國立交通大學

資訊科學與工程研究所

碩 士 論 文

嵌入式環車監控系統

An embedded system for vehicle surrounding monitoring

研 究 生：涂淵耀

指導教授：陳永昇　教授

中 華 民 國 九 十 八 年 六 月

嵌入式環車監控系統
An embedded system for vehicle surrounding monitoring

研 究 生：涂淵耀　　Student：Yuan-Yao Tu

指導教授：陳永昇　　Advisor：Yong-Sheng Chen

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

嵌入式環車監控系統

學生：涂淵耀　　指導教授：陳永昇

國立交通大學資訊科學與工程研究所碩士班

摘　　要

我的碩士論文的研究題目是視訊式的停車駕駛輔助系統，目的是希望透過整合多台攝影機畫面，產生汽車周圍 360 度的景象供駕駛人參考。這個問題牽涉到魚眼相機的影像校正、反扭曲、影像接合、以及嵌入式系統的影像處理等多項議題。

裝置在汽車前後左右的四台魚眼攝影機，畫面透過四分割器整合輸入到嵌入式系統(ADI 的 Blackfin561)，經過校正後在 PC 上先產生 Mapping table，嵌入式系統利用此 mapping table 產生圖二的結果，並及時的輸出在液晶螢幕給汽車駕駛人參考。

當障礙物出現在畫面接合處時，會因為攝影機視角的不同的關係產生鬼影或消失的狀況。針對這個問題，我也提出了解決的方法，並實作在嵌入式系統上。

An embedded system for vehicle surrounding monitoring

student：Yuan-Yao Tu　　Advisors：Dr. Yong-Sheng Chen

Institutes of Computer Science and Engineering

National Chiao Tung University

ABSTRACT

What surrounds a vehicle effects vehicle maneuvering. Since there are a lot of blind spots around a vehicle, which lead to difficult maneuvering and endanger passengers and pedestrians. In this thesis, we develop a low cost but efficiency driving assistant system which provides the surrounding image of a vehicle in bird's-eye view. By using a DSP chip and a fast-image-stitching algorithm as well as 4 fisheye cameras mounted around a vehicle, this system can instantly generate a vertical view of the vehicle from the top. This enables the driver to have a bird's-eye view of their car and its surroundings without any blind spots. We also propose a novel idea to ensure the obstacle's figure will show on the bird's-eye view image.

This vehicle surrounding monitoring system involves two major techniques, one is to find out the pixel mapping relationship between fisheye cameras and bird's-eye view image, and the other is the real time image processing in embedded system. We propose some simple methods to find out the mapping relationship, including the fisheye image distortion

model, fisheye image warping and rectifying. Then, we generate a lookup table for speeding up image processing in embedded system. On the other hand, we utilize the memory hierarchical structure and apply pipeline mechanism to enhance the throughput of embedded system. We also implement a dynamic boundary idea to let the system automatically switch image sources and reduce probability of missing obstacle's figure in bird's-eye view image. Finally, a bird's-eye view image is generated.

With this bird's-eye view surrounding monitoring system, drivers can quickly understand the surrounding environment around the vehicle. Driving and parking become more easily and safely.

誌　　謝

首先，我要感謝陳永昇老師這兩年來的熱情指導，除了在研究上持續指引我方向之外，老師認真執著、努力不懈的精神更是令我敬佩不已。兩年研究生活的點點滴滴歷歷在目，也讓我獲益良多，窮理致知的精神更是刻骨銘心，永難忘懷。感謝亮維與永煌，協助我研究及實驗不遺餘力，還有乙慈、育萱及育宏，讓我的研究生活充滿歡樂。安全監控組的學長學弟學妹，非常感謝你們在這兩年的研究討論中不斷提供想法與意見，讓我的研究更豐富，思考更多元。另外也感謝一直默默在背後支持我的媽媽與吳茵，謝謝你的陪伴及鼓勵，你是我成長奮發的動力來源。感謝振興小屋陪伴我走過風風雨雨，一起奮鬥的家維、邵穎，祝你們博士研究順利。最後，感謝曾經幫助過我的人，有你的幫助，我才能順利完成這篇論文。

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Driving Safety

Road safety is a critical issue. There are a lot of blind spots around the vehicle and it is especially dangerous that drivers fail to note the road conditions in blind spots. According to statistical information in 2007 year from Ministry of Transportation and communication R.O.C., there are total 2463 road traffic accidents. The second is fail to note road conditions that 460 counted. In recent years, vehicle-mounted cameras have grown in popularity as a tool for enhancing driver safety. There is growing global interest in car-mounted cameras as a means for providing an improved driver view to facilitate safer driving.

Lanes Detecting The information of the lanes ahead can be used to determine the lateral position of the vehicle and correct the position and heading direction of the vehicle to follow road and prevent running off the road. This technique includes lane detection and lane tracking [19]. Lane detection is the process of determining the location of lane boundaries and estimating the local geometry structure of the lane ahead. Lane tracking involves determining the location of lanes in a sequence of consecutive images using information about the lane location in previous images. Lanes detecting system consists in warning, suggesting or correcting the driver if he has not manifested the aim of a voluntary lane change or if his speed is not adapted with the road configuration (Figure 1.1). It is essential for many intelligent vehicle applications, including lane departure warning, lateral control and autonomous driving etc.

Front vehicle detecting In most crashes between vehicles, a vehicle bumps to another is in it heading direction. According to vehicle accident statistics, the main threats to a driver are from other vehicles. Consequently, robust and reliable front vehicle detection is useful to alert a driver about other frontal vehicle whose distance apart from our vehicle is less than the safe distance and to prevent accidents due to sudden braking of the other vehicle. This problem can be solved easy by using active sensors to measure the distance of the front vehicle. However, since these equipments are much expensive than the video cameras, the vision based approach using binocular stereo

Figure 1.1: **Examples of lane detection.** On the freeway, the lanes are detected to determine the lateral position of the vehicle.

analysis is preferred for vehicles detect [12, 20, 23]. The binocular stereo can also be done by fisheye camera [4], and the related fisheye epipolar geometry can reference [14]. A successful vehicle detection algorithm comprises vehicle recognition, vehicle tracking, and collision avoidance.

Surroundings monitoring Due to the structure of the vehicles, many areas around the vehicle become blind spots and those areas are threats to the driving safety, especially the area beneath rear window (Figure 1.2). Rear view cameras are becoming more of a standard feature these days. It is helpful when reverse car into a narrow parking space. But many other blind spots also exist along both sides of the vehicle, the right and left rear-view mirrors are used in the conventional driver support to help driver to know the environments where. But there exists a potential danger of bumping into the front obstacle, since the driver is distracted on the rear-view mirrors. Consequently, a surroundings monitoring system composed of multiple cameras eliminates blind spots from around the vehicle. It provides the driver awareness of what surrounds a vehicle without turning his head by displaying all images of blind spots behind him on the central monitor.

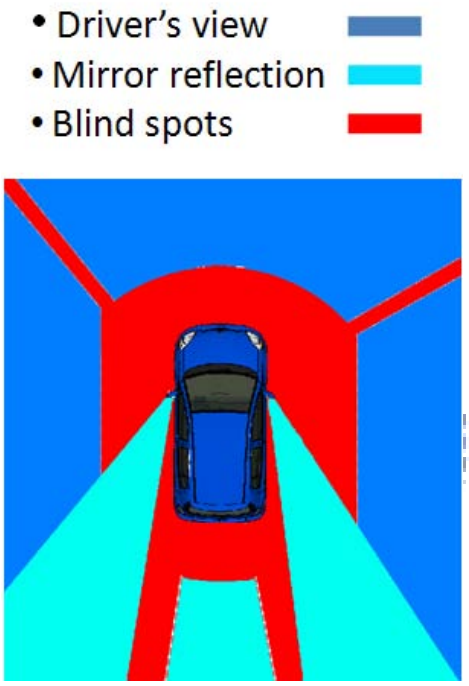


Figure 1.2: **Blind spots of a vehicle.** Many blind spots exist around a vehicle, especially on the rear and both sides of a vehicle. In this figure, the blind spots and visible area are marked in red and blue respectively.

The geographic information provided through a camera view is less, since the images are displayed separately on the central monitor. If an obstacle appears in one of images suddenly, the driver may take a few seconds to estimate the azimuth and distance of the obstacle relative to the vehicle, and may lose the golden time for maneuvering the vehicle to avoid collision with the obstacle.

In this thesis, we focus on the advanced surroundings monitoring which provide a bird's-eye view around a vehicle. The surroundings monitoring system could display the images of the blind spots by utilize four cameras mounted around the vehicle that film the roadway surrounding the vehicle. Their images are processed into a virtual bird's-eye view 2-D image that is shown on a monitor. By those vision based assistance system, drivers can easily understand surrounding environment and avoid barriers as well as accidents.

1.2 Survey of vehicle surrounding monitoring system

We will introduce the related works in this section. The related works of vehicle surrounding monitoring system including :

1 Nissan around view monitoring system

2 Honda multi-view camera system

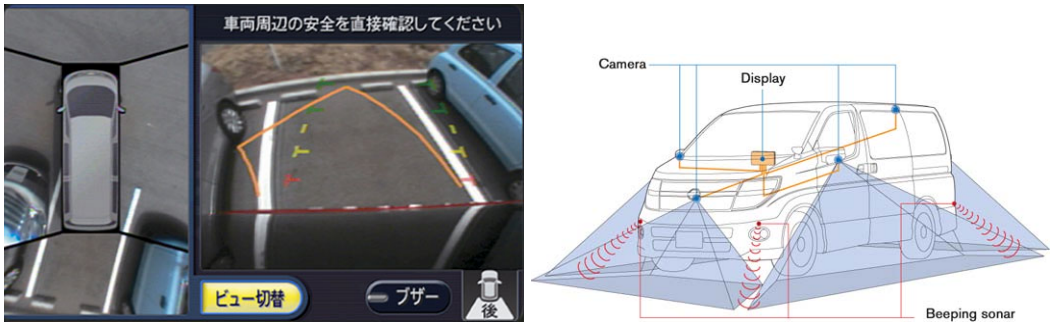
3 Fujitsu warparound view of vehicles in real time

4 Truck driving assistant system with catadioptric camera

5 Bird's-eye view vision system for vehicle surrounding monitoring

1.2.1 Around view monitoring system - Nissan

Around view monitoring system (Figure 1.3) is a parking assistant system that developed by NISSAN. The set-up uses four ultrawide-angle cameras mounted on the front, sides and rear of the vehicle, which combine to give a complete 360-degree view around the vehicle.

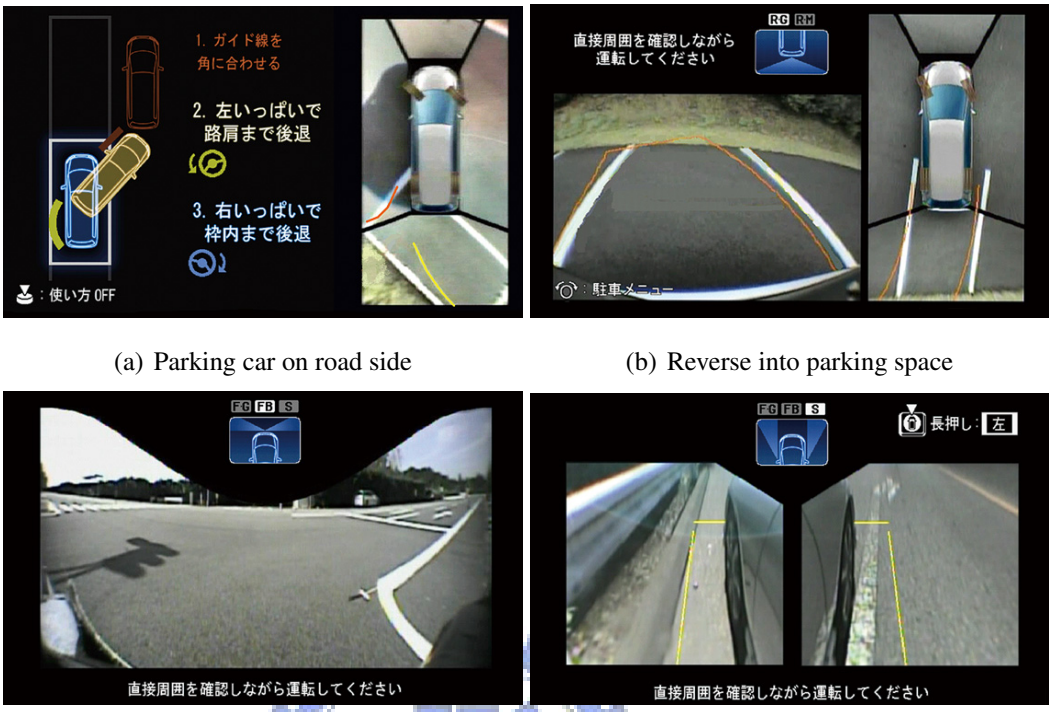


(a) Around View Monitoring system (b) AVM system setup

Figure 1.3: **Around View Monitoring system developed by NISSAN.** (a) The Around View Monitoring system. (b)The system setup of AVM system.

(a) Frontal camera (b) Side camera

Figure 1.4: **Cameras of AVM system.** (a) The frontal camera of AVM system. (b)The right side camera of AVM system.



(a) Parking car on road side

(b) Reverse into parking space

(c) Frontal image mode to see the incoming car from left and right side

(d) Side image mode for driving through a narrow way

Figure 1.5: **Honda Multi-view camera system.** (a) A mode of parking. (b) Another parking mode for reverse parking. (c) This mode is a warped frontal image. (d) This mode is used for driving through the narrow street.

That's take a look of the camera setup in AVM system. The frontal camera is mounted on the hood, just above the mark of NISSAN (Figure 1.4 (a)). The side camera is mounted on the side mirror of both side (Figure 1.4 (b)). According to (Figure 1.3 (a)), they use a flat plane to model the surrounding roadway and divide four display areas which corresponding to four cameras. There exist a black line between each display area, which dose not have image information but using sensors to detect obstacles. If there are obstables near the corner, the sersor will alert the driver.

(a) Camera setup (b) Curved plane (c) Warparound view of vehicle

Figure 1.6: **Warparound view developed by fujitsu laboratories.** (a) The camera setup. (b) The surrounding environment model is a curverd plane. (c) The warparound view of vehicle.

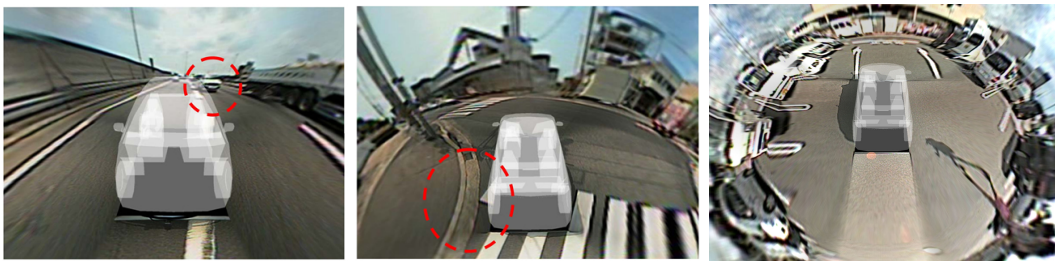
1.2.2 Multi-view camera system - Honda

Multi-view camera system is developed by Honda. The camera setup and the system display area division is similar to AVM, but this system can manually switch the display mode for different situations. While parking the car on road side or reversing, driver can switch to the parking mode to reference the bird's-eye view image (Figure 1.5 (a) (b)). When driving through a narrow street, there is a side image mode can see the surrounding image of the front wheels(Figure 1.5 (c)). While driving in lanes, driver's view field may be blocked by walls near the car. There is also a frontal image mode, which can help driver to see the incoming vehicles from left or right directions(Figure 1.5 (d).

The multi-view camera system also use a flat plane to model surrounding roadway, and the display areas of each camera is fixed.

1.2.3 Warparound view of vehicle - Fujitsu

This is a driving assistant system developed bu fujitsu laboratories. They also develop a SOC chip that includes all the parts and functions usually found in a complete system, such as a microprocessor, chipset, video chip, and memory. This system also support OpenCL-ES libary to handle 3D graphics and display. Comparing with previous system, the camera



(a) Rear view (b) Turning left (c) Warparound view of vehicle

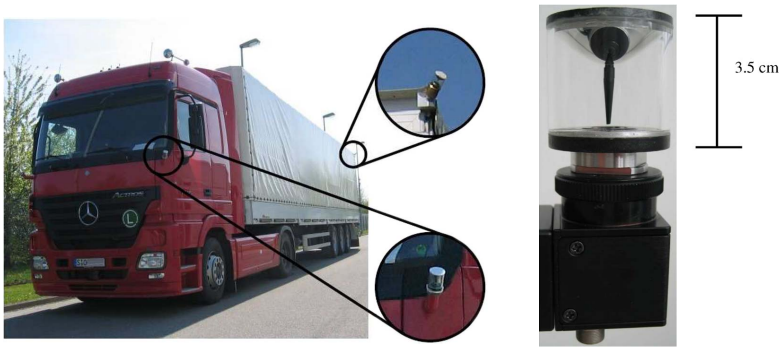
Figure 1.7: **Warparound view of vehicle in different direction.** (a) Sample view at merging point on a highway : perspective from front facing of vehicle. (b) Sample view when turning left : perspective from above-rear. (c) The warparound view of the vehicle.

setup approach and camera type are similar. The main different is the surrounding roadway model. This system use a curved plane to model the surrounding environment, and project camera images onto the curved plane. They average the image pixels of the intersection of different camera's image rather than use a black line to divide each camera's display area.

There are some sample views of this system. (Figure 1.7 (a)) is a sample view at merging point on a highway. The perspective view from front facing of vehicle clearly show the rear view with available space and approaching vehicles. (Figure 1.7 (b)) is a sample view when turning left, which is perspective from above-rear, and the curb clearance of left rear can be seen clearly. (Figure 1.7 (c)) is the warparound view in a parking lot, and this view is perspective from above rear.

