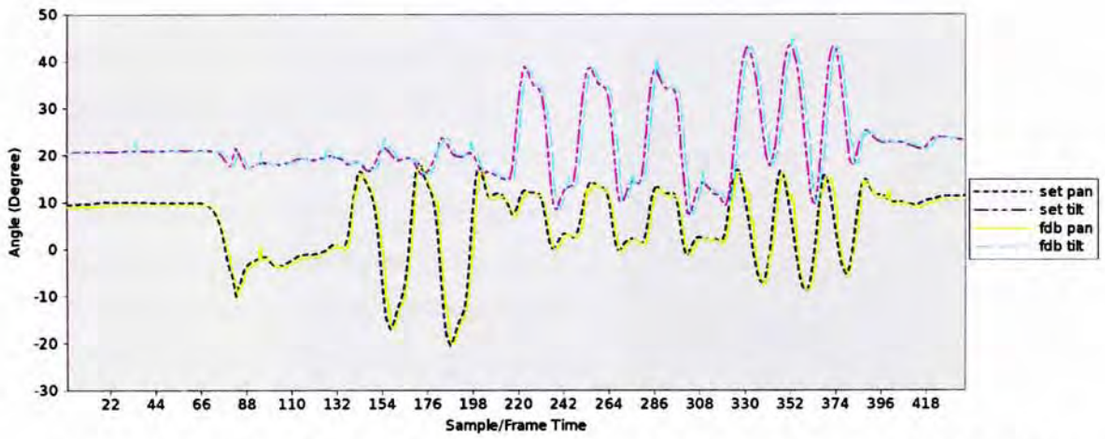
Figure 4.12 shows the centering error of the red color blob being tracked with the mean-shift tracker in 3 cycles of uniform motion generated by a MATLAB program. The maximum centering error for the pan axis is 33.92 pixels, and that for the tilt axis is 42.57 pixels. The standard deviation for pan and tilt axis is 20.26 pixels, and 24.20 pixels respectively. Here we can not have fair comparison on the performance of our tracking system using our modified angular displacement PID controller to the original angular velocity PID controller applied in the tracking system appearing in [62], as we are not able to produce the same tracking target setup and condition that performed in [62]. The target centering errors are also affected by the speed limit of the pan-tilt platform besides the controller being used. However, we still list both of our experimental results in Table 4.4 for a general evaluation on our tracking and target centering performance with the embedded vision system.

Figure 4.13a to Figure 4.13c also show the centering error, the angular positions commanded to the pan and tilt channel and the actual angular positions of the pan and tilt axis recorded in the experiment for tracking a red color box that moved freely holding by hand. The maximum centering error for the pan axis is 115.28 pixels, and that for the tilt axis is 91.79 pixels. The standard deviation for pan and tilt axis is 30.46 pixels, and 31.56 pixels respectively. The results also show that there is a latency of maximum 9.67° on the pan axis, and a latency of maximum 9.71° on the tilt axis for the current angle position commanded and that it could reach in the next sampling time. Therefore, a large error can be included in calculating the target geo-location if we take the current commanded panning and tilting angles for use instead of the immediate angular position of servos measured through the use of voltage value obtained over the potentiometer in the servo itself.

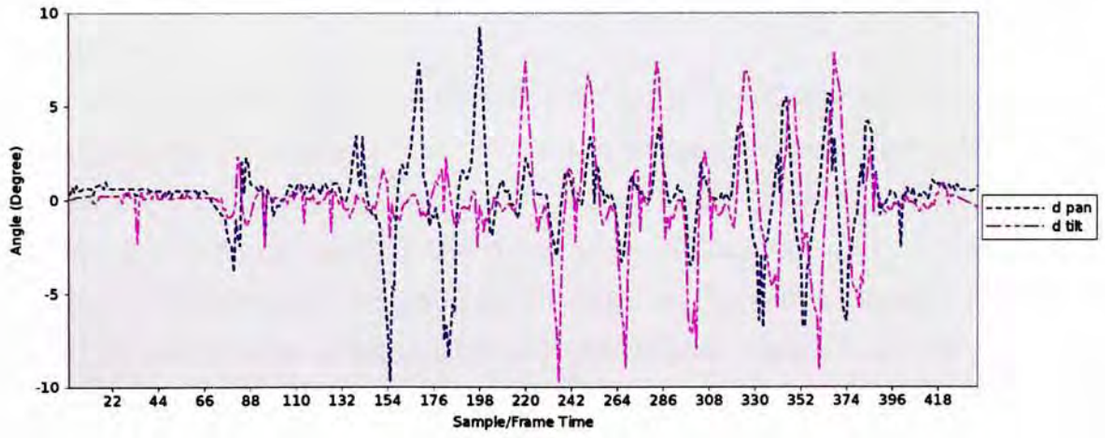
Moreover, there are glitches appeared in the measurement for the angular position of RC servos through the potentiometer voltage obtained from ADC (Analog to Digital Converter) on micro-controller unit. Therefore, further signal filtering circuit (e.g. RC circuit) would be required to remove the signal noise in order to reduce errors that could have introduced to the target geo-locating process when we make use of these measurement.



(a) Target centering error on pan and tilt axis



(b) Angular positions commanded and actual values recorded on pan and tilt axis



(c) Latency by comparing angular positions commanded and actual values recorded

Figure 4.13: Experiment record on the centering error, position commanded and the control latency of the tracking system for tracking free moving color box

Geo-locating an IR Illuminator on Ground

At the beginning of Section 4.4, we have mentioned that the geo-location information of ground target would be useful for search and rescue mission such as the competition appearing in [20]. In the UAV search and rescue challenge, participants have to develop an UAV that is capable of locating a dummy (wearing high visibility shirt, jeans, with an infra-red heat source nearby) and delivering a bottle of water (no less than 500ml) to it. In order to detect and locate the dummy, participants can handle this with manual detection and locating it using the analog video down streamed from UAV to ground station or do it automatically for extra points. Therefore, we would like to know if it is possible to use our vision system to tackle such a practical task for searching and geo-locating the ground target.

Figure 4.14: Dummy used in search and rescue challenge

Figure 4.15: Setup for detecting IR point source in indoor

In order to geo-locate the dummy, we have to identify the visual cue for determine and detecting it first. Since the dummy posture and color of shirt or jean are undetermined, but on the other hand, there is a fixed infra-red heat source (850nm) around the dummy which can be used to identify and locating it. So, we have designed an IR illuminating point detector to manage the detection process using our vision sensor that capable of detecting light source in the near infra-red band (Appendix B).

A. Infra-red Illuminating Point Detection

Here, we have performed indoor experiments that use an infra-red LED (Light Emitting Diode) as the tracking target (see Figure 4.15). We are able to detect the most possible illuminating point source through the following image

(a) Thresholding with low threshold value

(b) Thresholding with high threshold value

(c) Bounding box on external contour (low threshold image)

(d) Bounding box on external contour (high threshold image)

Figure 4.16: Image thresholding and bounding external contour in the illuminating point detection process (IR point at middle of image)

processing steps:

1. Extract regions in perceived image with high light intensity by image thresholding (see Figure 4.16a and Figure 4.16b) using two threshold values (one large and one small)

2. Use image dilation to join nearby high light intensity regions in the two thresholded images

3. Extract external contours of the two thresholded and dilated images and then bound the contour regions using the minimum area bounding box (see Figure 4.16c and Figure 4.16d)

4. Filter out regions that does not meet the predefined area and size requirements for the illuminating point source (can be obtain through observing

the light source with the required distance apart), and match those regions left that have close bounding box centers

5. The high light intensity region with the closest bounding box center and smallest area difference would be the most possible illuminating point source

In order to keep track with the detected IR illuminating point, instead of performing the detection steps over the full image again, we try to estimate the center position of the detected point in future frame using the velocity calculated from the frame time differences and the positions of IR point in the past frames, and then perform the detection steps within a neighborhood region on the estimated position. By such, we can achieve a detection and tracking speed of around 7.5Hz.

B. Outdoor Geo-locating Experiment

After implementing both the IR illuminating point detector and tracker on the biaxial pan-tilt embedded vision system, we have performed outdoor experiment for obtaining tracking and geodetic data to verify our geo-locating approach based on the vector addition and ray tracing methods suggested in [24].

(a) Ground position for placing the IR illuminator

(b) GPS setup for measuring IR illuminator position

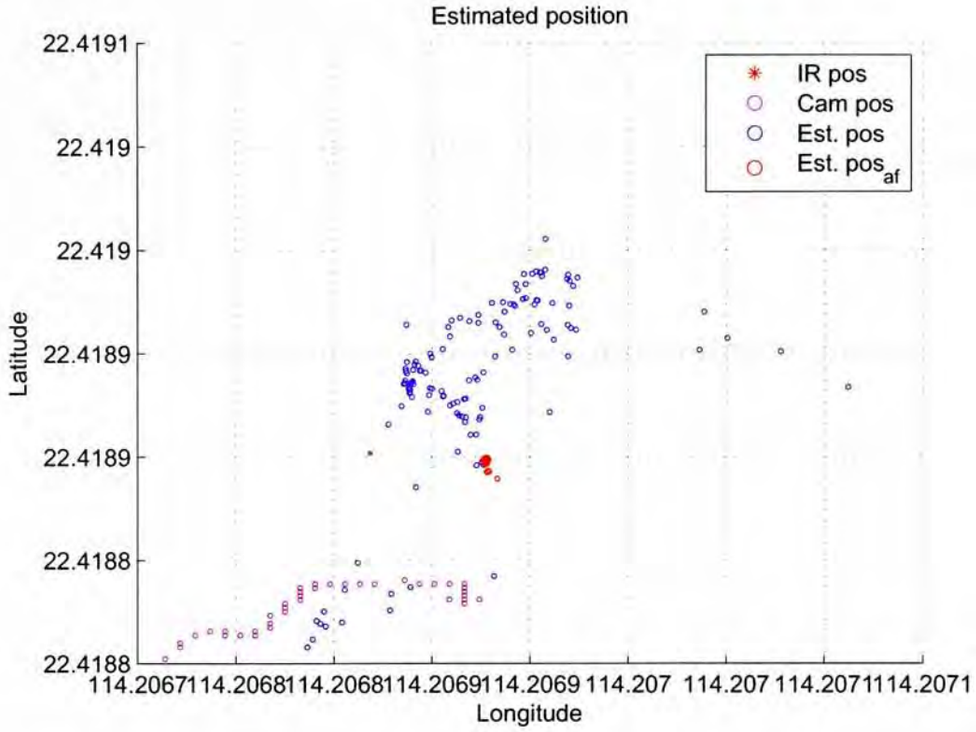
Figure 4.17: Place for putting the IR illuminator and position measurement with GPS

The experiment was conducted in the CUHK campus. We had placed an infra-red illuminator (generates near infra-red source of 850nm) at the ground floor of the Lady Shaw Building (see Figure 4.17a) and we had used our vision system to detect and track the IR source from the third floor (around 10 meters height from ground). We recorded all necessary data for geo-locating the IR source, including the image position of the IR source, the geographical position of the vision system, the system's altitude, body orientation and the angular position of the RC servos. We had also used a high accuracy GPS to measure the actual geodetic coordinates for the IR source where we placed (see Figure 4.17).

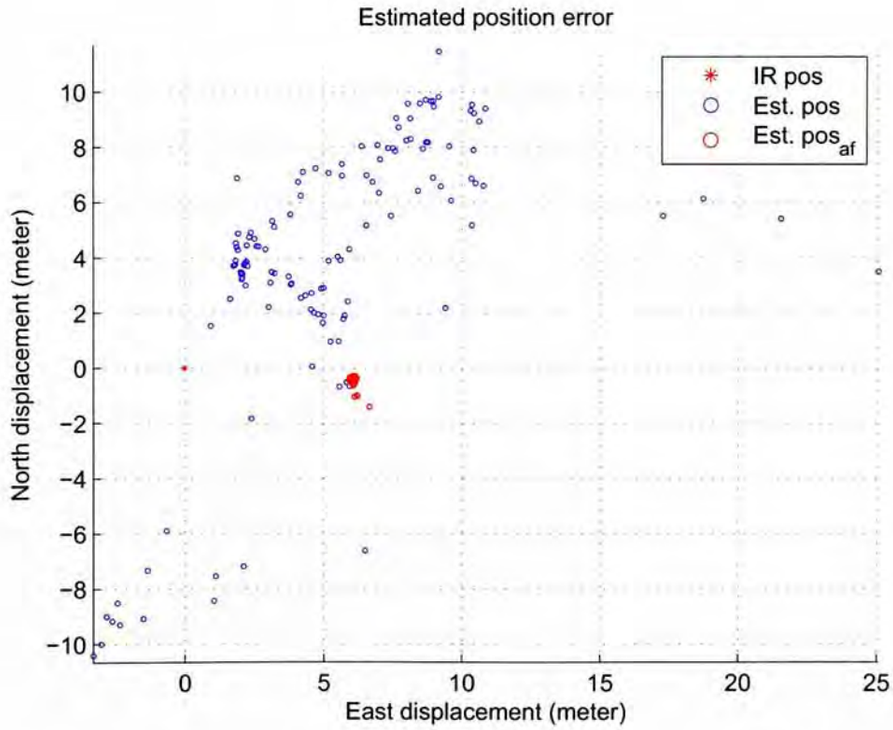
Figure 4.18 and Figure 4.19 show the estimated geodetic coordinates of the IR source and the position error for two sets of experiment we had performed with the use of the geo-locating approach implemented in a MATLAB program. The red star indicates the actual position of the IR source in geodetic coordinates and the pink circles indicate the position of the vision system in geodetic coordinates, while the blue circles are the immediate estimates of the IR source's position with the perceive image and the recorded information, and the red circles are the estimates of the IR source's position after applying the running average filter.

