OKLAHOMA TRANSPORTATION CENTER

ECONOMIC ENHANCEMENT THROUGH INFRASTRUCTURE STEWARDSHIP

OKCARS: OKLAHOMA COLLISION ANALYSIS AND RESPONSE SYSTEM

QI CHENG, PH.D.
DAMON CHANDLER, PH.D.
WEIHUA SHENG, PH.D.

OTCREOS9.1-15-F

Oklahoma Transportation Center
2601 Liberty Parkway, Suite 110
Midwest City, Oklahoma 73110

Phone: 405.732.6580
Fax: 405.732.6586
www.oktc.org

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

TECHNICAL REPORT DOCUMENTATION PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENTS CATALOG NO.
OTCREOS9.1-15-F		

4. TITLE AND SUBTITLE	5. REPORT DATE
OKCARS: Oklahoma Collision Analysis and Response System	October 31, 2012
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT
Qi Cheng, Damon Chandler and Weihua Sheng	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. WORK UNIT NO.
School of Electrical and Computer Engineering Oklahoma State University 202 Engineering South Stillwater, OK 74078	11. CONTRACT OR GRANT NO. DTRT06-G-0016
12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT AND PERIOD COVERED
Oklahoma Transportation Center (Fiscal) 201 ATRC Stillwater, OK 74078 (Technical) 2601 Liberty Parkway, Suite 110 Midwest City, OK 73110	Final July 2009- October 2012
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES
University Transportation Center

16. ABSTRACT

By continuously monitoring traffic intersections to automatically detect that a collision or near-collision has occurred, automatically call for assistance, and automatically forewarn oncoming traffic, our OKCARS has the capability to effectively reduce emergency response time, and in turn potentially save thousands of lives and millions of dollars each year. We have designed and developed an affordable hardware platform consisting of four smart audio visual (SAV) nodes and a cellular modem. For networking of multiple nodes, we have also developed a software platform. To meet the critical and challenging system requirements, we have developed a near realtime vehicle detection and tracking algorithm requiring modest computing power. As an alternative and complement detection system, we have developed modules for efficient collision sound recognition and localization. We have shown that fusion of data from multiple microphone arrays and/or fusion of results from audio-video subsystems can significantly improve detection accuracy. We have developed a small-scale testbed for validating and verifying OKCARS and associated algorithms. OKCARS is non-intrusive, does not require specialized in-car equipment, operates using existing 3G communication technologies, and is relatively low-cost. It is a significant improvement from traffic monitoring systems currently available, where a human analyst has to make decisions by constantly monitoring several video stream inputs. Through improvement of service monitoring and emergency response preparedness, OKCARS has the potential to enhance roadway traffic safety and security.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Collision analysis and response, audio-video sensor, detection/tracking/ recognition/localization, testbed	No restrictions. This publication is available at www.oktc.org and from the NTIS.

19. SECURITY CLASSIF. (OF THIS REPORT)	20. SECURITY CLASSIF. (OF THIS PAGE)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	105 + covers	

SI (METRIC) CONVERSION FACTORS

Approximate Conversions to SI Units

Symbol	When you know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	25.40	millimeters	mm
ft	feet	0.3048	meters	m
yd	yards	0.9144	meters	m
mi	miles	1.609	kilometers	km
	AREA			
in^2	square inches	645.2	square millimeters	mm
ft^2	square feet	0.0929	square meters	m^2
yd^2	square yards	0.8361	square meters	m^2
ac	acres	0.4047	hectares	ha
mi^2	square miles	2.590	square kilometers	km^2
	VOLUME			
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.0283	cubic meters	m^3
yd^3	cubic yards	0.7645	cubic meters	m^3
	MASS			
oz	ounces	28.35	grams	g
lb	pounds	0.4536	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams	Mg
	TEMPERATURE (exact)			
°F	degrees Fahrenheit	(°F-32)/1.8	degrees Celsius	°C
	FORCE and PRESSURE or STRESS			
lbf	poundforce	4.448	Newtons	N
lbf/in^2	poundforce per square inch	6.895	kilopascals	kPa

Approximate Conversions from SI Units

Symbol	When you know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.0394	inches	in
m	meters	3.281	feet	ft
m	meters	1.094	yards	yd
km	kilometers	0.6214	miles	mi
	AREA			
mm^2	square millimeters	0.00155	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.196	square yards	yd^2
ha	hectares	2.471	acres	ac
km^2	square kilometers	0.3861	square miles	mi^2
	VOLUME			
mL	milliliters	0.0338	fluid ounces	fl oz
L	liters	0.2642	gallons	gal
m^3	cubic meters	35.315	cubic feet	ft^3
m^3	cubic meters	1.308	cubic yards	yd^3
	MASS			
g	grams	0.0353	ounces	oz
kg	kilograms	2.205	pounds	lb
Mg	megagrams	1.1023	short tons (2000 lb)	T
	TEMPERATURE (exact)			
°C	degrees Celsius	9/5+32	degrees Fahrenheit	°F
	FORCE and PRESSURE or STRESS			
N	Newtons	0.2248	poundforce	lbf
kPa	kilopascals	0.1450	poundforce per square inch	lbf/in^2

ACKNOWLEDGMENTS

The authors gratefully thank and acknowledge the financial support provided by the Oklahoma Transportation Center (OkTC) and the RITA University Transportation Center Program.

OKCARS: OKLAHOMA COLLISION ANALYSIS AND RESPONSE SYSTEM

Final Report

October, 2012

Qi Cheng, Ph.D.

Principal Investigator

Damon Chandler, Ph.D.

Weihua Sheng, Ph.D.

Co-Principal Investigators

Oklahoma State University

School of Electrical and Computer Engineering

202 Engineering South

Stillwater, OK 74078

October 2012

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

The purpose of this report is to present the results of our research and development of *OKCARS: Oklahoma Collision Analysis and Response System*. By continuously monitoring traffic intersections to automatically detect that a collision or near-collision has occurred, automatically call for assistance, and automatically forewarn oncoming traffic, the system has the capability to effectively reduce emergency response time, and in turn potentially save thousands of lives and millions of dollars each year. Specifically, we have designed and developed an affordable hardware platform consisting of four smart audio visual (SAV) nodes and a cellular modem. Each SAV node is equipped with an omnidirectional vision sensor, a microphone array and the associated data acquisition board, a compact computer. For networking of multiple nodes, we have also developed a software platform, which is based on the ROS (Robot Operating System), an open source software framework for robots and sensors. To meet the critical and challenging system requirements, i.e., near real-time video analysis (10+ frames/second) and modest computing power, we have developed a vehicle detection and tracking algorithm based on low-level features and a low-level measure of visual dissimilarity developed to mimic the human visual system (HVS), which is demonstrated effective for vehicle tracking for the first time. As an alternative and complement detection system, we have developed modules for efficient collision sound recognition (mainly based on important audio features *Mel Frequency Cepstral Coefficients*) and localization (mainly based on beamforming). We have shown that fusion of data from multiple microphone arrays and/or fusion of results from audio-video subsystems can effectively reduce ambiguity and significantly improve detection accuracy. For validating and verifying OKCARS and associated algorithms, we have developed a small-scale testbed which consists of an arena to mimic traffic environments, an indoor localization system, automated radio controlled (RC) cars and our OKCARS. The developed system is non-intrusive, does not require specialized in-car equipment, operates using existing 3G communication technologies, and is relatively low-cost. It is a significant improvement from traffic monitoring systems currently available, where a human analyst has to make decisions by constantly monitoring several video stream inputs. Through improvement of service monitoring and emergency response preparedness, OKCARS has the potential to enhance roadway traffic safety and security.

CHAPTER 1
INTRODUCTION

1.1 Background and Objectives

As drivers, we all fear the possibility of being involved in a traffic accident. We are concerned about damages to our vehicles, injuries to our passengers, and injuries to ourselves and others. Yet, in the back of our minds, we all have a sense of security that should an accident occur, the proper authorities will be there to provide assistance. Unfortunately, for many motor vehicle crashes, this assistance arrives too late.

When an accident occurs, response time is critical: Every extra minute that it takes for help to arrive can mean the difference between life and death. Studies have shown that the number of traffic-related fatalities is highly dependent on emergency response time [1]. This fact is especially true for states such as Oklahoma in which first responders have a large geographical area to cover. According to the World Health Organization, traffic-related injuries represent the leading cause in worldwide injury-related deaths, claiming an estimated 1.2 million lives each year [2]. Traffic-related injury is among the top-ten causes of worldwide death, a list that includes tuberculosis, heart disease, and HIV/AIDS. In the United States, it is estimated that vehicle accidents account for over 40,000 deaths and cost over $164 billion dollars each year. Among these, passenger-vehicle crashes accounted for the vast majority of deaths [3]. Without preventative intervention, these figures are estimated to increase by 65% over the next 20 years.

Given these statistics, we put forth the following question: With today's advanced monitoring technology, shouldn't it be possible to automatically detect if an accident has occurred and automatically call for assistance? The ability to reduce emergency response time by even a small amount can potentially save thousands of lives and millions of dollars each year. In this project, we propose to research and develop such a system which we have titled OKCARS: Oklahoma Collision Analysis and Response System. The system consists of video cameras and microphones mounted upon traffic posts at intersections. The system operates by continuously analyzing the acquired audio and video to extract salient features, determining correspondences between auditory and visual regions of interest, and then integrating the data within the regions of interest to determine if an accident has occurred.

The specific aims of our system are (1) to reduce the time required for proper assistance (not just first responders) to arrive at the scene, and (2) to mitigate further accidents by forewarning on-coming traffic. It is important to note that no artificial system can replace the accuracy of a human observer. In automated monitoring, there is always the possibility of a false alarm or the possibility of failing to detect a true accident. Accordingly, our system uses a combination of autonomous and human monitoring. If the system deems that the probability that a collision or detrimental near-collision has occurred surpasses a predetermined threshold, the system will notify authorities by providing both the probability value and the captured audio and video. Then, by viewing and listening to the audio-visual data, a human monitor can pre-screen the data and call for appropriate response (e.g., ambulance or fire). We believe this scheme represents a logical and feasible tradeoff between sensitivity (low miss rate) and specificity (low false-alarm rate). The developed system is non-intrusive, does not require specialized in-car equipment, operates using existing 3G communication technologies, is relatively low-cost, does not disrupt traffic during maintenance, can provide footage to facilitate accident reconstruction, and has the potential to stimulate Oklahoma's economy by facilitating jobs and industry in wireless communication technology.

1.2 Main Contributions

- In this project, we have developed four smart audio-video (SAV) sensor nodes. Each of them consists of an omnidirectional vision sensor, a microphone array and the associated data acquisition board. We have two versions of this omnidirectional vision sensor: catadioptric camera and fish-eye camera. We have developed the software to interface with the vision in both Windows and Linux OSes. The microphone array consists of multiple microphones and a plastic ring which is light and adjustable is radius. Two types of data acquisition boards have been tested. One is USB7202 and the other is NI 9234. We also developed the software platform for the networking of multiple audio/video sensor nodes. The software platform is based on the ROS (Robot Operating System), which is an open source software framework for robots and sensors.
- We have successfully developed a near real-time video-analysis system for accident detection. For detection and tracking of the vehicles, our algorithm operates based on low-level features and a low-level measure of visual dissimilarity developed to mimic the

human visual system (HVS). Low-level features (e.g., color, orientation, size) are used because of their low computational complexity. Although HVS models have found widespread use in a variety of consumer image processing applications, our work was the first to demonstrate the effectiveness of HVS models for vehicle tracking. For accident detection, we implemented a real-time version of an existing algorithm.

- We have tested our algorithm offline on video sequences (saigon01 and saigon02) obtained from a traffic intersection in downtown Saigon, Vietnam; these videos do not contain collisions, but are useful for verifying the detection and tracking stages. Ground truth labels are manually determined for these two videos. Our algorithm achieves a detection rate of 90-93% and a tracking rate of 88-92%. We further test our system on a database of videos of a mock intersection obtained from our testbed; these videos contain both normal driving and collisions. Our algorithm yields an accident detection precision of approximately 87.5% on these videos.
- We have developed an audio data processing flow chart for accident sound detection and localization, which includes source separation, accident sound detection and direction of the accident sound estimation. The frequency domain blind source separation method turns out to be efficient. Based on our study of the collected online car accident/collision sound tracks, important audio features such as Mel Frequency Cepstral Coefficients are identified for accident/collision detection, which can provide high detection accuracy. We have implemented and tested the method in both simulations and the testbed. We have implemented the basic sound localization algorithm on the built microphone array and its performance is limited. Several advanced direction of arrival estimation algorithms are investigated under different signal and environmental conditions (low and high SNRs, single and multiple sources). A fusion algorithm that can handle data from multiple microphone arrays has been proposed. A Bayesian audio/video fusion scheme for general region of interest detection has been developed.
- We have developed a small-scale testbed to conduct the experiments that can be used to validate our proposed collision detection algorithms. Our testbed has four main parts: an arena; an indoor localization system; automated radio controlled (RC) cars; and roadside monitoring facilities. First, to mimic traffic environments we built an arena with a

wooden floor, mock buildings and streets. Second, to facilitate feedback control for trajectory following, an indoor localization system was set up to track the RC cars. Third, both autonomous driving RC cars and human driving RC cars were developed based on an automated RC car design. The automated RC cars can receive control signals from a computer through an Xbee RF module and control the front and rear wheels through motors. A new control algorithm was developed to allow the RC cars to track predefined trajectories. Finally, the roadside monitoring system is a collection of SAV nodes which can collect both the audio and video data from the collision scenario.

1.3 Report Organization

The rest of the report is organized as follows. Prior studies on automatic incident detection are summarized in Chapter 2 of Literature Review. In Chapter 3, we explain the architecture and hardware setup of our proposed system. The video based collision analysis system is presented in Chapter 4. The audio based collision analysis system and multisensor/multimodal fusion are presented in Chapter 5. In Chapter 6, we present the development and implementation of a small-scale testbed. We draw the concluding remarks and provide discussion in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

Systems for detecting motor vehicles crashes and other traffic incidents are generally referred to as *Automatic Incident Detection* (AID) systems, which are often deployed as part of a broader intelligent transportation system (ITS). The earliest work on AID dates back to the 1970s [4]. Since that time, several ITS traffic management systems have employed AID; some notable examples include the Los Angeles Freeway Surveillance and Control Project, the John Lodge Freeway system, the Tokyo Expressway Control system, and the Tangenziale di Napoli (TANA) system.

Most incident detection systems are based on the macroscopic behavior of traffic flow. Traffic is considered as a stream of vehicles, and measurements are made on the global properties of the stream. For example, loop detectors are commonly used to measure average velocity, flow rate, and average vehicle occupancy. Unusual deviations from these averages are then presumed to be an incident. The concept is similar to that of locating a stone in a stream by looking for disruptions in water flow. Unfortunately, there are several limitations of this flow-based approach.

First, the averaging process introduces a time delay, which in turn, adds to the overall emergency response time. Most AID algorithms take several minutes to alert the monitoring center. For example, the "California Algorithm 7" technology used in the Los Angeles Freeway Project has an average reaction time of 4.5 minutes [5]. Second, even though AID systems which rely on flow measurements have proved useful for highways, such systems are not suited for intersections which necessarily impose a disruption in flow. For example, the TANA system requires a flow of at least 1,000 vehicles per hour and a minimum speed of 37 miles/hour, requirements which obviously cannot be met at an intersection. Finally, many AID systems are considered flawed. Either they yield too many false alarms, too many missed detections, or they are not adaptable to changing environmental conditions. According to [5], "some Departments of Transportation have shut down their AID algorithms altogether because of the problems that they have had."

These limitations have driven recent efforts toward the use of video-based systems, which employ video cameras and analysis algorithms to monitor traffic [6-8]. The application of

machine vision for traffic monitoring has previously been used mainly for acquiring traffic figures such as numbers of vehicles, speed data, and amount of congestion [7]. More recently, these techniques have been applied to automatic incident detection [8]. For example, in [9], Ikeda *et al.* propose a video-based system for detecting stopped/slowed vehicles and fallen objects. In [10], Veeraraghavan *et al.* propose the use of a Kalman filter for detecting accident-prone incidents. In [6], Kamijo *et al.* employ a spatiotemporal Markov random field for tracking, and a Hidden Markov Model for incident detection; this is one of the very few systems designed specifically for intersections.

Yet, despite its promise, video-based AID suffers from one very serious shortcoming: It is prone to false alarms when environmental conditions change. For example, rain creates glare and reflections which are consequently misinterpreted as vehicles and/or obstructions. Shadows and occlusion are particularly problematic for vehicle tracking. Moreover, at nighttime, normally reliable features such as shape and color are unavailable. According to [11], changes in environmental conditions have been shown to give rise to a false-alarm rate as high as 50-80%.

A simple and effective solution to these problems is to use other senses beyond just vision. In particular, for collision detection, auditory information can be a very important indicator of whether a crash has occurred. Typical collisions are accompanied characteristic audio signatures which can be detected and localized. When audio is coupled with video, much greater detection accuracy can be realized than using either modality alone. This strategy of combining audio and video falls under a research thrust known as multimodal fusion. Furthermore, to overcome issues related to occlusion and shadows, one can fuse video/audio captured from multiple viewpoints (multiviewpoint fusion).

CHAPTER 3

SYSTEM ARCHITECTURE AND HARDWARE SETUP

As shown in Figure 3.1, the OKCARS consists of a set of Smart Audio-Video (SAV) nodes, connected through an 802.3 Ethernet bus. Each SAV node consists of an omnidirectional camera, four mini microphones, and a small-form-factor eBox-3850 computer. One of the SAV nodes is designated as the master node and has interfaces to a wireless modem to access the cellular network for accident reporting purposes and for communicating with a roadside electronic sign for alerting. This OKCARS system hardware is reconfigurable and expandable. The minimum or basic configuration consists of only one master SAV node, a cellular modem and a roadside electronic sign. More SAV nodes can be added to allow an enlarged field of view and improved accuracy through multiviewpoint fusion. This system can be seamlessly integrated into the existing infrastructure of many intersection control and management systems.