1.2.4 Truck driving assistant system with catadioptric camera

This truck driving assistant system [6] is published in intelligent transportation systems of IEEE transactions. They also publish some related works of driving assistant system that using catadioptric cameras [7, 16, 21]. They use catadioptric cameras to capture the surrounding roadway of a truck (Figure 1.8 (a) (b)). The view field of catadioptric camera is greater than the fisheye camera. This system setup those catadioptric cameras at four corners of a truck. The ground model is also a flat plane, but the display areas' division is asymmetric. The asymmetric division can ensure the obstacle's figure dose not vanish in the bird's-eye view image. (Figure 1.9 (a)(b)(c)(d)) is the sameple views of this catadioptric



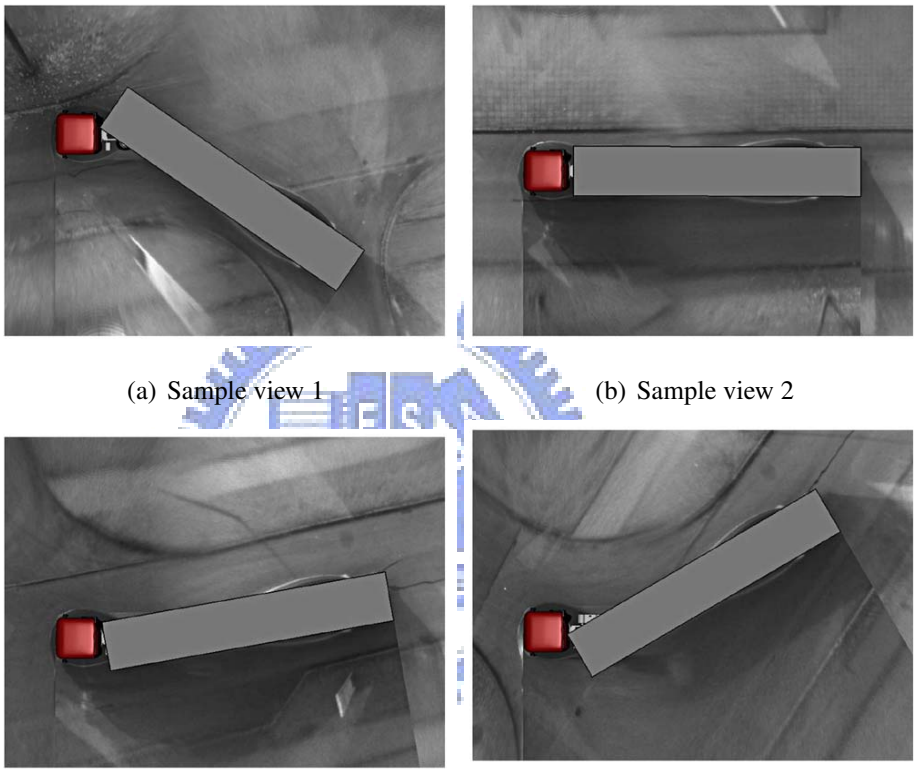
(a) Camera setup (b) Catadioptric camera

Figure 1.8: **Truck driving assistant system.** (a) The camera setup of truck driving assistant system. (b) The catadioptric camera.

camera truck driving assistant system.

1.2.5 Bird's-eye view vision system for vehicle surrounding monitoring

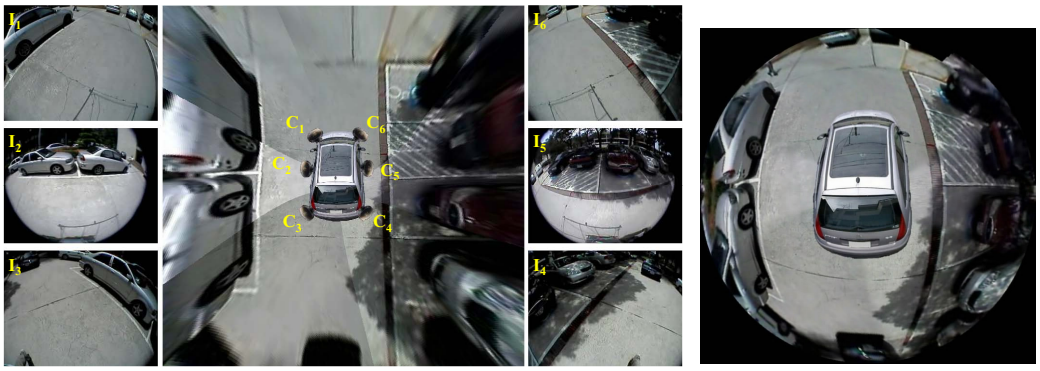
This bird's-eye view vision system [13] is published in Robvis 2008 (Figure 1.10(a)). There are six fisheye cameras used in this system (Figure 1.10(b)), which are mounted on the middle of both sides and four corners of a vehicle. This system use a flat plane to stitch those fisheye cameras' images, and this system divide each fisheye camera's display area according to the optimal seams, which are dicided by the distribution of each fisheye camera's calibration error. Moreover, they use dynamic programing skill to align the image signals on the optimal seams and spread out the aligning result by wendland function. In order to enhence the image resolution and quality at farther monitoring areas, they apply a distortion model to warp the virtual perspective image into a fisheye image. (Figure 1.11(a)(b)) are the sample results.



(a) Sample view 1 (b) Sample view 2

(c) Sample view 3 (d) Sample view 4

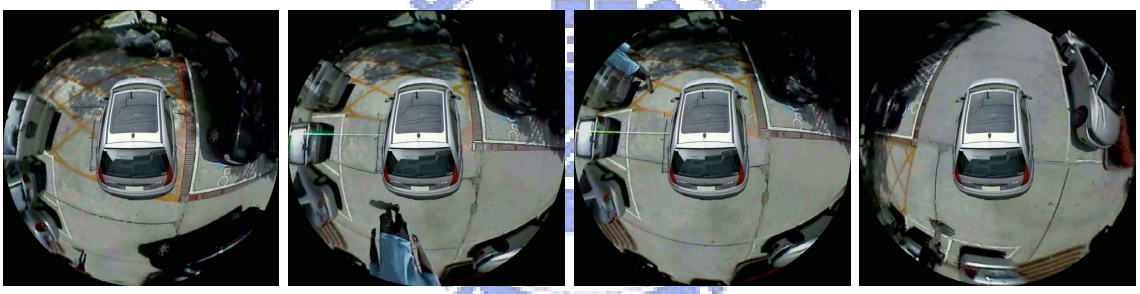
Figure 1.9: **Sample views of truck driving assistant system.**



(a) Sample view of bird's-eye view vision system

(b) Virtual perspective view and cameras setup of this system

Figure 1.10: **Sample views of bird's-eye view vision system.** (a) The camera setup and the virtual perspective bird's-eye view image. (b) The sample view of bird's-eye view vision system.



(a) Sample views of bird's-eye view vision system

(b) Sample views of reversing parking

Figure 1.11: **Sample views of bird's-eye view vision system in parking lot with a pedestrian].** (a) The sample view of bird's-eye view vision system. (b) The sample view of bird's-eye view vision system while reversing parking.

1.3 Overview of the proposed method

We propose a low cost but efficiency embedded vehicle surveillance system that only need four fisheye cameras, a quad splitter and a low cost DSP embedded system to generate bird's-eye view image in real time. Also we propose a dynamic boundary idea to switch image source according the image contents, and ensure driver can see figures of obstacles in bird's-eye view image.

Our work can be divided in two parts. First task is to build a lookup table, which defined the mapping relationship from bird's-eye view image to four fisheye camera images (Figure 1.12). The procedures of generating such a table include fisheye image distortion model, fisheye image rectification, 2D image projection (homography), and image format conversion between PC and embedded system. The second task is the implementation of real time image processing in embedded system. The lookup table is used to speed up the image processing procedures in embedded system, and we also apply the pipeline mechanism to reduce the data transition time between cache and SRAM. Moreover, we implement the dynamic boundary idea to this system to ensure the driver can see obstacles near the vehicle. (Figure 1.12) is the flow chart of lookup table generation, and (Figure 1.13) is the flow chart of real time image processing in embedded system.

1.4 Thesis Organization

In chapter 2, we introduce the first part of the proposed method for finding the mapping relationship between fisheye images and the bird's-eye view image. We also generate a lookup table for embedded system to reduce the computation of image processing. In chapter 3, we introduce the embedded system and the real time image processing application. The implementation details are revealed in this chapter, too. Chapter 4 is the results of our work. Finally, the conclusions are in last chapter.

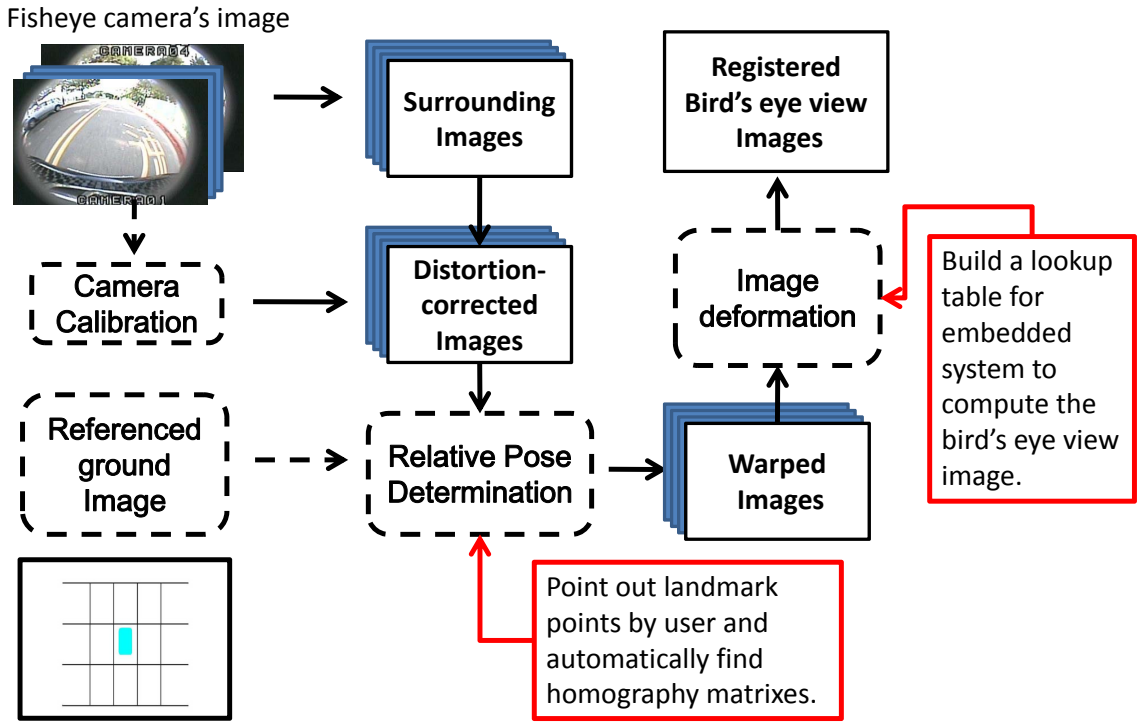


Figure 1.12: **Flow chart of generating the lookup table for embedded system.** This flow chart illustrate the procedures for generating the lookup table. The procedures includes image format conversion, fisheye distortion principle, fisheye image rectification, image warping, and 2D image projection.

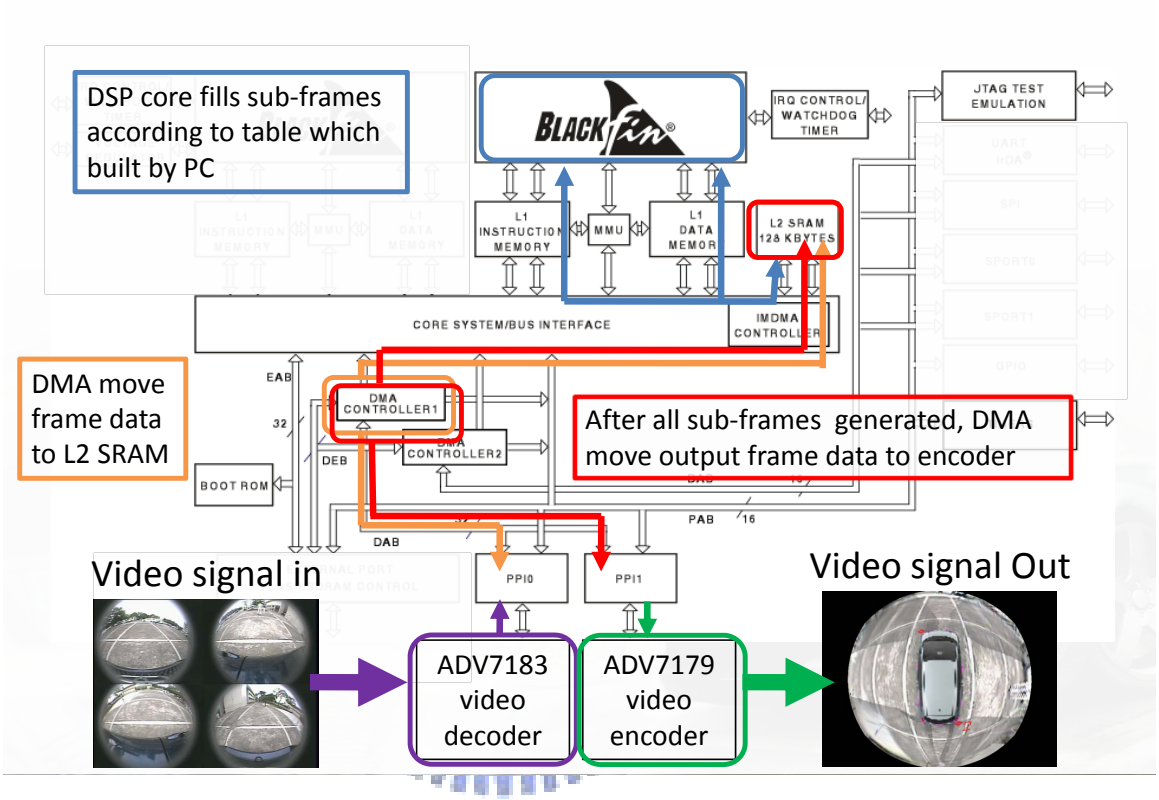


Figure 1.13: **BlackFin-561 video processing flow chart.** This flow chart illustrate the video processing sequence in blackfin-561. When receive the input frame from quad splitter, the decoder decode video signals into itu-656 format frame data. Then, the DMA move the frame data into the SRAM through the PPIs(parallel peripheral interfaces). The DSP core generate the bird's-eye view image and saved in SRAM. After the whole bird's-eye view image has been generated, another DMA will move the result to the encoder. Then, the bird's-eye view image frame immediately show on screen.


E
S
A
1896

Chapter 2

System Setup of Embedded Bird's-eye View Driving Assistant System

In this chapter, we introduce the system setup and preprocessing procedures of the embedded vehicle surrounding monitoring system. First of all, we introduce the system setup and image rectification. Then we illustrate the 2D image homography with projection model. After that, we generate a lookup table according to itu656 image formate used in embedded system. Finally, we give the summary for this chapter.

2.1 System Setup and Image Rectification

We will introduce the system setup of this embedded vehicle surrounding monitoring system from each component to the system connections. With those components, we can obtain the video signals around a vehicle , compose four cameras' signal into one channel video signal, which will be taken as the processing input of embedded system. The input video signals will be processed frame by frame and output to the screen for driver to reference. At the end of this section, we also show how to rectify the image that distorted by fisheye lens.

(Figure 2.1). is the layout of the vehicle surrounding monitoring system. Four 180 degree fisheye cameras mounted on front, both sides and rear of a car to monitor surrounding environment. Those four channels will be combined into one channel by a quad splitter. The output of quad splitter will be taken as embedded system's input. While receiving the input video frame, the embedded system will use this input frame to synthesize the result and output to a screen for the driver to reference.

In order to capture the environment around a vehicle and reduce the number of cameras, we use cameras with 1.7mm focal length fisheye lens (Figure 2.2). Each fisheye camera have 180 degree view angle that can cover one side of the vehicle. Here we use Blackfin-561 (Figure 2.3) as our embedded platform for real-time video processing. BF561 is a dual cores embedded system with large on-chip memory and high data throughput, which tailored for the needs of imaging and consumer multimedia applications. But BF561 only have one video input and one video output, so we need a real-time color quad splitter (Figure 2.4) to combine four fisheye cameras' video signals into one channel video signal.

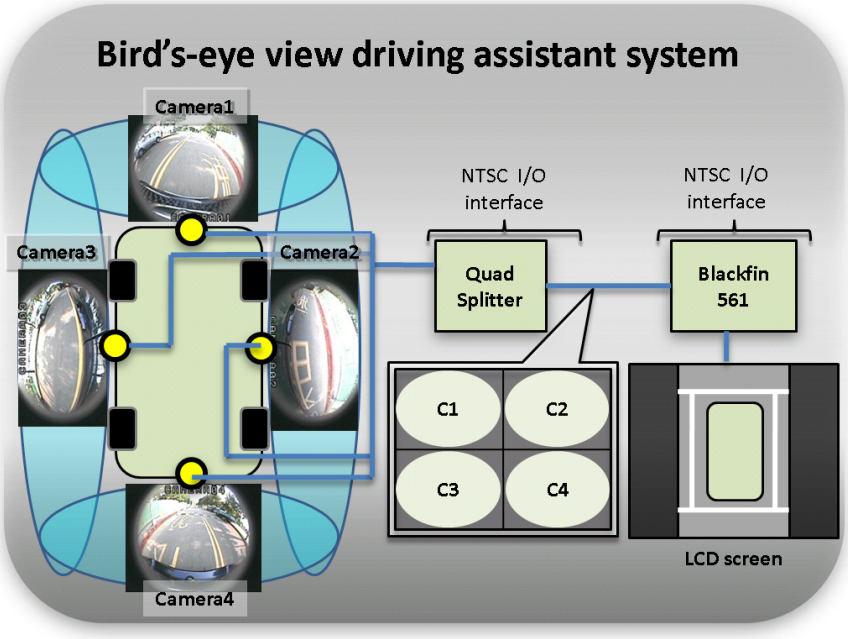
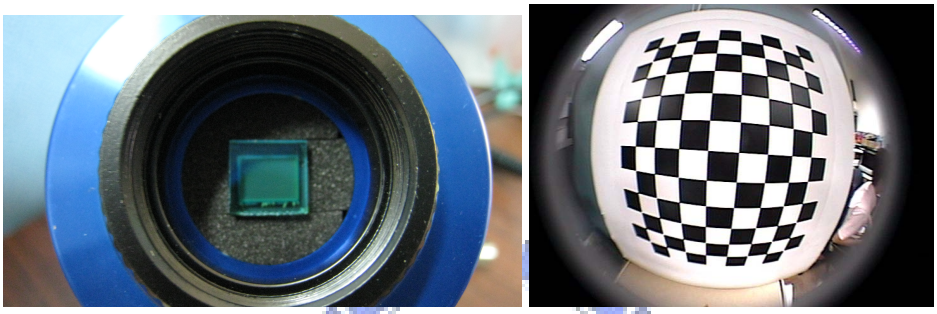


Figure 2.1: **Layout of the vehicle surrounding monitoring system.** Components and connections of the vehicle surrounding monitoring system.

2.1.1 Mount fisheye cameras on a vehicle

The position where we mount cameras on the vehicle directly effect the reconstruction of the bird's-eye view result. The shape of a car in the top view can be simplify as a rectangle. Each fisheye camera have 180 degree view field, which can exactly cover a view of one side of the vehicle. Because the bird's-eye view reconstruction is generated by the back projection of cameras' image frames, to mount cameras on the middle of each side can maximize the utilities of camera CCDs and keep important contents in the middle of image. Because the fisheye camera calibration and rectification processes may introduce some errors and the errors will increase according to the distance between target point and principle point [6, 13], mounting cameras on the middel of each side can reduce the errors caused by calibration and rectification. (Figure 2.5) are the positions where we mounting cameras on a vehicle, (Figure 2.6) are the captured scenes. We use Toyota Yaris as the experiment vehicle.

(a) Fisheye camera (b) Fisheye lens



(c) Camera CCD (d) Fisheye image

Figure 2.2: **Fisheye camera, 1.7mm focal length lens, 1/2 inch CCD and image shoot by fisheye camera.** (a) This is the fisheye camera we use. (b)(c) The focal length of the fisheye lens is 1.7mm, and inside the camera is a 1/2 inch CCD. This combination allow our camera to capture a 180 degree viewing angle. (d) This image is captured by our fisheye camera. The chess board calibration pattern is distorted due to the refraction of the fisheye lens.