In the first set of experiment we are able to geo-locate the IR source with a position error of around 6 meters while in the second set of experiment the position error is around 15 meters. There is quite a large deviation on the estimation error. And after analyzing the collected data sets, we found that there is a large difference (around 5 meters) in the altitude recorded in the two sets of data, which is possibly due to difference in number of satellites that could be tracked on the GPS device. And we find that the resolution of the altitude recorded by the GPS device is in meter grade which may not be accurate enough for the geo-locating approach we used that rely much on the altitude information of the platform. Therefore, in future experiments, we will make use of the air pressure sensor available on the off-the-shelf autopilot system for obtaining more accurate altitude data (resolution in centimeter grade).

Besides the possible error induced by the sensor noise, there appears a bias in the measurement along the direction of movement of the vision platform. We noticed that this could be induced by the centering error for not being able

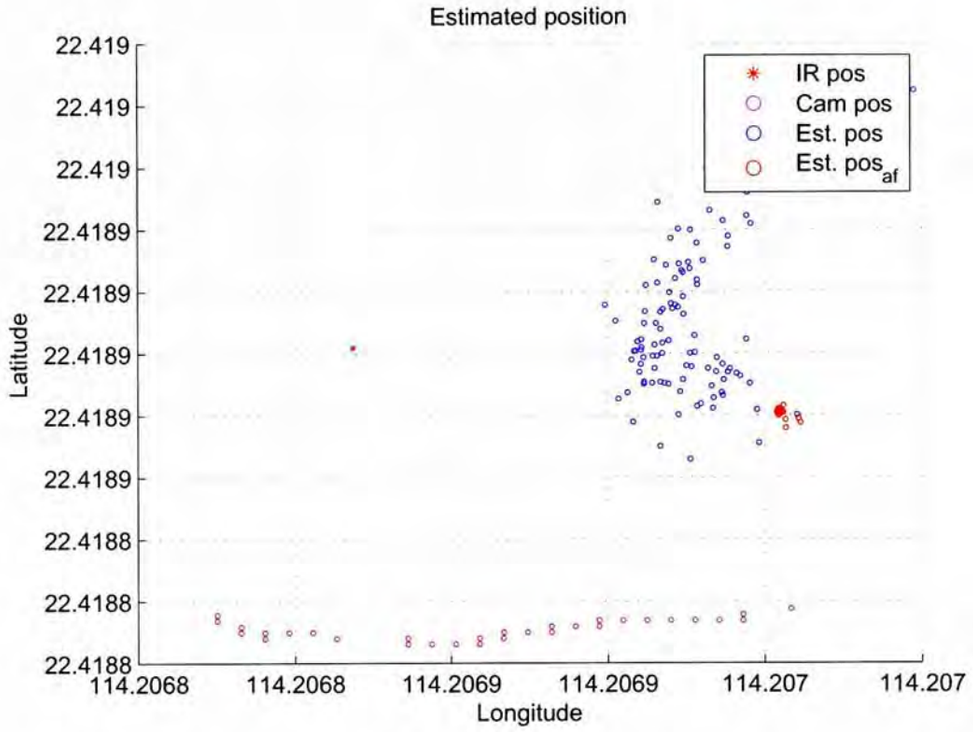


(a) Geodetic coordinates

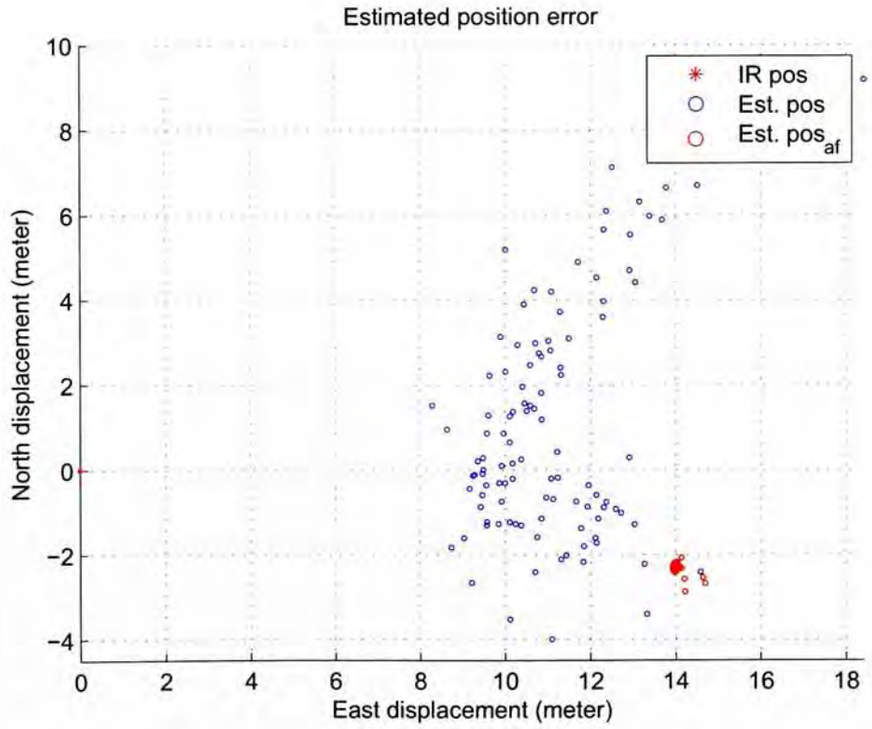


(b) Local navigation frame

Figure 4.18: Estimated position of the IR illuminator (Data set 1)



(a) Geodetic coordinates



(b) Local navigation frame

Figure 4.19: Estimated position of the IR illuminator (Data set 2)

to move the detected IR source to the middle of perceived images. Therefore, we would have to adjust the control gains for the PID controller on the camera pan-tilt platform to achieve better centering performance. And, according to [24], the bias error can also be reduced through controlling the UAV for flying with a circular pattern over the ground target. Moreover, method for filtering the zero-mean Gaussian sensor noise, such as the RLS (Recursive Least Squares) filter, can also be used for achieving better geo-locating results.

Although we still not implement the geo-locating approach to the vision system and perform experiment on UAV for geo-locating ground target, however, the above experiments show that our vision system should be capable for performing such tasks. We only require to improve our aerial platform for unmanned flight and find a suitable place for carrying out experiment.

□ End of chapter.

Chapter 5

Real-time Aerial Mosaic Building

Apart from object tracking, another focus for developing our vision system is to apply it for performing regional surveillance task on the UAV platforms that we have. Usually, surveillance with UAV are performed directly by down streaming the analog video captured with on board camera to a ground station for human monitoring. That means only the current view directing by the on board camera can be seen on ground. It would be hard to have a full knowledge on ground features (e.g. landscape, constructions or traffics) within the region looking at. Therefore, building a regional map with the perceived aerial image in real-time would greatly facilitate the surveillance task. Moreover, the aerial mosaic map created can also be used for change assessment, for example, before and after natural disaster.

Image stitching algorithms are widely used nowadays in digital imaging for producing wide-angle panoramas. In this chapter, we show these general algorithms for which we use for building aerial mosaic in real-time on a ground laptop PC using the digital images perceived with the vision system installed on our aerial platforms and transmit to ground through Wi-Fi connection. First, we present the motion model we used for relating the inter-frame images. Then, after applying undistortion to the received images, we show how we match and align the images by a feature-based method with use of RANSAC algorithm [40]. Finally, image feathering technique is used for compositing aligned images to create aerial mosaic. Simulation with use of aerial images

captured in Google Earth and mosaicking results in outdoor experiments are given at end sections of the chapter.

5.1 Motion Model Selection

Since we are capturing images with camera on a moving aerial platform, ground features appear to move in the perceived images. In order to align images with matched ground features, we have to establish the relationship that maps pixels in one image to another. This mapping is referred as a motion model [75], and there are a numbers of them available, for example, 2D transformation, planar perspective motion model, 3D rotation model (usually used in creating 360° panorama), etc. Therefore, it is important that we have chose an appropriate motion model which suits our application, as simple model may not give good results while complex model would increase the processing time and deter the real-time applicability.

5.1.1 Planar Perspective Motion Model

With the situation that UAV is commonly being flew at fixed and high altitude, we can assume ground features in perceived images to be lain on the same planar surface. By that assumption, we can use a planar perspective motion model for the pixel mapping and aligning the images to composite an aerial mosaic.

In the planar perspective model, image pixels are mapped from one image to another by the following transformation [75]:

$$\mathbf{x}' \cong \mathbf{H}\mathbf{x} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.1}$$

where $\cong$ indicates equality up to scale, $\mathbf{H}$ is a 3×3 homography matrix and, $\mathbf{x}' = [x', y', 1]^T$ and $\mathbf{x} = [x, y, 1]^T$ are the pixel positions corresponding to the two aligning images in homogeneous coordinates.

After obtaining the homography matrix, we can warp image $\mathbf{x}$ to the coordinates in image $\mathbf{x}'$ after normalization to get the inhomogeneous results as

shown below:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{5.2}$$

In order to reduce perspective distortion arises when creating aerial mosaic with the planar perspective motion model, we have to ensure the images are taken by vertically downward looking camera using small view angle lens. For that, we have developed a view-stabilizing biaxial platform (in Section 3.3), and one major application of it is to maintain the camera in a downward looking orientation.

5.2 Feature-based Image Alignment

Generally, there are two common approaches to align images: direct pixel-based alignment and feature-based alignment [74]. Direct pixel-based method computes the parameters for the motion model through comparing the pixel-to-pixel similarity of the image pair. An error metric is used for estimate the motion in a coarse-to-fine searching framework on multi-level image pyramid. The direct method usually can produce accurate alignment for sequential image frames in a video [73]. However, if there is only partial overlapping (i.e. large motion) between the image pair or high intensity change in images, the direct method often fails. While on the other hand, feature-based method with robust keypoint descriptors (e.g. SIFT, SURF) and matching schemes (e.g. RANSAC) available nowadays, can handle images with large separate view, different in scales and orientations. It is also regarded to be operate faster than the direct method with appropriate implementation.

In the following parts, we will show how the feature-based method is used for matching and aligning the aerial images we perceived with the vision system on our aerial platform.

5.2.1 Image Preprocessing

Straight lines usually appear curved due to camera lens imperfection, and that deviates the linear projection model we used for image mapping. Therefore, before we move on to feature registration for image alignment, we have to

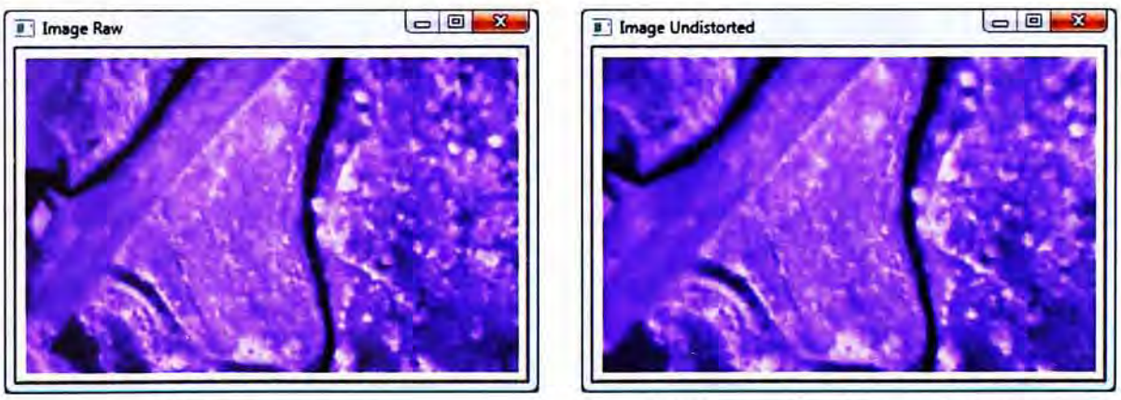
correct the lens distortion first.