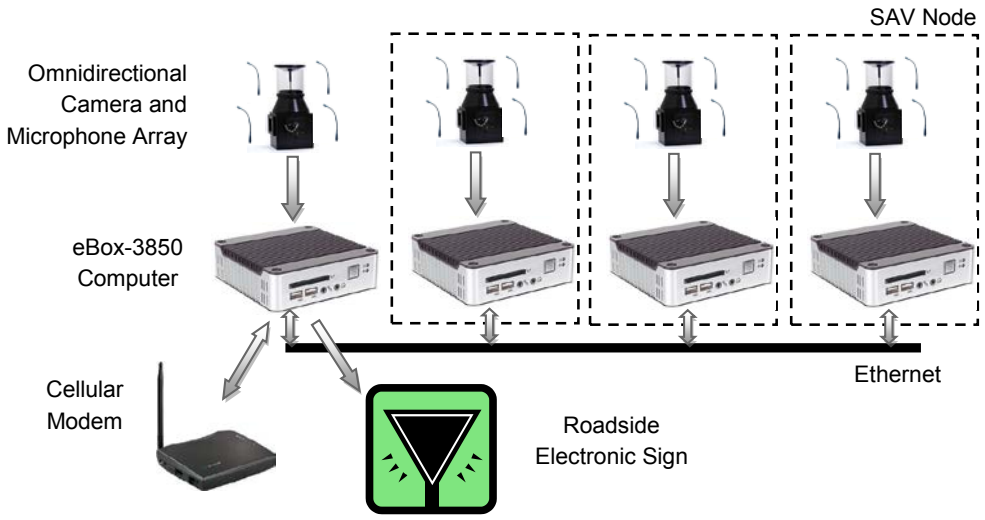


Figure 3.1 The overall architecture of OKCARS

3.1 Development of the Visual Sensor Node

Two different types of cameras are used, which include catadioptric cameras and fish-eye cameras. Both are omni-directional cameras. The catadioptric camera consists of a hyperbolic

mirror and a Firefly MV sensor from Point Grey Research. The fish-eye camera is a surveillance camera named Q24 from Mobotix Company.

The cameras are shown in Figure 3.2. For the purpose of real-time processing, we develop the C/C++ interface for the cameras to obtain the data without the software provided by the companies, and then to process the data using open source libraries such as ROS and OpenCV.

Figure 3.2 The catadioptric camera (left) and the fish-eye camera (right)

3.1.1 Catadioptric Camera

Similar to most catadioptric optical systems, the catadioptric camera consists of two parts: the small and affordable imaging camera Firefly MV and the hyperbolic mirror. A mini USB2.0 interface is provided for the data transmission and power supply. The FlyCapture software development kit (SDK) is provided by the company. So we use it directly to obtain the raw data from the camera. First we need to add the FlyCapture library to the project path so we can use the functions and then we put the obtained image in the RAM and then different algorithms can be applied on the raw data. We use the OpenCV library for video processing.

One problem for the catadioptric system is that the original image captured from the camera is distorted as shown in Figure 3.3, which is not convenient for image processing. So a simple algorithm is developed to unwrap the distorted image to a panoramic image. The algorithm is based on the projection from a ring area to a rectangle area. The result is shown in Figure 3.4. It can achieve a frame rate of around 4 fps with a resolution of 1280 x 960 (original) and 2500 x 330 (unwrapped) pixels which ensures real-time processing. The unwrapped images contain a

360 degree view of the environment which enables the robots to perform some tasks better, such as searching and navigation.

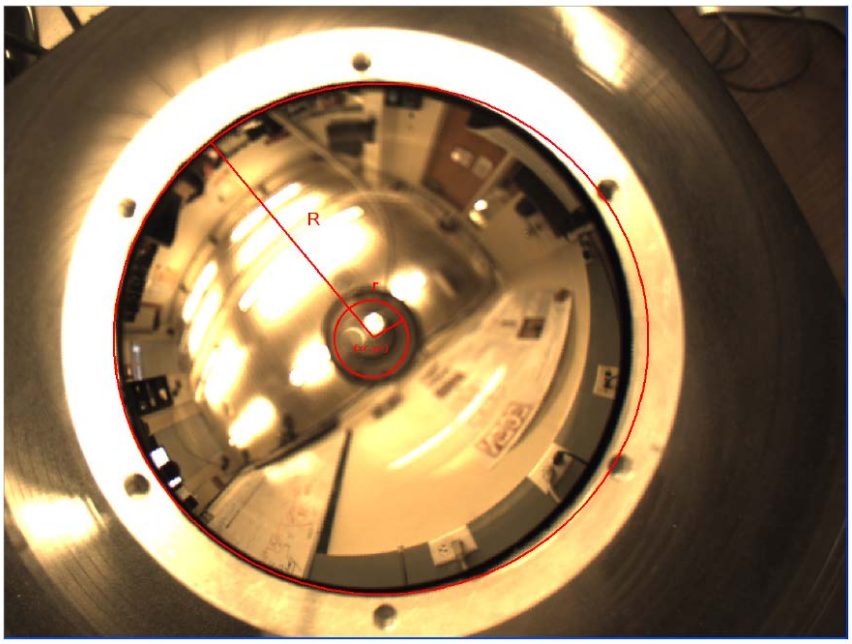


Figure 3.3 The ring area for undistortion of catadioptric camera

Figure 3.4 Image from the camera after unwrapping

3.1.2 Fish-eye Camera Q24

The fish-eye camera (Q24) is capable of providing four different views simultaneously. The panoramic view is selected so that it can cover the surrounding area of the mobile platform. The camera provides a highest resolution of 3 Megapixels and color images scalable from 160 x 120 to 2048 x 1536, and it uses an Ethernet-based interface. The features of the camera (including resolutions, frame rates, etc.) can be easily adjusted by sending a web request. Moreover, the zooming and panning of the camera lenses can be done by the virtual PTZ function. The camera

itself is a web server so that the stream of live images can be obtained by setting up a socket connection.

The Q24 camera is a commercial product for the purpose of security and surveillance so the SDK for the camera is not available. Our solution to get the raw images from the Q24 camera is to use the Libcurl library to establish an Internet connection to grab the current image from the web server. Libcurl is a free and easy-to-use client-side URL transfer library, supporting different Internet protocols and services. Libcurl is highly portable, and it builds and works identically on numerous platforms. In this way, we can store the image in the RAM of the computer. After that, different algorithms can be applied on the raw data. Here we use OpenCV library for the vision processing. It can achieve a frame rate of around 7-8fps with a resolution of 800 x 600 pixels which ensures the real-time processing.

3.2 Development of the Microphone Array

The audio part of the audiovisual node is shown in Figure 3.5. The hardware platform supports up to 8 channels microphones. The DAQ (Data Acquisition Equipment) in the platform supports maximum 100 KSamples/second throughput sampling rate, or 50 KSamples/second on any one channel. The DAQ USB-7202 used in this project is a general purpose DAQ. Therefore, a preamplifier board is designed to amplify the signal acquired from the microphones and output to the DAQ. An example code based on C++ is developed to read the data from USB-7202. The code also provides a Matlab interface to call Matlab functions.

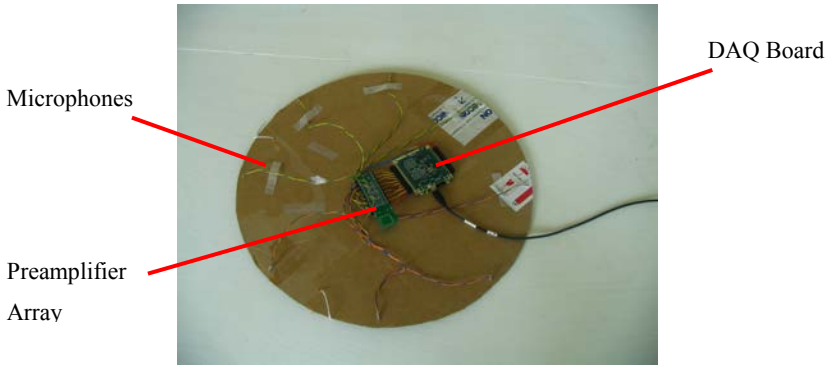


Figure 3.5 The audio part of a prototype of the SAV node

3.2.1 The USB-7202 DAQ

USB-7202 is a USB bus-powered DAQ module with eight, 16-bit analog inputs and eight digital I/O lines; for Message-Based DAQ - Designed for OEMs.

- 8 channels of 16-bit analog input
- 100 kS/s max total throughput (200 kS/s Burst Mode), 50 kS/s on any one channel
- 8 digital I/O lines
- One 32-bit event counter
- Simultaneous sampling (1 A/D converter per input)
- Stackable 3.55" x 3.75" board dimensions
- Develop on one computing platform, deploy on many with out-of-the-box support for Windows® and Linux®
- Simple messaging protocol
- Small software footprint
- Included accessories: USB cable and a CD containing the DAQFlex DAQ Software API, a Windows® installer file (msi), and a Zip file containing installation files for Linux® operating systems
- RoHS compliant

3.2.2 Design of the Microphone Array

Figure 3.6 shows the simplified block diagram of the hardware platform.

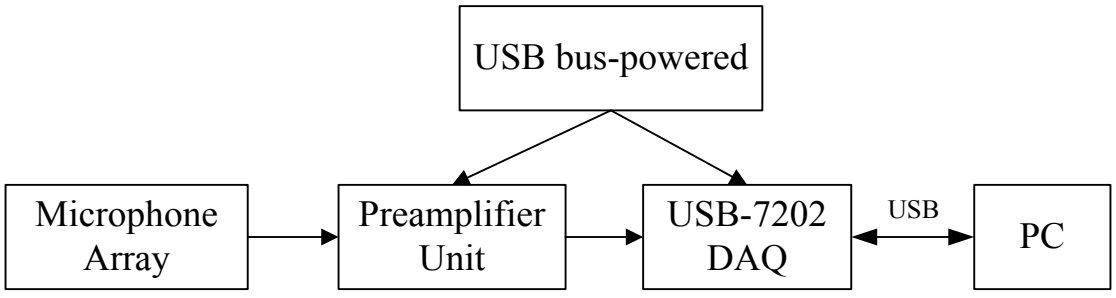
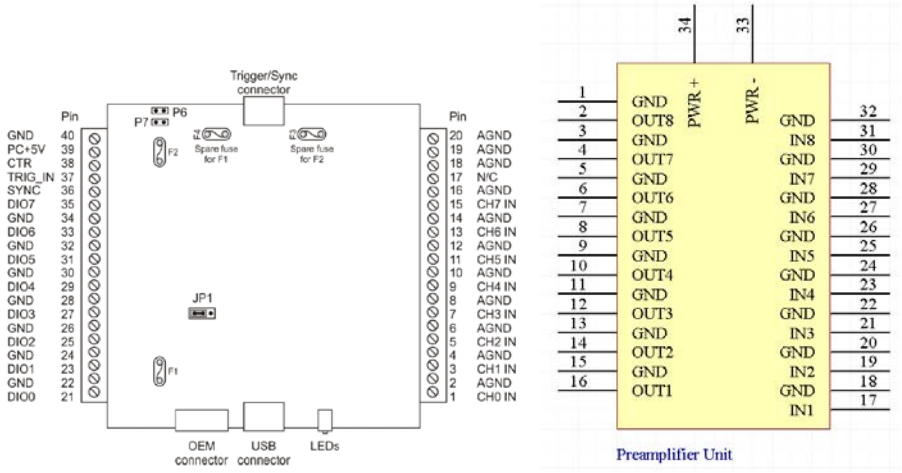


Figure 3.6 The Block Diagram of the Microphone Array Platform

3.2.3 Steps to Use the Microphone Array

1. Install USB driver and software MCC 7000 (DAQ Software ver 1_3_0_0.ZIP).
2. Connect the preamplifier unit board to the USB DAQ as shown in Figure 3.7 and Table 3.1.
3. Connect the microphones to the input of the preamplifier unit. Make sure the positive pin of the microphone connect to IN* and the negative pin connect to GND as shown in Figure 3.8**.**
4. Power on the hardware platform. Plug the USB cable to the PC.
5. Read the sampling data.



USB-7202 DAQ Preamplifier unit

Figure 3.7 Pins definition of DAQ and preamplifier unit

Table 3.1 Pins connection

DAQ	Pre Unit	DAQ	Pre Unit	DAQ	Pre Unit	DAQ	Pre Unit
1	16	6	11	11	6	16	1
2	15	7	10	12	5	39	34
3	14	8	9	13	4	40	33
4	13	9	8	14	3		
5	12	10	7	15	2		

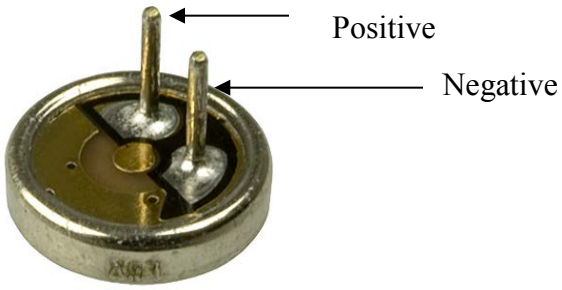


Figure 3.8 The polarity of the microphone

We also developed a flexible ring mount for the microphones. This improved microphone array is shown in Figure 3.9.

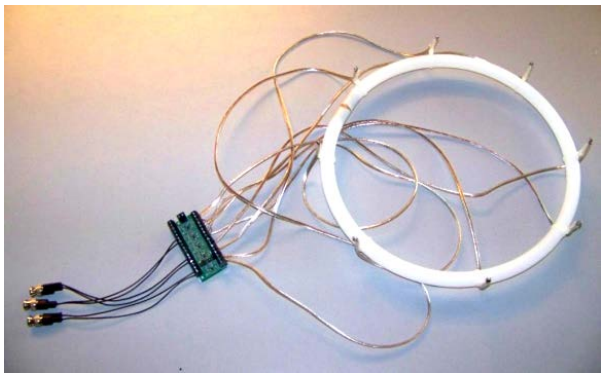


Figure 3.9 The audio part of the SAV node: the microphone arrays on a flexible ring

3.3 FitPC2

The FitPC2 is used as the computation engine for the integrated SAV node. The FitPC2 is a small, light netbook computer which includes an Intel Atom Z5xx Silverthorne processor (1.1/1.6/2.0 GHz options), up to 2GB of RAM and 160GB SATA Hard Drive. We run the Ubuntu Linux OS on this FitPC2 and the Robot Operating System (ROS) to provide the networking capability.

3.4 Integration of the Smart Audio Visual Sensor Node

Finally we integrate both the audio and video part to make a whole node which is shown in Figure 3.10.

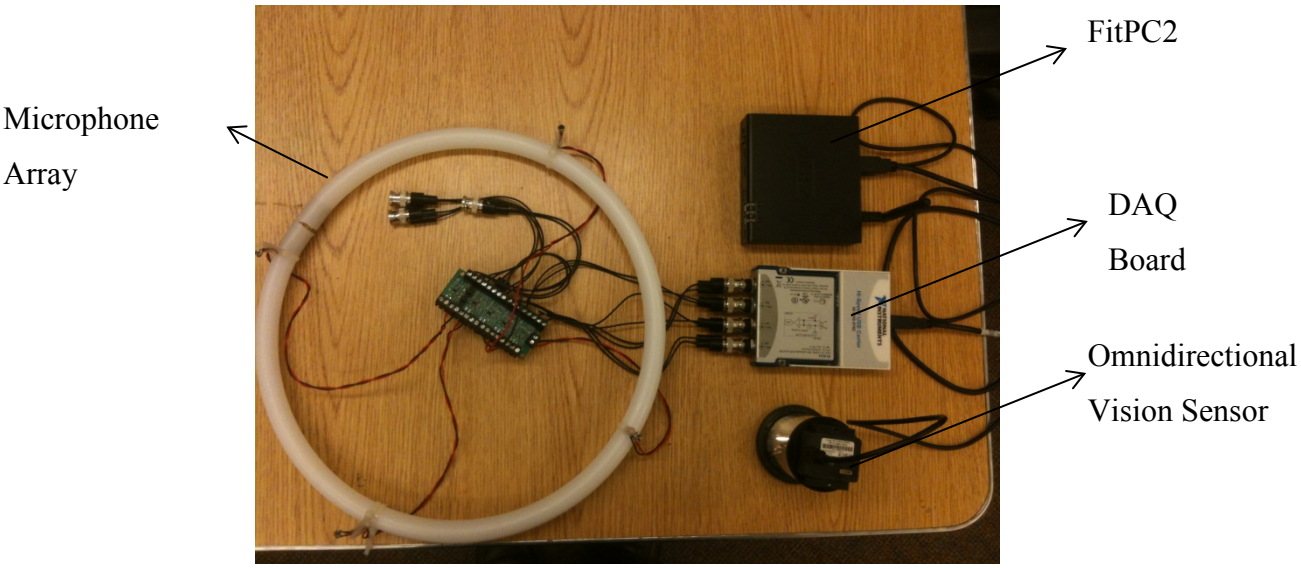


Figure 3.10 The prototype of the vision/audio sensor node

3.5 Communication

The cellular modem will deliver a video clip of the suspected collision or near-collision to a remote computer located in a control center monitored by local authorities.

CHAPTER 4

VIDEO BASED ACCIDENT ANALYSIS

An important stage in an automatic vehicle monitoring system is the detection of accidents via video analysis. By analyzing frames of the captured video, it is possible to track the movements of vehicles. With such tracking, data about the vehicles, such as speed, change in speed, and change in orientation, can be determined. Finally, these data can be used to estimate the presence of an accident (collision).

In this chapter, we describe the video-analysis portion of the accident detection system that was researched and developed in this project. Figure 4.1 shows a block diagram of the system.