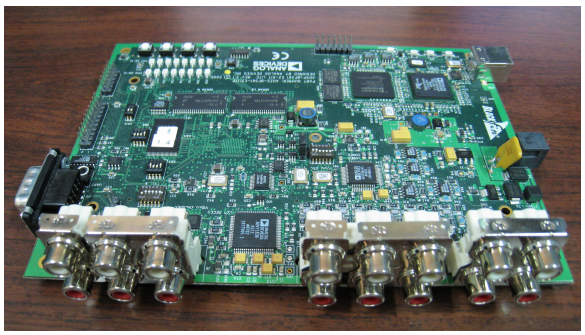


Figure 2.3: **ADSP Blackfin-561** This is BlackFin-561, which have dual cores and large On-chip memory with high data throughput. With those features, BF561 is tailored for the needs of imaging and consumer multimedia applications.

(a) Quad (b) QuadBackSide

Figure 2.4: **Quad splitter can resize four channels' video signals and combind into one channel video.** (a) This is the color quad splitter we use. (b)This is the back side of quad splitter, there are four video-in channels and three different functional video-out channels.

2.1.2 Fisheye projection principle

The principle of fisheye projection is that the distance between a distorted image point and the principal point is roughly proportional to the angle between the corresponding incoming ray and the optical axis [5, 17] (Figure 2.7 (b)). The distance between a distorted image point and the principal point is called radial radius. For a fisheye lens with ω FOV, a ray with an incident angle $\omega/2$ projected onto the outer border of the circular fisheye image. This means that the resulting image point has the maximal radius $\breve{R}$ to the optical axis. The relation between the angle of incidence and the resulting radial radius of the image point is constant for the whole image. Consequently the following ratio equation can be set up as basic equation for the fisheye projection,

$$\frac{\breve{r}}{\breve{R}} = \frac{\theta}{\frac{\omega}{2}} \tag{2.1}$$

where $\breve{r}$ is the distorted radial radius of a certain image point and θ is the incident angle of whose corresponding 3D ray.

2.1.3 Distortion model and fisheye image rectification

We use the Field of View (FOV) [5] as the model of radial distortion(Figure 2.8). Without loss of generality, we assume an object point M in the world coordinate, and it project on the image point $\breve{m} = (\breve{u}, \breve{v}$ with incident angle θ obeying the fisheye projection. Since $\breve{m}$

(a) Camera 1 on hood (b) Camera 2 on right side

(c) Camera 3 on left side (d) Camera 4 on rear

Figure 2.5: **Four fisheye cameras are mounted on the middle of each side to obtain the surrounding scenes.** (a) Camera 1 for front view. (b) Camera 2 for the view of right side. (c) Camera 3 for the view of left side. (d) Camera 4 for rear view.

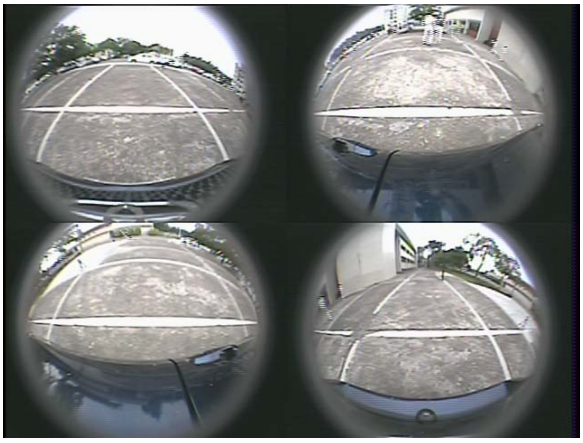
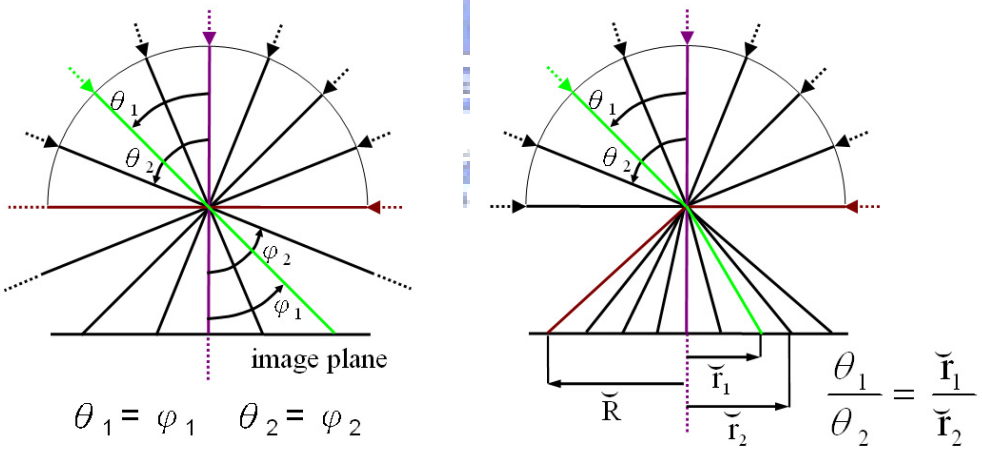


Figure 2.6: **Surrounding scenes captured by four fisheye cameras.** This is the surrounding scene that captured by four fisheye cameras, these cameras are mounted on the middle of each side of a vehicle. The video output of four cameras will be composed by the quad splitter.



(a) Perspective projection (b) Fisheye projection

Figure 2.7: **Comparison between the perspective projection and the fisheye projection.** (a) The refraction angle φ of any captured ray is equal to it incident angle θ in perspective projection. (b) The radial radius $\breve{r}$ of a projection is proportional to the incident angle θ of corresponding light ray.

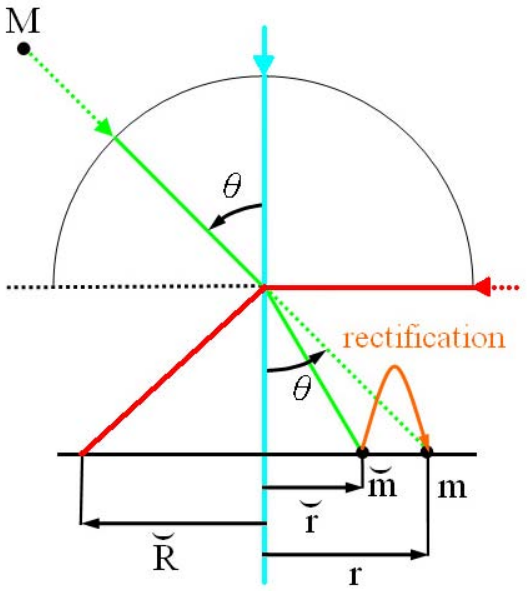


Figure 2.8: **Distortion correction for fisheye images.** The distortion correction is a process for rectifying a fisheye projection $\breve{m}$ of a object point M to its perspective projection m.

and its distortion-free image point $m = (u, v)$ are in the same radial direction, the relation between m and $\breve{m}$ can be describe by the relation between their radial radius r and $\breve{r}$

$$r = \tan\theta \cdot f \tag{2.2}$$

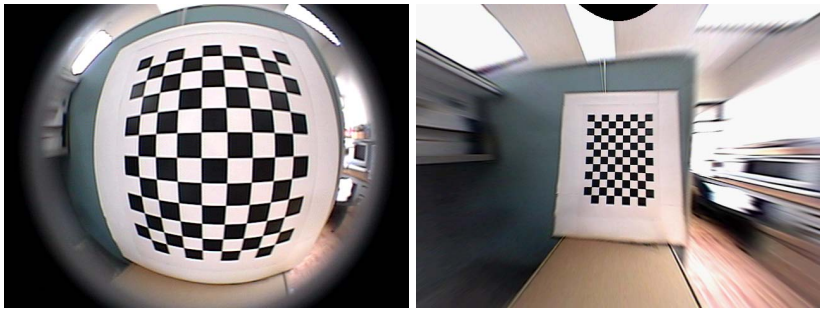
$$= \tan\left(\frac{\omega\breve{r}}{2\breve{R}}\right)\frac{\breve{R}}{\tan\left(\frac{\omega}{2}\right)} \tag{2.3}$$

$$= \frac{\breve{R}\tan\left(\frac{\omega\breve{r}}{2\breve{R}}\right)}{\tan(\frac{\omega}{2})} \tag{2.4}$$

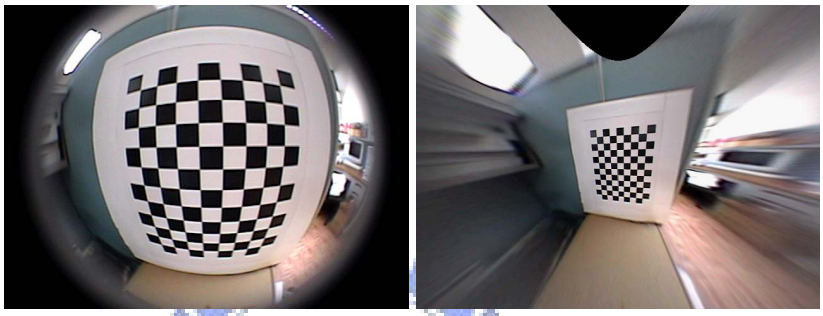
and its inverse are

$$\breve{r} = \frac{1}{\omega}2\breve{R}\arctan\left(\frac{r\tan\left(\frac{\omega}{2}\right)}{\breve{R}}\right) \tag{2.5}$$

The estimation and optimization of model parameters can reference [13, 22]. (Figure 2.9) are the original fisheye images and rectified results.



(a) The original fisheye image 1. (b) The rectified fisheye image 1.



(c) The original fisheye image 2. (d) The rectified fisheye image 2.

(e) The original fisheye image 3. (f) The rectified fisheye image 3.

(g) The original fisheye image 4. (h) The rectified fisheye image 4.

Figure 2.9: **Fisheye images and virtual perspective image.** (a) Original fisheye image 1. (b) The virtual perspective image reconstructed from image 1. (c) Original fisheye image 2. (d) The virtual perspective image reconstructed from image 2. (e) Original fisheye image 3. (f) The virtual perspective image reconstructed from image 3. (g) Original fisheye image 4. (h) The virtual perspective image reconstructed from image 4.

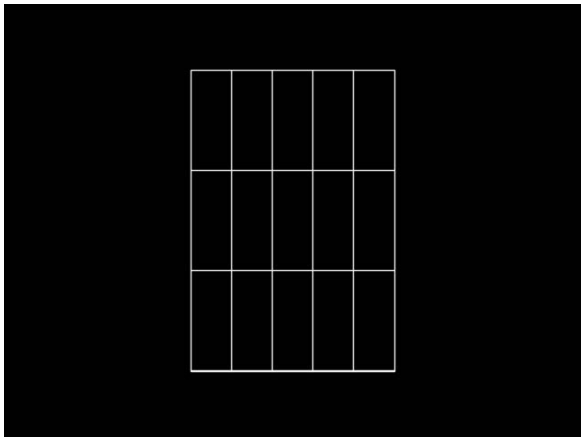


Figure 2.10: **Calibration patten.** The calibration patten is generated according to the size of parking space.

2.2 Homography - 2D image projection

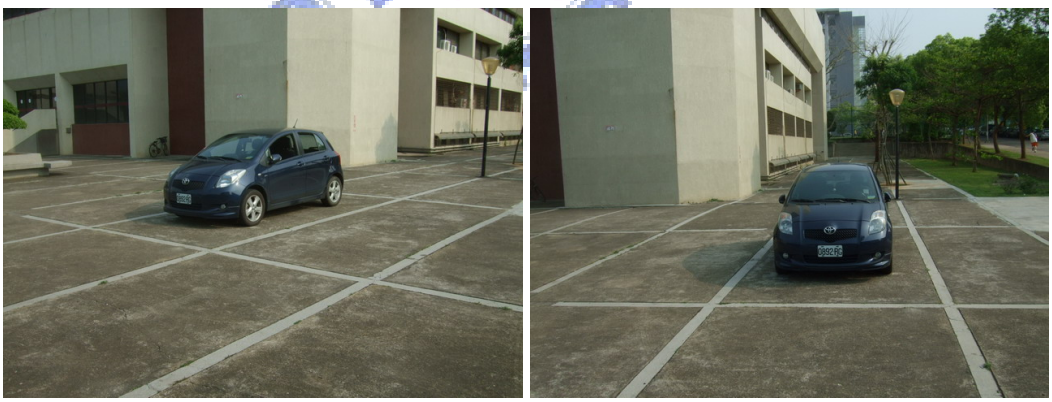
Once the fisheye camera has been calibrated, all images taken by this camera can be rectified to the virtual perspective images without the distortion. Straight lines remain straight in the distortion-free images, it provides convenience for image stitching.

We apply the same approach in [8–11] to get the corresponding points in both reference plane and virtual perspective images of each camera. Utilizing straight lines in the parking lot, four corners of rectangle parking space are taken as the feature points. Each parking space (Figure 2.12) is 5 meters in length and 2 meters wide. The calibration patten (Figure 2.10) is generated according the size. We mark out those corresponding feature points manually to estimate homography matrixes [18]. Each homography matrixes estimation is using feature points of the parking space in front of the fisheye camera. This manually process only have to do once, and the process is like (Figure 2.11). The feature points are marked in same sequence. We also take this calibration patten as the reference plane, the size of parking space effect the view field of the bird's-eye view result directly.

Because vehicles are always driving on the road, we can simply use a flat plane to model the roadway surrounding the vehicle. Without loss of generality, we assume the ground around a vehicle is flat and use a virtual flat plane to model it. Each rectified fisheye image will apply a 2D image projection or homography to map on the reference plane. We serve

(a) Corresponding points in reference plane (b) Corresponding points in rectified fisheye camera image

Figure 2.11: **Comparison between the perspective projection and the fisheye projection.** (a) The corresponding points are marked out manually in the reference plane. (b) The corresponding points are marked out manually in rectified fisheye camera image with the same sequence of (a).



(a) Parking lot (b) Parking lot 2

Figure 2.12: **Parking lot.** The parking lot with straight lines. Each parking space is 5 meters in length and 2 meters wide.

such a plane as reference plane and assume this plane is on the plane $Z = 0$, the relative pose between every pair of cameras is invariant, and the Z coordinates of every cameras Z_i are preserved. By Assuming reference plane is on $Z = 0$ of the world coordinate system. We can relate a 3D object point $M = [X, Y, 1]^T$ on the reference plane and its 2D image point $m = [u, v, 1]^T$ by the perspective transformation. it is also known as homography. It can be written as $s\tilde{m} = H\tilde{M}$ or

$$s\tilde{m} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tilde{M} \tag{2.6}$$

where s is a scalar which normalizes the resulting homogeneous coordinate. We can obtain the inhomogeneous coordinate m by

$$u = \frac{h_{11}X + h_{12}Y + h_{13}}{s} \tag{2.7}$$

$$v = \frac{h_{21}X + h_{22}Y + h_{23}}{s} \tag{2.8}$$

$$s = h_{31}X + h_{32}Y + h_{33} \tag{2.9}$$

The homography matrix can be estimated by applying SVD (Singular Value Decomposition) method.

With four pairs of corresponding points, we can estimate the homography matrix which can project points from the pattern coordinate to the rectified fisheye images. For convenient, we can generate a calibration patten just like the lines of the parking lot and park the experiment vehicle in the middel of parking space. After getting fisheye images, we can apply image rectification processes mentioned in section 2.1 to correct those fisheye images. Then, the homography matrices estimation can be done by marking out the corresponding points in virtual perspective images and calibration pattern manually. Since the homography matrix H is a 3 by 3 matrix, and $h_{33} = 1$, there are eight parameters need to be estimated. Because each point have x value and y value, by manually marking out four pairs of corresponding points the homography matrix can be estimated.

The implementation details of applying SVD method to estimate homography matrix are revealed below. According to equation 2.6 We have

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{2.10}$$

$P_{ri} = (u_i, v_i)$ and $P_{vi} = (x_i, y_i)$ refer to corresponding points in reference plane and virtual perspective image, respectively. H_{rv} donate the homography matrix corresponding to one fisheye camera that can transform points from reference plane to the virtual perspective image (rectified fisheye image) of the fisheye camera.

Before applying SVD method, the coordinate of corresponding feature points and known values of homography matrix must be arranged into the equation like $AX = B$. First, unknown values in H_{ri} are arranged into a 8x1 vector X, where

$$X = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{32} & 1 \end{bmatrix}$$

, and the x and y values of each feature point in virtual perspective image can be arranged in a 8x1 vector B.

$$B = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 & y_0 & y_1 & y_2 & y_3 \end{bmatrix}$$

The corresponding feature points in reference plane will be arrange into a 8x8 matrix A. The elements in A are generated by belowing equations. For i = 1 to 4 ,where

$$\begin{aligned} a_{i0} = a_{(i+4)3} &= u_i, \\ a_{i1} = a_{(i+4)4} &= v_i, \\ a_{i2} = a_{(i+4)5} &= 1, \\ a_{i3} = a_{i4} = a_{i5} &= 0, \\ a_{(i+4)0} = a_{(i+4)1} = a_{(i+4)2} &= 0, \\ a_{i6} &= -u_i x_1, \\ a_{i7} &= -v_i x_1, \\ a_{(i+4)6} &= -u_i y_1, \\ a_{(i+4)6} &= -v_i y_1, \end{aligned}$$

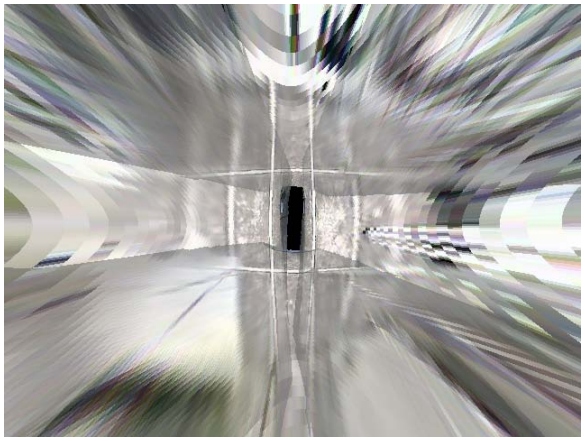


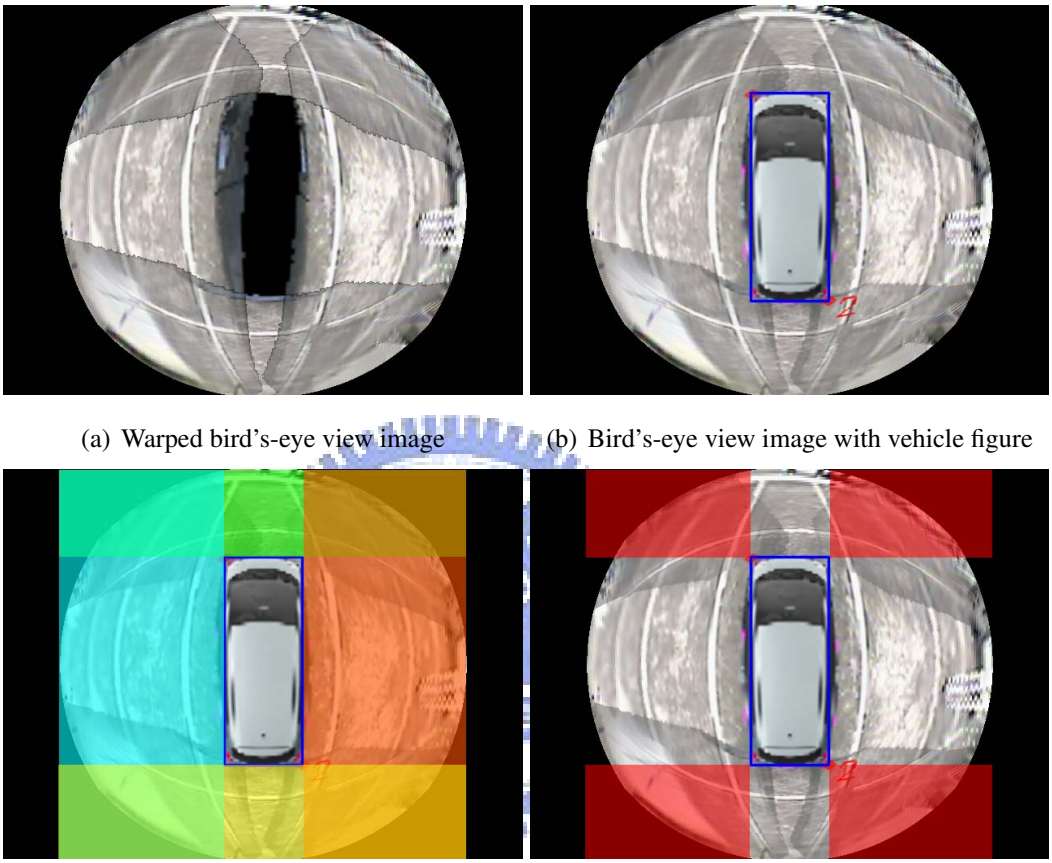
Figure 2.13: **Virtual perspective result.** This virtual perspective bird's-eye view image is generated by stitching four fisheye cameras' virtual perspective images into the reference plane.