As mentioned in Section 4.1, the Camera Calibration Toolbox for MATLAB is used for finding the principal point and distortion coefficients for correcting lens distortion. And they are found to be:

Principal point: [200.30611, 112.97155]
Distortion: [−0.39674, 0.16797, 0.00019, −0.00009, 0.00000]

as shown in Table 4.1 after the calibration.

By applying Equation 4.6 in Section 4.1 to the image with the calibration results we obtained, we can get undistorted image as shown in Figure 5.1b. The images we show here were captured by the vision system on our aerial platform in the airfield of the Hong Kong Model Engineering Club (HKMEC) in Yuen Long. Since we had covered the camera lens with an IR pass filter, so the color of images appeared different from those visible light images.



(a) Raw input image (b) Image after lens undistortion

Figure 5.1: Lens undistortion on perceived aerial image

5.2.2 Feature Extraction and Matching

In feature-based image alignment, a sparse set of distinctive feature correspondence between the image pairs are used for obtaining transformation between images with the selected motion model. There are two state-of-the-art robust keypoint descriptors namely SIFT (Scale-Invariant Feature Transform) [53] and SURF (Speeded Up Robust Features) [25] for registering invariant feature

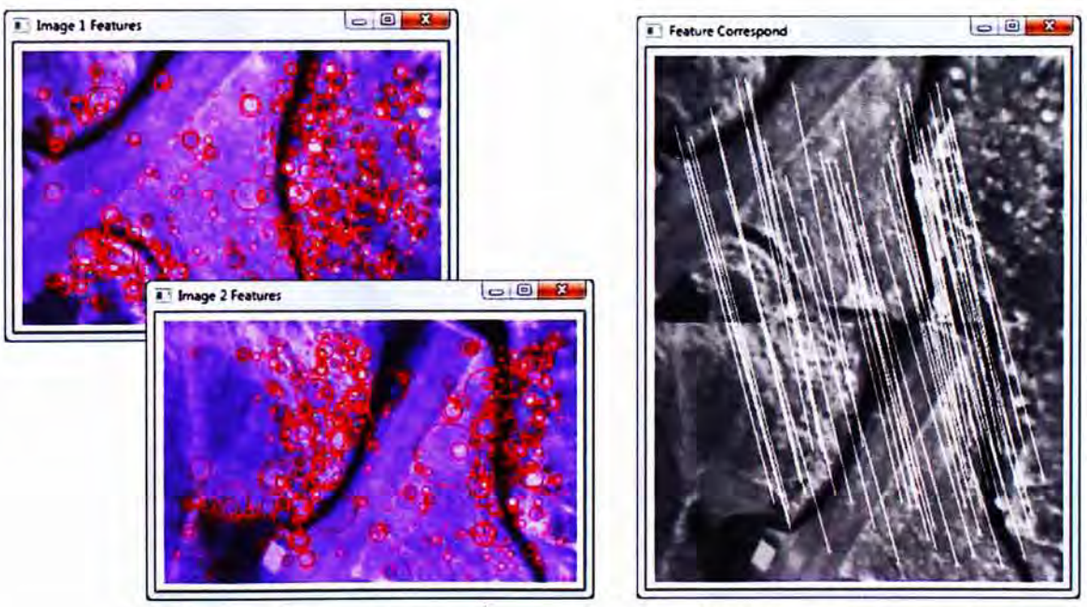
points and their correspondence in image pairs. In our project, we have made use of the SURF keypoint descriptor to match and align images for the its good repeatability and fast extraction speed comparing to the SIFT algorithm.

Here, we do not implement the SURF algorithm on our own as there already exist several reliable SURF library, e.g. the OpenSURF library [38] and SURF routines in the OpenCV library. And, in our project, the extraction process for SURF features is performed directly with the keypoint extraction routine (`cvExtractSURF`) provided in the OpenCV library.

After extracting the SURF keypoints (usually represented by 64-dimensional vectors or 128-dimensional vectors in the extended mode), we have to find their correspondence in the image pair. Simple method such as the nearest neighbor will do the job. Generally, the nearest neighbor distance ratio is used for computing features correspondence by the feature vectors [59]:

$$NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|} < t \tag{5.3}$$

where d_1 and d_2 are the nearest and second nearest neighbor distances, D_A is the target descriptor, D_B and D_C are its closest two neighbors, and t is a threshold value (the value 0.6 is commonly used).



(a) Extracted image features (b) Feature correspondence

Figure 5.2: SURF feature extraction and feature correspondence

Feature points are considered as matched pairs when the ratio of their Euclidean distance to that of its second closest feature point is less than a threshold value of 0.6. For example, according to the above equation, D_B is regarded as the matched keypoint of D_A if the ratio of their Euclidean distance to that of the Euclidean distance between D_A and its second nearest neighbor D_C is less than 0.6.

In Figure 5.2a and Figure 5.2b, we show the SURF feature points extracted in a pair of image and their correspondence through line segments joining the feature positions in the images.

5.2.3 Image Alignment using RANSAC Algorithm

After obtaining the feature correspondence, generally, least-squares approach can be used to estimate the parameters in the image homography for aligning images. However, if there appears too many outliers (mismatched feature pairs) in the feature correspondence, it would not be possible to obtain a global optimum solution. To due with the outliers problem, matching scheme such as the RANdom SAmpling Consensus (RANSAC) [40] approach can be used.

Following gives a summary on the RANSAC algorithm that used for image alignment [75]:

1. A set of feature correspondence (usually at least 5 feature pairs are needed) is randomly selected for computing the image homography by least-squares algorithm (can refer to Appendix A.2 of [75])

2. The homography obtained is then applied to all features and check the number of inliers (if the estimated feature position is within usually, 1-3 pixles of the detected feature position, it is counted as an inlier) that agree with the estimated motion

3. The random selection process is repeated certain times and the homograhpy obtained by the sample set with the largest number of inliers is kept as the final solution and further refined with the Levenberg-Marquardt method [49, 57] for reducing the reprojection error

In our project, the OpenCV routine (`cvFindHomography`) is used to find the homography matrix using the RANSAC process. After obtaining the image

homography, we can transform the image pixels in one image to the image coordinate of the other reference image according to the planar perspective motion model (Section 5.1.1) we used. By then, we would be able to warp the transformed image and the reference image together to form a mosaic with wider angle of view.

5.3 Image Composition

To warp the aligned image together, we have to decide the composition surface (e.g. flat, cylindrical, spherical) for which we want the image mosaic to build upon. And it usually depends on the number of images that would be included in the mosaic and the motion model that used for image mapping. For example, a spherical surface is usually used for composite wide angle or 360° panorama image with the 3D rotation model. In our application, we would like to build aerial mosaic of ground features with a planar perspective motion model. Therefore, a simple flat surface can satisfy our requirement.

The simplest method to blend two aligned images on a mosaic image with flat composition surface is by taking average value at each valid pixel position on the aligned images [75]:

$$C(x,y) = \sum_k \omega_k(x,y)\, I_k(x,y) \,/\, \sum_k \omega_k(x,y) \tag{5.4}$$

where $I_k(x,y)$ are the warped images, $\omega_k(x,y)$ is 1 at valid pixels and 0 elsewhere, and k is the number of images.

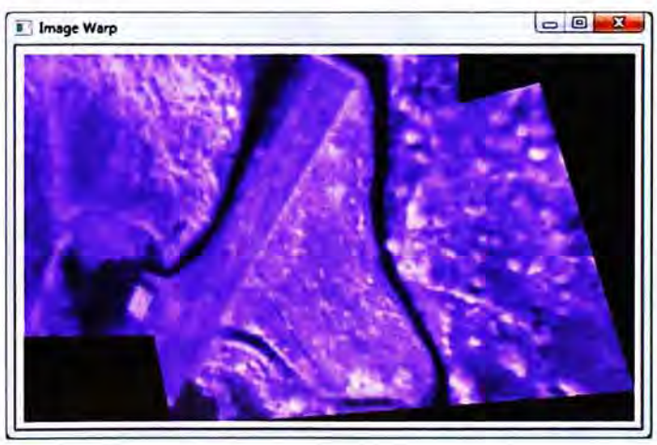


Figure 5.3: Blending of aligned images by simple averaging

However, this simple averaging method usually not works well, as there are seams appearing on the mosaic image due to exposure differences (see Figure 5.3). Moreover, mis-registration and scene movements are also easily visible.

Figure 5.4: Distance map that used for image blending (whiter color indicates larger weight)

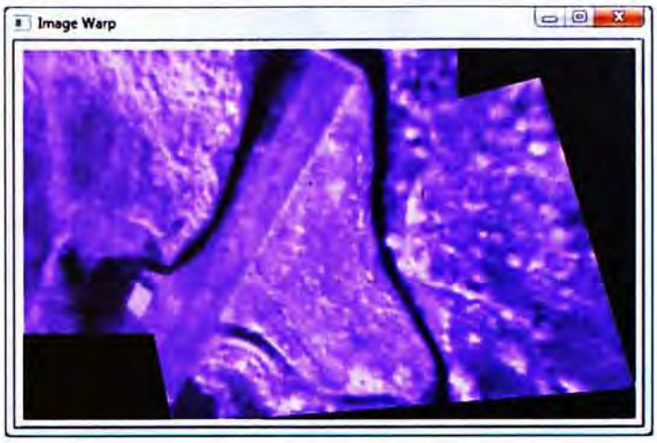
5.3.1 Image Blending with Distance Map

A better approach to perform image blending is to use weighted average with a distance map. The process is often called image feathering [75]. It blends images by weighting more heavily on the pixels when they get closer to the image center while weighted less when nearer to the image edge, that is, i.e. replacing $\omega_k(x, y)$ in Equation 5.4 with the following weighting function:

$$\omega_k(x, y) = \arg\min_D \{\|D\|\} \tag{5.5}$$

where D is the distance to the invalid (i.e. edge) pixel.

We can obtain appropriate weightings for the image by computing the distance map (also called distance transform) where each valid pixel is tagged with its Euclidean distance to the closest invalid (i.e. edge) pixel [75]. In our project, we have used the routine function (`cvDistTransform`) of the OpenCV library that based on the distance transform algorithm in [27] for finding the weightings. And the computed distance map is stored in the alpha channel of



(a) Image blending result

(b) Accumulated distance map in alpha channel

Figure 5.5: Image blending by weighted average using distance map

each image and accumulated in the composted mosaic image.

In Figure 5.4, we show the pixel weightings that computed with distance transform as an image, where regions with whiter color indicate larger weights. While Figure 5.5a shows the blending result with use of distance map on the aerial images and Figure 5.5b shows the accumulated distance map storing in alpha channel of the mosaic image.

It is noticeable that the seams originally appearing in Figure 5.3 due to exposure differences had disappeared with use of distance map for image blending as shown in Figure 5.5a.

5.3.2 Overall Stitching Process

In previous sections, we have shown the approach for stitching an image pair to a mosaic image. Similarly, for creating a large mosaic with a sequence of aerial image, we just have to repeat the above processes but with the mosaic created as the reference image for aligning newly received image frame. However, instead of using the full mosaic image, only nearby portion within region of the newly stitched image will be used for warping. This would be sufficient for stitching sequential image frames captured by vision system on our aerial platform, as movement (position change) of ground features in perceived images would be small when viewing from the UAV at high altitude.

Figure 5.6 shows the program flow on the stitching process we have implemented for building aerial mosaic in real-time. Most of the parts are just the approaches that we mentioned above, but in addition, we can create a separate mosaic image as a sub-scene when the perspective distortion appeared to become too large for aligning the newly received image frame to the original mosaic image in the main scene or when there is not enough feature correspondence for computing the homography matrix. And later, we can stitch the two mosaic images together for building a larger aerial mosaic if the same situation is met again. Certainly, we can also set the stitching program just to skip those image frames with large perspective distortion when aligned to the mosaic image.