4.1 General Approach

As shown in the Figure 4.1, image sequences are obtained from the video camera mounted on a pole at the traffic intersection. The image sequences are fed to the accident detection system where the occurrence of an accident is determined. The accident detection system consists of the following stages: (1) vehicle detection, (2) features analysis, (3) vehicle tracking, and (4) vehicle

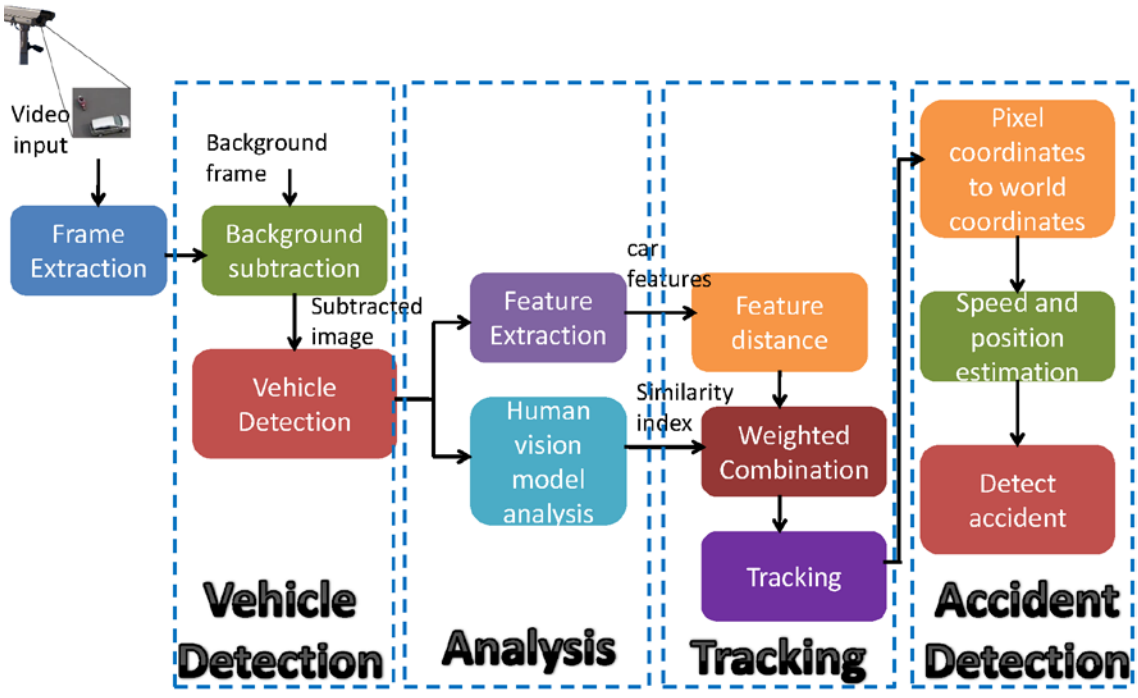


Figure 4.1 Overview of the video analysis portion of the accident detection system

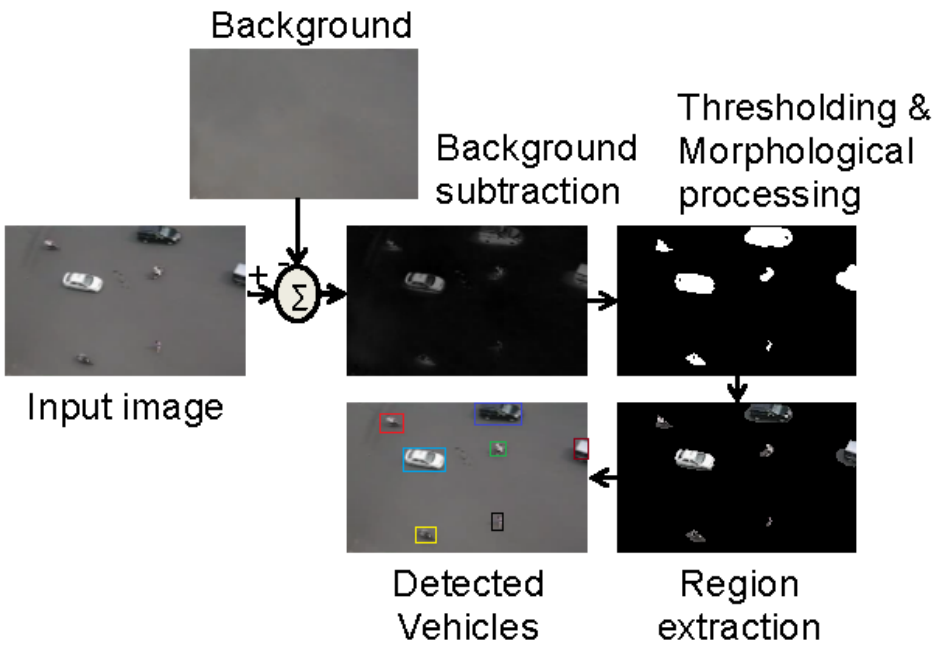


Figure 4.2 Block diagram of the processing performed during the vehicle detection stage

parameter extraction and accident detection.

In addition to the image input, some side information is input to the system: the stored background image, threshold values for the image processing, information about the position and orientation of the camera, camera calibration parameters, and frame rate of the video sequence. After analyzing the image sequence, the system identifies the moving vehicles in the image and tracks them using low-level features. After the vehicles are tracked in each frame, the speed, orientation, position, and area of each tracked vehicle are used to estimate the occurrence of an accident. If an accident is detected, the system signals the detection to a monitoring station.

4.2 Vehicle Detection

Vehicle detection is an important stage of the accident detection system in which the moving vehicles are segmented from the background. Figure 4.2 shows a block diagram of the vehicle detection subsystem.

The method that is used for detecting moving vehicles is *background subtraction*, a very computationally efficient technique. Because our research has focused on real-time video analysis, other background modeling techniques that have high computational cost were attempted, but not actually employed. Because the testing of the algorithm was done offline, and

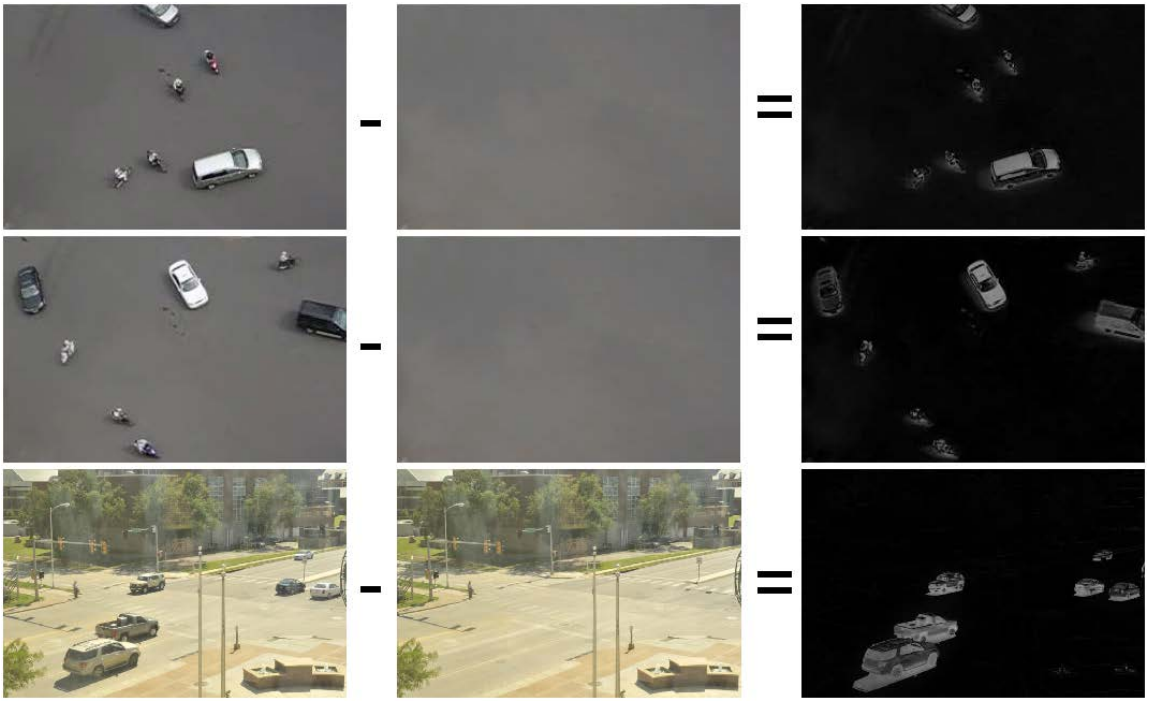


Figure 4.3 Left: Input frames. Middle: Static blank background. Right: Result of subtraction

because the position of the camera was fixed, we used a stored background frame for use in background subtraction. After the vehicle regions are detected, suitable low-level features are extracted from the vehicle regions. The process of vehicle detection is explained in detail in the following sections.

4.2.1 Background Subtraction

The first step in the vehicle detection algorithm is to subtract the background from the current input frame to detect the vehicles. Figure 4.3 shows examples of background subtraction method. Here, a frame at time *t* from the input video along with the previously acquired background frame (containing no vehicles) is fed as input to the algorithm. The algorithm subtracts the intensity value of each pixel in the frame $I_t(x,y)$ from the background image $I_{bk}(x,y)$ resulting in a difference image $I_{diff}(x,y)$ given by

$$I_{diff}(x,y) = |I_t(x,y) - I_{bk}(x,y)|$$

As mentioned, this background subtraction step is performed to detect moving objects since the static objects are part of background. Thus, we are left with the intensity values of moving objects in the difference image $I_{diff}(x,y)$.

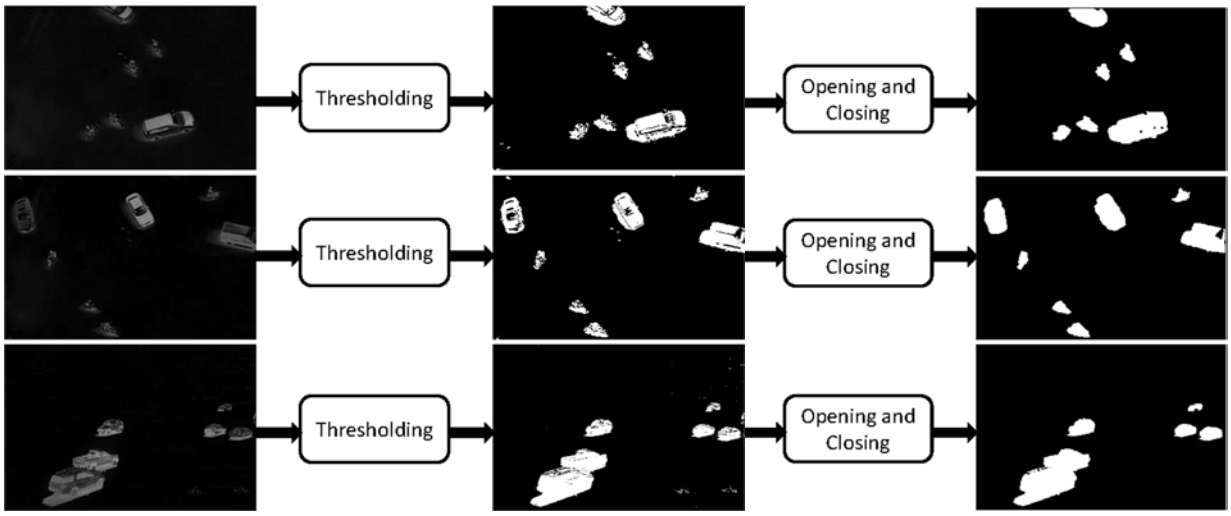


Figure 4.4 Thresholding and morphological processing are used to obtain a binary map of vehicle pixels

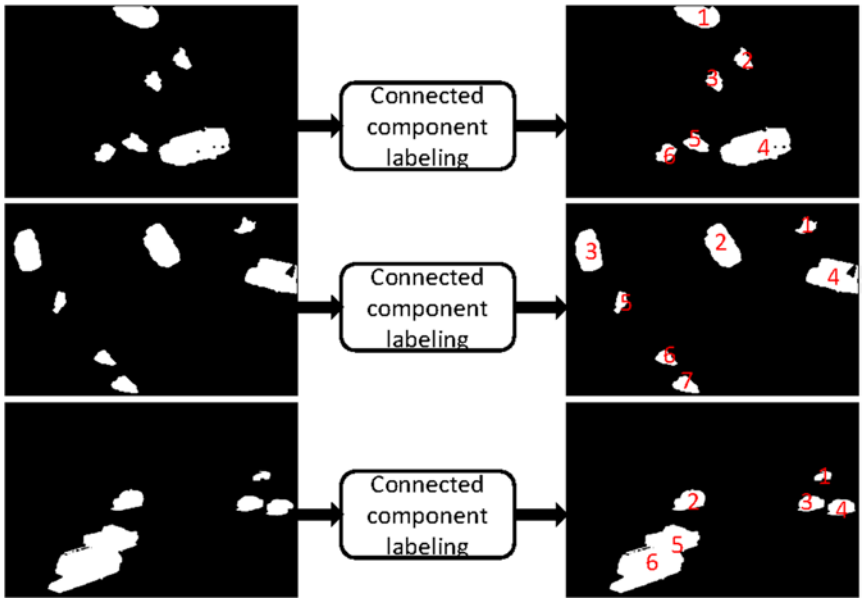


Figure 4.5 Connected components labeling is used to segment the binary map and assign a unique label to each detected vehicle

4.2.2 Thresholding and Morphological Processing

The difference image $I_{diff}(x,y)$ is converted into a binary image $bw(x,y)$ using a specific threshold value T as follows:

$$bw(x,y) = \begin{cases} 1, & I_{diff}(x,y) \geq T \\ 0, & I_{diff}(x,y) < T \end{cases}$$

The value of T was empirically chosen to be 0.1.

The binary image $bw(x,y)$ obtained from thresholding suffers from noise and unwanted pixels. Therefore, the morphological operations of opening followed by closing are applied to the binary image $bw(x,y)$ to obtain a final binary image $bw_{final}(x,y)$. Figure 4.4 shows the thresholding and morphological processing operations on example frames. The final binary image $bw_{final}(x,y)$ indicates pixels corresponding to detected vehicles.

4.2.3 Connected-Component Labelings and Region Extraction

The regions in the binary image $bw_{final}(x,y)$ are labeled using connected-components labeling. This process assigns a label to each region in the binary image (see Figure 4.5). From this process, the number of vehicles detected in the image is estimated. After connected-components labeling, the binary map is used to guide analysis of the original input frame $I_t(x,y)$; we specifically focus only on those regions in which the vehicle are detected.

4.3 Feature Extraction

After the regions containing vehicles are detected, suitable low-level features are extracted from these vehicle regions. Five features are used: area, centroid, orientation, luminance, and color. These features were chosen due to their low computational complexity. Let X_i $\{i= 1, 2, 3...\}$ denote the individual vehicle regions detected in the input image $I_t(x,y)$, and let $f_k(X_i)$ denote the k^{th} feature.

4.3.1 Bounding Box

From the connected component labeled image, the bounding-box coordinates of each vehicle region are calculated. From the bounding-box coordinates, the height and width information of the vehicle region is estimated. These bounding-box coordinates are used to calculate the features of a particular vehicle region. Figure 4.6 shows an example of extracted vehicle regions.

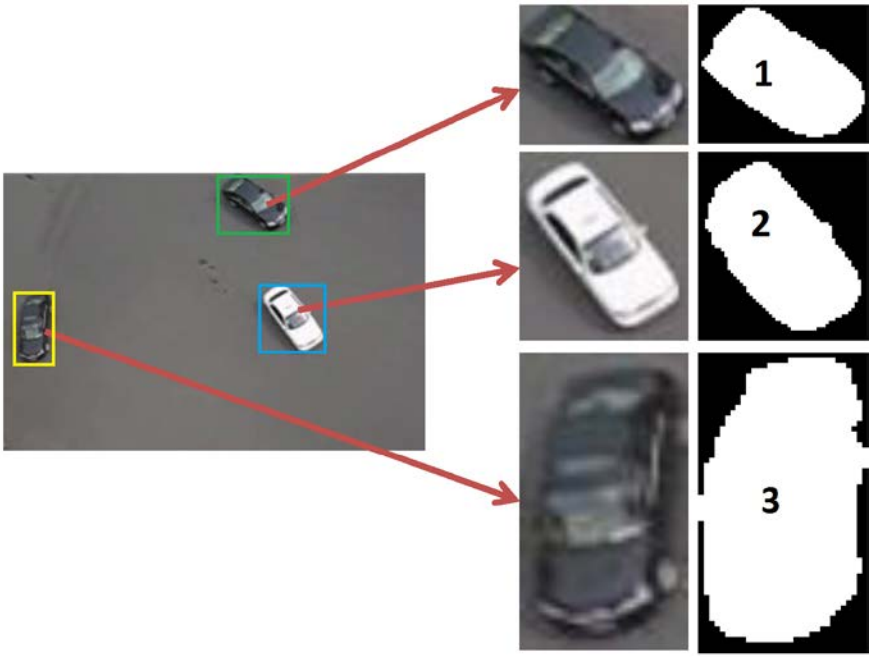


Figure 4.6 Vehicle regions are extracted from each frame via multiplication with the frame's corresponding binary map.

4.3.2 Area

Let $f_1(X_i)$ denote the area of region X_i. Area is defined as the total number of pixels N in the region X_i. The expression for $f_1(X_i)$ is given by

$$f_1(X_i) = N \in X_i$$

The area of a particular vehicle region is given by the number of white pixels in the binary map of each vehicle (see Figure 4.6).