Then we can apply SVD method to solve $AX = B$. The X vector can be arrange to homography matrix H_{ri}, and H_{ir} is the inverse of H_{ri}. After we find homogeaphy matrixes, the relations between fisheye cameras and the reference plane are decided. The projection result is in (Figure 2.13).

Observe (Figure 2.13), the low resolution image near the boundary is caused by amplified vibration. Compare the original fisheye images (Figure 2.9 (e)) and (Figure 2.9 (f)) with the virtual perspective bird's-eye view image, the rectify process reduce the information contant and decrease the image quality. So after apply image homography, the low resolution virtual perspective image will be amplified especially in the areas that are far to the middle of reference plane. On the other hand, the black area in the middle part of virtual perspective bird's-eye view image is the vehicle, we can overlap a car figure to illustrate and let the driver understand the bird's-eye view image easily.

In order to improve image resolution and increase view field, we apply the same method in [13]. By using the FOV model mentioned before to warp the virtual perspective image into a fisheye image (Figure 2.14 (a)). Furthermore, to get the warped image and keep the resolution and quality, the interpolation must be done in the original fisheye image rather than virtual perspective one.

On the other hand, we need to manually mark out the left top and right bottom corners



(a) Warped bird's-eye view image (b) Bird's-eye view image with vehicle figure

(c) Partitions of each camera image's display areas (d) Overlapping areas in bird's-eye view image

Figure 2.14: **Bird's-eye view image, overlapping vehicle figure and the partition of each camera image's display field.** (a) Warped bird's-eye view image. By applying FOV model, the warped bird's-eye view image has better image resolution. (b) The warped bird's-eye view image with the vehicle figure inside. (c) The partition of each fisheye camera image's display field. (d)The overlapping areas in the bird's-eye view image.

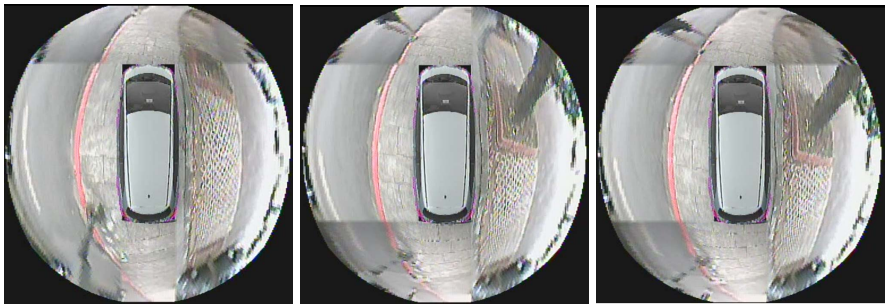
of the car figure in the virtual perspective bird's-eye view image. With this information, we can also apply the warp effect to a vehicle's top down view. Then, we put the warped vehicle figure on the warped bird's-eye view image (Figure 2.14 (b)). On the other hand, the left top corner and bottom right corner in virtual perspective bird's-eye view image can not only decide the rectangle size of the vehicle, but also decide each camera's display area. Since each fisheye camera have almost 180 degree view field and exactly cover one side the vehicle, we can use these two point to divide the display area of each camera in the virtual perspective bird's-eye view image and the warped one. In warping process, these two point just deside a rectangle in the middel of bird's-eye view image like (Figure 2.14 (b)), and also divide the bird's-eye view image into 9 blocks. The middle block is the vehicle. Each camera provides one side's video signal(top, right, left and bottom, total four sides), and there are four areas are overlapped by each two neighbor fisheye cameras (the red areas in Figure 2.14 (d)).

2.2.1 Problems caused by non-ground level objects

As mentioned before, we simply use a flat plane to model the roadway around the vehicle and apply homography to project fisheye image onto the flat plane model. The figures of lanes, road signs, manhole covers and road which are belong to ground level are projected well on the plane model. But in practice, there are full of obstacles around the vehicle. We call them non-ground level objects. While camera images are projected onto the flat plane, those obstacles are also been projected, too. Because the bottom part of a obstacle is connected with ground, the projection does not distort it. But other patrs above obstacle will be stretched.

If we fix the camera image display areas of bird's-eye view image, the figures that belong to ground level can be stitched together. But those figures of non-ground level objects are stitched only in bottom parts, and other parts may be covered by neighbor camera's image and cause the missing effect. If we average the image in overlapping area, the bottom parts of obstacles will stitch together, but other parts will fork out from the bottom, we call it ghosting effect.

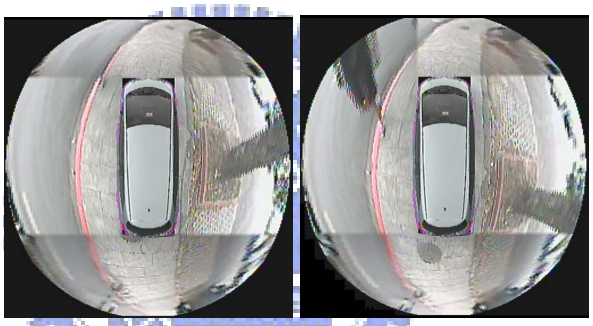
This is a serious of bird's-eye view images (Figure 2.15) while driving car on the pave-



(a) A tree is missing in bird's-eye view image.

(b) The tree's figure shows up.

(c) The pedestrain is near the image boundary.



(d) The pedestrian is missing in bird's-eye view image.

(e) The pedestrian's figure shows up again.

Figure 2.15: **A serious of frames demonstrate the projection of obstacles and the object missing effects.** (a) There is a tree on the right side of vehicle but missing in the bird's-eye view image. (b) By moving right side's camera image on the top level, the tree's figure is show up, and there is a pedestrian coming on the left side. (c) In the bird's-eye view image, the partition is walking and near the boundary between front camera and camera on left side. (d) The partition is missing in the bird's-eye view image. (e) The partition is show up again.

ment. There is a tree on the right side of the car. On the left side, there is a pedestrian coming. The object missing effects are happened when objects' figures is just on the boundary between different cameras' display areas in bird's-eye view image. In (Figure 2.15 (a)), the tree's figure is captured by the camera on right side, but can not be seen by the frontal camera. Since object's figure will be projected along the line from camera to the bottom of object. The tree's figure is stretched into the overlapping area on the right top of bird's-eye view iamge and exactly covered by frontal camera's image. By moving the right side's camera image on the top level, the tree's figure is show up(Figure 2.15 (b)).

The missing effects on (Figure 2.15 (c),(d),(e)) is same as previous. When the pedestrian is walk on the image boundary, the figure of pedestrian is missing in the bird's-eye view image, and will show up after car moving or changing the overlapping area's video source. No matter how we divide the display areas, the missing effect still exist. Also, average the overlapping areas will cause ghosting effect to confuse the driver. Here we propose a dynamic boundary idea to eliminate both missing and ghosting effects. If we can dynamically change the image boundary rather then fix the boundary, the missing effect will no longer misleading the driver, and there are no ghosting effect since we do not average the overlapping areas.

(Figure 2.16) is the variation of pixels difference in right top overlapping area, where is overlapped by frontal camera and right side camera. This serious frame record a pedestrian walking through, the blue and red line stand for the frontal and right side camera respectively. The image pixels variation is computed by suming the pixels difference between current frame and previous frame. More variation means the image content is changing more violent. This case usually happened when something is coming in front of the camera, and we use this property to decide which camera's image should be projected onto the overlapping area. Since one object approaching to a camera is relatively removed from another camera, the figure in differen camera's image will also relatively changed.

We use the variation to decide the overlapping area's video source, (Figure 2.17) illustrate the result. A pedestrian will walk through the overlapping area, where is overlapped by camera 1 (frontal camera) and camera 2 (right side camera). When frame number is 18(Figure 2.17 (b)), the pedestrian is in front of camera 1. When frame number is 18, the pedestrian is walking near the overlapping area, the variation of camera 1 is decreasing

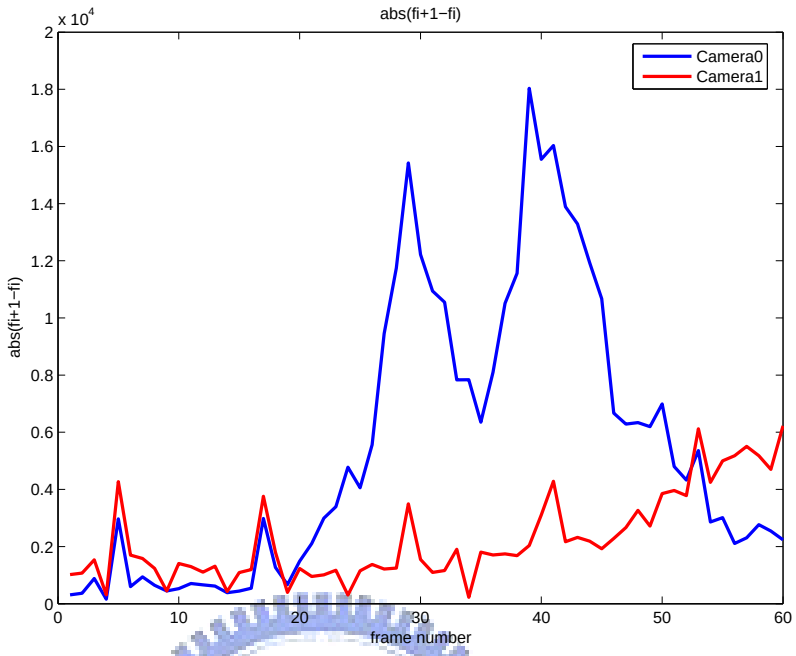


Figure 2.16: **Variation histogram of right top corner overlapping area.** This is the variation of sum of pixels difference between current frame and previous frame in the top left corner overlapping area. The blue and red line stand for the frontal camera and right side camera respectively. The horizontal axle is the frame number, and the vertical axle is the quantity of variation of sum of pixels difference between current frame and previous frame.

(a) Frame number 5 (b) Frame number 18 (c) Frame number 20

(d) Frame number 52 (e) Frame number 54

Figure 2.17: **A serious of frame image about the pedestrian walking through the top right overlapping area.** (a) frame number = 5. (b) frame number = 18. (c) frame number = 20. (d) frame number = 52. (e) frame number = 54.

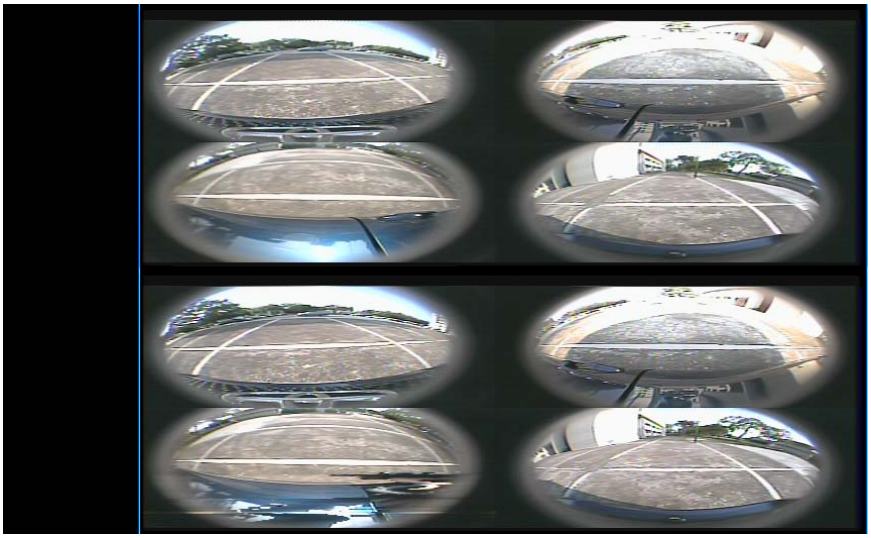


Figure 2.18: **NTSC frame format.** This picture is converted from a NTSC frame. The scan lines are seperated into even and odd fields. The horizontal and vertical synchronization signals are also binding in the frame (the blue line), and the frame data format is YCbCr. Two pixels combind into a micro pixel, each pixel has its own Y value and share the Cb and Cr with another pixel.

because the pedestrian is walking away. On the other hand, the variation of camera 2 is increasing because the pedestrian is coming. When frame number is 20(Figure 2.17 (c)), the variation of camera 2 is greater than camera 1, the video source of the right top overlapping area is changed to camera 2, and we can see the pedestrian's figure in the bird's-eye view image clearly. If we still use frontal camera's image, the pedestrian will missing in the bird's-eye view image.

2.3 Embedded system lookup table generation

In our application, we not only simulate the processing in PC, but also implement this work in an embedded system. We use Blackfin-561 as our embedded video processing system. In the embedded system, the frame information is itu-656 NTSC formate. Comparthed with PC image processing, the main differents is the scan lines are seperated into even and odd fields, and the color space is YCbCr format. In order to speed up the im-

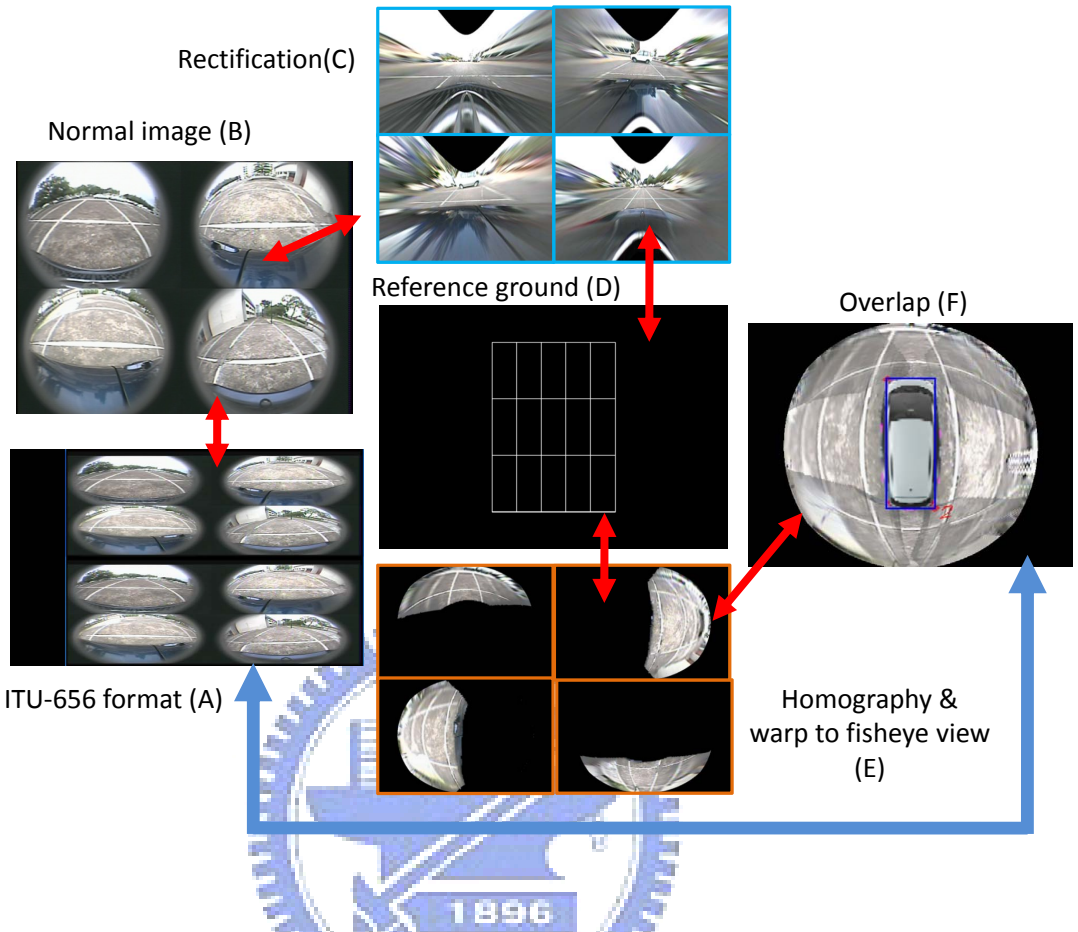


Figure 2.19: **Image transition.** This is the transition flowchart of the vehicle surrounding monitoring system. The red arrows represent the image processing flow in PC. The blue arrow stands for the transition of lookup table which is used in embedded system.

age processing process, we generate a lookup table for embedded system to find the pixel values of the bird's-eye view image. Therefore, the lookup table must be seperated into even field and odd field. Also, the pixel data format in the embedded system is different from PC site. Two pixels compose a micro pixel, each two pixels has their own Y value and share same Cb and Cr value. The frame data is saved in embedded memory just like (Figure 2.18).

One of our goals is to generate a lookup table for embedded system to speed up the image processing procedure.(Figure 2.19) illustrate the image processing flow of the vehicle surrounding monitoring system. The red arrow stands for the image transition flow in PC, and the blue line represent the transition relationship of the lookup table. At first,

the input image (Figure 2.19(A)) is exist in the memory of Blackfin-561 and saved as the ITU-656 format. We use developing tool from PC site to dump out the image data and convert to normal image (Figure 2.19(B)). Then, we rectify fisheye images in (Figure 2.19(B)) by applying the FOV distortion model, (Figure 2.19(C)) is the rectified result. By marking out the corresponding points between the calibration patten (Figure 2.19(D)) and each virtual perspective fisheye image in (Figure 2.19(C)), we obtain the homography matrixes. Furthermore, we also apply the FOV midel to warp the projection result (Figure 2.19(E)). Finally, we overlapping those warped projection results and put a vehicle figure in the middle, the bird's-eye view image is complete.

However, to obtain the lookup table for video processing in embedded system, the direction of transition flow must be inverted. In order to get each pixel value of the bird's-eye view image (Figure 2.19(E)), we first need to know which partition that the pixel belong to, and apply FOV model to find the related pixel in the virtual perspective image. With the location of our torget point in virtual perspective image, we use the homography matrix to back-project this point to (Figure 2.19(B)). Again, by using FOV model and 2D image transition, we can find the pixel value of bird's-eye view image point in original ITU-656 frame data. To reduce above transitions, we directly record the relations between (Figure 2.19(A) and (E)) in the lookup table, and the mapping relation just like the blue line. With this lookup table, the embedded system can directly obtain the pixel values of bird's-eye view image rather than compute fisheye image rectifications and image homography.