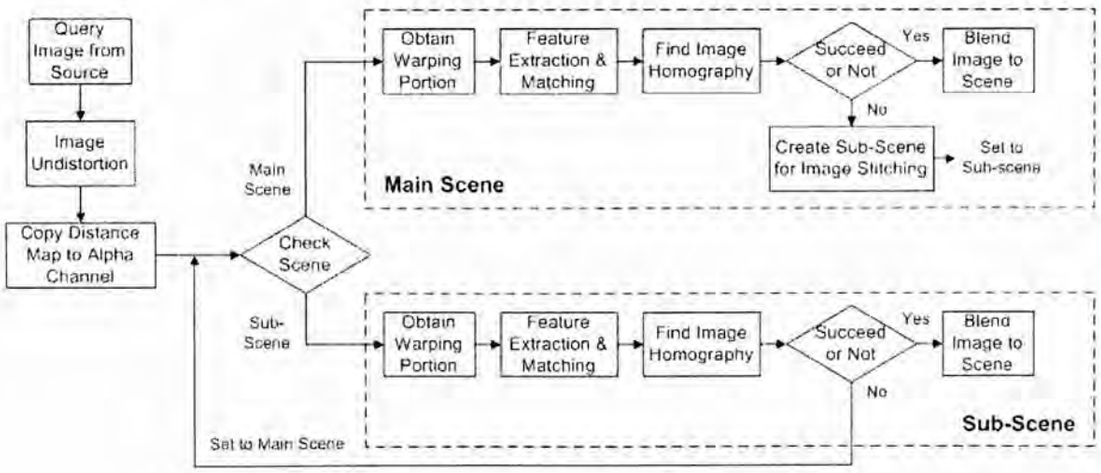


Figure 5.6: Flow chart for aerial image stitching process

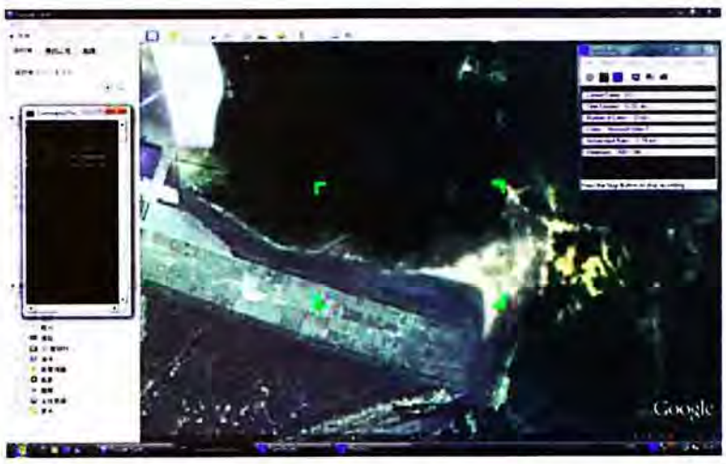


Figure 5.7: Capturing aerial image from Google Earth

5.4 Mosaic Simulation using Google Earth

As it always takes long time for setting up the aerial platform for performing flight test to collect sample aerial images that can be used for testing and debugging with our image stitching program, we therefore try to make use of the readily available aerial images in Google Earth for our test and mosaicking simulation during the program development process.

The Google Earth COM (Common Object Model) API [6] allows external applications to query information from and send commands to Google Earth, for example, to query and control the 3D viewpoint on Google Earth, obtaining the geographical coordinates (latitude, longitude, altitude) of current viewpoint, etc. Thus, by using the COM API, we can obtain aerial images in Google Earth with controllable viewports for performing tests on the image stitching approach we have implemented. Besides, we can also obtain reliable geodetic information registered with the aerial image that would be useful for producing aerial map with registered geographical information.

Here, we have used a screen capturing program to record a video with same image resolution (368×240) as our vision system on the viewport of Google Earth that is being controlled by an external program we wrote with the COM API (see Figure 5.7). Such that, the image frames are captured as if an UAV is flew under a predefined (helix like) flight path with camera keeps looking down vertically to the ground and with a fix altitude. Result of the mosaic simulation is shown in next section.

5.5 Results and Discussion

In this section, we present the results on mosaic simulation using aerial images captured from Google Earth and also, both off-line and on-line mosaicking using aerial images captured in outdoor experiments with the image stitching algorithms we mentioned in previous sections. The capability for real-time mosaicking on a ground laptop PC using aerial images captured and transmitted from the embedded vision system through Wi-Fi connection is also verified.

Mosaic Simulation Result

As shown in Figure 5.8, we are able to produce a regional mosaic image using aerial video recorded from Google Earth with a changing viewport. The video frames are recorded as if an UAV is being flown under a helix like path at the airfield of HKMEC in Yuen Long. The playback frame rate for the video is 25fps and we have set the stitching program to skip new incoming video frames if the current image frame is still under processed. Generally, the stitching program takes around 50ms to extract about 800 feature points in an image frame and the average stitching speed is around 2Hz for building the mosaic image on a desktop PC. And the speed could be further increased if we can set a maximum number of feature points to be extracted in the program.

From the simulation result with Google Earth, we found that image mis-

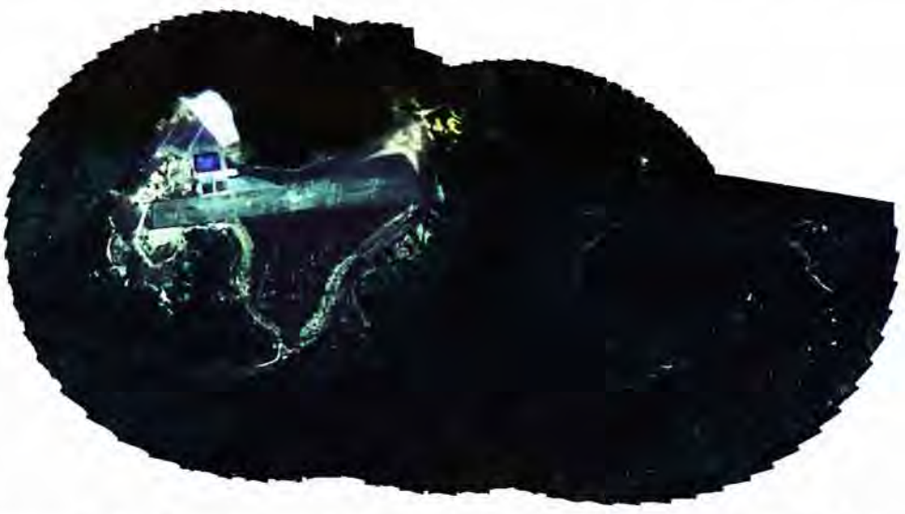


Figure 5.8: Mosaic created using aerial image from Google Earth

alignment occurred when the flight path came to a loop closure. We think the problem is possibly due to accumulation of image reprojection errors through out the flight path. Therefore, new stitched images will drift to a biased direction and cannot align with previous stitched image in the scene when the flight path comes to a closure.

In order to solve the problem that may also appear in real flight experiments, we can perform specific flight path (e.g. zigzag path) such that no loop closure is involved. The bundle adjustment [77] approach that usually used for adjusting image alignment in full view panorama can also be applied with use of some software package available [52]. We can perform local bundle adjustment on the collected images when loop closure is detected [79]. However, the computational cost for bundle adjustment process could be so high if a lot of images are involved, it would possibly affect the real-time performance of the stitching program. And, a less computational expensive approach we can use is to remap the stitched scene to an underlying geodetic grid map with registered geographical coordinates. The geographical information obtained from GPS can be used and registered as control points on the stitched image.

Outdoor Mosaicking Experiments

The mosaic simulation result shows that the image stitching algorithms we have used for building aerial mosaic in our program can function quite well, although there are still some improvements can be made. Therefore, we try to perform both off-line and on-line mosaicking on aerial image obtained in real flight experiments.

I. Mosaic of Airfield in Yuen Long

Last year, we had got some chances to perform manual flight tests at the airfield of HKMEC in Yuen Long with the 1/4 scale Super Cub fix-wing airplane (see Figure 3.23). And at that time, we had recorded some aerial images directly using the vision system installed on our aerial platform. So, we try to perform off-line mosaicking with those image sequences we get. But as we still had not implemented the view-stabilizing mechanism on the biaxial pan-tilt platform in these flight tests, the aerial images obtained could be in large perspective view that deviate from the downward looking camera assumption.

Figure 5.9 shows the mosaicking result of the aerial images captured above the airfield in Yuen Long. The flying altitude of the aerial platform is around 300 meters to the near ground. And the image recording rate on the vision system we could achieve at that time was around 10Hz (we are now able to achieve 25Hz with use of GStreamer [7] - a multimedia framework on Linux system). As the recorded image frame rate is not that high as simulation, we have intentionally reduced the playback frame rate to 2Hz for the mosaicking test.

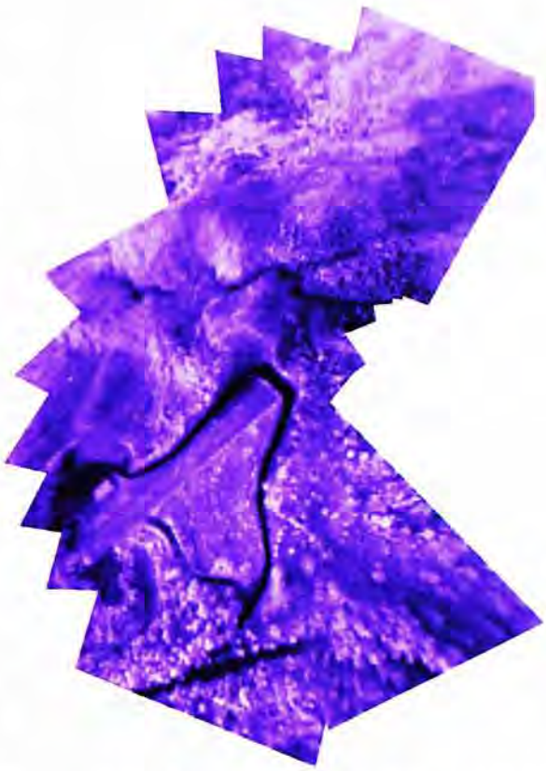


Figure 5.9: Aerial mosaic of the airfield in Yuen Long

Since the tests were performed under manual flight and we did not follow any predefined flight path, so we were not able to obtain a nice looking regional aerial map as that shown in the mosaic simulation. There are also prominent perspective distortions appearing in the aerial mosaic due to violation of the downward looking camera assumption. However, middle part of the mosaic image still shows good stitching results with only little influence from the perspective distortion.

II. Aerial Mosaic in CUHK Campus

As mentioned in Section 3.4, we have also performed manual flight test experiments on the gasoline helicopter with the embedded vision system for real-time aerial mosaicking. We had set up a ground station using a laptop PC connected with a USB wireless adapter for receiving aerial images transmitted from the embedded vision system through Wi-Fi connection. Both of the vision system and the ground PC used the same model of a high power USB wireless adapter (TL-WN7200ND: support IEEE 802.11b/g/n standards, up to

150Mbps transmission rate, max. 500mW transmission power and 5dBi antenna). For captured image with resolution of 368×240 and compressed with JPEG library, depending on image content, but the file size would generally be less than 25KBytes. So, it would require a data rate of 6Mbps for transmitting image frames at 30Hz. And from the wireless calculator [19] provided on the web page of TP-LINK, for wireless device with transmission power and antenna gain similar to that we used, excellent link condition can still be maintained for a maximum distance of around 1km apart. Use of antenna with higher receiving gain or directional antenna can further increase the transmission distance. But the current wireless link condition would be enough for performing some initial tests for real-time aerial mosaicking using our vision system and aerial platform.

Figure 5.10: Aerial mosaic over United College in CUHK campus

Figure 5.10 shows the mosaicking result of the aerial view around 150 meters above a car park in the United College of CUHK. We were able to achieve real-time aerial mosaicking in the single flight test experiment. However, due to a crash when landing and lost of wireless connection, we were not able to

store the final mosaicking result in the experiment with the mosaicking program. The aerial mosaic shown in Figure 5.10 is reproduced from the aerial images received on the ground laptop PC and playback at 10Hz in off-line manner.