4.3.3 Centroid

Let $f_2(X_i)$ denote the area of region X_i. The centroid is defined as the center of mass of the region X_i. The expression for $f_2(X_i)$ is given by

$$f_2(X_i) = \left(\frac{x_1 + x_2 + \cdots x_n}{N}, \frac{y_1 + y_2 + \cdots y_n}{N}\right) = (\bar{x}, \bar{y})$$

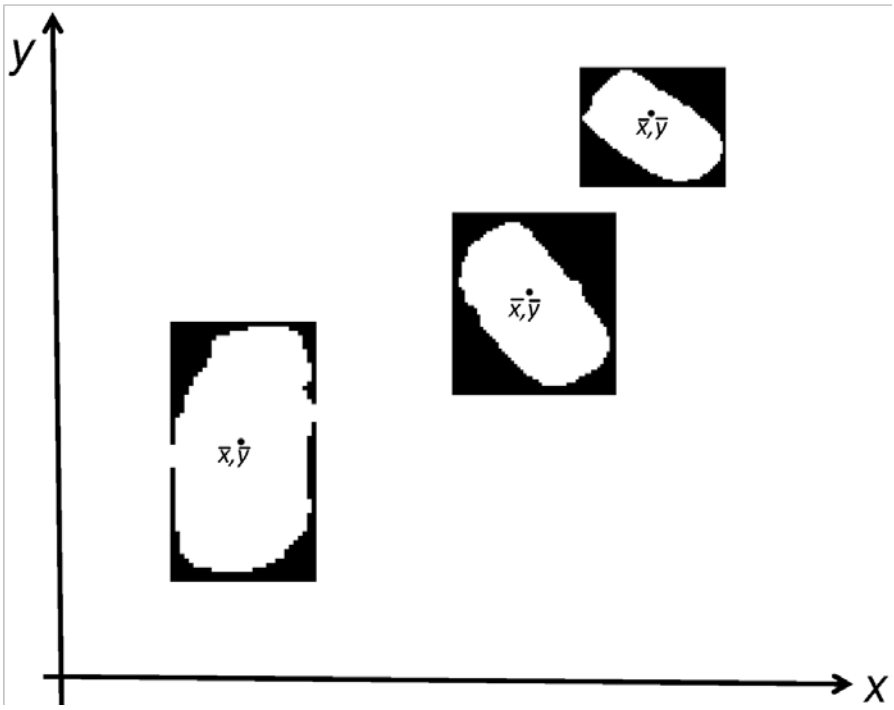


Figure 4.7 The centroid of each vehicle is estimated based on the detected vehicle regions

where x_1, x_2,…. x_n denote the points along the horizontal plane of the image and y_1, y_2,…. y_n denote the points along the vertical plane of the image. Figure 4.7 shows an example of the centroid of vehicle regions.

4.3.4 Orientation

Let $f_3(X_i)$ denote the orientation of region X_i. The orientation is determined by the bounding box of each vehicle region. Orientation is defined as the angle in degrees between the x axis and major axis of the ellipse that has the same second moments as region X_i. Figure 4.8 illustrates the orientation of a vehicle region. Figure 4.8 (*left*) shows a vehicle region and its corresponding ellipse. Figure 4.8 (*right*) shows the same ellipse, with features indicated graphically; the solid black lines are the axes. The orientation is given by the angle between the horizontal dotted line and the major axis of the ellipse.

The expression for $f_3(X_i)$ is given by

$$f_3(X_i) = \begin{cases} \frac{1}{2} cot^{-1}\left(\frac{a-c}{b}\right), & b \neq 0 \; and \; a < c \\ \frac{\pi}{2} + \frac{1}{2} cot^{-1}\left(\frac{a-c}{b}\right), & b \neq 0 \; and \; a > c \end{cases}$$

where a, b is the semi-length of the of the major axis and minor axis of the ellipse, respectively and $c = \sqrt{a^2 - b^2}$.

4.3.5 Luminance and Color

Let $f_4(X_i)$ and $f_5(X_i)$ denote the average luminance and average color of the region X_i. These two features are given by

$$f_4(X_i) = \overline{L^*}(X_i)$$

$$f_5(X_i) = \left(\overline{a^*}(X_i), \overline{b^*}(X_i)\right)$$

where $\overline{L^*}, \overline{a^*}, \overline{b^*}$ denote the average L^*, a^*, b^* measured in the CIE 1976 (L^*, a^*, b^*) color space (CIELAB). The value of L^* ranges between 0 and 100, while the values of a^* and b^* ranges between negative to positive values. Figure 4.9 shows the RGB image converted to L^* a^* b^* color space.

4.3.6 Feature Vector

All the above features discussed earlier are grouped together in a feature vector of a particular region X_i. The feature vector is given as $\boldsymbol{f}_t(X_i)$

$$\boldsymbol{f}_t(X_i) = [f_1(X_i), f_2(X_i), f_3(X_i), f_4(X_i), f_5(X_i)]$$

Examples of regions extracted from a frame and the corresponding feature values are shown in Figure 4.10.

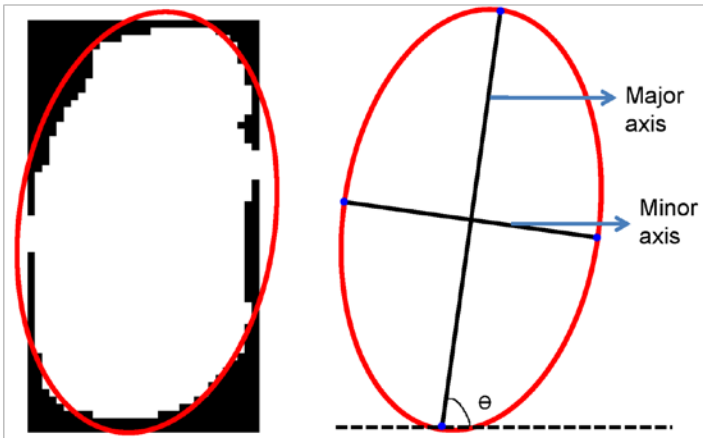


Figure 4.8 The orientation of each vehicle is estimated via ellipse-fitting on the detected vehicle regions

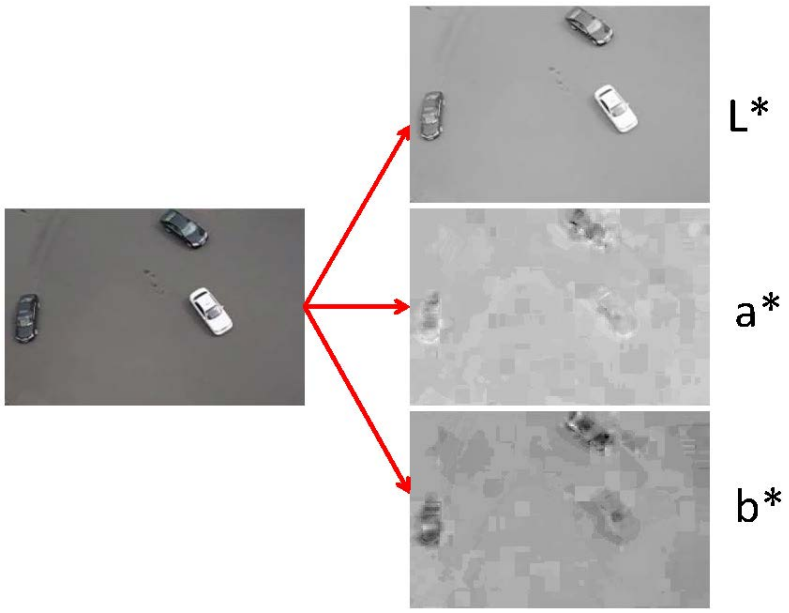


Figure 4.9 The lightness and color of each vehicle is estimated via an RGB to CIELAB color-space conversion on each frame followed by averaging the L*, a*, and b* values within each vehicle region

Features		**(a)**	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**	**(g)**
Area (*pixels*)		237	1208	217	56	220	1455	1382
Centroid (*pixel*)	$\bar{x}$	44	89	84	200	195	190	299
	$\bar{y}$	25	78	178	171	73	15	67
Orientation (*degrees*)		-22.5	-10.5	-15.7	57	44	-5.1	-13
Luminance		64	83.1	61	67.7	69.8	56.5	69.2
Color	$\overline{a^*}$	1.06	-0.1	-0.21	3.72	1.3	-0.4	0.4
	$\overline{b^*}$	-0.78	-0.4	1.26	-0.83	-0.15	-2.1	-0.2

Figure 4.10 Extracted vehicles from a particular frame and table of feature values

4.4 Human Visual System (HVS) Model Analysis

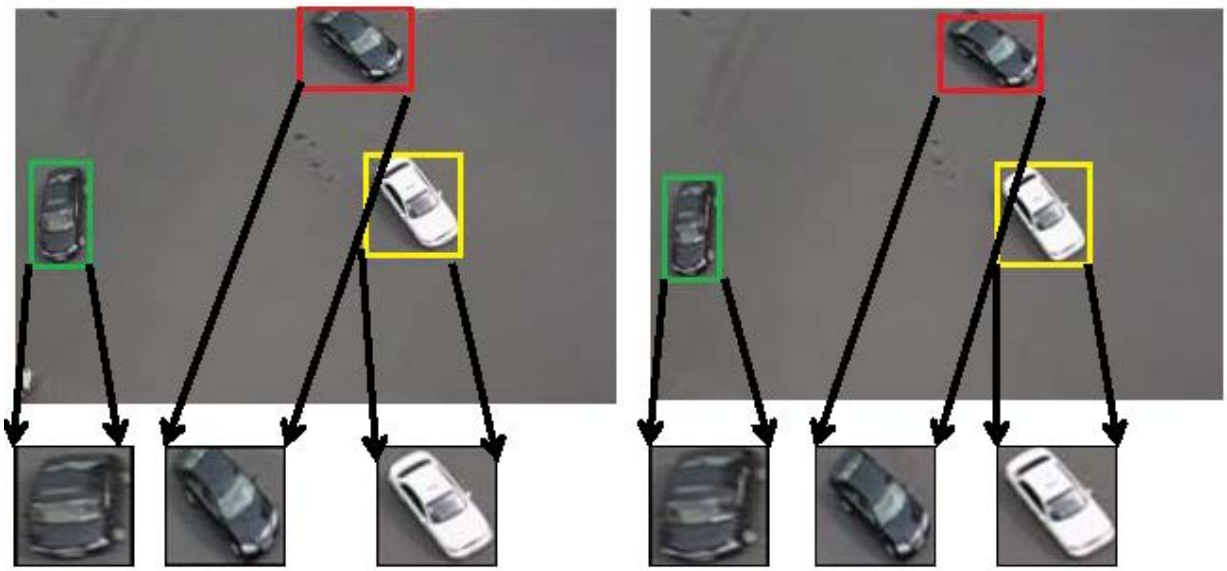
The features described in the previous section can assist in tracking vehicles across multiple frames. However, these features do not explicitly take into account the overall visual appearance of each vehicle as gauged by the human eye. To model this aspect, we employ a visual similarity estimator, called MAD (Most Apparent Distortion) [12]. Given two images or image regions, MAD will return an index which is proportional to how dissimilar the two images appear to a human observer. MAD operates by using a combination of a visual detection model and a visual appearance model. The detection-based stage models of the human contrast sensitivity function, luminance masking, and contrast masking to gauge subtle (near-threshold differences). The appearance-based stage employs a log-Gabor transform and local comparisons of log-Gabor coefficient statistics in an attempt to model the visual appearance of clearly visible differences. The MAD index is computed via a weighted geometric mean of these two model outputs.

Here, we use MAD to assist in tracking by searching for the vehicle in the next frame that mostly closely matches (i.e., yields the lowest MAD index) for a given vehicle in the current frame. Specifically, after the regions in the frames I_t and I_{t+1} are detected, they are subjected to MAD analysis. In this step each of the regions X_i $\{i= 1, 2, 3...\}$ frame I_t are compared one-by-one with each of the regions X_j $\{j= 1, 2, 3...\}$ in frame I_{t+1}. Thus the regions in I_t forms the first input to MAD algorithm and the regions in frame I_{t+1} form the second input to MAD algorithm.

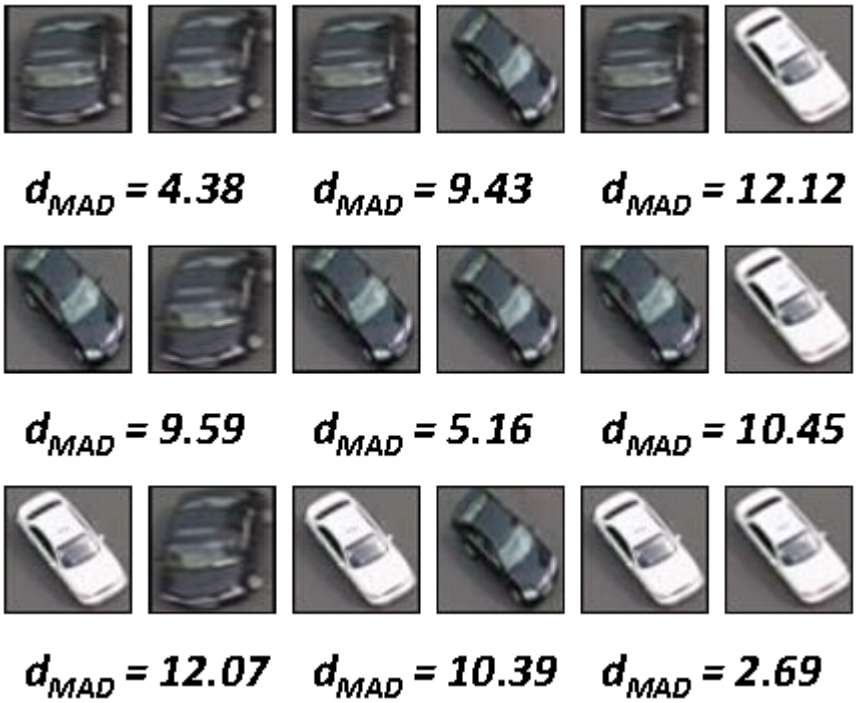
The output for each comparison is denoted by $d_{MAD}(X_i, X_j)$. If a particular region in frame I_{t+1} matches with a region in frame I_t, MAD should yield an index close to zero, meaning that the vehicles detected in frame I_{t+1} and I_t are the same. Examples of MAD analyses are shown in Figure 4.11 and Figure 4.12. The regions have been resized to a common size (of at least 64x64 pixels) as required by MAD. A lower MAD index denotes a closer visual match between the vehicles. As demonstrated in Figure 4.12, MAD can yield decent matching results even when some features of the regions are of deficient (such as color and luminance).

4.5 Vehicle Tracking

The tracking is done via corresponding via: (1) Searching for the region in frame I_{t+1} whose features most closely match the features of the given region in frame I_t and (2) searching for the region in frame I_{t+1}, with the lowest MAD index as compared to the given region in frame I_t.

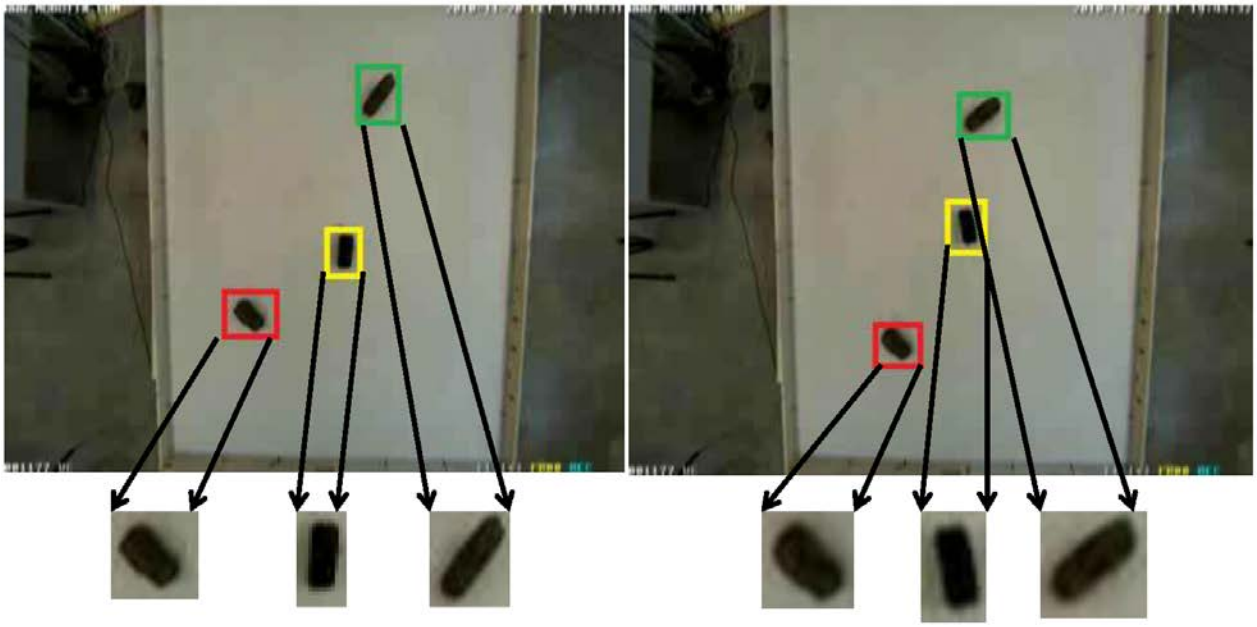


(a) (b)

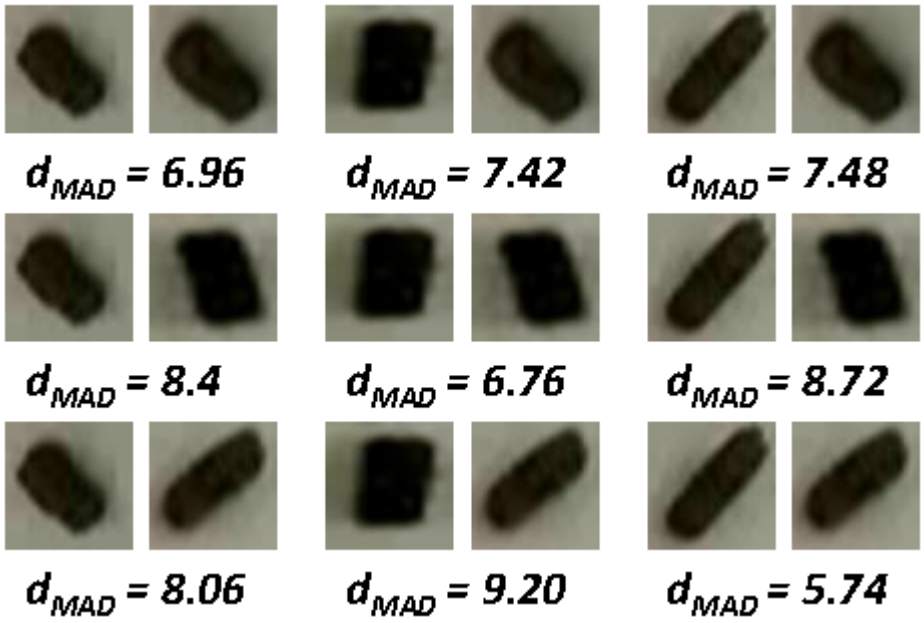


(c)

Figure 4.11. Example of HVS model analysis: (a) Frame at time *t*. (b) Frame at time *t*+1 (c) MAD index for different vehicle comparisons—smaller values denote closer matches.