2.4 Summary of this chapter

This chapter introduce the system setup and configuration of our vehicle surrounding monitoring system, including fisheye cameras, camera position, quad spillter, and embedded system. Also describe the FOV model, fisheye image rectification, image homography and image stitching. We also mention about the missing and ghosting effect which caused by non-ground level objects, and we propose a dynamic boundary approach to eliminate those defects. Finally, we build a lookup table in PC site, which will speed the image processing procedures in embedded system.

Chapter 3

Embedded System and Implementation

In this chapter, we introduce the embedded system and the image processing procedures. First, that's have an overview of the ADSP Blackfin-561 multimedia processing embedded system, and also introduce the real time image processing procedures in blackfin-561. Secondly, we will apply a double buffer method to take advantages of memory hierarchical structure and reduce the data movement delay. Then, we implement the dynamic boundary idea into the surrounding monitoring system to enhance the visibility of our monitoring system. Finally, we introduce the implementation of our system.

3.1 ADSP Blackfin-561

In this section, we will introduce the features of blackfin-561 and understand the processing units in this embedded system. Also, we will introduce the developing environment and some important tool to help developer to implement their application. Moreover, we will describe the real time video processing procedures and related hardware units of our application.

3.1.1 Blackfin-561

The ADSP-BF561 processor [2] is a high performance member of the BlackfinR family of products targeting a variety of multimedia, industrial, and telecommunications applications. At the heart of this device are two independent Analog Devices Blackfin processors [1]. These Blackfin processors combine a dual-MAC state-of-the-art signal processing engine, the advantage of a clean, orthogonal RISC-like microprocessor instruction set, and single instruction, multiple data (SIMD) multimedia capabilities in a single instruction set architecture. The ADSP-BF561 processor has 328K bytes of on-chip memory. Each Blackfin core includes:

- 16K bytes of instruction SRAM/cache
- 16K bytes of instruction SRAM
- 32K bytes of data SRAM/cache

- 32K bytes of data SRAM
- 4K bytes of scratchpad SRAM

Additional on-chip memory peripherals include:

- 128K bytes of low latency on-chip L2 SRAM
- Four-channel internal memory DMA controller
- External memory controller with glueless support for
- SDRAM, mobile SDRAM, SRAM, and flash.

Here we also list features of ADSP blackfin-561:

- Dual symmetric 600 MHz high performance Blackfin cores 328K bytes of on-chip memory
- Each Blackfin core includes
  - Two 16-bit MACs, two 40-bit ALUs, four 8-bit video ALUs, 40-bit shifter
  - RISC-like register and instruction model for ease of programming and compiler-friendly support
  - Advanced debug, trace, and performance monitoring
- 0.8 V to 1.35 V core VDD with on-chip voltage regulator
- 2.5 V and 3.3 V compliant I/O
- 256-ball CSP-BGA (2 sizes) and 297-ball PBGA package options

Here is the peripherals of blackfin-561:

- Dual 12-channel DMA controllers (supporting 24 peripheral DMAs)
- 2 memory-to-memory DMAs
- 2 internal memory-to-memory DMAs and 1 internal memory DMA controller

- 12 general-purpose 32-bit timers/counters with PWM capability SPI-compatible port
- UART with support for IrDA
- Dual watchdog timers
- Dual 32-bit core timers
- 48 programmable flags (GPIO)
- On-chip phase-locked loop capable of 0.5 to 64 frequency multiplication
- 2 parallel input/output peripheral interface units supporting ITU-R 656 video and glueless interface to analog front end ADCs
- 2 dual channel, full duplex synchronous serial ports supporting eight stereo I2S channels

The block diagram of blackfin-561 is in (Figure 3.1). ADSP-BF561 system peripherals include the following:

- Parallel Peripheral Interfaces (PPIs)
- Serial Ports (SPORTs)
- Serial Peripheral Interface (SPI)
- General-Purpose Timers
- Universal Asynchronous Receiver Transmitter (UART)

All of the peripherals, except for general-purpose I/O and timers, are supported by a flexible DMA structure, including two DMA controllers (DMA1 and DMA2) and an internal memory DMA controller (IMDMA). In each of DMA1 and DMA2, there are twelve programmable peripheral DMA channels and two separate memory DMA streams dedicated to data transfers between the DSP's memory spaces, which include external SDRAM and asynchronous memory. Multiple on-chip buses provide enough bandwidth to keep the

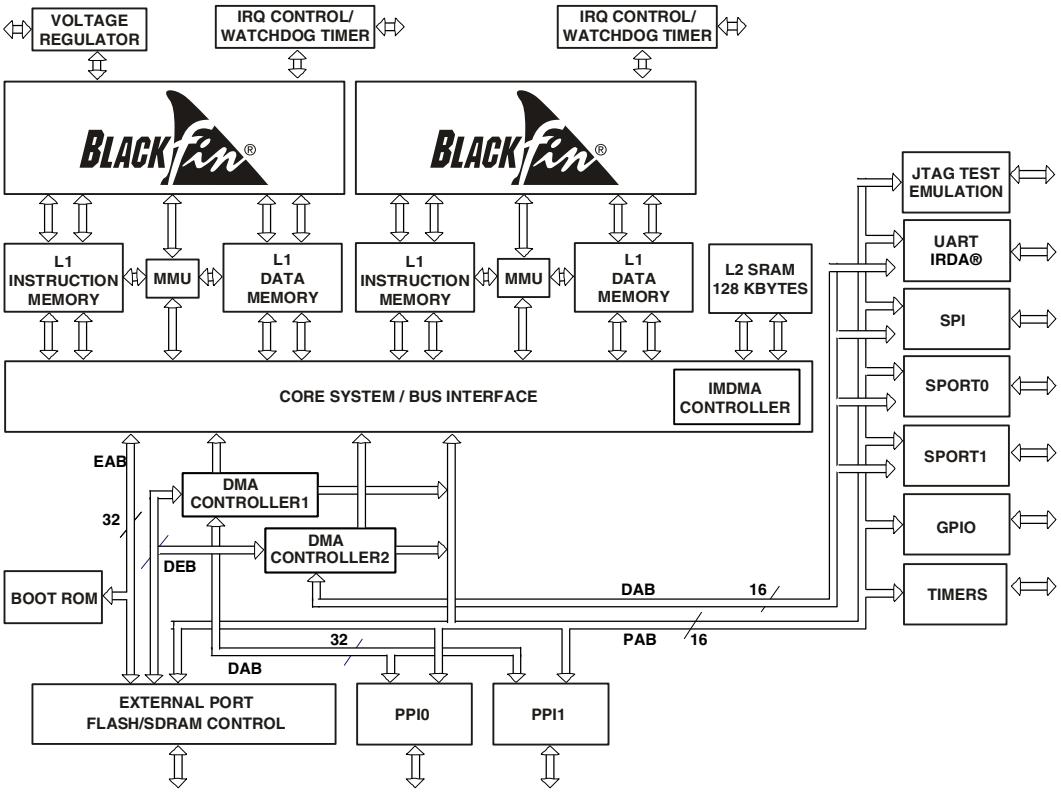


Figure 3.1: **ADSP-BF561 Block Diagram.**

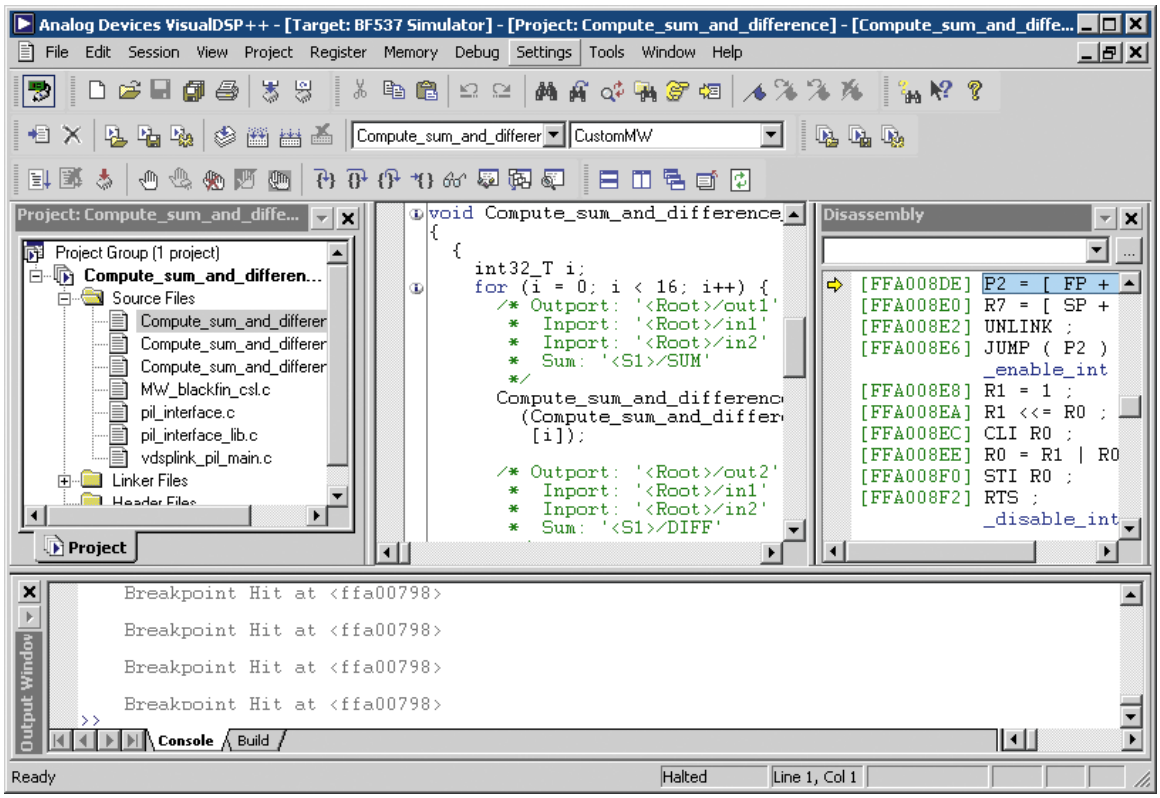


Figure 3.2: **VisualDsp++.** This is the IDE (Integrated Development Environment) of ADSP blackfin-561. The environment is like windows visual studio 2005, and there are a lot of developing tool for developers to debug or measure the performance of applications.

processor core running even when there is also activity on all of the on-chip and external peripherals.

(Figure 3.2) is the IDE (Integrated Development Environment) of ADSP blackfin-561. The programming language is similar to C, and the environment is similar to visual studio 2005. There are some useful tools that can help developers to debug, adjust, and improve the application performance. Furthermore, the instructions and implementation details can reference [1–3].

3.1.2 Real time image processing procedures in blackfin-561

The most important features of blackfin-561 is the dual Parallel Peripheral Interfaces (PPIs). With PPIs, blackfin-561 can receive and transmit video frame data simultaneously. Moreover, there are multiple DMAs can not only speed up the data movement procedures, but also cooperate with hierachical memory to improve data throughput.

(Figure 3.3) is the real time video processing flow chart of ADSP blackfin-561 embedded system. The video input signal is the output signal from the quad spillter. The signal is transmited by the coaxial cable, and the video siganl is ITU-656 NTSC format. NTSC format have 525 scanlines, and each scanline have 1716 bytes including vertical and horizontal synchorization signals, blanks, and image data. The image data are seperated into even and odd fields, each have 243 scanlines, and each scanline is 1440 bytes that contain 360 micro pixels. The image data is in YCbCr color space. Each micro pixel is composed by two pixels and each pixel has its own Y value and share the same Cb and Cr with other pixel. Y value is the illumination of a pixel, Cb and Cr are the color values.

The video input signals will be decoded by a ADV7183 decoder, the decoder will convert the received signals into frame data (the purple path in Figure 3.3). So the developer only have to config the registers to control the decoder and use a DMA to move the frame data into the SRAM (the orange path in Figure 3.3). After we move the whole frame data into the SRAM, the DSP core can access the frame data and generate the pixel values of bird's-eye image (the blue path in Figure 3.3).

As mentioned before, we use a table to record the pixel mapping information between the input frame data and the bird's-eye view result. The DSP core can fill the result image

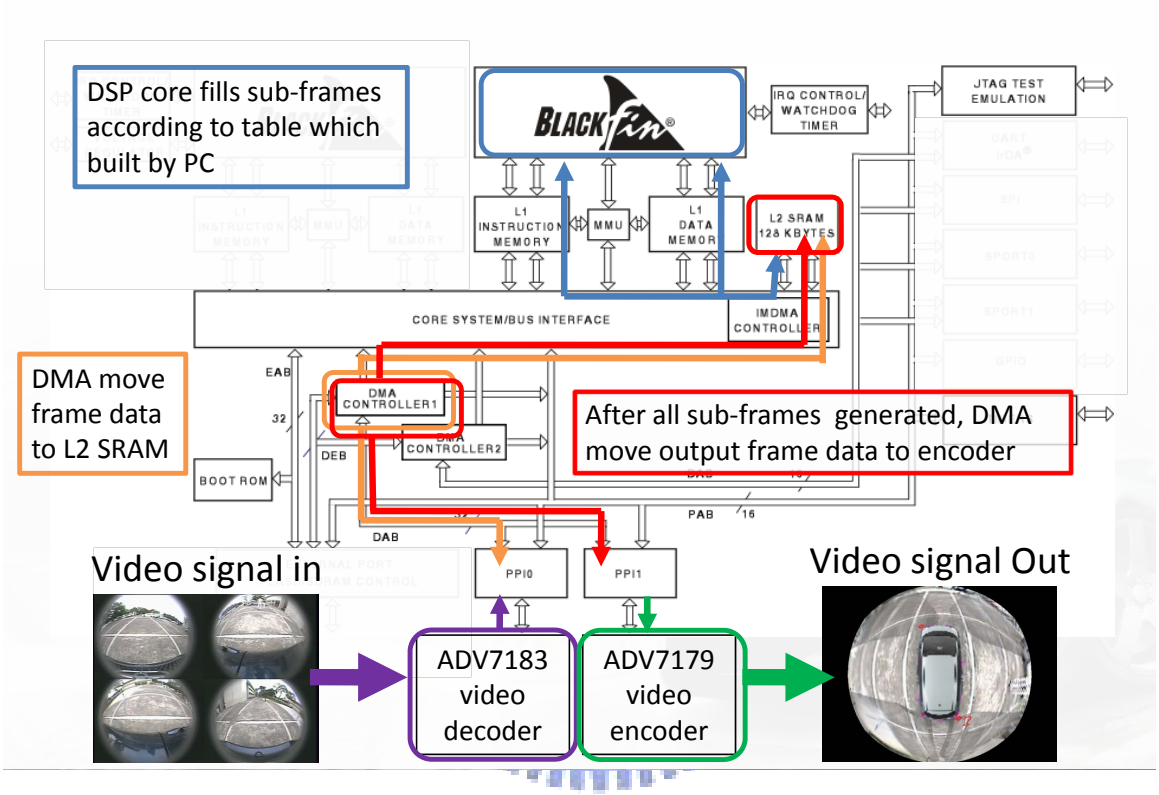
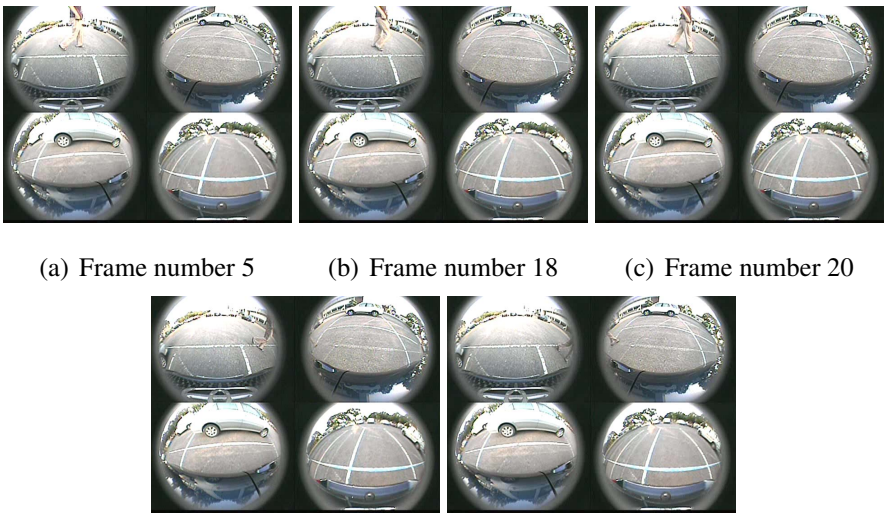


Figure 3.3: **BlackFin-561 video processing flow chart.** This flow chart illustrates the video processing sequence in blackfin-561. When receiving the input frame from the quad splitter, the decoder begin decoding video signals into itu-656 format frame data. Then, the DMA move the frame data into the SRAM through the PPIs(parallel peripheral interfaces). The DSP core generate the bird's-eye view image and saved in SRAM. After the whole bird's-eye view image has been generated, another DMA will move the result to the encoder. Then, the bird's-eye view image frame immediately show on screen.

according the lookup table. The work of DSP core can be divide into two parts, one is looking up the mapping table to get the pixel values from input frame image, and other is to save the generated result into the output frame buffer in SRAM. Because the DSP core need more clock cycles to access data in SRAM, we can utilize the on-chip cache to reduce the access time. On the other hand, the search range of pixel mapping is the whole frame image, and the searched image data is less spatial and temporal locality. It is necessary for DSP core to directly access the input frame in L2 SRAM. But the output result of bird's-eye view image has sequential and have the spatial locality. Rather than directly save result data in L2 SRAM, we use two small buffer in L1 cache to contain partial results of the bird's-eye view image. The data moving from L1 cache to L2 SRAM can be dealed by memory DMA. Since DSP core accessing cache faster than SRAM, the clock cycles can be reduced. The data moving and data processing can simultaneously be done by DMA and DSP core, respectively.

Due to the limit of cache size, each buffer can only contain a part of the whole frame. Therefore, we divid a frame into 36 sub-frames, which means each buffer has 120x81 pixels (since the whole frmae has 720x480 pixels). On the other hand, if there is only one cache buffer, the buffer is available after the lookup table result has been moved to SRAM. To reduce the data transiting time, we apply the pipeline mechanism [15] to overlap the time of filling buffer and moving data. The pipeline mechanism is using two buffers at the same time. Assume there are buffer 0 and buffer 1. At first, one buffer is containing partial result of bird's-eye view image, and another buffer is idling. After buffer 0 is full, the core keep looking up table and filling the result in buffer 1. The data in buffer 0 is moved by a MDMA (memory DMA) simultaneously. Because the data moving is a sequential task, but filling buffer according to the table is not. So filling buffer take more cycles than data moving. We use the time of filling up buffer to switch two buffers, and the processing time and moving time can be overlapped. Moreover, we can also apply the pipeline mechanism in the frame receiving. Since the DMA (for frame input) is idling while the DSP core is processing. We can also use two buffers for receiving data from decoder and for DSP core accessing. The processing time and frame receiving time can be reduced. This pipeline mechanism greatly enhence the data throughput and increase the processing speed.

After the whole frame of bird's-eye view image is generated, the frame data must moved



(a) Frame number 5 (b) Frame number 18 (c) Frame number 20

(d) Frame number 52 (e) Frame number 54

Figure 3.4: **Frame images when a pedestrian is walking through the right top corner of vehicle.**

to encoder to encode and output (the green path in Figure 3.3). Here is another DMA copy the frame data from SRAM to the ADV7179 encoder through PPI (the red path in Figure 3.3). The embedded system keep doing the same work for next incoming frame and also output previous result simultaneously.