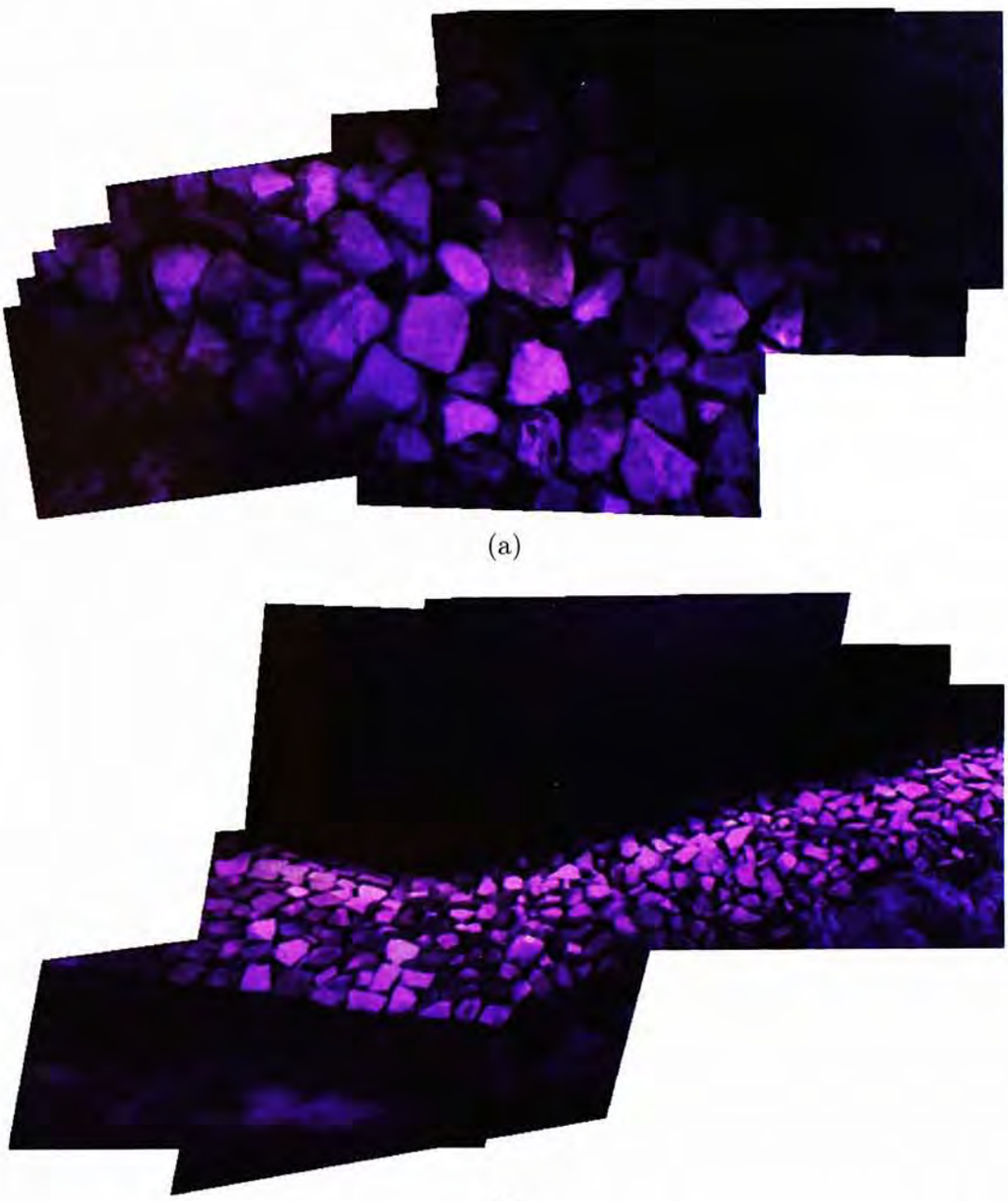
The perspective distortion in the mosaic image has been reduced with the use of the view-stabilizing pan-tilt platform and rejection from the image stitching process when the perspective parameters in the homography matrix get too large. Images that nearly fully overlapped with the mosaic image are also skipped from stitching process. The overall aerial mosaic looks fine but there are blurring regions near the image perimeter. It is possibly due to the circular artifacts appeared in the perceived images as a result of the use of an IR cutoff filter and the stretching of images after lens undistortion. Previously, we have shown the use of a black color IR pass filter. But this time, we are using a transparent IR cut filter. That means only visible light can reach the image sensor that also sensitive to near infra-red light (refer to Appendix B). However, as the IR cutoff filter is not attached to the camera lens directly, light source may still pass the lens through RP (Rapid Prototype) mounting parts and resulted as circular artifacts in images. In order to completely solved the problem, camera lens with IR cutoff coating and smaller field of view may be used in future flight experiment.

III. Aerial Mosaic in Tseung Kwan O

Another set of mosaicking experiment has been performed with the vision system installed on our foam fix-wing airplane (see Figure 3.24). We have flown the aerial platform manually above a flat land near the coast in Tseung Kwan O industrial area. However, due to excessive loading (instead of two, a single battery pack can be used for both system and motor power supply for reducing weight) and insufficient elevator control of the airplane, we were only able to fly at a maximum of 50 meters above the ground. The view change in perceived images is also large due to low flying altitude but high flying speed. And, we also can not create regional aerial mosaic with this set of flight experiment as the flat land are mainly grasses, it is difficult to get enough stable feature keypoints for aligning the perceived images with the low flying altitude. Apart from these, the vibration isolation mechanism for

the autopilot (it controls the pan-tilt platform for view-stabilization, refer to Section 3.4) was also not working well and it leaded to abrupt view changes on the biaxial camera platform.

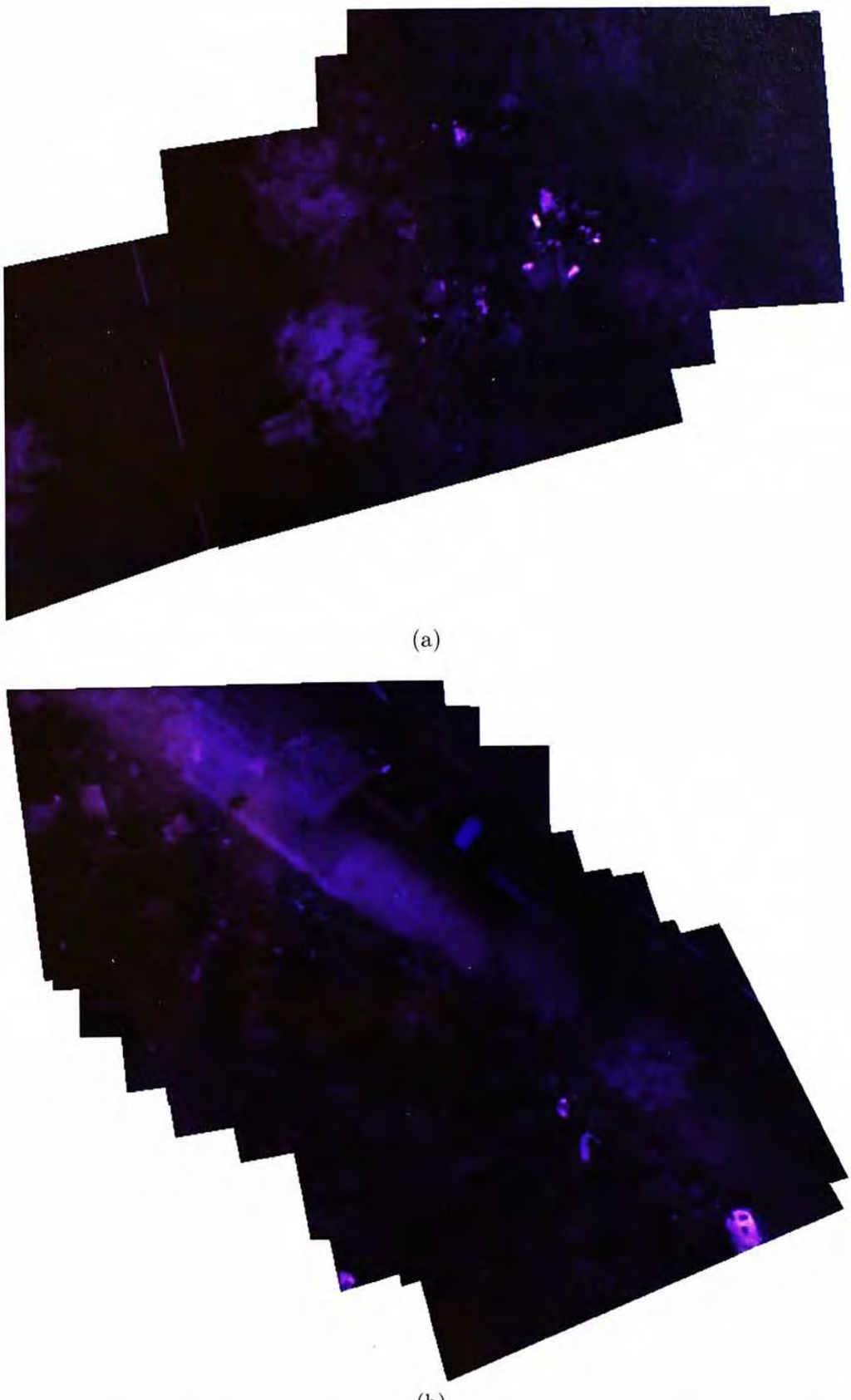
However, we are still able to perform off-line aerial mosaicking with the images received on the ground laptop PC that being playback at 1Hz. Figure 5.11 and Figure 5.12 show the off-line mosaic images we build. They are mainly



(a)

(b)

Figure 5.11: Aerial mosaic near the coast of TKO industrial area



(a)

(b)

Figure 5.12: Aerial mosaic over flat land roads in TKO industrial area

regions near the coast and the road which have more stable feature keypoints. As we have used a camera lens with a smaller viewing angle (no IR cutoff coating so images appear reddish), image stretching arose from lens distortion correction has been reduced. Much clearer mosaic images are obtained. But due to the abrupt motion of the view-stabilizing platform and mis-registration of instable image features at low flying altitude, there are still blurring regions appearing in the aerial mosaic images.

The above mosaicking results in outdoor environment show that the image stitching approaches we applied in our aerial mosaicking program can works well if we can maintain the camera in a downward-looking orientation and a high flying altitude with little variations. Although satisfactory wide regional aerial mosaic can not be obtained in the experiments, the capability of the vision system for transmitting digital images to ground laptop PC for real-time aerial mosaic building has been verified and achieved. And certainly, there are still parts in the stitching approaches we can have further improvements, for example, solve the loop closure problem and handle object movement in scene.

□ End of chapter.

Chapter 6

Conclusion and Further Work

The main goals of the thesis are to develop a light weight modularized embedded vision system for small UAVs that capable of 1)providing clear images at interactive frame rate under severe vibration on aerial platforms and 2)operating vision algorithms on perceived images for object tracking, geo-locating and reconnaissance in outdoor environment with high processing speed and low power consumption; and to verify the system performance through both indoor and outdoor applications. Although not all desired applications are fully implemented on the embedded system, major tasks like on-board object tracking and real-time aerial mosaicking have been successfully achieved and reached the goals that we have stated.

In the preceding chapters, we show that CMOS image sensor featuring global shutter can sustain severe vibration induced by gasoline engine or electrical motor on aerial platform to provide clear and shear-free images, and with initial tests on sensor configuration and image data retrieval through USB peripheral, we are able to interface such image sensor to a low power and light weight embedded system offering high performance heterogeneous multi-core processor for vision processing. A camera view-stabilizing platform has also been developed with use of off-the-shelf autopilot and RC servos. Although abrupt camera motion occurred in flight experiments due to ineffective sensor vibration isolation, the biaxial pan-tilt platform still helps to maintain the camera in a downward-looking orientation and reduces perspective distortions arise in stitching process for aerial mosaicking.

Indoor experiments using the biaxial vision system for on-board color blob

and infra-red point tracking and target auto-centering show that general vision algorithms can be applied to the system and operate at interactive frame rate (color blob tracking @15fps and infra-red point tracking @8Hz) without co-processing of DSP and GPU cores in the system. However, computational intensive feature-based algorithms should be modified and offloaded to DSP or GPU for co-processing in order to attain real-time performance. Target geo-locating has not been implemented on-board yet, but the approach has been verified with outdoor tracking data collected using the vision system and achieved a minimum position error of 6 meters computed in a MATLAB program.

Outdoor experiments using the vision system for transmitting digital aerial images from flying aerial platform to ground laptop PC for real-time mosaicking have been established. Comparing to analog camera that commonly used for real-time surveillance, our vision system can provide much clearer and noise free digital images. Although regional-wide aerial mosaic for surveillance application can not be obtained successfully due to uncontrolled flight path and flying altitude of aerial platforms, the performance of real-time mosaicking and the image stitching approaches are verified to work well and we still obtain some nice aerial mosaic with wide field of view.

Further Work

There are still works to do both on hardware, software and vision algorithms implemented for further improving the embedded vision system to be used on UAVs for performing real-time target tracking, geo-locating and surveillance through aerial mosaic.