(a) (b)



(c)

Figure 4.12 Example of HVS model analysis: (a) Frame at time *t*. (b) Frame at time *t*+1 (c) MAD index for different vehicle comparisons—smaller values denote closer matches

4.5.1 Feature Distance

In this step the feature vector of the regions extracted from frames I_t and I_{t+1} are used. For the purpose of tracking the vehicles accurately across each frame, we use the Euclidean distance between the feature vector of each region X_i $\{i= 1, 2, 3...\}$ in I_t and the feature vector of each region X_j $\{j= 1, 2, 3...\}$ in I_{t+1}.

Let $\boldsymbol{f}_t(X_i)$ denote the feature vector of the i^{th}_region extracted from frame I_t and let $\boldsymbol{f}_{t+1}(X_i)$ denote the feature vector of the j^{th}_region extracted from frame I_{t+1}. Let $d_{features}(X_i, X_j)$ denote the distance between $\boldsymbol{f}_t(X_i)$ and $\boldsymbol{f}_{t+1}(X_i)$, which is given by

$$d_{features}(X_i, X_j) = \sqrt{\left(\boldsymbol{f}_t(X_i,) - \boldsymbol{f}_{t+1}(X_j)\right)^2}$$

Thus, the smaller the value of $d_{features}(X_i, X_j)$, the more the two vehicle regions (X_i and X_j) match in terms of area, centroid, orientation, lightness, and color.

4.5.2 Weighed Combination of Feature Distance and MAD analysis

The above equation for $d_{features}(X_i, X_j)$ provides one measure of dissimilarity between vehicle regions in frame I_t and I_{t+1}. This dissimilarity measure can be improved by combining $d_{features}(X_i, X_j)$ with a MAD index $d_{MAD}(X_i, X_j)$ to compute an overall dissimilarity measure between the vehicle regions. For this purpose $d_{features}(X_i, X_j)$ and $d_{MAD}(X_i, X_j)$ are combined using weights. The overall similarity measure $d(X_i, X_j)$ is given by:

$$d(X_i, X_j) = \alpha d_{MAD}(X_i, X_j) + (1 - \alpha) d_{features}(X_i, X_j)$$

where α is the weighting factor. The value of α was empirically chosen to be 0.9. Finally the closest matching between vehicle regions in frame I_t and frame I_{t+1} is determined by searching for the minimum value resulting from the weighted combination output $d(X_i, X_j)$ and is given by

$$j^* = \arg min_j \left(d(X_i, X_j)\right)$$

Tracking is done typically between two consecutive frames I_t and I_{t+1}. Figure 4.13 shows results of this technique for two representative frames.

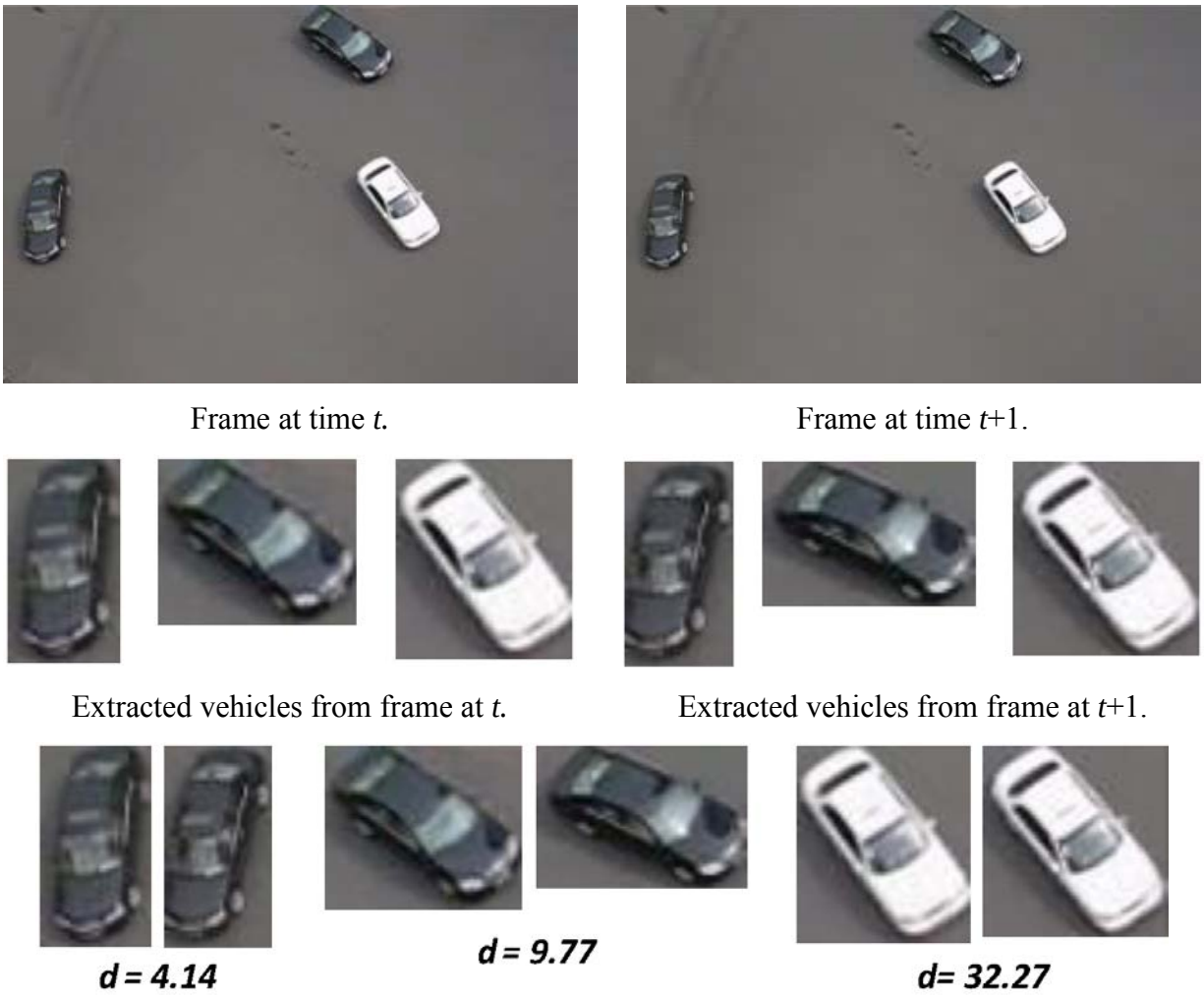


Figure 4.13 Overall measured *d* for matched vehicles in two consecutive frames

Since the tracking is done between two consecutive frames at a time, for rest of the instances of the system, the information of the vehicles in frame I_{t+1} are carried over to the next tracking step between frames I_{t+1} and I_{t+2}, and thus it is not to extract the vehicle information in frame I_{t+1}, since this information is already computed in the previous step, thus saving time and computation. In subsequent steps, the vehicle information in the upcoming frames (I_{t+2}, I_{t+3}, I_{t+4}, …) are extracted and are compared with the vehicle information obtained in the previous frames (I_{t+1}, I_{t+2}, I_{t+3}, ...) to track the vehicle as explained earlier. For the purpose of tracking, when the vehicle information computed in the previous frames (e.g., I_{t+1}) are carried over in the next step,

the vehicles in I_{t+1} are labeled in the order they occurred in the frame I_t and then compared with the vehicle information in the frame I_{t+2}, thus the system will be able to know which vehicle it is tracking. This process is repeated for the upcoming frames.

If a match for a particular vehicle region in frame I_t cannot be found in the frame I_{t+1}, the vehicle is assumed to have left the scene, and thus the vehicle is no longer tracked and its information is not carried over to the next step. Similarly, if a new vehicle region is detected in frame I_{t+1}, the features of the vehicle are extracted and used for tracking in the upcoming frames.

Specifically, when searching for the minimum index from the overall similarity measure $d(X_i, X_j)$, to find the matching vehicles between frames I_t and I_{t+1}, suppose a vehicle found in frame I_t has left the scene in frame I_{t+1}. In this case, the system still tries to find the closest matching vehicle in frame I_{t+1} corresponding to a vehicle region in frame I_t , which is not a true match of the vehicle in I_t since it has left the scene in I_{t+1}, but this scenario is unknown to the system. In order to overcome this situation, when the system finds a false match to a vehicle in I_t, the area and centroid position of the vehicle in search belonging to I_t is compared to the falsely matched vehicle in I_{t+1}. Since they are false matches the Euclidean distance between the area and centroid of these vehicles should be larger than some predefined threshold value. If this condition is satisfied, the system is informed that there was a false match and instructed that the vehicle under search has left the scene.

There may also be situations in which the Euclidean distance may be lesser than the threshold value. In these cases, other features such as the orientation and color information of the vehicles are used in addition to the area and centroid. Since the comparisons between the vehicles are done between consecutive frames, the area and centroid information of the vehicles are effective in finding the false matches in most of the cases. Similarly, when a new vehicle region is detected in I_{t+1} but not previously detected in I_t, the scenario of false matches cannot be encountered since the system searches for close matches only for the vehicles belonging to I_t in I_{t+1}.

Examples of tracked vehicles across nine frames are shown in the Figure 4.14. A colored rectangle is drawn around each vehicle. It can be seen that the color of the rectangle around each vehicle across the frames remains the same, indicating that the vehicles are detected and tracked correctly.

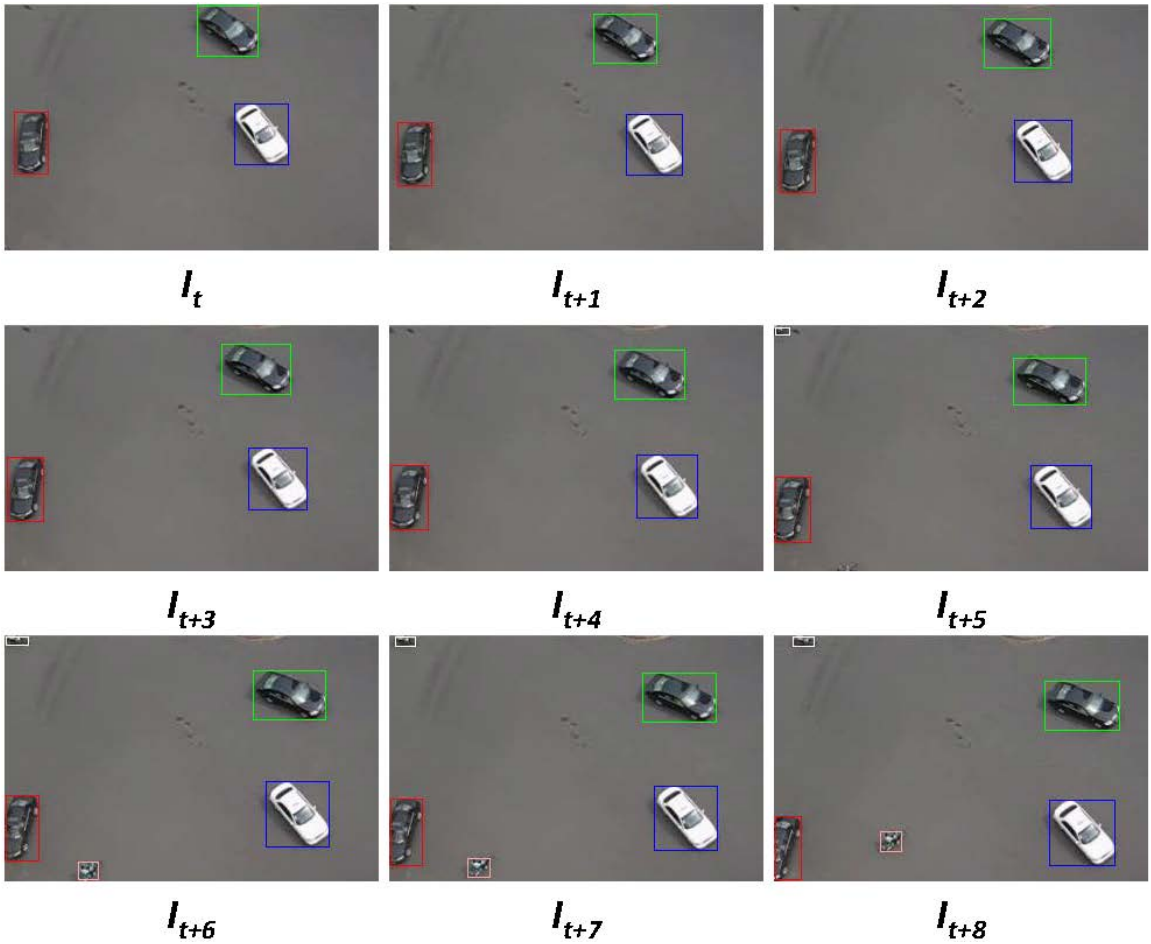


Figure 4.14 Vehicle tracking across nine consecutive frames. The color of each rectangle denotes the same vehicle across different frames

4.6 Computation of Vehicle Parameters

After the vehicles are detected and tracked, vehicle parameters such as speed and trajectory are computed using some of the features extracted during tracking.

4.6.1 Speed of the Vehicles

The speed of a particular vehicle region X_i in frame I_t is computed using the distance traveled by the vehicle in frame I_{t+1} and the frame rate of the video from which the image sequence are extracted. The distance traveled by the vehicle is computed using the centroid positions of the vehicle in I_t and I_{t+1}.

Let X_i denote a particular vehicle detected in I_t, and X_j denote the same vehicle detected in I_{t+1}, assuming the correspondence between the vehicles is determined using the vehicle tracking stage. The speed of a particular vehicle region X_i is given by:

$$Speed(X_i) = \frac{\sqrt{\left(\bar{x}(X_i) - \bar{x}(X_j)\right)^2 + \left(\bar{y}(X_i) - \bar{y}(X_j)\right)^2}}{\frac{1}{frame\ rate}}$$

The above equation gives the speed of the vehicle in terms of *pixels/sec*. In order to determine the speed of the vehicle in terms of real-world units (*miles/hr*), a camera calibration process is used. Based on the calibration mapping, the centroid positions of a particular vehicle in I_t and I_{t+1} are converted from pixel coordinates to real-world coordinates. From this step, the speed of the vehicle in terms of (*miles/hr*) is determined. A similar process is repeated for all the vehicle regions detected and tracked. As an example, for 15 frames/sec video, the speed of three vehicles (in pixels/second) are computed as follows:

$$Speed\ of\ vehicle\ (a) = \frac{3.16\ pixels}{\frac{1}{15} seconds} = 47.4\ pixels/sec$$

$$Speed\ of\ vehicle\ (b) = \frac{4.48\ pixels}{\frac{1}{15} seconds} = 67.2\ pixels/sec$$

$$Speed\ of\ vehicle\ (c) = \frac{2.83\ pixels}{\frac{1}{15} seconds} = 42.45\ pixels/sec$$

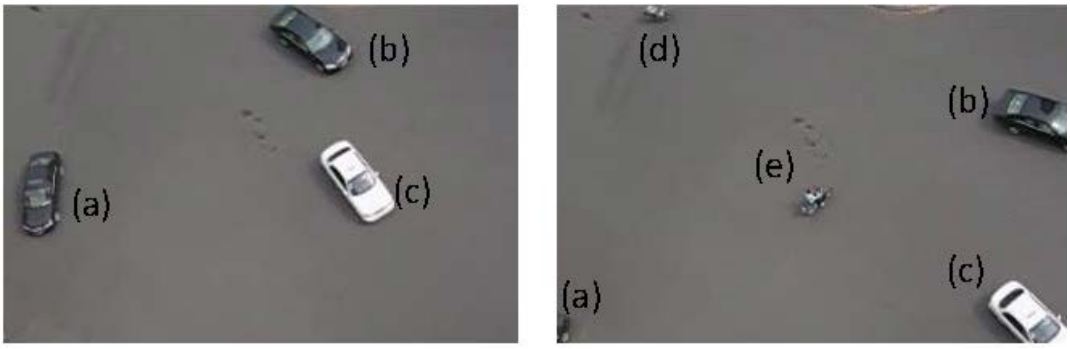
Since the camera is assumed to be parallel to the ground plane, a scaling factor was obtained by comparing the known vehicle width and height with the pixel width and height of the vehicle as found in the image sequence. The scale factor that relates pixel distance to real-world distance was approximately found to be (1 *pixel* $\simeq$ 0.00001 km). The speed of the vehicle is in terms of *miles/hr* is then given by:

$$Speed\ of\ vehicle\ (a) = 11\ miles/hr$$

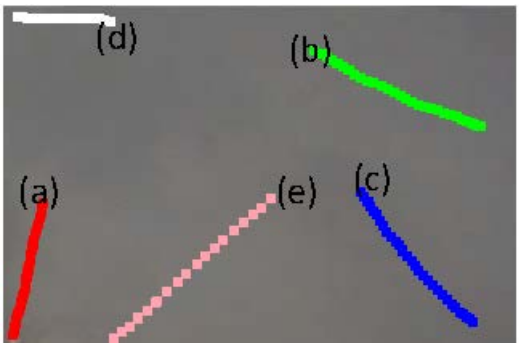
$$Speed\ of\ vehicle\ (b) = 15\ miles/hr$$

$$Speed\ of\ vehicle\ (c) = 10\ miles/hr$$

These speeds are typical of vehicles at traffic intersections. Therefore the assumption of the camera placed parallel to the ground plane to capture the video holds good in this case.



Frame at time *t*. Frame at time *t*+1.



Estimated trajectory for each vehicle.