3.2 Implementation of dynamic boundary

This section describe the implementation of dynamic boundary mechanism. Since the fixed image boundary will cause object missing (Figure 2.17 (a)(d)), we propose this idea to change the image source of overlapping area, and also move the image boundary.

For example, the pedestrian belong to the non-ground level object. When a pedestrian is walking around the right top corner of a vehicle, the camera of frontal and right side record the pedestrian's figures. (Figure 3.4). We use the sum of difference quantity to trigger the change of image source in the right top overlapping area (Figure 3.5).

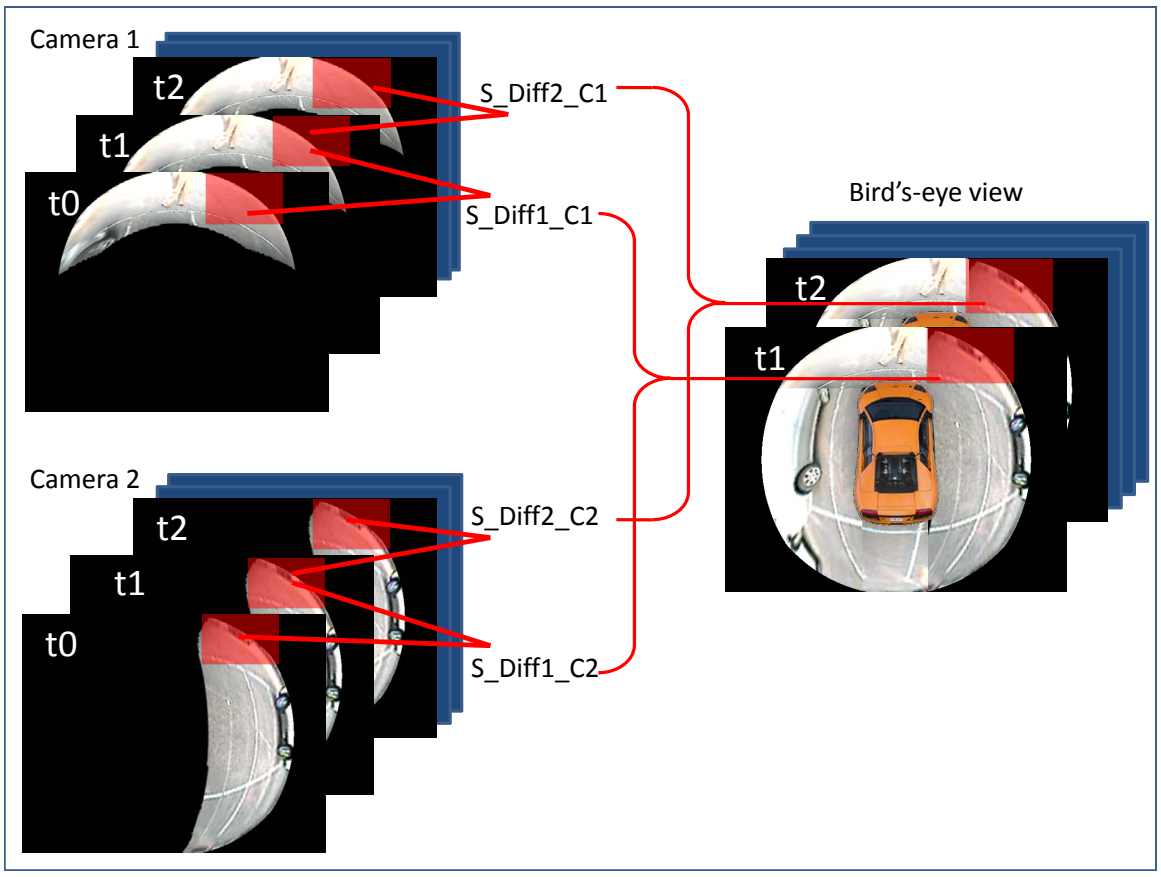
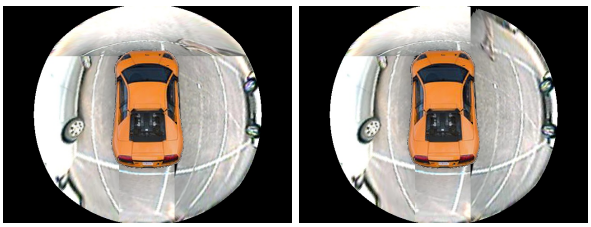


Figure 3.5: **Sum of difference quantity in overlapping area.** This is the flow chart for switching the image source of overlapping area.

(a) Frame number 5 (b) Frame number 18 (c) Frame number 20



(d) Frame number 52 (e) Frame number 54

Figure 3.6: **Bird's-eye view image when a pedestrian is walking through the right top corner of vehicle.**

$$S(n,t) = \sum_{i} \sum_{j} \left| P(t)_(i,j) - P(t-1)_(i,j) \right| \tag{3.1}$$

Where n is the camera number, t is the frame sequence. $P(t)_(i,j)$ is the pixel value of camera number n at frame number t, i and j is the coordinate index of pixels in overlappint area. $S(n,t)$ is the sum of difference between frame t and frame $t-1$ of overlapping area of camera n. Then we use $S(n,t)$ to compare the vatiation of different image source of the overlapping area. (Figure 2.16) is the variation of sum of difference when the pedestrian is walking around the right top corner.

(Figure 3.6) is the corresponding bird's-eye view results of (Figure 3.4).

In order to compare current frame and previous frame, we use two buffers to contain current frame and previous frame. After current frame have been processing, those two buffers switched. On the other hand, there are four overlapping areas need to compute. Using the whole overlapping area's data to compute the difference quantity is a time consuming task. Since the difference of neighbor pixels is small, we can down sampling the overlapping areas to compute the sum of difference quantity and also reduce the buffer size.

On the other hand, the camera image may contain a lot of noises, if we directly use the differences summation to trigger the switching, the switching result may be influenced by noises, and the change frequence may become too fast. In order to keep the stability of image boundary, we use a counter to counting the results of comparing difference quantity, and also define a threshold about the condition of changing image source. Comparing the summation of difference quantity can be treated as the competition of image source. The image source will supply by the side with more difference quantity. In other word, the way win more times of the competitions, the way dominate the image source of the overlapping area.

Assume there are A and B two image sources and the overlapping area is dominated by A. The initial of the counter is equal to the threshold value. If source A win the competition, the counter will increase by one and can not greater than the threshold. But if source B win the competition, the counter is dreased by one. If the counter is equal to zero, the image source will switch and the counter will initial to the threshold value. While receiving a frame, there is a new run of competition. If B can not win the competition more times than the threshold value, the image source is still dominate by A, but if the number of times that B win the competitions is greater than the threshold value, the image source will change to B and the counter also initial to the threshold value. With this competition and the threshold, the image source will not influenced by noises easily. However, the threshold value become the trade off between stability and sensitivity, and we set five to the threshold value in our experiment.

3.3 Implementation of surrounding monitoring system

As previous mentioned, we divide a frame into 36 sub-frames. Because the display areas are not fixed, the pixel mapping relations is changed according to the comparing results of 3.2. Furthermore, the location of buffer will change, so we have to record those information to enhence the processing speed of embedded system. We use a table to record the information, and use the buffer number and partition number as the index of table. The buffer and the bird's-eye view image partition can be treat as two rectangles and have their

own position in the frame. If they have intersection, the DSP core need to fill the buffer with the image data of the intersection. Those information can be computed previously by PC, which including:

- Have intersection?
- X and Y coordinate of the left top corner of the intersection
- X and Y coordinate shift for buffer
- The width and height of the intersection
- X and Y index of the lookup table
- X and Y coordinate shift for table

During the buffer filling procedure, the DSP core need to check whether the buffer and bird's-eye view image partition have intersection? If yes, the intersection in the buffer will be filled with the result according to the lookup table.

3.4 Summary of this chapter

This chapter introduce the embedded system we used and the the implementation details of real time video processing in embedded system. The features of ADSP blackfin-561, and the developing environment of embedded system are also mentained. We also reveal the key points of real time video processing in blackfin-561 and apply the pipeline mechanism to reduce the redundant time of data moving. We also describe the implementation of dynamic boundary, which can dynamically decide the image source of overlapping areas with stability. Finally, we describe the procedures of filling the cache buffer and list out important informations of lookup table that can be obtained previously in PC site.

Chapter 4

Experimental Results

In this chapter, we demonstrate the performance of proposed method and system. We implement the proposed method for homography matrixes calibration and display areas seperation of bird's-eye view image. The procedures are implemented in C/C++ language, compiled by Microsoft Visual Studio 2005 compiler and executable on WindowsXP platform. On the other hand, the application of embedded system is implemented and compiled by Visual-DSP++ 5.0. The executable file will be loaded into the ADSP BlackFin-561 platform to execute.

In first section, we demonstrate the results of fisheye image rectification and homography projection. Then, we generate a bird's-eye view image according to the rectification and projection results, and we use this bird's-eye view image to divide the display areas. Once the display areas are decided, the mapping relationship will be recorded in a mapping table. In the second section, we use a real-time timer to measure the performance of our system. Moreover, we also demonstrate the functionalities of our surrounding monitoring system in different situations.

4.1 Results of Image rectification and 2D image projection

After we mount four fisheye cameras on the vehicle, those four fisheye cameras' video signals will be collected by a quad spillter. As shown on (Figure 4.1) each channel will be scaled and arranged. The left top image belongs to frontal camera, the right top image belongs to right side camera, the left bottom image belongs to left side camera, and the right bottom image belongs to the rear camera.

As mentained before, we park the vehicle in the parking lot, and use lines of parking spaces to calibrate the homography matrixes. But lines in these four fisheye images are distorted by fisheye lens, so we apply FOV model to rectify these fisheye image. We first clip each cameras' image from (Figure 4.1), and resize to 640x480 (The resolution of clipped image is 320x240.). (Figure 4.2 (a)(b)(c)(d)) is the resized result. After getting each cameras' image, we apply the FOV model to rectify (Figure 4.2 (a)(b)(c)(d)) to (Figure 4.3 (a)(b)(c)(d)).

The lines of parking space are corrected, then we can use rectified results to calibrate

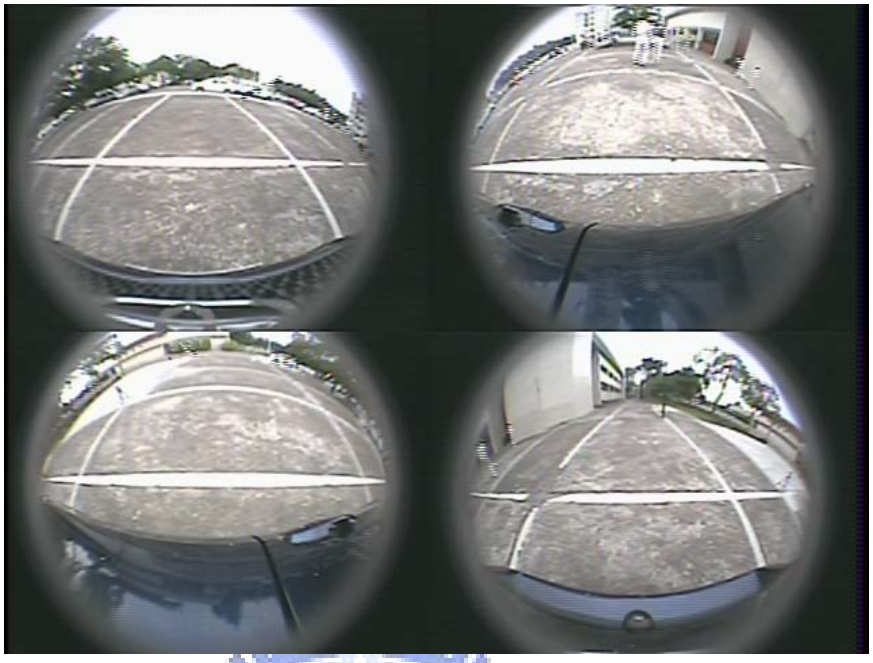
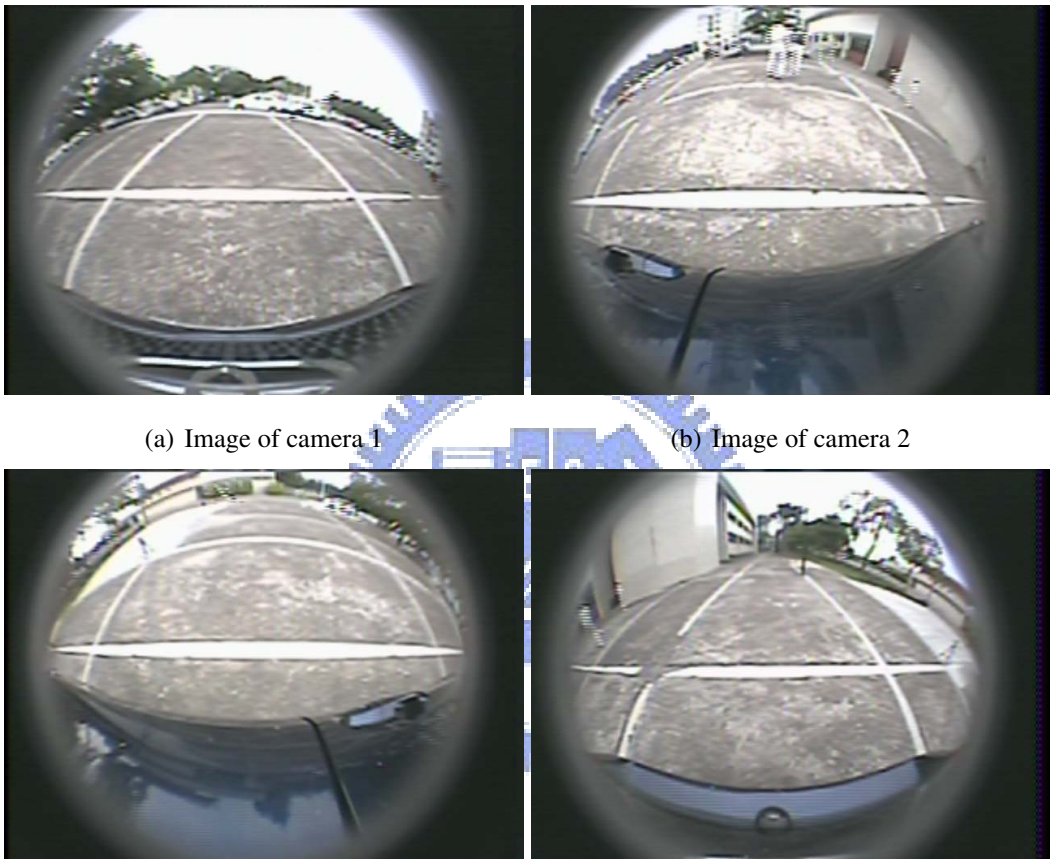


Figure 4.1: **Output image of the quad spillter.** This is a output image the of quad spillter, four fisheye cameras' images will be scaled and arranged by the quad spillter.

homography matrixes. (Figure 4.4), (Figure 4.5), (Figure 4.6), (Figure 4.7) are the corresponding points in four rectified bird's-eye view images and the calibration pattern. By marking out those corresponding points, we can estimate the homography matrixes. (Table 4.1), (Table 4.2), (Table 4.3), (Table 4.4) list out the X and Y index values of corresponding points and the homography matrixes that can project points in calibration pattern to rectified fisheye image.

The lines of parking space are corrected, then we can use rectified results to calibrate homography matrixes.

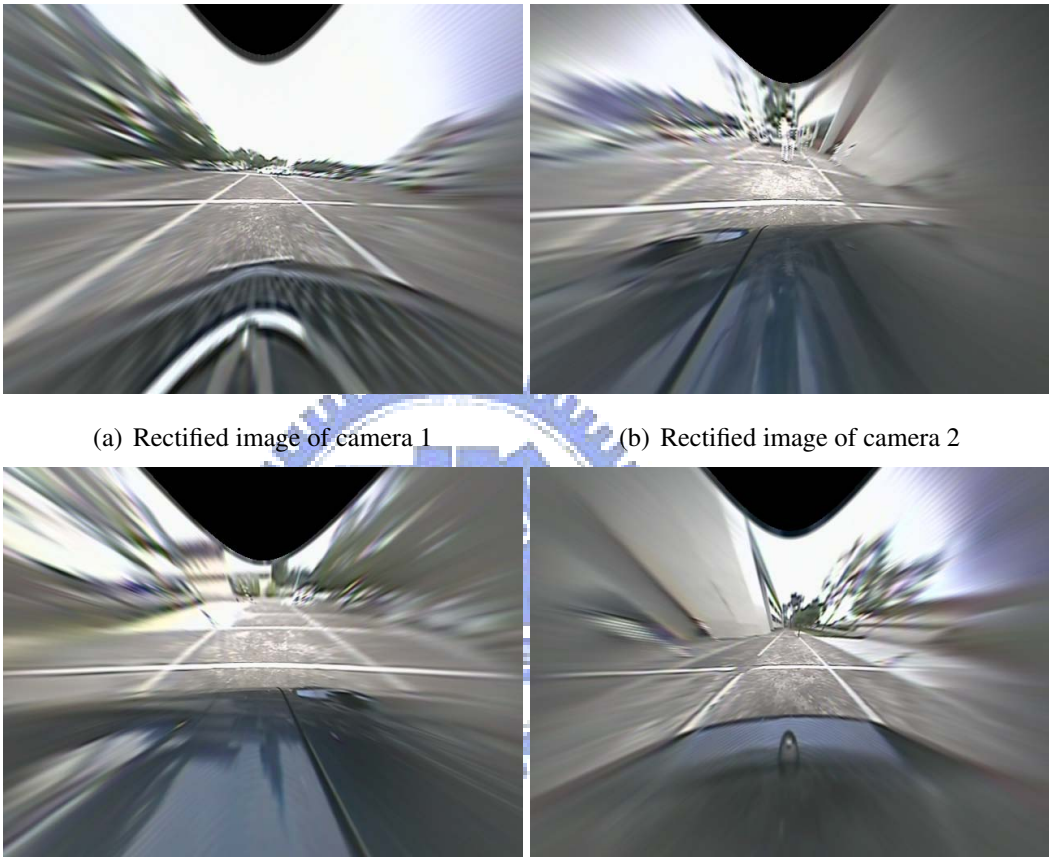
After finishing homography matrixes estimation, we project four rectified fisheye images onto the calibration pattern. The homography matrix can translate the point index of calibration pattern into the point index of a rectified fisheye image. We can find the correspoinding pixel value in a rectified fisheye image. (Figure 4.8) is the projection results of four cameras' images. These cameras' images are all projected to the same calibration pattern, and we can align them simply by overlapping. The black parts in the projection



(a) Image of camera 1 (b) Image of camera 2

(c) Image of camera 3 (d) Image of camera 4

Figure 4.2: **Fisheye images.** (a) Camera 1 for front view. (b) Camera 2 for the view of right side. (c) Camera 3 for the view of left side. (d) Camera 4 for rear view.