Vibration isolation method on autopilot for view-stabilization has to be redesigned. Instead of using rubber band on the supporting platform, soft rubber or sponge material might provide better isolation to the engine or motor vibration that lead to abrupt motion of the pan-tilt platform. Currently, simple image encoding with use of DSP on the embedded system can be achieved. Later on, we would have to migrate and rewritten the program code of the vision algorithms to be run on the system with co-processing of DSP and/or GPU. Especially, it would be beneficial to be able to implement (SIFT or SURF) feature extraction on DSP or GPU for more robust object tracking in

real-time. Moreover, it is hard to ensure that the downward-looking camera assumption and high flying altitude requirement are always met during aerial mosaicking. Therefore, further work should consider using full 3D motion model for aligning the aerial images during the image stitching process. The inertial data on sensors can also be used for aiding the estimation process of image alignment.

□ **End of chapter.**

Appendix A

System Schematics

Following pages show the system schematics for printed circuit board fabrication on the camera module, image acquisition board, camera interfacing boards, expansion and micro-controller modules for the embedded vision system.

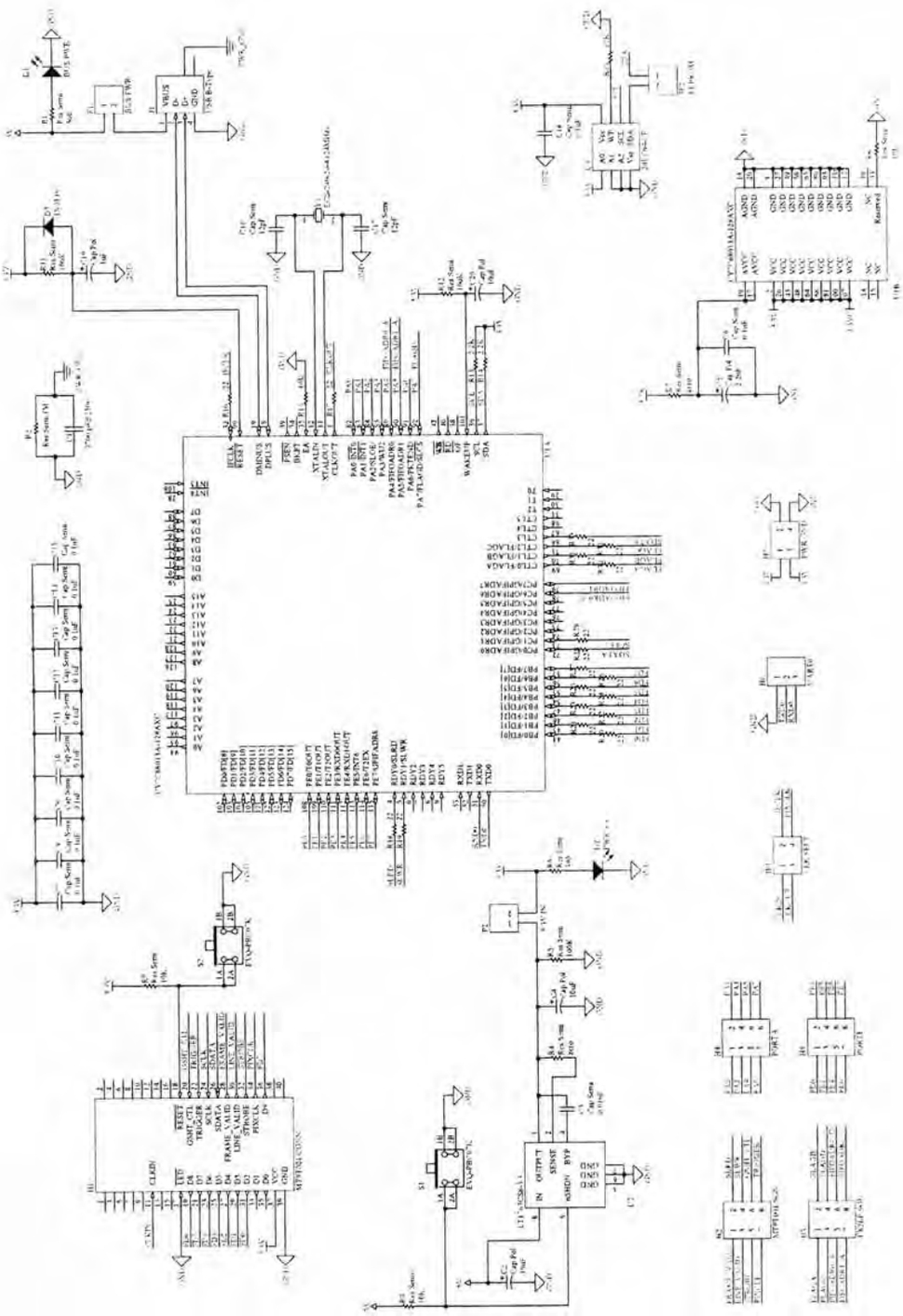


Figure A.1: Schematic of the USB image acquisition board

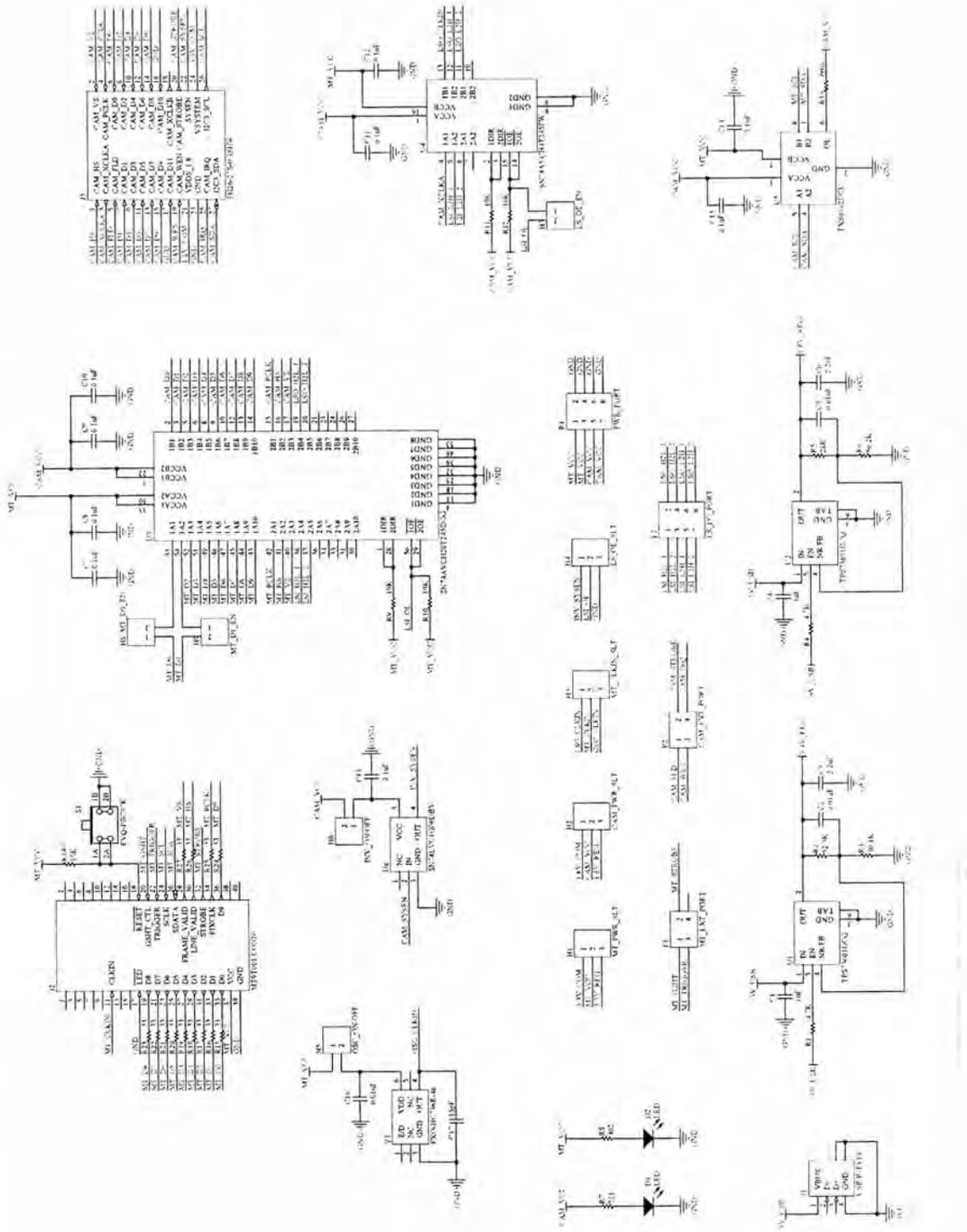


Figure A.2: Schematic of camera module adapting board for Gumstix CoM

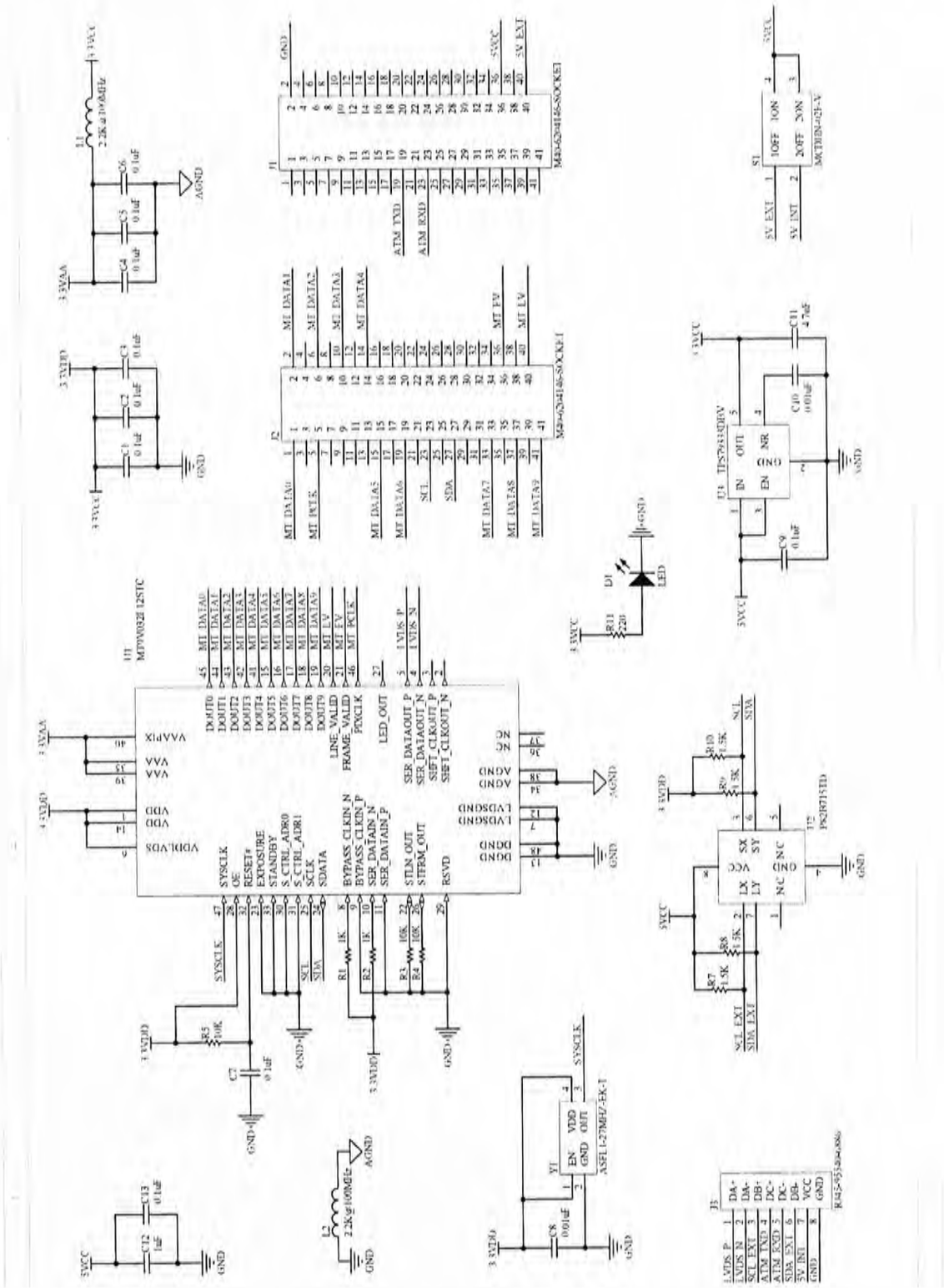


Figure A.3: Schematic of the new MT9V032 camera module

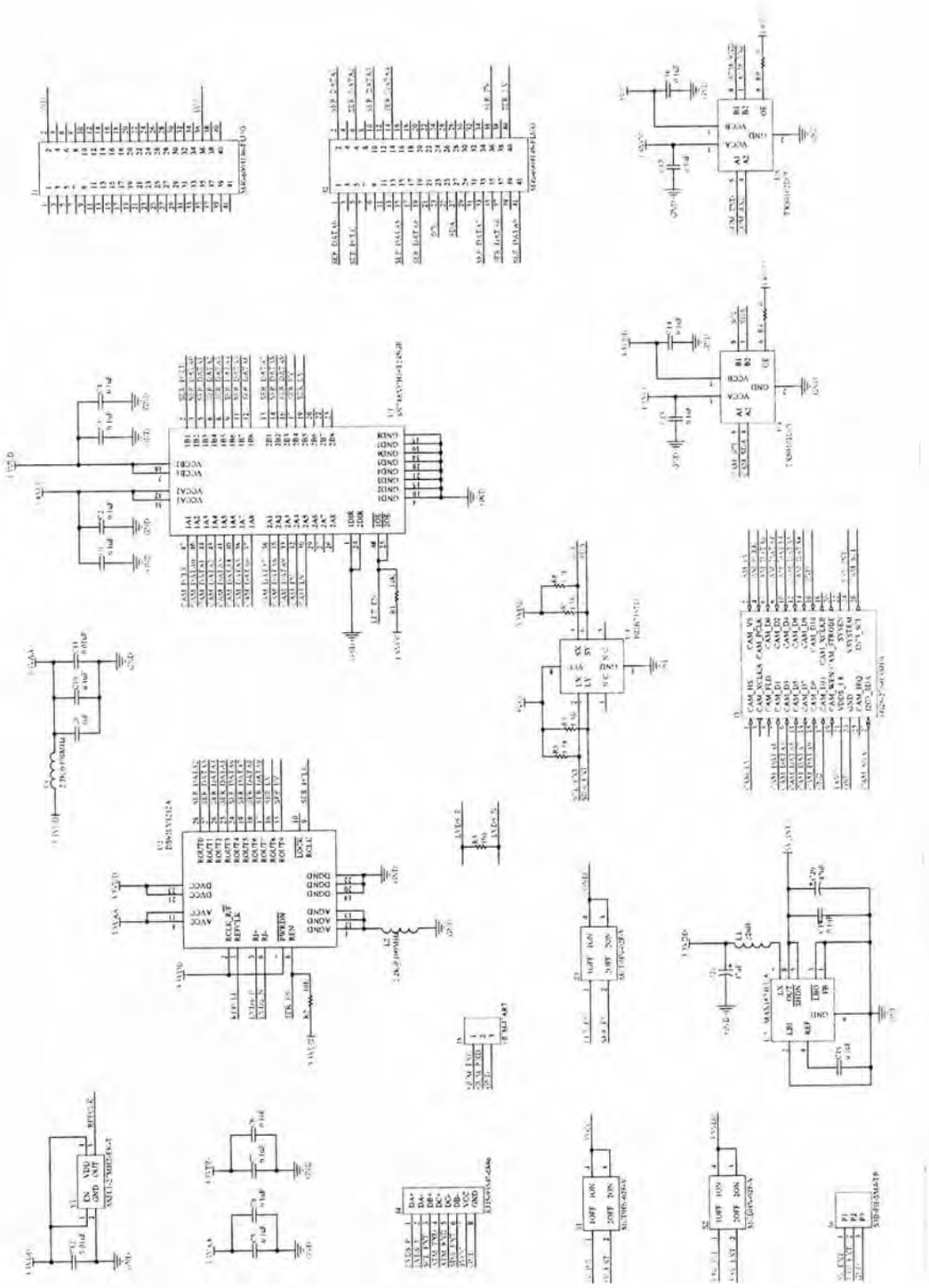


Figure A.4: Schematic of the interfacing module for new camera system

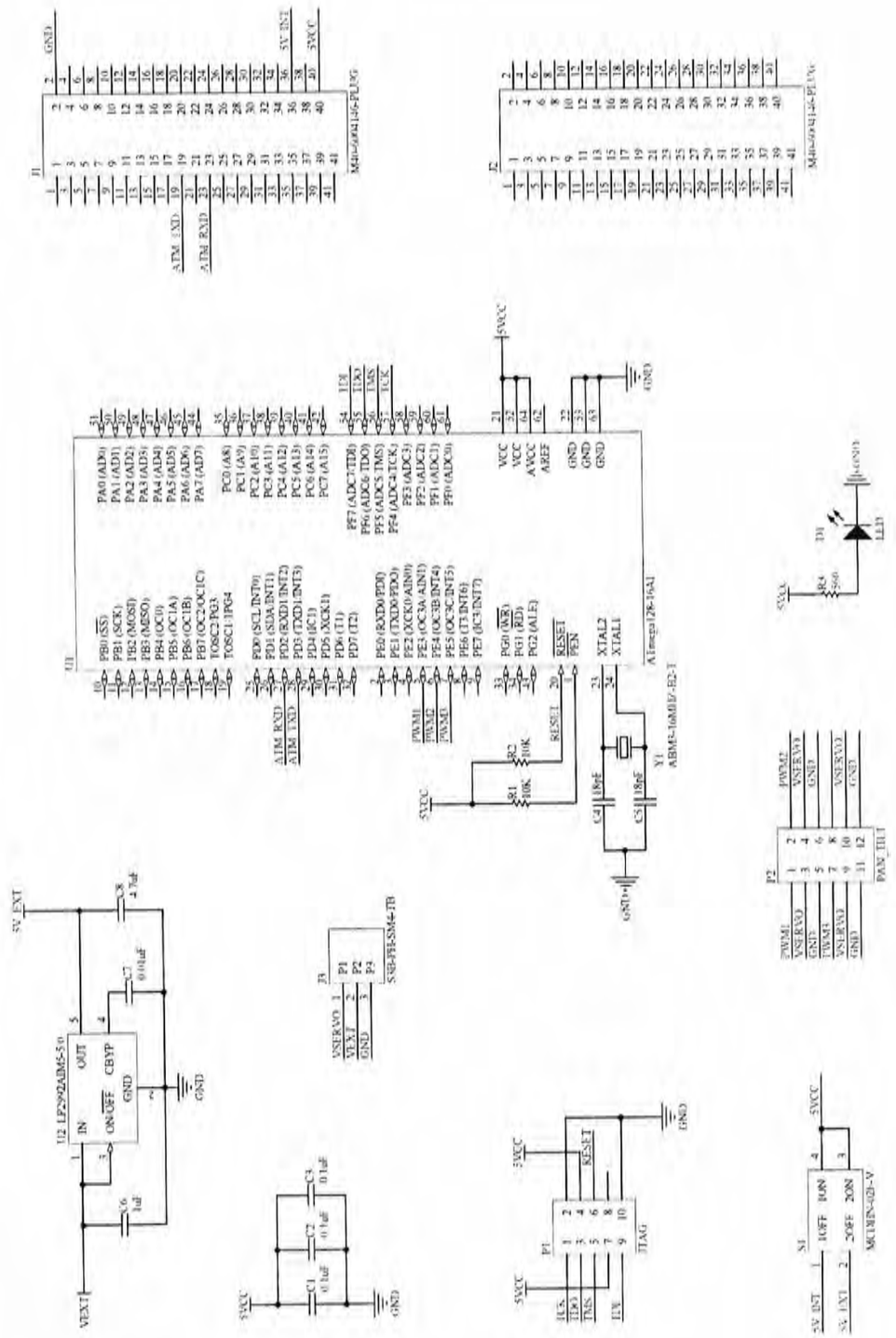


Figure A.5: Schematic of the MCU module for new camera system

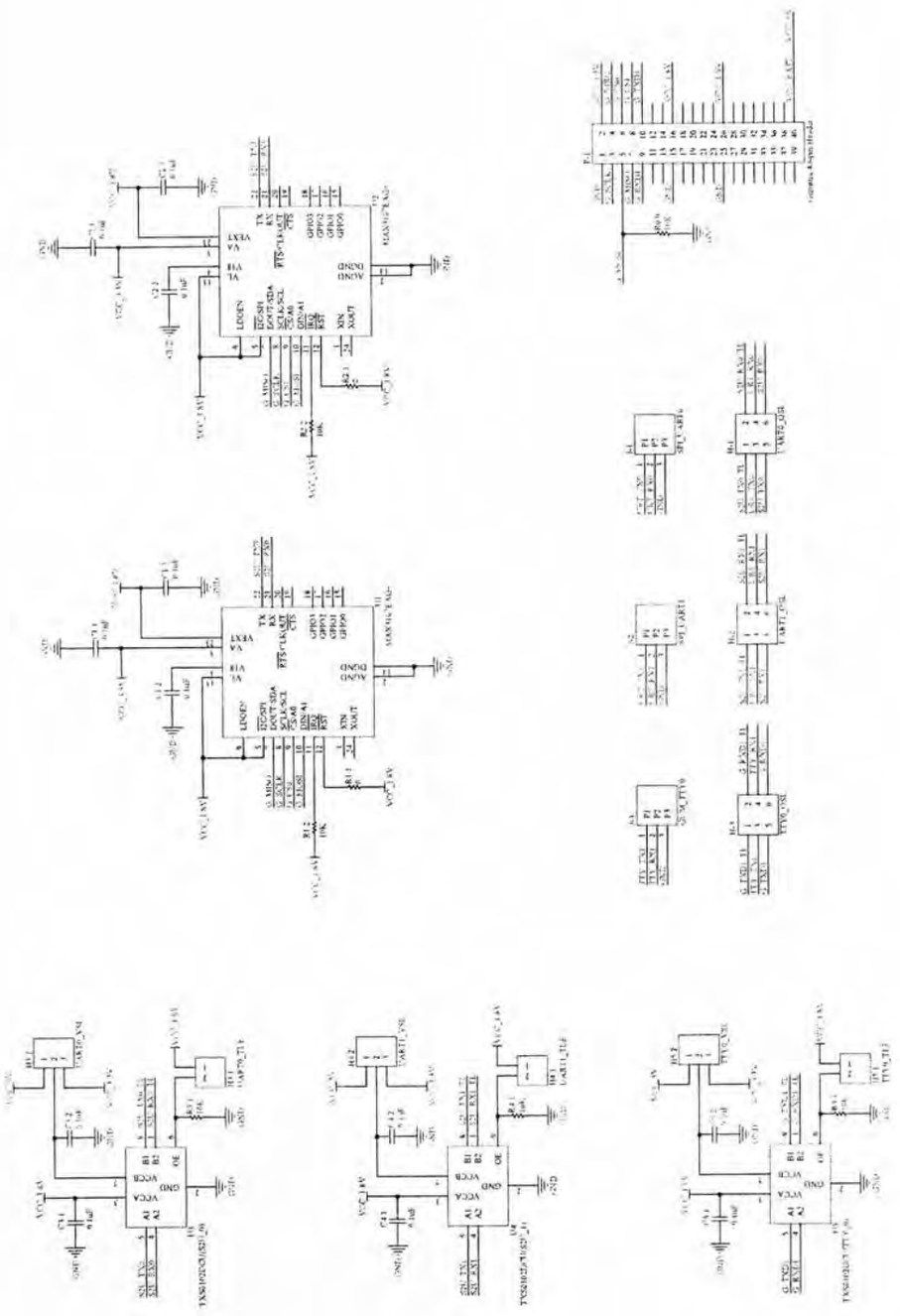


Figure A.6: Schematic of the SPI-to-UART serial expansion board

Appendix B

Image Sensor Sensitivity

Following page show the figures on the quantum efficiency (a measurement of sensor sensitivity to light) [14] of the image sensors we use for our vision system. The light sensitivity of MT9V032 image sensor in the near infra-red band explains why special color images are obtained when IR pass filter or no IR filter is used with the vision system.