Figure 4.15 Vehicle trajectories are estimated by connecting the centroids of tracked vehicles across multiple frames.

4.6.2 Trajectory of the Vehicles

Similar to finding the speed of the vehicles, the trajectories of the vehicles are also estimated using the centroid information of the vehicles in frame I_t and I_{t+1}. A line fit is made connecting the centroid of a particular vehicle detected and tracked correctly in I_t and I_{t+1} from the instant the vehicle enters the scene until it leaves the scene. Figure 4.15 illustrates the vehicles' trajectories obtained for frame I_t to I_{t+n}.

4.7 Accident Detection System

After the parameters are extracted, the next step is to determine the occurrence of an accident by using the parameters. We employed the algorithm of Ki and Lee [13] for the implementation of the accident detection system. Figure 4.16 shows a flowchart of the accident detection algorithm.

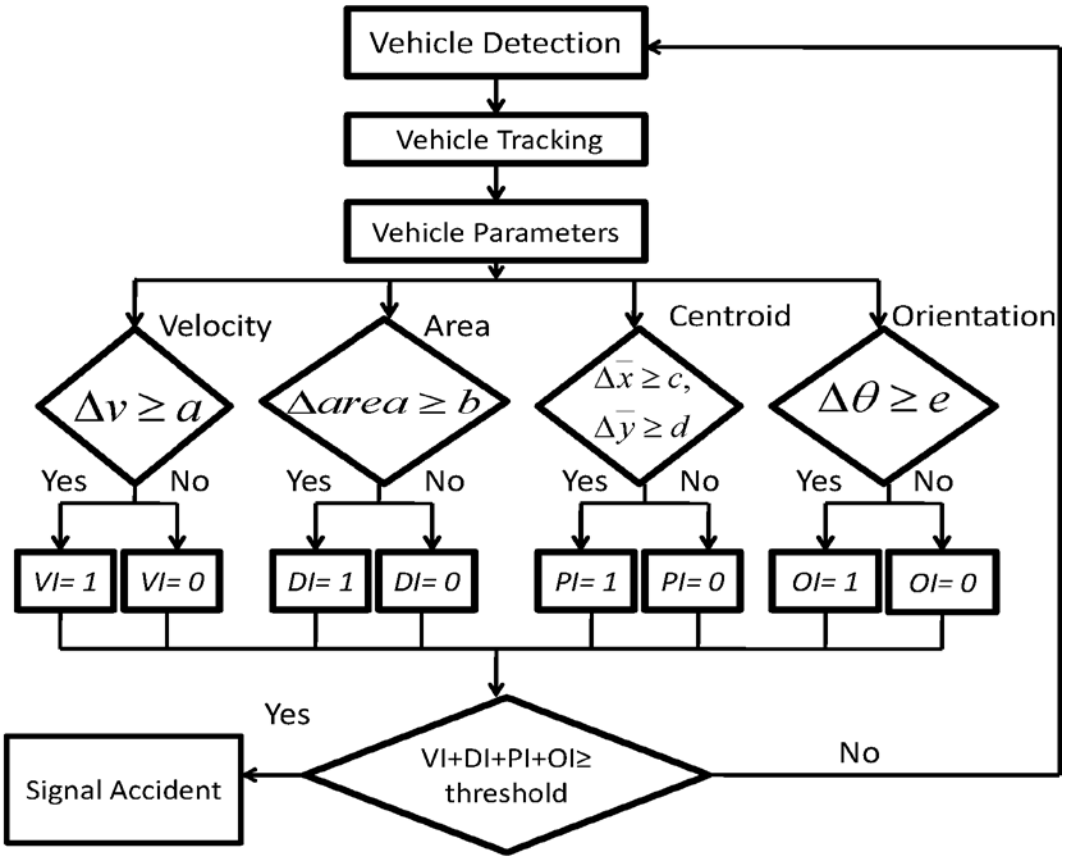


Figure 4.16 Flowchart of the Accident Detection Algorithm

4.7.1 Variation in Speed of the Vehicles

The speed of a vehicle is an important factor toward determining the occurrence of crashes at a traffic intersection. A rapid change in speed is a useful descriptor for a traffic accident. For example, if a particular vehicle travels with a particular velocity, after an occurrence of an accident, there is rapid change in the velocity. Therefore variation in the velocities of the vehicles across frames is used as a factor for estimating the occurrence of crashes by the system. In the accident detection system, vehicles are detected and tracked correctly and their velocity information is extracted at each frame the vehicle occurs.

After successful tracking of a vehicle in two consecutive frames I_t and I_{t+1}, the velocity information of the tracked vehicle obtained from I_t and I_{t+1} is compared with that obtained from I_{t-1} and I_t. Since it is assumed that the vehicles moves at an approximately constant velocity, if a vehicle crashes with another vehicle in frame I_{t+1}, the velocity of the vehicles is expected to go down drastically. So when the velocity of the vehicle determined in I_{t+1} is compared with that

obtained in I_t, there should be a large difference in the velocity of the vehicle indicating that a crash has occurred. Thus, to determine the occurrence of an accident, the difference in velocity of vehicles obtained between two consecutive frames is compared with a predefined threshold: $Change\ in\ velocity = \Delta v = v\ at\ I_{t+1} - v\ at\ I_t$. The following expression is used for the traffic accident detection algorithm:

$$VI = f(x) = \begin{cases} 1, & if\ \Delta v \geq a \\ 0, & otherwise \end{cases}$$

where *VI* is the velocity index and a is the speed threshold. Figure 4.17 shows a scenario for accident detection.

4.7.2 Variation in Area of the Vehicles

A rapid change in the area of the vehicles is also used to aid in accident detection. When an accident occurs, two vehicles come into contact and there is a possibility that the bounding box of the vehicles may intersect; in this case, there is a rapid change in the area of the vehicles detected. To detect accidents, the area of the vehicles detected and tracked in It and It+1 are compared; if the change in area of the vehicles exceeds an area threshold, then there may be possibility of accident: Change in area= Δarea=area at I_(t+1)- area at I_t. Thus, the following expression is used as a factor for traffic accident detection:

$$DI = \begin{cases} 1, & if\ \Delta area \geq b \\ 0, & otherwise \end{cases}$$

where *DI* is the area index and *b* is the area threshold. Figure 4.18 shows a scenario for accident detection using the change in area.

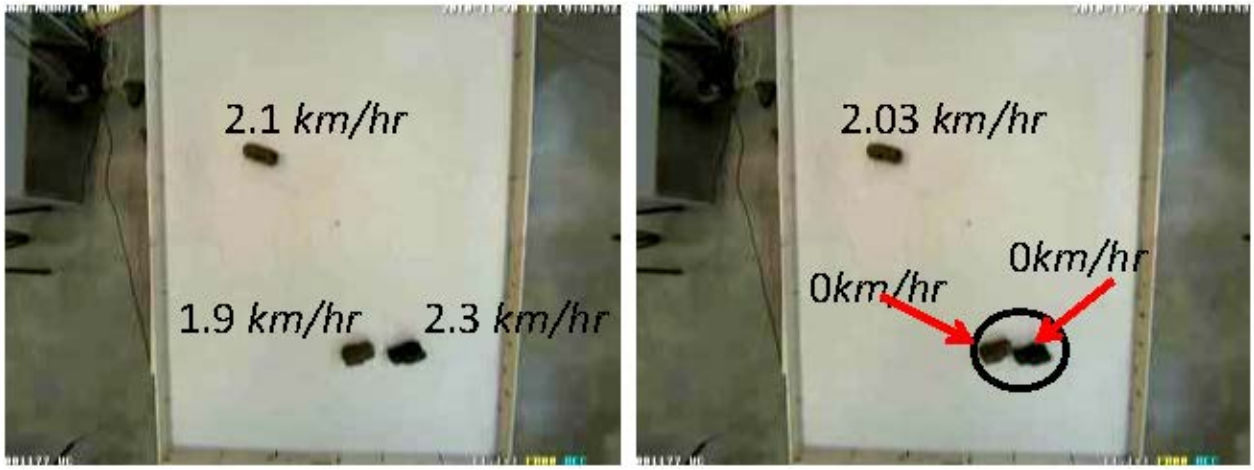


Figure 4.17 Illustration of accident detection by identifying rapid changes in speeds

Figure 4.18 Illustration of accident detection by identifying the change in area that occurs when two vehicles are detected as a single, combined vehicle

4.7.3 Variation in Position of the Vehicles

Change in the centroid position of the vehicles in frames I_t and I_{t+1} can be used as a factor to determine the occurrence of accident. As with the change in area, when an accident occur,s the bounding boxes of two vehicles intersect, causing a change in the estimated positions of the vehicles. Therefore a change in the centroid of a vehicle in consecutive frames can be used as a descriptor to determine the occurrence of an accident. The change in centroid is given by:

$$\Delta\bar{x} = \bar{x}\ at\ I_{t+1} - \ \bar{x}\ at\ I_t$$
$$\Delta\bar{y} = \bar{y}\ at\ I_{t+1} - \ \bar{y}\ at\ I_t$$

The following expression is used as a factor for traffic accident detection:

$$PI = \begin{cases} 1, & if\ \Delta\bar{x}\ \geq c, if\ \Delta\bar{y}\ \geq d \\ 0, & otherwise \end{cases}$$

where *PI* is the position index and *c,d* are thresholds.

4.7.4 Variation in Orientation of the Vehicles

Variation in orientation of the vehicles can also be used as a factor to determine the occurrence of an accident. As in the case of speed, area, and centroid, the orientations of a particular vehicle in frames I_t and I_{t+1} are compared: $Change\ in\ orientation = \ \Delta\theta = \theta\ at\ I_{t+1} - \ \theta\ at\ I_t$. The orientation index *OI* is given by:

$$OI = \begin{cases} 1, & if\ \Delta\theta\ \geq e \\ 0, & otherwise \end{cases}$$

where *e* is the threshold for change in orientation of the vehicles.

4.7.5 Overall Accident Index

After computing the velocity, area, position, and orientation index of the vehicles, the overall accident index is determined by the sum of individual indices. The overall accident index is then compared with a preselected threshold to determine the occurrence of accident. If the accident index exceeds the threshold, then an occurrence of accident is signaled, otherwise the system determines that there is no accident and the process is repeated until an accident is detected. The overall accident index (*AI*) is given by:

$$AI = VI + DI + PI + OI$$

The occurrence of accident is determined by:

$$Signal\ Accident = \begin{cases} 1, & if\ AI \geq accident\ threshold \\ 0, & otherwise \end{cases}$$

The accident detection algorithm is summarized as follows:

1. Vehicle regions are detected in image frames.
2. Low-level features such as area, orientation, centroid, luminance and color of the detected vehicles are extracted.
3. Vehicles are tracked using the tracking algorithm.
4. Speeds of the tracked vehicles are calculated.
5. Velocity, Area, Position and Orientation Indexes are calculated.
6. Overall Accident Index is calculated using the sum of individual indexes and occurrence of accident is identified.

4.7.6 Locating the Accident

Upon the occurrence of an accident, the next step is to locate the point at which the accident has occurred. This information can be obtained by using the positions of the vehicles that were involved in the accident at a particular frame. By using this information, the end user is not only informed about the occurrence of the accident, but the user is also provided the point at which the accident has occurred within the recorded video clip. This location information can be used by the end-user's display software to direct the user's attention to the crash location. This location information can also be used to encode the recorded video, for example, via region-of-interest encoding to facilitate rapid transmission of the video over lower-bandwidth networks (e.g., 3G networks).

4.8 Improving Matching Robustness via F-MAD

To better deal with motion blur and shadows, we recently developed a new feature-based image quality assessment algorithm. Our goal was to use features which are robust to changes in shape and lighting, and are therefore less sensitive to occlusions and shadows which can degrade the vehicle detection performance.

We specifically developed a feature-based variant of MAD—called *F-MAD*—which uses five feature maps: sharpness, luminance, edge strength, color distance, and contrast. All five

features maps are combined in order to accomplish the work of MAD by calculating the peak signal to noise ratio between feature maps of the two to-be-compared vehicle images. The following subsections described the computation of the features. In this discussion, let X denote an image containing a vehicle, and x denote a block of X. Let $f_i(x)$ denote a feature measurement for block x, and let $f_i(X)$ denote the corresponding feature map.

4.8.1 Lightness and Color Distance

Let $f_1(x)$ denote the Euclidean distance between the average lightness of block x and the average lightness of the image. Let $f_2(x)$ denote the Euclidean distance between the average color of block x and the average color of the image. These two features are given by:

$$f_1(x) = |\bar{L}^*(x) - \bar{L}^*(B)|$$

$$f_2(x) = \sqrt{[\bar{a}^*(x) - \bar{a}^*(B)\,]^2 + [\bar{b}^*(x) - \bar{b}^*(B)\,]^2}$$

where $\bar{L}^*$, $\bar{a}^*$, $\bar{b}^*$denote the average L*, a*, b* measured in the Commission Internationale de i'Eclairage (CIE) 1976 (L*, a*, b*) color space (CIELAB). Let R', G', B' denote the nonlinear RGB values of the image, the conversion from RGB color space to L*a*b* is implemented by first linearizing the R', G', B' values to be proportional to light energy, assuming sRGB values:

$$A = \begin{cases} \dfrac{A'}{12.92}, & A' < 0.04045 \\ [(A' + 0.055)/1.055]^{2.4}, & A' > 0.04045 \end{cases}$$

where $A = R$, G, or B.

The linearized R, G, B values are then converted to the CIE XY Z color space as:

$$X = 0.412453 * R + 0.357580 * G + 0.180423 * B,$$
$$Y + 0.212671 * R + 0.715160 * G + 0.950227 * B,$$
$$Z = 0.019334 * R + 0.119193 * G + 0.950227 * B.$$

Finally the L*, a*, b* values are given by

$$L^* = 116 * g\left(\frac{Y}{Y_r}\right) - 16,$$

$$a^* = 500 * [g(X/X_r) - g(Y/Y_r)]\,,$$

$$b^* = 200 * [g(Y/Y_r) - g(Z/Z_r)],$$

where X_r = 0.950456, Y_r = 1, Z_r = 1.088754 are the CIE *XY Z* tristimulus values of the *D*65 reference white point; and the function g is given by:

$$g(t) = \begin{cases} t^{1/3}, & t > 0.008856, \\ 7.787 * t + \frac{16}{116}, & otherwise \end{cases}$$

4.8.2 Contrast

Local contrast can also be an important factor which influences an image's visual appearance. To measure this, we first convert the image into the luminance domain. Then, the root mean square (RMS) contrast of each block is given by ratio of standard deviation to the mean of pixel values of the respective block. The result is a map in which each value represents local contrast.

Specifically, let $l(x)$ denote the luminance of an image block x. The contrast feature, denoted by $f_3(x)$, is given by:

$$f_3(x) = \begin{cases} \sigma_{l(x)}/\mu_{l(x)}, & \mu_l(x) > 0 \\ 0, & \mu_l(x) = 0 \end{cases}$$

where $\sigma_{l(x)}$ and $\mu_{l(x)}$ denote the standard deviation and mean of $l(x)$, respectively.

4.8.3 Edge Strength

To quantify similarity between object boundaries, we use maps of local edge strength. First, edges are detected by using Robert's edge detector. Then the edge strength of each block is computed by averaging the number of detected edge pixels within that block. The result is a map in which each value represents local edge strength.

Specifically, let $f_5(x)$ denote the edge strength of block x. Let E denote the binary edge map computed by running the Roberts edge detector on the entire frame. The feature $f_5(x)$ is then given by:

$$f_5(x) = \mu_{E(X)} = \frac{1}{m^2}\sum_j e_j$$

where $E(x)$ is the corresponding block of x in E, and e_j is a pixel of $E(x)$.

4.8.4 Sharpness

The sharper an image the better is its quality. If the image is blurred, we are not able to clearly distinguish between neighboring objects. Blurring also reduces the ability to visually recognize objects. Thus, sharpness can prove to be a useful feature for estimating image similarity.

For measuring local sharpness, we employ our own S3 sharpness map algorithm [14] in which local sharpness is measured in both the frequency domain and the spatial domain. In the frequency domain, the image is divided into 32x32 pixel blocks with 75% overlap. We then measure the slope of the power spectrum averaged across all orientations. In the spatial domain, we divide the image into 8x8 pixel blocks and measure local total variation. The two sharpness measurements are then combined via a geometric mean. The result is a map in which each value represents local sharpness.

4.8.5 Computation of F-MAD

For determining image similarity between two images X_1 and X_2, we first compute the five feature maps for both the reference image and the distorted image. Next, we compute the peak signal to noise ratio (PSNR) between the feature maps of the two images. We also compute the linear correlation coefficient between the corresponding maps from the two images. Finally, we multiply correlation coefficients with corresponding PSNRs and then sum them up, as follows:

$$sumPSNR = \sum_{i=1}^{5} PSNR\big(f_i(X_1), f_i(X_2)\big)$$

where $f_i(X_1)$ and $f_i(X_2)$ denote the i^{th} feature map for images X_1 and X_2, respectively.

To compute the final image dissimilarity index, we combine $sumPSNR$ with the output from the original MAD algorithm via a weighted geometric mean:

$$F\_MAD = (d_{detect})^{\alpha}(d_{appear})^{\beta} sumPSNR^{-\gamma}$$

where d_{detect} and d_{appear} denote the outputs of MAD's detection-based and appearance-based stages, respectively. The parameters β and γ are given by $\beta = \frac{1-\alpha}{2}$ and $\gamma = 1 - \alpha - \beta$, where α is the blending parameter computed in the original MAD algorithm (see [12]).