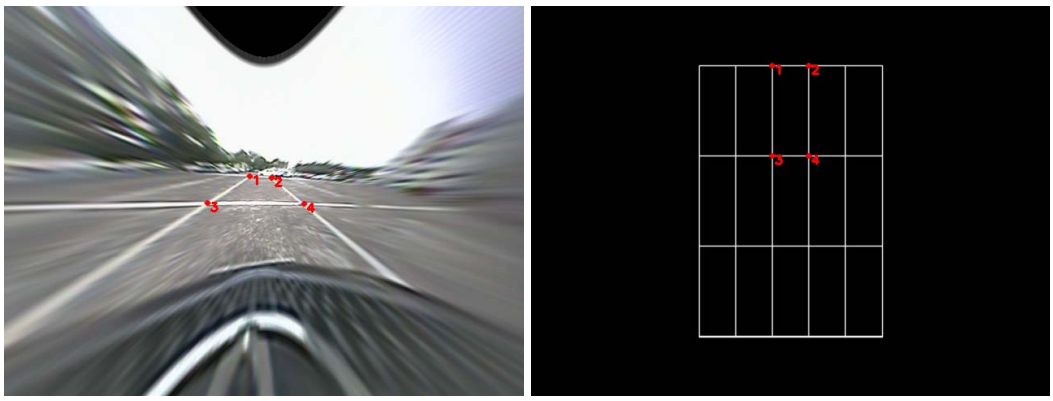
(a) Rectified image of camera 1

(b) Rectified image of camera 2

(c) Rectified image of camera 3

(d) Rectified image of camera 4

Figure 4.3: **Rectified fisheye images.** (a) Rectified image of camera 1 for front view. (b) Rectified image of camera 2 for the view of right side. (c) Rectified image of camera 3 for the view of left side. (d) Rectified image of camera 4 for rear view.



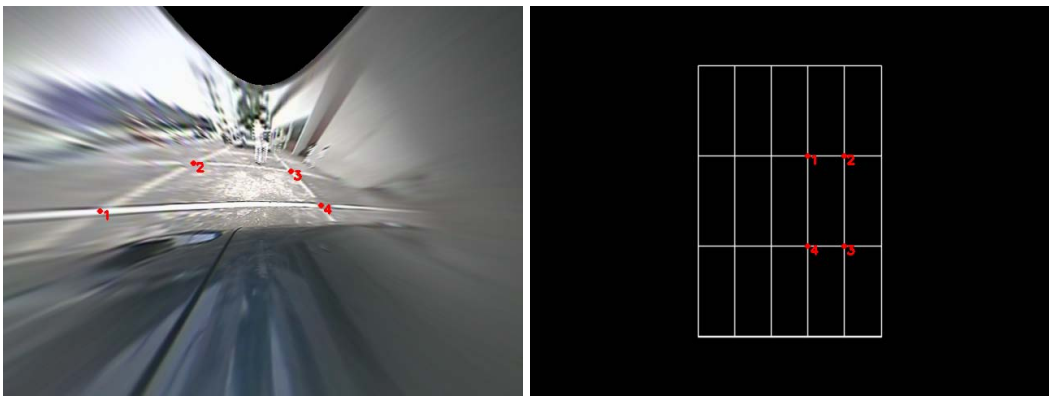
(a) Rectified image of camera 1 with four corresponding points.

(b) Calibration pattern with four corresponding points.

Figure 4.4: **Corresponding points** (a) Four corresponding points stand for four corners of a parking space in front of the vehicle. (b) The corresponding points in the calibration pattern.

Table 4.1: **Corresponding points of Camera 1**

Camera 1	Rectified fisheye image		Calibration pattern	
	X	Y	X	Y
pair 1	302	209	297	73
pair 2	329	211	342	73
pair 3	250	242	297	184
pair 4	369	243	342	184
H_1	$\begin{bmatrix} 0.876804 & -1.95625 & 144.722 \\ 0.266681 & -1.2415 & 192.97 \\ 0.00108092 & -0.00619728 & 1 \end{bmatrix}$			



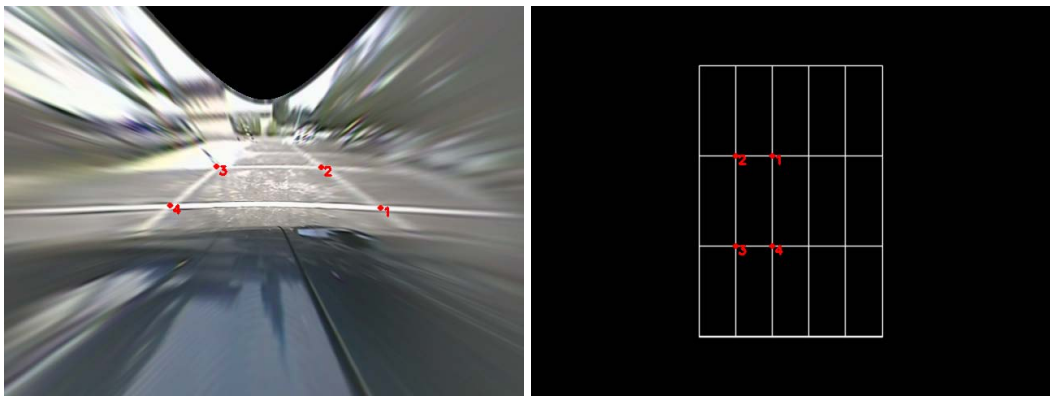
(a) Rectified image of camera 2 with four corresponding points.

(b) Calibration pattern with four corresponding points.

Figure 4.5: **Corresponding points** (a) Four corresponding points stand for four corners of the parking space on the right side of the vehicle. (b) The corresponding points in calibration pattern.

Table 4.2: **Corresponding points of Camera 2**

Camera 2	Rectified fisheye image		Calibration pattern	
	X	Y	X	Y
pair 1	119	252	342	184
pair 2	234	193	387	184
pair 3	354	203	387	295
pair 4	391	245	342	295
H_2	$\begin{bmatrix} -0.887621 & -0.439827 & 374.303 \\ -0.439301 & -0.138697 & 154.18 \\ -0.00285795 & -0.000588154 & 1 \end{bmatrix}$			



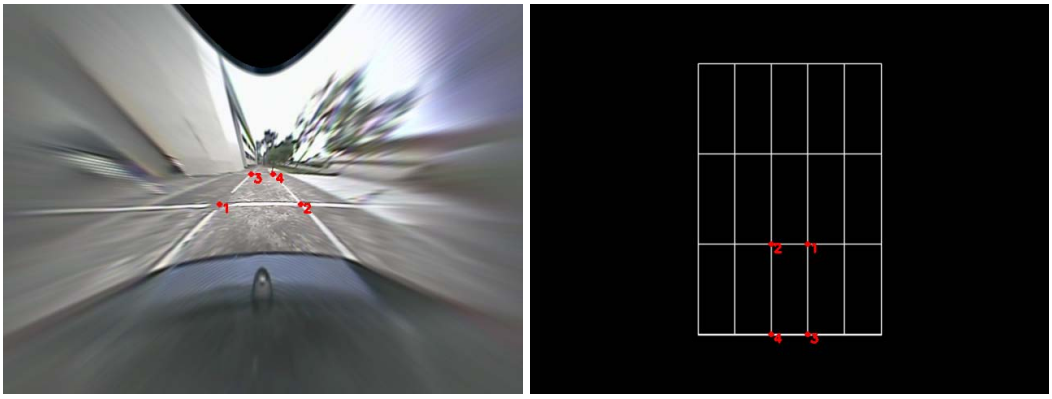
(a) Rectified image of camera 3 with four corresponding points.

(b) Calibration pattern with four corresponding points.

Figure 4.6: **Corresponding points** (a) Four corresponding points stand for four corners of the parking space on the left side of the vehicle. (b) The corresponding points in calibration pattern.

Table 4.3: **Corresponding points of Camera 3**

Camera 3	Rectified fisheye image		Calibration pattern	
	X	Y	X	Y
pair 1	463	248	297	184
pair 2	390	198	252	184
pair 3	261	197	252	295
pair 4	204	245	297	295
H_3	$\begin{bmatrix} -0.9365791 & -0.299037 & 393.497 \\ -0.438042 & 0.00236863 & 161.967 \\ -0.00294331 & 2.40374e-005 & 1 \end{bmatrix}$			



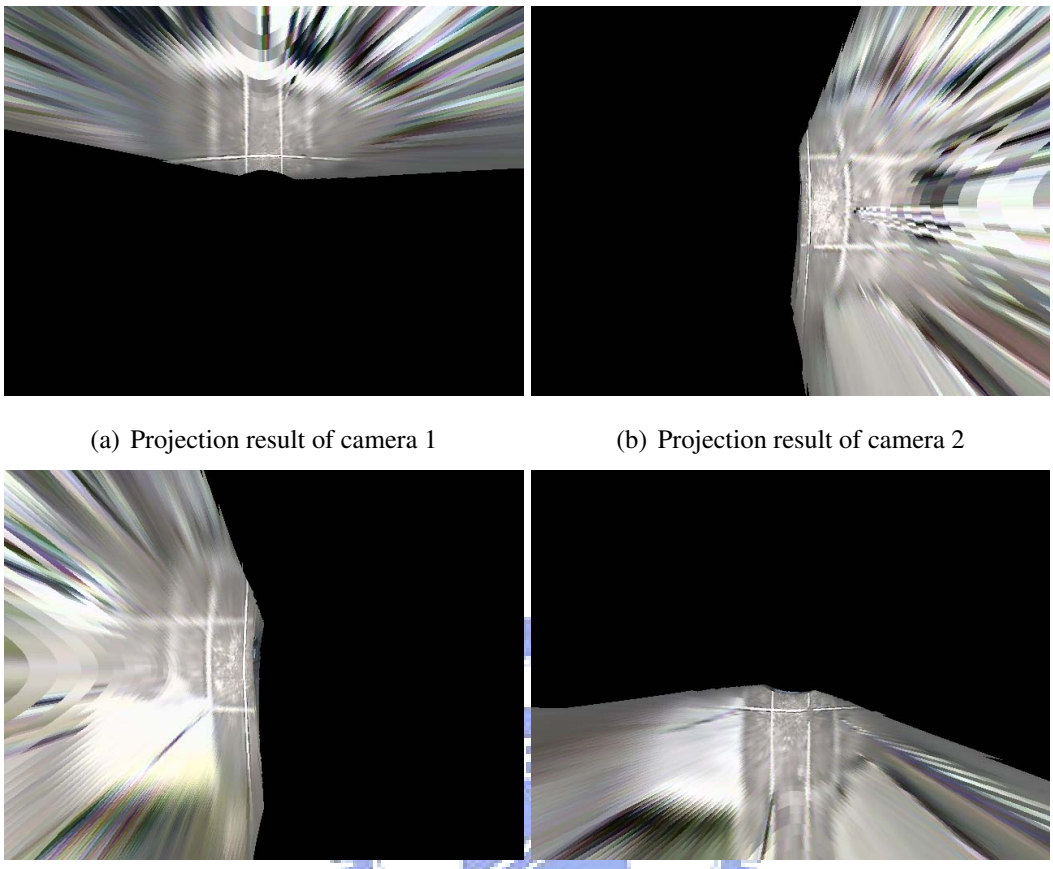
(a) Rectified image of camera 4 with four corresponding points.

(b) Calibration pattern with four corresponding points.

Figure 4.7: **Corresponding points** (a) Four corresponding points stand for four corners of the parking space in rear of the vehicle. (b) The corresponding points in calibration pattern.

Table 4.4: **Corresponding points of Camera 4**

Camera 4	Rectified fisheye image		Calibration pattern	
	X	Y	X	Y
pair 1	266	246	342	295
pair 2	366	246	297	295
pair 3	305	209	342	406
pair 4	332	209	297	406
H_4	$\begin{bmatrix} 0.359262 & -1.25785 & 205.194 \\ -8.71716e-011 & -0.769123 & 187.121 \\ -5.37521e-013 & -0.00393786 & 1 \end{bmatrix}$			



(a) Projection result of camera 1

(b) Projection result of camera 2

(c) Projection result of camera 3

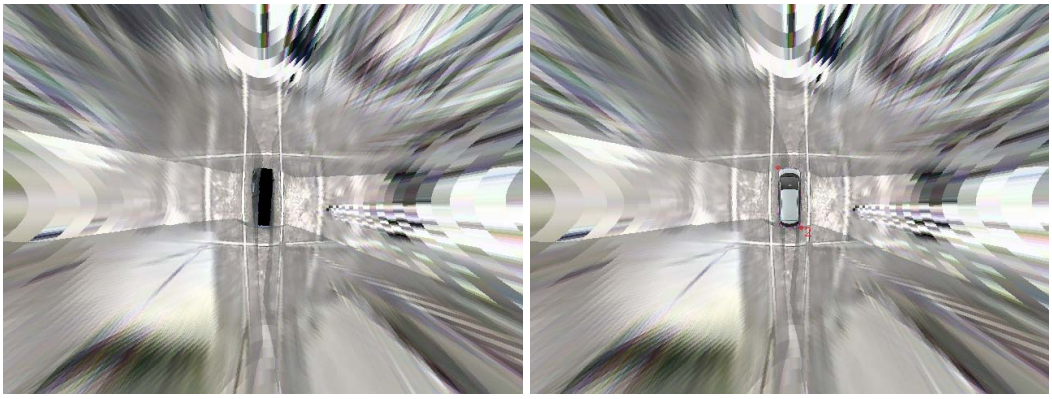
(d) Projection result of camera 4

Figure 4.8: **Projection results.** (a) Projection result of camera 1. (b) Projection result of camera 2. (c) Projection result of camera 3. (d) Projection result of camera 4.

results means there are no image contents in those area. While we overlapping those projection results, we can ignore those black parts by giving a zero weighting while blending. (Figure 4.9 (b)) is the result of overlapping.

We decide the location and size of vehicle's figure before warping. The blank part in the middle of (Figure 4.9 (a)) is the body of our vehicle. We use corners in left top and right bottom to represent the location and size of vehicle's figure (Figure 4.9 (b)). In next step, we apply FOV model to warp those projection results (Figure 4.10(a)(b)(c)(d)).

After warping, the left top and right bottom corner of vehicle figure (Red points in (Figure 4.9 (b)) also transit away principle center. These two warped points will be used



(a) Result without the vehicle figure (b) Result with the vehicle figure

Figure 4.9: **Overlapping results.** Overlapping for camera images' projection results. (b) Overlapping result with the figure of a vehicle.

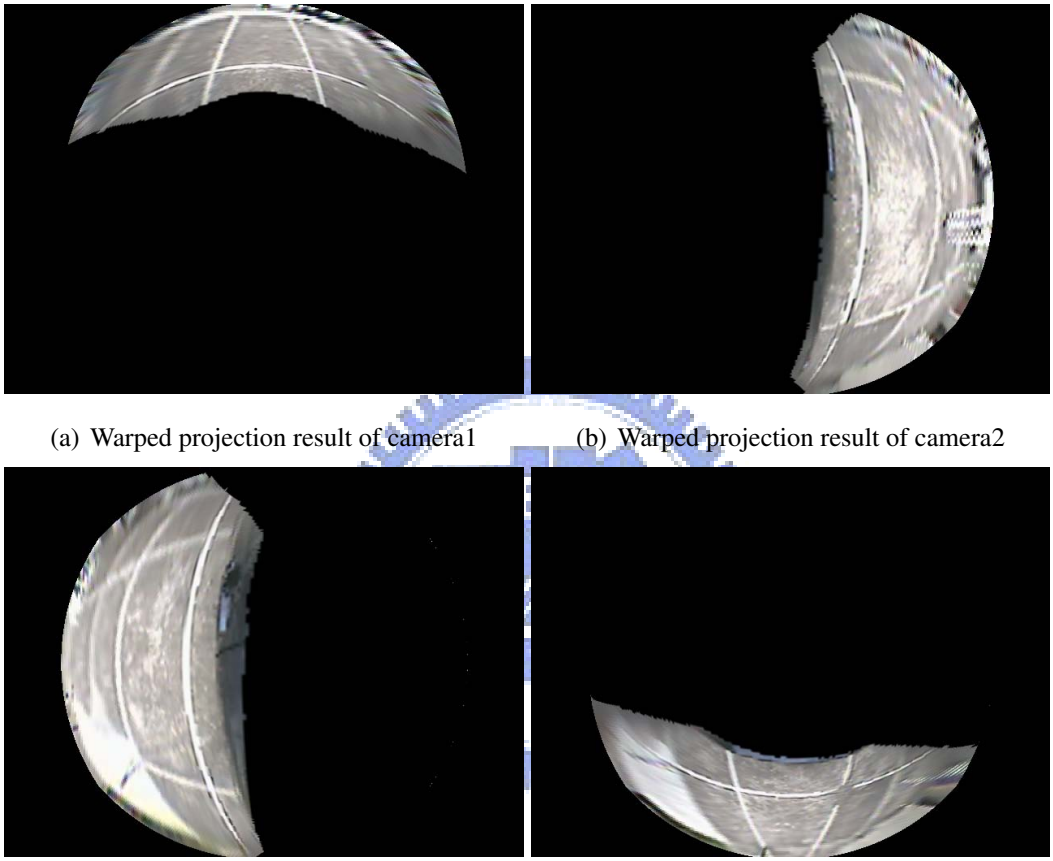
to divide the display areas. The projection image of each camera will be seperated into three parts (Figure 4.11 (a)(b)(c)(d)). The first and third parts is in overlapping areas, and the middle parts are not. By overlapping these partition results, the bird's-eye view image are seperated into total nine areas (Figure 5.12), the middle part is vihicle's figure. Images of those middle parts on each side are direct projected by each side's camera. The four corners are the overlapping areas, where the image sources are supported by neighbor two cameras.

4.2 Results of surrounding monitoring system and system performance

We demonstrate some sample views of vehicle surrounding monitoring system in different situations and measure the performance of our system.

4.2.1 Reverse Parking

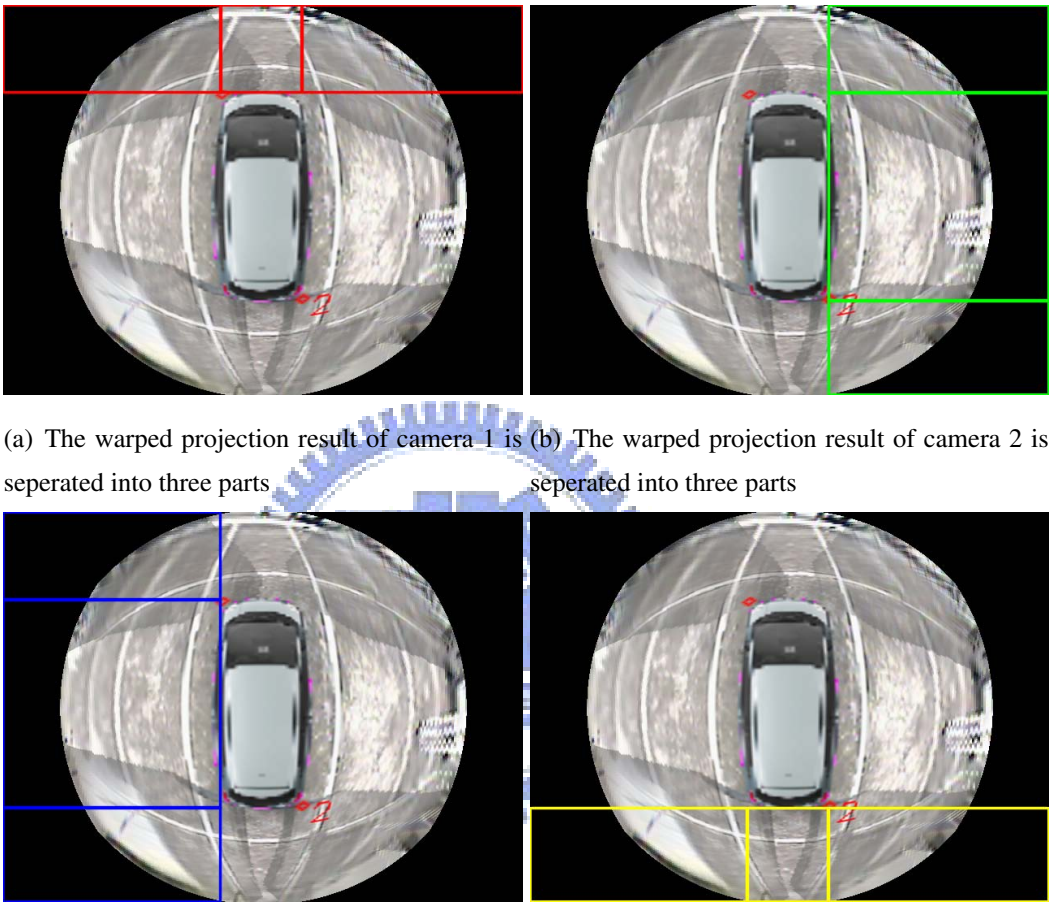
In first case, we demonstrate a reverse parking scene (Figure 4.13 (a)). While the driver sitting in the driving seat, the visibility is restricted (Figure 4.13 (b)). It is difficult for the



(a) Warped projection result of camera1 (b) Warped projection result of camera2

(c) Warped projection result of camera3 (d) Warped projection result of camera4

Figure 4.10: **Warped projection results.** (a) Warped projection result of camera 1. (b) Warped projection result of camera 2. (c) Warped projection result of camera 3. (d) Warped projection result of camera 4.