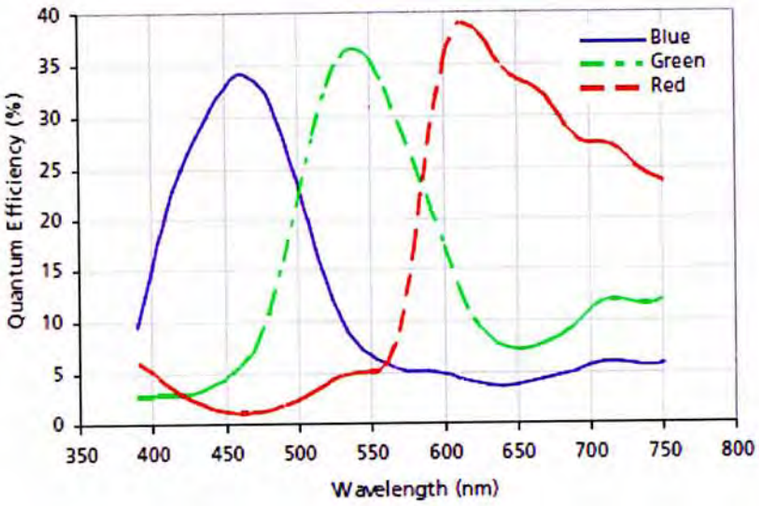


Figure B.1: Quantum efficiency of MT9T001 image sensor, from [9]

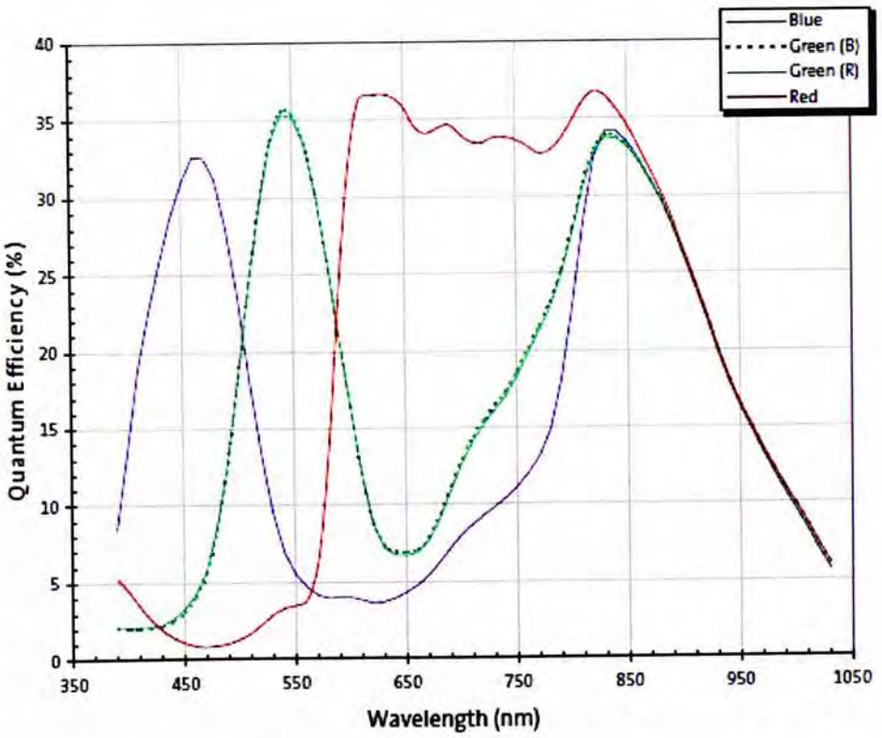


Figure B.2: Quantum efficiency of MT9V032 image sensor, from [10]

Bibliography

[1] BitBake Build Tool. http://developer.berlios.de/projects/bitbake/.

[2] CCD vs. CMOS. http://www.teledynedalsa.com/sensors/products/ccd_vs_cmos.aspx.

[3] Color Histogram. http://en.wikipedia.org/wiki/Color_histogram.

[4] EZ-USB FX2LP High Speed USB Peripheral Controller, Data Sheet. http://www.cypress.com/?docID=27092.

[5] Geodetic System. http://en.wikipedia.org/wiki/Geodetic_system.

[6] Google Earth COM API Documentation. http://earth.google.com/comapi/.

[7] GStreamer: open source multimedia framework. http://gstreamer.freedesktop.org/.

[8] Gumstix Overo Fire CoM. http://www.gumstix.com/store/product_info.php?products_id=227.

[9] MT9T001: 1/2-Inch 3-Megapixel Digital Image Sensor, Data Sheet. http://download.micron.com/pdf/datasheets/imaging/MT9T001_3100_DS.pdf.

[10] MT9V032: 1/3-Inch Wide-VGA Digital Image Sensor, Limited Data Sheet. http://www.aptina.com/assets/downloadDocument.do?id=78.

[11] OMAP35x Applications Processors Product Bulletin (Rev. B). http://focus.ti.com/lit/ml/sprt457b/sprt457b.pdf.

[12] OnPoint Targeting - Procerus Technologies, Video on target following and mosaicking. `http://www.procerus.com/video/TargetFollowingLong.wmv`.

[13] OpenEmbedded Project. `http://wiki.openembedded.net/index.php/Main_Page`.

[14] Quantum Efficiency. `http://en.wikipedia.org/wiki/Quantum_efficiency`.

[15] Rolling Shutter. `http://en.wikipedia.org/wiki/Rolling_shutter`.

[16] Streaming Data Through Isochronous/Bulk Endpoints on EZ-USB FX2 and EZ-USB FX2LP, Application Note. `http://www.cypress.com/?docID=27700`.

[17] The I^2C-Bus Specification, Version 2.1. `http://www.nxp.com/documents/other/39340011.pdf`.

[18] The OpenCV Library. `http://opencv.willowgarage.com/wiki/`.

[19] TP-LINK: Wireless Calculator. `http://www.tp-link.com/en/support/calculator.html`.

[20] UAV Challenge - Search and Rescue Challenge. `http://www.uavoutbackchallenge.com.au/2011/index.cfm?contentID=5`.

[21] Understanding Digital Camera Histograms: Luminosity & Color. `http://www.cambridgeincolour.com/tutorials/histograms2.htm`.

[22] O. Ait-Aider, N. Andreff, J. Lavest, and P. Martinet. Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. 2006.

[23] C. Arth and H. Bischof. Real-time object recognition using local features on a DSP-based embedded system. *Journal of Real-Time Image Processing*, 3(4):233–253, 2008.

[24] D. Barber, J. Redding, T. Mclain, R. Beard, and C. Taylor. Vision-based target geo-location using a fixed-wing miniature air vehicle. *Journal of Intelligent and Robotic Systems*, 47(4):361–382, 2006.

[25] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision ECCV 2006*, pages 404 417, 2006.

[26] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. `http://www.ces.clemson.edu/~stb/klt/`, 1997.

[27] G. Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344 371, 1986.

[28] J. Bouguet. Pyramidal implementation of the affine Lucas Kanade feature tracker - Description of the algorithm. Technical report, Intel Corporation, 2001.

[29] J. Bouguet. Camera Calibration Toolbox for MATLAB. `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`, 2004.

[30] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[31] D. Casbeer, R. Beard, T. McLain, S. Li, and R. Mehra. Forest fire monitoring with multiple small UAVs. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3530–3535. IEEE, 2005.

[32] W. Chen, Y. Xiong, J. Gao, N. Gelfand, and R. Grzeszczuk. Efficient extraction of robust image features on mobile devices. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1 2. IEEE Computer Society, 2007.

[33] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, pages 603–619, 2002.

[34] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, page 2142. Published by the IEEE Computer Society, 2000.

[35] G. Conte, M. Hempel, P. Rudol, D. Lundström, S. Duranti, M. Wzorek, and P. Doherty. High accuracy ground target geo-location using autonomous micro aerial vehicle platforms. In *Proceedings of the AIAA-08 Guidance, Navigation, and Control Conference*. Citeseer, 2008.

[36] J. Corbet, A. Rubini, and G. Kroah-Hartman. *Linux Device Drivers*. O'Reilly Media, Inc., 2005.

[37] K. Derpanis. The Harris corner detector. *Technical Report, York University*.

[38] C. Evans. Notes on the OpenSURF Library. *University of Bristol, Tech. Rep. CSTR-09-001, January*, 2009.

[39] W. Faig. Calibration of close-range photogrammetric systems: Mathematical formulation. *Photogrammetric engineering and remote sensing*, 41(12), 1975.

[40] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381 395, 1981.

[41] D. Frolova and D. Simakov. Matching with invariant features. *The Weizmann Institute of Science*, 2004.

[42] M. Goodrich, B. Morse, D. Gerhardt, J. Cooper, M. Quigley, J. Adams, and C. Humphrey. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89 110, 2008.

[43] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[44] R. Hartley. An algorithm for self calibration from several views. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 908 912. IEEE, 1994.

[45] R. Hartley. *Multiple view geometry in computer vision*. Cambridge university press, 2008.

[46] T. Instruments. OMAP35x Applications Processor: Technical Reference Manual, 2010.

[47] M. Kölsch and S. Butner. Hardware considerations for embedded vision systems. *Embedded Computer Vision*, pages 3 26, 2009.

[48] T. Lau, Y. Liu, and K. Lin. An experimental study of hierarchical autopilot for untrimmed hingeless helicopters. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3874 3879. IEEE, 2009.

[49] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164 168, 1944.

[50] C. Liang, L. Chang, and H. Chen. Analysis and compensation of rolling shutter effect. *Image Processing, IEEE Transactions on*, 17(8):1323 1330, 2008.

[51] D. Litwiller. CCD vs. CMOS: Facts and Fiction. *Photonics Spectra*, 2001.

[52] M. Lourakis and A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):1 30, 2009.

[53] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91 110, 2004.

[54] B. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, pages 674 679. Citeseer, 1981.

[55] Q. Luong and O. Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *International Journal of Computer Vision*, 22(3):261 289, 1997.

[56] Y. Ma, S. Soatto, J. Kosecka, Y. Ma, S. Soatta, J. Kosecka, and S. Sastry. *An invitation to 3-D vision*, volume 6. Springer, 2004.

[57] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431 441, 1963.

[58] L. Meier. Towards visual sensor networks related research on micro air vehicles. *Distributed Systems Seminar, Mobile Sensing*, 2009.

[59] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, pages 1615–1630, 2005.

[60] J. Nakamura. *Image sensors and signal processing for digital still cameras*. CRC, 2006.

[61] T. Nakamura, Y. Sakata, T. Wada, and H. Wu. High performance control of active camera head using PaLMtree. In *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on*, pages 963–968. IEEE, 2005.

[62] H. OIKE, T. KATO, T. WADA, and H. WU. A high-performance active camera system for taking clear images. *Joho Shori Gakkai Kenkyu Hokoku*, 2004(40):71–78, 2004.

[63] S. Phang, J. Ong, R. Yeo, B. Chen, and T. Lee. Autonomous mini-UAV for indoor flight with embedded on-board vision processing as navigation system. In *Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), 2010 IEEE Region 8 International Conference on*, pages 722–727. IEEE, 2010.

[64] R. Ramanath, W. Snyder, G. Bilbro, and W. Sander III. Demosaicking methods for Bayer color arrays. *Journal of Electronic Imaging*, 11:306, 2002.

[65] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh. CMUcam3: an open programmable embedded vision sensor. *CMU-RI-TR-07-13, Robotics Institute, Carnegie Mellon University*, 2007.

[66] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A second generation low cost embedded color vision system. 2005.

[67] S. Saha and S. Bhattacharyya. Design methodology for embedded computer vision systems. *Embedded Computer Vision*, pages 27–47, 2009.

[68] T. Sakamoto, C. Nakanishi, and T. Hase. Software pixel interpolation for digital still cameras suitable for a 32-bit MCU. *Consumer Electronics, IEEE Transactions on*, 44(4):1342–1352, 1998.

[69] Z. Sarris. Survey of UAV applications in civil markets. In *IEEE Mediterranean Conference on Control and Automation*, page 11, 2001.

[70] M. Schimek. V4L2 API Specification. `http://v4l2spec.bytesex.org/spec/`, 2008.

[71] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1993.

[72] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[73] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, 1996.

[74] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[75] R. Szeliski. *Computer vision: algorithms and applications.* Springer-Verlag New York Inc, 2010.

[76] C. Tomasi and T. Kanade. *Detection and tracking of point features.* Citeseer, 1991.

[77] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

[78] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[79] E. Turkbeyler and C. Harris. Mapping object movement to aerial mosaics. In *Crime Detection and Prevention (ICDP 2009), 3rd International Conference on*, pages 1–5. IET.

[80] T. Wada, N. Ukita, and T. Matsuyama. Fixed viewpoint pan-tilt-zoom camera and its applications. *The Transactions of the Institute of Electronics, Information and Communication Engineers J81DII (6)*, pages 1182–1193, 1998.

[81] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.

[82] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 57–64. IEEE, 2009.

[83] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 38(4):13, 2006.

[84] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, page 666. Published by the IEEE Computer Society, 1999.

[85] J. Zhu. Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates. *Aerospace and Electronic Systems, IEEE Transactions on*, 30(3):957–961, 1994.