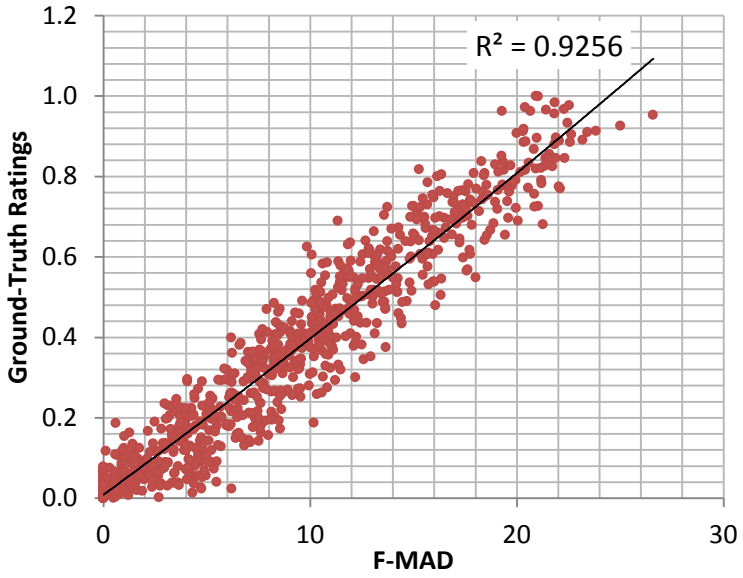


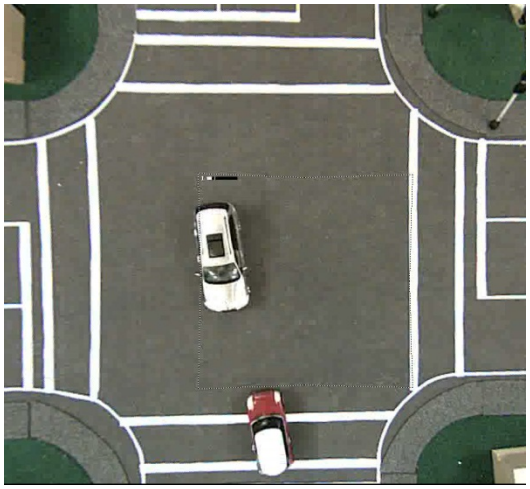
Figure 4.19 Scatterplot of ground-truth ratings of visual dissimilarity vs. F-MAD's predictions on image's from the CSIQ image database

Figure 4.19 shows a scatterplot of ground-truth ratings of visual dissimilarity vs. F-MAD's predictions on images from the CSIQ image database [12]. On this database, F-MAD is able to yield estimates of visual dissimilarity which correlate highly with the ground-truth data.

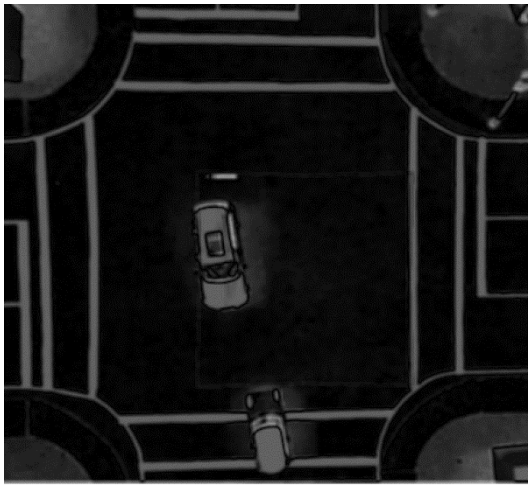
4.8.6 Results of F-MAD on Vehicle Tracking

Figure 4.20 shows the five feature maps obtained for a representative video frame obtained from our indoor testbed. Figure 4.21 shows a frame corresponding to the occurrence of an accident. Here, because of the motion blur and presence of shadows, MAD was unable to correctly track both vehicles and was thus unable to detect the collision. F-MAD, on the other hand, was successful at tracking both cars, and thus the use of F-MAD allowed the system to correctly detect the accident. On average, F-MAD is able increase tracking accuracy by 10-15%.

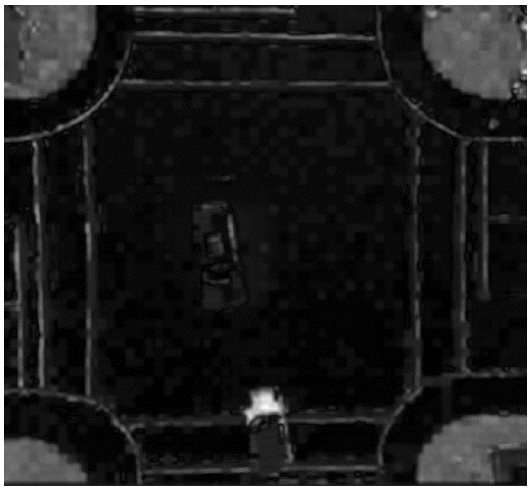
Yet, despite the improved tracking accuracy afforded by F-MAD, the algorithm is too computationally complex for a real-time implementation. We have devoted a considerable amount of research, in collaboration with a computer engineer, toward improving the computational efficiency of both MAD and F-MAD. Although we were able to accelerate MAD, the computation of the feature maps in F-MAD—particularly, the sharpness map—remains a



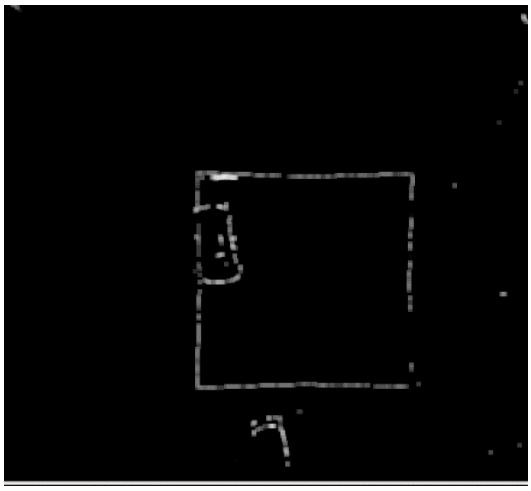
Original frame



Lightness Distance map

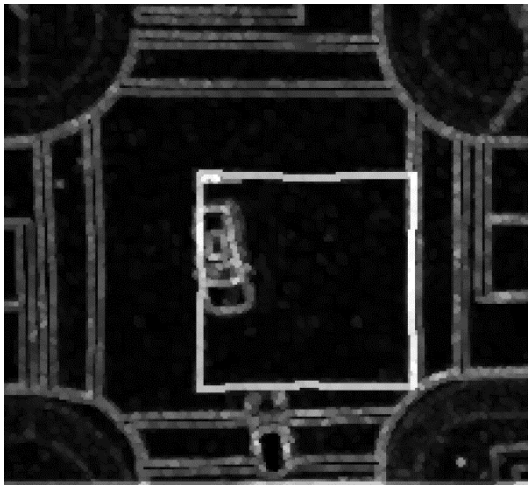


Color Distance map



Contrast map

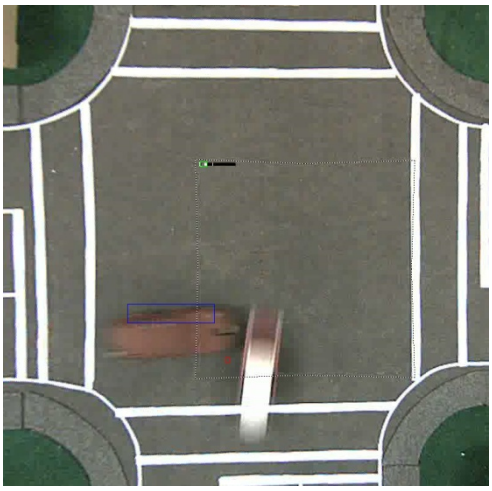
Edge Strength map



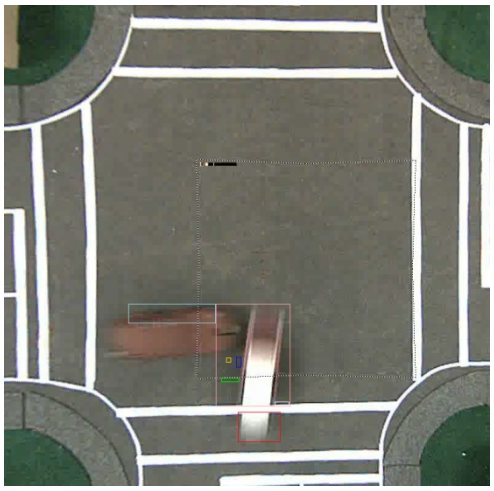
Sharpness map

Figure 4.20 Feature maps used in the F-MAD algorithm for determining vehicle matches

bottleneck in F-MAD. Currently, F-MAD requires 2-3 seconds per frame. Developing a real-time version of F-MAD remains a goal of our future research.



MAD tracking result



F-MAD tracking result

Figure 4.21 MAD vs. F-MAD tracking results. F-MAD is better able to handle shadows and motion blur

4.9 Improving Tracking via Kalman Filtering

The segmentation and tracking algorithm described in the previous sections uses only spatial information; no motion information deduced from multiple frames is used. To investigate the effectiveness of using such motion analysis, we implemented and evaluated a Kalman-filter-based algorithm, and outline of which is shown in Figure 4.22. The goal is to segment the foreground (vehicles) from the background while simultaneously tracking the foreground objects. We model the background statistically from the first few frames instead of mere background subtraction. The algorithm first loads the data and some parameters for background modeling (Nedler-Mead Optimization parameters for background adaptation) and tracking (Kalman Parameters). Based on the set parameters, the algorithm starts modeling the background. Next, the

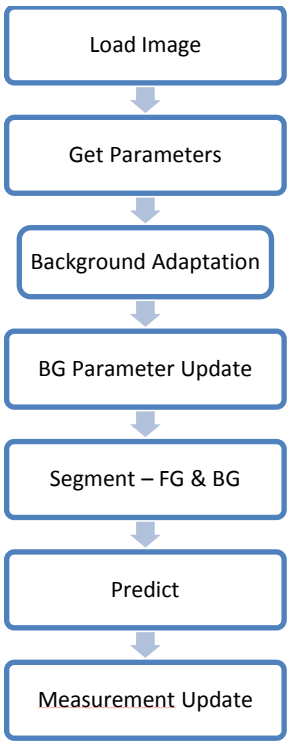


Figure 4.22 Stages of the Kalman-filter-based tracking algorithm

image is segmented into foreground and background. The foreground objects are then given to the Kalman filter for prediction and tracking. Finally, measurement of the actual positions of objects in the frame is performed, and tracking (Kalman) parameters are updated accordingly.

Demonstrative results of the Kalman Filtering algorithm are shown in Figure 4.23. We observed that separation of frame sequences into background and foreground starts by the second frame, and object detection and tracking occurs within three frames. (The number in the top-right corner of each frame in Figure 4.23 represents the frame number.) The implemented Kalman algorithm can be used for wide number of settings. It is largely independent of the specific environment because the background is modeled and continuously updated; thus, lighting and weather conditions have little impact on the algorithm. However, as with the F-MAD algorithm, the background modeling and prediction stages of the Kalman-filter-based algorithm are too computationally expensive to be practical, particularly when numerous vehicles must be tracked. Our attempts to accelerate this process via simplifications of the algorithm reduced tracking accuracy to the point that the benefits afforded by the algorithm were negligible. Developing a version of the Kalman-filter-based tracking which can operate in real-time, yet still yield acceptable tracking performance, remains a goal of our future research.

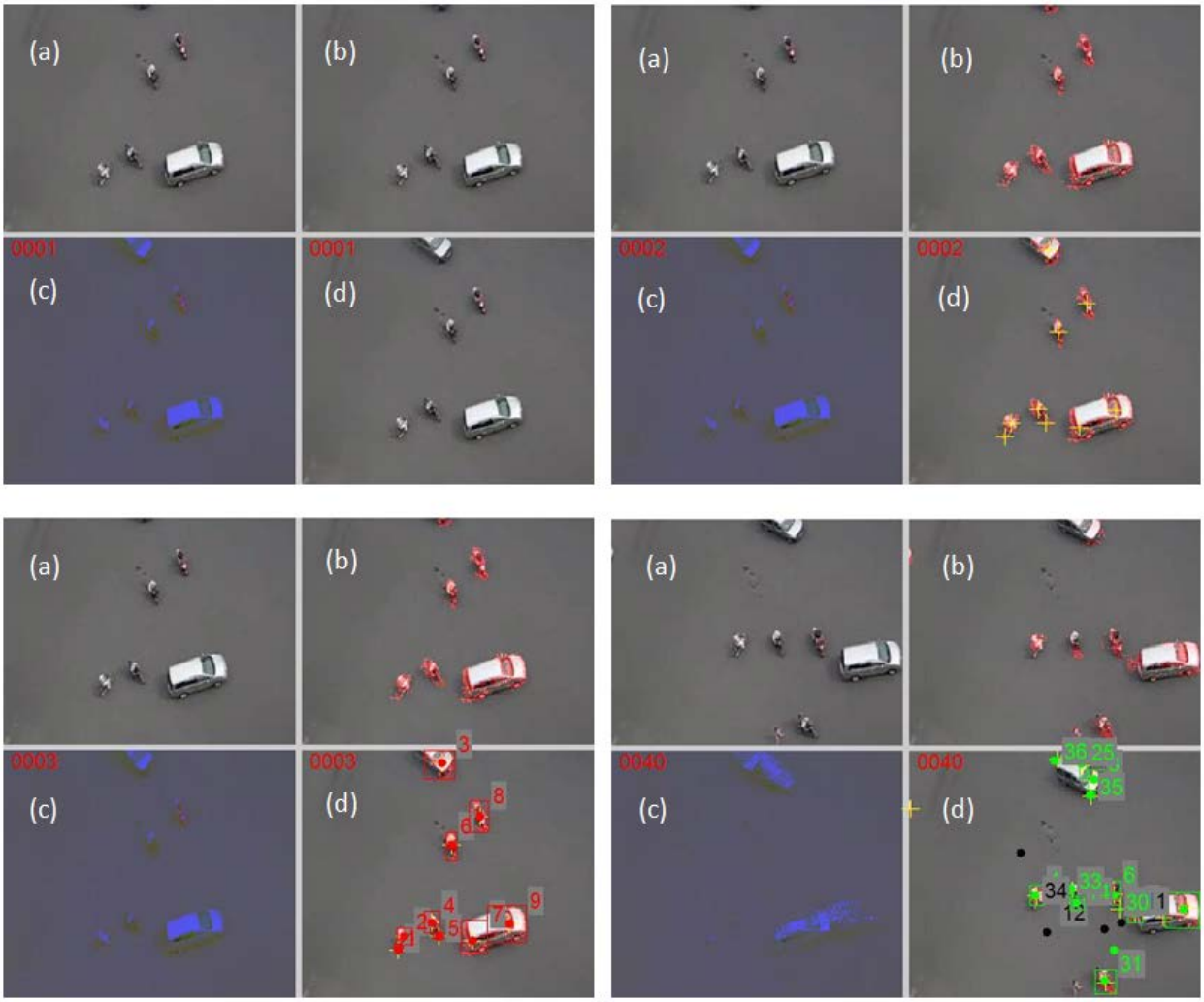


Figure 4.23 Results of Kalman Filtering algorithm. (a) Input Frame. (b) Segmented into background and foreground (marked red). (c) Image used for modeling background and foreground. (d) Tracking result.

CHAPTER 5

AUDIO BASED ACCIDENT ANALYSIS

Video data alone cannot guarantee high accuracy in an accident detection system in some cases, because it could be easily affected by occlusions, changes in illumination, and limitations of segmentation and identification algorithms [15]. As one of the most common sensing modalities, audio information is widely used in traffic monitoring systems [16]. Audio sensors, such as a simple microphone, are cost effective, and can act as a computationally efficient alternative detection device. Signals collected by the microphones contain important audio information when traffic accidents occur. In real world environments, crash sounds are usually mixed with other distinct sound sources in the monitored environment. For example, horn blaring or multiple collisions at different locations may occur at the same time.

In order to collect high-quality audio information and improve the performance of the collision detection system, an array of microphones can be used to replace a single microphone, due to the following advantages. Firstly, it may be electronically aimed to collect a high quality signal from a desired source. In this regard, a microphone array has the potential to outperform a single and highly-directional microphone. Secondly, a microphone array does not need local placement of transducers, which will not require any physical movement to alter its direction of reception. Besides, it has capabilities that a single microphone does not have, such as the automatic detection, localization, and tracking of sound sources in its receptive area. In traffic scene analysis, the coordinates of accidents may help emergency services locate potentially injured persons. Since traffic accidents are usually accompanied with a high-pitched sound resulting from sudden deceleration and/or a loud impact sound due to collision, these features can be extracted and used for traffic accident detection.

In this chapter, an audio processing system is presented to monitor traffic and identify potential collisions by collaborating with the video processing system described in the previous chapter. The system consists of a module for separation and localization of received audio signals at microphone arrays, and a module for crash sound detection of the separated signals. The whole system is summarized in Figure 5.1.