(a) The warped projection result of camera 1 is seperated into three parts

(b) The warped projection result of camera 2 is seperated into three parts

(c) The warped projection result of camera 3 is seperated into three parts

(d) The warped projection result of camera 4 is seperated into three parts

Figure 4.11: **Partitions of each cameras' warped results.** (a) Partitions of frontal image. (b) Partitions of right side's image. (c) Partitions of left side's image. (d) Partitions of rear image.

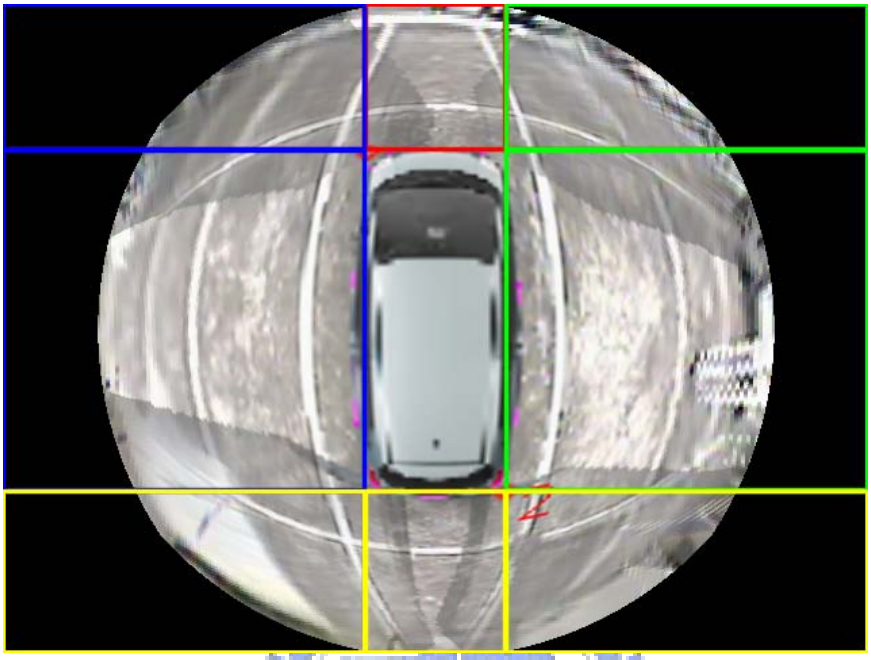
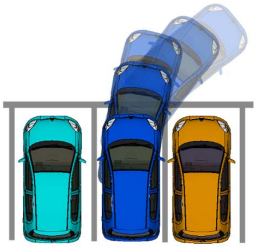


Figure 4.12: **Partitions of bird's-eye view image.** The bird's-eye view image is seperated into nine parts by left top and right bottom corners of the warped vehicle figure.

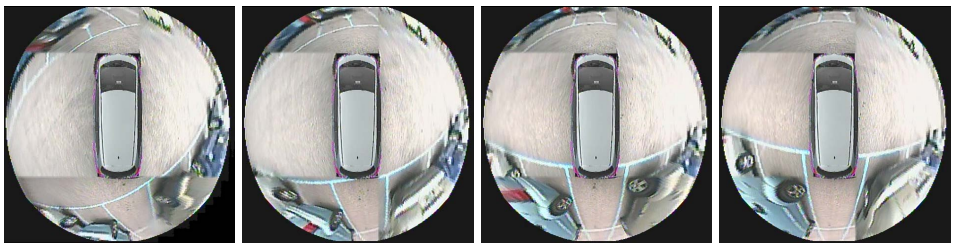


(a) Reverse parking

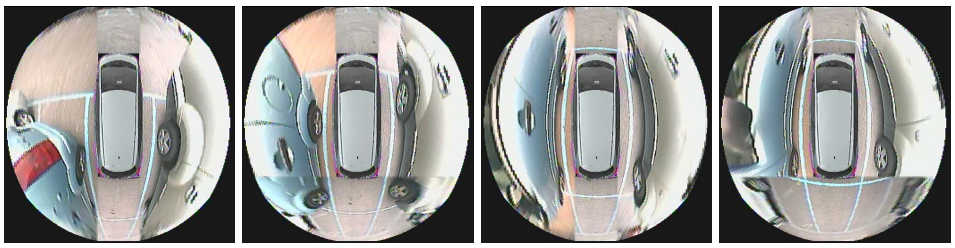
(b) The real scene of reverse parking

(c) Resticted visibility of driver

Figure 4.13: **Reverse parking.** (a) A chart of reverse parking. (b) A real scene of driver's view. (c) Resticted visibility of driver.



(a) Reverse parking 1 (b) Reverse parking 2 (c) Reverse parking 3 (d) Reverse parking 4



(e) Reverse parking 5 (f) Reverse parking 6 (g) Reverse parking 7 (h) Reverse parking 8

Figure 4.14: **Frame serious of reverse parking.**

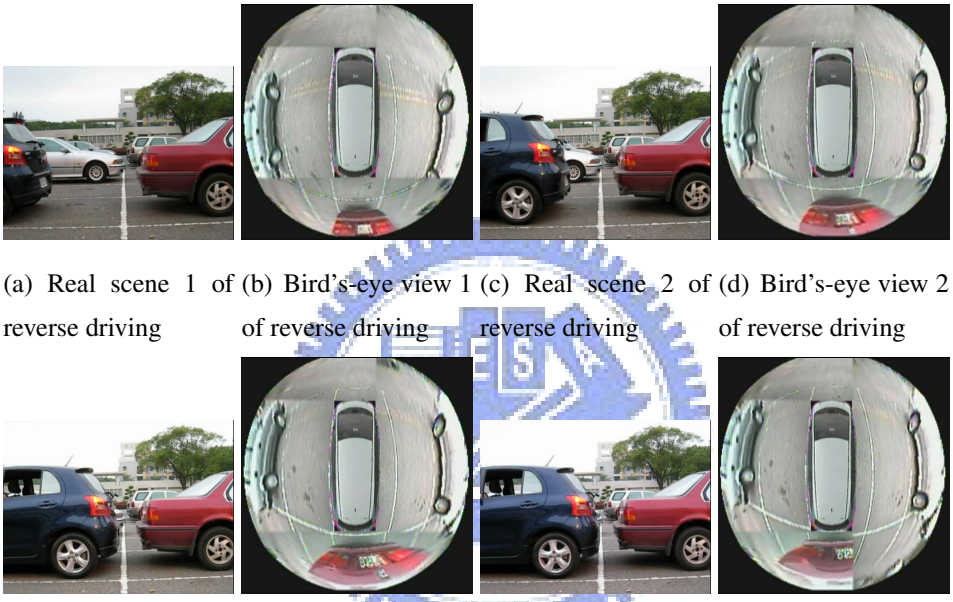
driver to estimate the distance between vehicle and other cars. But driver can reference the bird's-eye view image through the vehicle surrounding monitoring system, and can clearly see another car as well as lines of parking space (Figure 4.14).

4.2.2 Reverse driving

The second case is reverse driving. The visibility of driver is same as reverse parking. We can compare the real scene images (Figure 4.15 (a)(c)(e)(g)) with the bird's-eye view images (Figure 4.15 (b)(d)(f)(h)).

4.2.3 Driving in the narrow alley

When driving through a narrow alley, the driver's eyesight is restricted by buildings or walls on both sides. Bird's-eye view image is useful in this situation. It is easily to see the buildings and walls clearly and can avoid collision. Moreover, our system can automatically change image to see obstacles in the intersection of the alley. Not only eliminate blind spots, but also give driver more time to response for pedestrians or cars from left or right



(a) Real scene 1 of reverse driving (b) Bird's-eye view 1 of reverse driving (c) Real scene 2 of reverse driving (d) Bird's-eye view 2 of reverse driving

(e) Real scene 3 of reverse driving (f) Bird's-eye view 3 of reverse driving (g) Real scene 4 of reverse driving (h) Bird's-eye view 4 of reverse driving

Figure 4.15: **Frame serious of reverse driving.**

(a) Intersection 1

(b) Bird's-eye view image 1 of intersection in alley

(c) Intersection 2

(d) Bird's-eye view image 2 of intersection in alley

(e) Intersection 3

(f) Bird's-eye view image 3 of intersection in alley

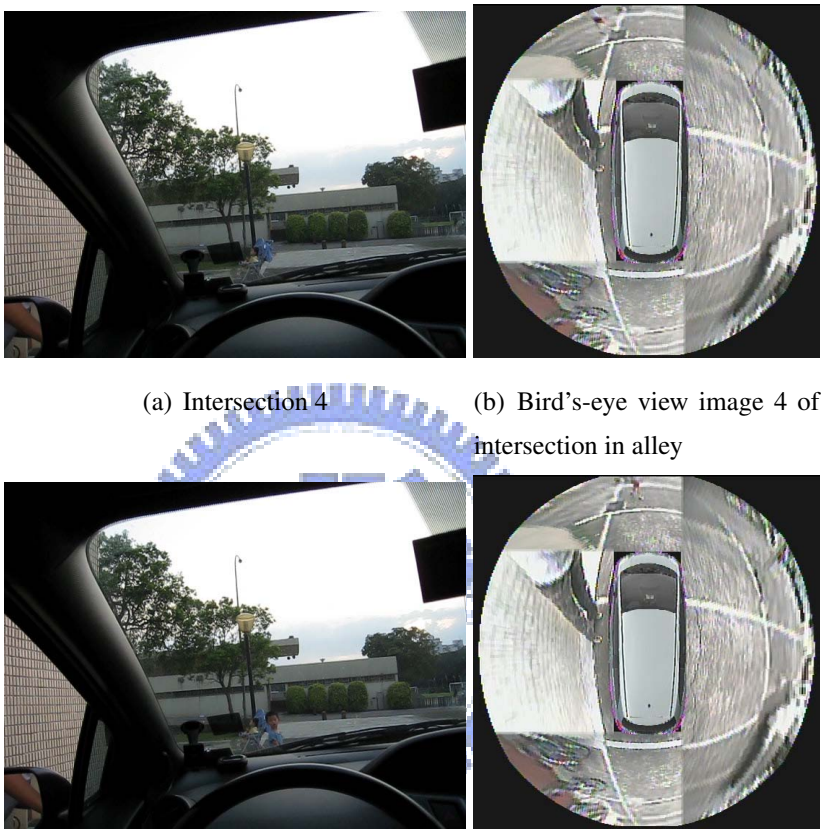
Figure 4.16: **Frame serious of driving through the intersection in an alley.** (a) There is a pedestrian behind the building, but can not be seen by the driver. (b) The bird's-eye view image shows the figure of the pedestrian. (c) Two seconds later, the pedestrian can be seen by the driver. (d) The bird's-eye view at the same time of (c). (e) The pedestrian is walking in front of the driver. (f) The bird's-eye view at the same time of (e).

side. (Figure 4.16 (a)(b)(c)(d)(e)(f)) is the frame serious of driver's eyesight and bird's-eye view images at the same time.

(Figure 4.17 (a)(b)(c)(d)) is a child walking through the intersection. The child is shorter than the pedestrian in (Figure 4.16), and more difficult to be seen by the driver. In (Figure 4.17 (a)), when the child is walking to the same position of the pedestrian in (Figure 4.16 (c)),the driver still can not seen. But we can find the child's figure in (Figure 4.17 (b)). When the child is in front of the driver (Figure 4.17 (c)(d)), driver may just notice that there is a child in front of the car.

4.2.4 System performance

We measure the system performance by using a timer to count execution time and a counter to record the frame number we processed. The maximum system clock rate of ADSP BlackFin-561 is 133MHz, and the timer counting in the same frequency. We set the timer to count $1.33x10^7$ times to count 10 seconds. After time out, the timer will send a interrupt to DSP core and output the processed frame number. The processing speed of this vehicle surrounding monitoring system is 14.3 frame per second. Each frame will need 0.06993 second to process. We also use a statistical profiling tool to observe the time cost of each functionality. The two major parts is filling buffer and decide image sources of overlapping area. It takes 46.44% of execution time to fill buffers by looking up table, 41.49% to compute the sum of pixels' differences in overlapping areas as well as decide the image sources of these four areas.



(a) Intersection 4

(b) Bird's-eye view image 4 of intersection in alley

(c) Intersection 5

(d) Bird's-eye view image 5 of intersection in alley

Figure 4.17: **Frame serious of driving through the intersection in an alley.** (a) There is a child behind the building, but can not be seen by driver. (b) The bird's-eye view image show the figure of the child. (c) Four seconds later, the child can be seen by driver. (d) The bird's-eye view at the same time of (c).


E
S
A
1896

Chapter 5

Conclusion

In this thesis, we develop a driving assistant system which provides the bird's eye view image of vehicle surrounding. In this system, four fisheye cameras are mounted around a vehicle to capture images of the surroundings from four sides of a vehicle. All fisheye images are rectified to the virtual perspective images according to the fisheye distortion model and these rectified images are stitched together to synthesize the bird's-eye view image. This bird's-eye view surrounding monitorint system involves two major techniques, finding pixel mapping relationship and real time image processing in embedded system. We propose a simple method to define the mapping relationship, and novel methods to ensure the visibility of obstacles around the surrounding of a vehicle.

For finding the pixel mapping relationship, we first rectify the fisheye image with the FOV distortion model. Then, we proposed a simple method by using lanes in parking lot to calibrate the homography matrixes. The fisheye distortion model help us to rectify or warp image easily. The curved line become straight in the rectified fisheye image, and the image quality and resolution of warped bird's-eye view image is better than virtual perspective image. The calibration procedures of homography matrixes is simple and effective, only requires to mark out the corresponding points in calibration pattern and in rectified fisheye images once. Then, the pixel mapping information can be decided, and this mapping relation can be recorded into a look up table to reduce the computation of embedded system and complexity of application implementation in embedded system.

For real time image processing in embedded system, we proposed a novel method to dynamically decide the image source of overlapping area. With this feature, we can eliminate the missing and ghosting effects of the bird's-eye view. We also utilize memory hierarchical structure and apply pipeline mechanism to speed up the image processing in embedded system.

The fisheye images taken from four fisheye cameras on front, both sides, and rear are combined in a bird's eye view image. The blind-spot-less image of vehicle surroundings is synthesized and displayed on a screen for driver to reference. With the support of this bird's-eye view vehicle surrounding monitoring system, drivers can easily maneuver their vehicles by surveying the surrounding area of their vehicles on a single screen. Driver can avoid collision with the obstacles which appear in the blind spot area around their vehicles. Parking a vehicle into a narrow parking space also becomes an easy task. While there are

obstacles approaching the vehicle, the bird's-eye view image will show out immediately, The proposed vehicle monitoring system not only provides an easy and efficient way for maneuvering vehicles but also provides the higher safety level for passengers and pedestrians.


E
S
A
1896

Bibliography

[1] Adsp-bf561 blackfin processor hardware reference, revision 1.1. 2007.

[2] Adsp-bf561 data sheet rec.c. 2007.

[3] Adsp-bf561 programming reference, revision 1.2. 2007.

[4] Steffen Abraham and Wolfgang Frstner. Fish-eye-stereo calibration and epipolar rectification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(5):278–288, 2005. doi: DOI: 10.1016/j.isprsjprs.2005.03.001.

[5] F. Devernay and O. Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13(1):14–24, 2001.

[6] T. Ehlgen, T. Pajdla, and D. Ammon. Eliminating blind spots for assisted driving. *Intelligent Transportation Systems, IEEE Transactions on*, 9(4):657–665, 2008.

[7] T. Ehlgen, M. Thorn, and M. Glaser. Omnidirectional cameras as backing-up aid. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–5, 2007.

[8] T. Gandhi and M. M. Trivedi. Motion based vehicle surround analysis using an omnidirectional camera. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 560–565, 2004.

[9] T. Gandhi and M. M. Trivedi. Dynamic panoramic surround map: Motivation and omni video based approach. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3, 2005.

[10] T. Gandhi and M. M. Trivedi. Vehicle surround capture: Survey of techniques and a novel omni-video-based approach for dynamic panoramic surround maps. *Intelligent Transportation Systems, IEEE Transactions on*, 7(3):293–308, 2006.

[11] Tarak Gandhi and Mohan Trivedi. Parametric ego-motion estimation for vehicle surround analysis using an omnidirectional camera. *Machine Vision and Applications*, 16(2):85–95, 2005. 10.1007/s00138-004-0168-z.

[12] S. Gehrig, C. Rabe, and L. Krueger. 6d vision goes fisheye for intersection assistance. In *Computer and Robot Vision, 2008. CRV '08. Canadian Conference on*, pages 34–41, 2008.

[13] Yu-Chih Liu, Kai-Ying Lin, and Yong-Sheng Chen. Bird's-eye view vision system for vehicle surrounding monitoring. In *Robot Vision*, pages 207–218. 2008.

[14] B. Micusik and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–485–I–490 vol.1, 2003.

[15] David A. Patterson and John L. Hennessy. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann Publishers Inc., 2008. 1502247.

[16] Wolfgang Schulz, Markus Enzweiler, and Tobias Ehlgen. Pedestrian recognition from a moving catadioptric camera. In *Pattern Recognition*, pages 456–465. 2007. 10.1007/978-3-540-74936-3-46.

[17] E. Schwalbe. Geometric modelling and calibration of fisheye lens camera system. *Institute of Photogrammetry and Remote Sensing - Dresden University of Technology, Dresden*, 2005.

[18] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing: Analysis and Machine Vision*. Thomson-Engineering, 1998.

[19] A. Takahashi, Y. Ninomiya, M. Ohta, M. Nishida, and M. Takayama. Rear view lane detection by wide angle camera. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 148–153 vol.1, 2002.

[20] Dang Thao, C. Hoffmann, and C. Stiller. Self-calibration for active automotive stereo vision. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 364–369, 2006.

[21] C. Toepfer and T. Ehlgen. A unifying omnidirectional camera model and its applications. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–5, 2007.

[22] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[23] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *Int. J. Comput. Vision*, 27(2):161–195, 1998. 290094.