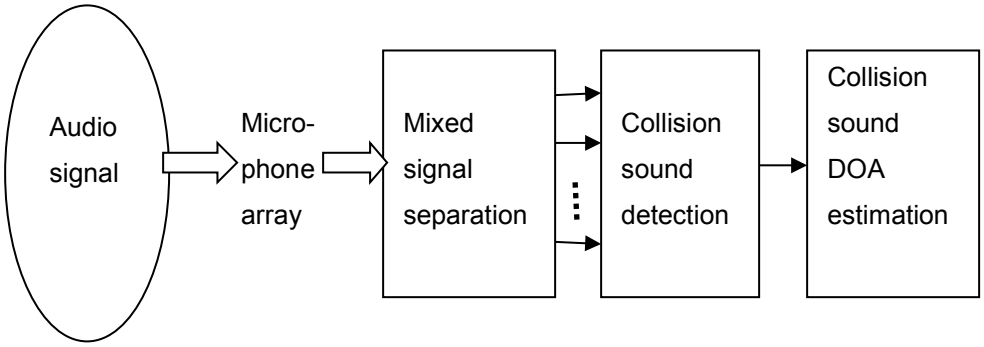
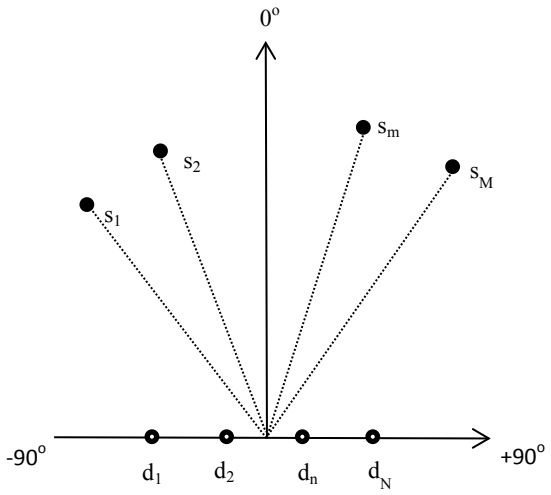


Figure 5.1 The diagram of the robust audio-based collision detection system

5.1 Blind Source Separation and Localization



2

Figure 5.2 Spatial configuration of sources and microphones

A linear array with N microphones is assumed and each microphone is with a known location d_n with respect to the center of the array. There are M audio sources, each with direction of arrival θ_m, $m = 1, \cdots, M$. Here, we deal with the overdetermined case, i.e., $M < N$. Figure 5.2 shows the spatial configuration of sources and microphones. Based on the central limit theorem (CLT), it's assumed that noise **n**(t) is zero mean additive white Gaussian noise across the microphones. The received signal for microphone n in an anechoic environment can be written as $x_n(t) = \sum_{m=1}^{M} a_{nm} s_m(t - \tau_{nm}) + n_n(t)$, where a_{nm} is the attenuation factor, τ_{nm} is the relative arrival lag between source m and microphone n, $\tau_{nm} = d_n \sin\theta_m / c$, and c is the propagation velocity of

the sound in the medium. The direction orthogonal to the array is 0 degree, and $\theta \in [-\pi/2, \pi/2]$. It is also assumed that a_{nm} is uniform for all source and microphone pairs, which is generally valid in a far filed. Therefore, $x_n(t) = \sum_{m=1}^{M} s_m(t - \tau_{nm}) + n_n(t)$.

Using a K-point short time discrete Fourier transform, we have

$$X_n(p, f) = \sum_{m=1}^{M} S_m(p, f) e^{-j2\pi f \tau_{nm}} + N_n(p, f)$$

where p is time frame index and f represents the frequency value.

$$S_m(p, f) = \frac{1}{K} \sum_{k=0}^{K-1} s_m(p, k) e^{-j2\pi k \frac{f}{K}},$$

$$N_n(p, f) = \frac{1}{K} \sum_{k=0}^{K-1} n_n(p, k) e^{-j2\pi k \frac{f}{K}},$$

$f = F_s(k-1)/K$, and k is the time index in a frame, and F_s is the sampling frequency. Due to the linearity of DFT, the noise is additive in the frequency domain. At one particular frequency, the noises at different microphones are uncorrelated. Moreover, signals and noises at this frequency are uncorrelated. The model can be written in a compact form:

$$\mathbf{X}(p, f) = \mathbf{A}(f)\mathbf{S}(p, f) + \mathbf{N}(p, f),$$

where $\mathbf{S}(p, f) = [S_1(p, f), S_2(p, f), \cdots, S_M(p, f)]^T$,

$\mathbf{X}(p, f) = [X_1(p, f), X_2(p, f), \cdots, X_N(p, f)]^T$, and $\mathbf{A}(f)$ is a $N \times M$ matrix with each column $\mathbf{a}(\theta_m) = [e^{-j2\pi f d_1 \sin\theta_m / c}, e^{-j2\pi f d_2 \sin\theta_m / c}, \cdots, e^{-j2\pi f d_N \sin\theta_m / c}]$.

The frequency-domain blind source separation separates the $\mathbf{X}(p, f)$ for all the frequencies bins to recover $\mathbf{s}(t)$. That is,

$$\tilde{\mathbf{S}}(p, f) = \mathbf{W}(f)\mathbf{X}(p, f),$$

where $\tilde{\mathbf{S}}(n, f) = [\tilde{\mathbf{S}}_1(p, f), \tilde{\mathbf{S}}_2(p, f), \cdots, \tilde{\mathbf{S}}_M(p, f)]^T$ is the recovered signal vector, and $\mathbf{W}(f)$ is the $M \times N$ demixing matrix. The main object here is to perform blind source separation and localization. That is, to get $\{\tilde{\theta}_m\}$, and the demixing matrix estimates $\tilde{\mathbf{W}}(f)$ for $f = 0, \cdots, F_s/2$. Finally, the inverse STFT is used to recover the source signals $\tilde{\mathbf{s}}(t)$.

5.1.1 Preprocessing

Before using subspace methods, the observed mixtures are normalized into zero mean and unit variance, which ensures that the signals at each frequency are also zero mean. It should be emphasized that although audio signals are generally non-stationary, a short duration of the signals are assumed to be approximately stationary. This is why subspace methods can be applied, which is corroborated by the results of computer simulation and real world experiments.

5.1.2 Subspace Methods

The covariance matrix of $\mathbf{X}(f)$ is $\mathbf{R}_{\mathbf{XX}}(f) = E\{\mathbf{X}(f)\mathbf{X}^H(f)\}$. In reality, it is approximated by $\mathbf{R}_{\mathbf{XX}}(f) = \frac{1}{P}\sum_{p=1}^{P}\mathbf{X}(p,f)\mathbf{X}^H(p,f)$ and $E\{\mathbf{X}(f)\} = \frac{1}{P}\sum_{p=1}^{P}\mathbf{X}(p,f)$, where P is the frame number. We can write

$$\mathbf{R}_{\mathbf{XX}}(f) = \mathbf{A}(f)E[\mathbf{S}(f)\mathbf{S}^H(f)]\mathbf{A}^H(f) + \mathbf{R}_{\mathbf{NN}}(f),$$

where $\mathbf{R}_{\mathbf{SS}}(f) = E[\mathbf{S}(f)\mathbf{S}^H(f)] = \frac{1}{P}\sum_{p=1}^{P}\mathbf{S}(p,f)\mathbf{S}^H(p,f)$.

The generalized eigenvalue decomposition is used to perform subspace computation as follows:

$$\mathbf{R}_{\mathbf{XX}}(f)\mathbf{V}(f) = \mathbf{R}_{\mathbf{NN}}(f)\mathbf{V}(f)\mathbf{\Lambda}(f),$$

where $\mathbf{V}(f) = [\mathbf{v}_1(f), \mathbf{v}_2(f), \cdots, \mathbf{v}_N(f)]$, $\mathbf{\Lambda}(f) = \text{diag}\{\lambda_1(f), \lambda_2(f), \cdots, \lambda_N(f)\}$, $\lambda_i(f) \geq \lambda_j(f)$, for $i > j$, and $\mathbf{v}_i(f)$ is the eigenvector corresponding to eigenvalue $\lambda_i(f)$. It is well-known that the largest M eigenvectors form the basis for the column space $R\{\mathbf{A}(f)\}$ of $\mathbf{A}(f)$ and the remaining $N-M$ eigenvectors form the basis of the orthogonal complement $R\{\mathbf{A}(f)\}^{\perp}$ of $R\{\mathbf{A}(f)\}$. The subspaces $R\{\mathbf{A}(f)\}$ and $R\{\mathbf{A}(f)\}^{\perp}$ are the signal subspace and noise subspace, respectively.

The ambient noise $\mathbf{N}(f)$ is almost omnidirectional and the correlation is small [17] . It is reasonable to assume that noise covariance matrix is $\mathbf{R}_{\mathbf{NN}}(f) = \sigma_f^2\mathbf{I}_M$, where σ_f^2 is an unknown constant for frequency f. Without loss of generality, we assume $\sigma_f^2 = 1$ for all frequencies.

Thus, the generalized eigenvalue decomposition becomes the standard eigenvalue decomposition $\mathbf{R}_{\mathbf{xx}}(f)\mathbf{V}(f) = \mathbf{V}(f)\mathbf{\Lambda}(f)$.

For arbitrary arrays, the MUSIC algorithm can be employed to estimate the DOAs. It computes the following pseudo-spectrum as a function of θ:

$$P_f(\theta) = \frac{\mathbf{a}_f^H(\theta)\mathbf{a}_f(\theta)}{\mathbf{a}_f^H(\theta)\mathbf{E}_N(f)\mathbf{E}_N^H(f)\mathbf{a}_f(\theta)},$$

where $\mathbf{E}_N(f) = [\mathbf{v}_{M+1}(f), \cdots, \mathbf{v}_N(f)]$ and $\mathbf{a}_f(\theta) = [e^{-j2\pi f d_1 \sin\theta/c}, e^{-j2\pi f d_2 \sin\theta/c}, \cdots, e^{-j2\pi f d_N \sin\theta/c}]^T$. The M largest peaks in the spectrum correspond to the M source directions. One drawback of the MUSIC algorithm is that it needs to compute the spectrum values for all directions, which results in huge computational burden. Also, the peak search algorithm further adds the computational cost.

Conversely, the ESPRIT algorithm directly gives DOA estimates after obtaining the signal subspace, while it applies only to a uniform linear array. It is conducted as follows [18]:

(1) Choose the eigenvectors corresponding to the M largest eigenvalues and form the matrix $\mathbf{S}_1(f) = [\mathbf{v}_1(f), \mathbf{v}_2(f), \cdots, \mathbf{v}_M(f)]$, $\mathbf{A}_1(f) = [\mathbf{I}_{N-1}\ \mathbf{0}]\mathbf{S}_1(f)$, $\mathbf{A}_2(f) = [\mathbf{0}\ \mathbf{I}_{N-1}]\mathbf{S}_1(f)$. We get the matrix $\boldsymbol{\mu}(f) = (\mathbf{A}_1^H(f)\mathbf{A}_1(f))^{-1}\mathbf{A}_1^H(f)\mathbf{A}_2(f)$, where $\mathbf{I}_{N-1}$ is the $N-1$ dimension identity matrix, and $\mathbf{0}$ is a $N-1$ dimension vector with all zero elements.

(2) It's known that the eigenvalues $\{\lambda_u(f)\}$ of $\boldsymbol{\mu}(f)$ correspond to $\{e^{j2\pi f d \sin\theta_1/c}, e^{j2\pi f d \sin\theta_2/c}, \cdots, e^{j2\pi f d \sin\theta_M/c}\}$. Therefore, the estimated DOAs can be computed according to $\arcsin\{\mathrm{Im}\{\ln(\lambda_u(f))c/(2\pi f d)\}\}$, where $\arcsin(\cdot)$ is the inverse sine function, $\mathrm{Im}(\cdot)$ gives the imaginary part of a complex number, and $\ln(\cdot)$ is the natural logarithm operator.

After having multiple DOA estimates $\tilde{\theta}_m(f)$ at various frequencies f, we will apply some rules to obtain final DOA estimates $\{\tilde{\theta}_m\}$.

5.1.3 Final DOA Determination

At each frequency, we have the DOA estimates of the sources. However, because of the differences in signal power, noise power and thus SNRs, the estimated DOAs can vary a lot.

Therefore, how to choose the frequencies with high SNRs and combine the estimates together is a crucial issue. Since high SNRs ensure better DOA estimates, we try to use the DOA estimates from frequency components with high SNRs. In reality, only mixtures are given, and thus true SNRs are unknown.

In simulations, noises are assumed to be additive white Gaussian. Therefore, noise power is almost equally distributed among different frequencies, while signal power is different at different frequencies. Thus, the mixtures' SNRs at different frequencies are generally proportional to the signal power at corresponding frequencies and thus to the mixture power. The mixture power at different frequencies here is represented by the sums of squared amplitudes (SSA) of the mixture spectrograms at corresponding frequencies. We choose the DOA estimates at the frequencies with high SSA values. However, the SSA's values are attributed to multiple source signals' contribution in signal power. Therefore, the separate SNRs may be quite different and so are their DOA estimates. We use the average of the DOA estimates or the weighted average normalized by the SSA values to mitigate this effect. In the experiments, noises are mostly at low frequencies and we choose relatively high frequencies in which noises are generally much less. The same method for simulations is then applied.

To summarize, we adopt the following rule: Firstly, use principle component analysis (PCA) to get M principle components at each frequency f. That is, we have $\bar{\mathbf{X}}(p, f) = S_1^H(f)\mathbf{X}(p, f)$, for $p = 1, \cdots, P$. Then, choose Q frequencies $\{f_q\}$, $q = 1, \cdots, Q$ with the largest P percent sum of squared amplitudes (SSA). Namely, SSA at frequency f: $SSA(f) = \sum_{p=1}^{P} \left\| \bar{\mathbf{X}}(p, f) \right\|^2$. Then, the final estimates are obtained by averaging the DOA estimates at these frequencies $\tilde{\theta}_m = \sum_{q=1}^{Q} \tilde{\theta}_m(f_q)$ or using the weighted average of the DOA estimates.

After getting the DOAs $\{\tilde{\theta}_m\}$, we can get $\tilde{\mathbf{A}}(f) = [a(\tilde{\theta}_1)\ a(\tilde{\theta}_2)\ \cdots\ a(\tilde{\theta}_M)]$. Demixing matrix $\tilde{\mathbf{W}}(f)$ for each frequency can be obtained by using the least square estimates with the constraint $\tilde{\mathbf{W}}(f)\tilde{\mathbf{A}}(f) = \mathbf{I}$. That is, $\tilde{\mathbf{W}}(f)$ is the pseudo-inverse of $\tilde{\mathbf{A}}(f)$. Then, we get $\tilde{\mathbf{S}}(p, f) = \tilde{\mathbf{W}}(f)\mathbf{X}(p, f)$ for all frequencies. Using the inverse DFT, we can recover the time domain source signals $\{\tilde{\mathbf{s}}(t)\}$.

5.1.4 Related Issues

Source Number Estimation

In the previous section, the number of sources is assumed to be known beforehand and smaller than the number of microphones. In fact, the source number can be determined by analyzing the eigenvalues $\{\Lambda_i(f)\}$ of the spatial correlation matrix of the mixtures at frequency f. That is, the number of dominant eigenvalues is equal to the number of sources. Therefore, a subjective threshold on the eigenvalues can be set to estimate the source number. Another approach is to use the information theoretical criteria. Akaike information criteria (AIC) [19] and minimum description length (MDL) [20, 21] are two common criteria for source number estimation. Theoretically, AIC is more likely to give an overestimation, while MDL gives unbiased estimation [22] .

Frequency Bin Selection

It is known in array signal processing that, for a broadband signal, its high frequency components prefer small array spacing, while its low frequency parts prefer large spacing. In reality, a given microphone array is fixed and signals of interest can be different. In [23], the authors have talked about using microphones with different spacing to handle different frequency ranges respectively. That is, considering that we need multiple microphones' observations to perform localization, we may select only a subset of microphones to perform the localization for lower frequency signals, as long as the source number is less than the number of chosen microphones. Therefore, a microphone array can handle a wider range of signals.

Microphone arrays sample signals in the space domain. Similar to the aliasing problem in time domain sampling, microphone array also experience spatial aliasing problem. The well-known Nyquist sampling theory tells that if we recover a signal with the highest frequency $f_{\max}$, we use sampling rate $2f_{\max}$ at least. In the spatial sampling, it requires that half of the minimum wavelength of a wideband signal should be larger than the interval of the array it impinges. To be more specific, for spacing d, the minimum wavelength it can capture is $d/2$ and therefore the maximum frequency is $2c/d$, where c is the velocity of sound in the medium. In our problem,

the phase delay between two microphones should be smaller than π in modulus, i.e., $|2\pi fd \sin\theta| \le \pi$. We only focus on the frequency range where no spatial aliasing occurs. On the other hand, if the frequency of a signal is too low and thus its wavelength is too long, the array can hardly capture the small amount of phase change of the sound signal. Therefore, a low cutoff frequency is also set. Although audio signals are naturally broadband, we can only consider some specific frequency range, at which the algorithm can estimate the DOAs more accurately.

Performance Measure

The performance measures for algorithm evaluation are the mean squared error (MSE) of DOA estimates and the signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR), and signal-to-artifacts ratio (SAR) for separation, which are firstly introduced in [24]. The MSE for source m at frequency f is computed as

$$\mathrm{MSE}_{m,f} = \frac{1}{I}\sum_{I}(\theta_m(f) - \theta_m)^2 ,$$

where I is the number of Monte Carlo runs. For separation, it's assumed that the recovered source signal can be decomposed as

$$\hat{s}_m = s_{\text{target}} + e_{\text{interf}} + e_{\text{noise}} + e_{\text{artif}} ,$$

where s_{target} is a modified version of s_m with allowed distortion, e_{interf}, e_{noise}, and e_{artif} are respectively the interferences, noise, and artifacts terms. The existing algorithms for recovered signal decomposition and SDR, SAR, and SIR computation can be found in [24].

5.1.5 Simulations

In the simulations, there are $N = 4$ microphones uniformly and linearly distributed with spacing $d = 0.05$. Two sources $M = 2$ are located at directions $\theta_1 = 40$ and $\theta_2 = 40$ degrees, respectively. Source signals include music and speech and they are normalized into signals with zero mean and unit variances. The velocity of sound in the air is $c = 340\text{m/s}$. The sampling rate is 16KHz. The noise is set to be additive white Gaussian noise (AWGN) with variance σ^2 across microphones. It is noted that for audio signals, the power at different frequencies are different, while for white noise, the power at different frequencies are ideally equal. The signal-to-noise ratio is set to be $10\log_{10}(M/\sigma^2)$. We only consider the frequency range without spatial aliasing.

That is, in our simulation, $f \le c/(2d)$. We run 10000 Monte Carlo runs for each SNR. Additionally, in all simulations, due to the inability of a microphone array to capture low frequency signals, we set a lower frequency threshold to be 1500 Hz. The parameter setting in simulations is summarized in Table 5.1.

Table 5.1 Parameter settings in simulations

Mixture Characteristics	Parameters to be Specified
Number of Sources M	2
Source Categories	Speech & Music
Source Length	4 Seconds
Source Angles	+/- 40 degrees
Noise Type	AWGN (sensor noise)
Number of Microphones N	4
Array Spacing d	0.05
Sampling Rate F_s	16KHz
Mixture Type	Pure Delay
Mixture Domain	Frequency Domain
Frame Length	256
Frame Shift	256
FFT Window	Rectangular
Monte Carlo Runs	10000

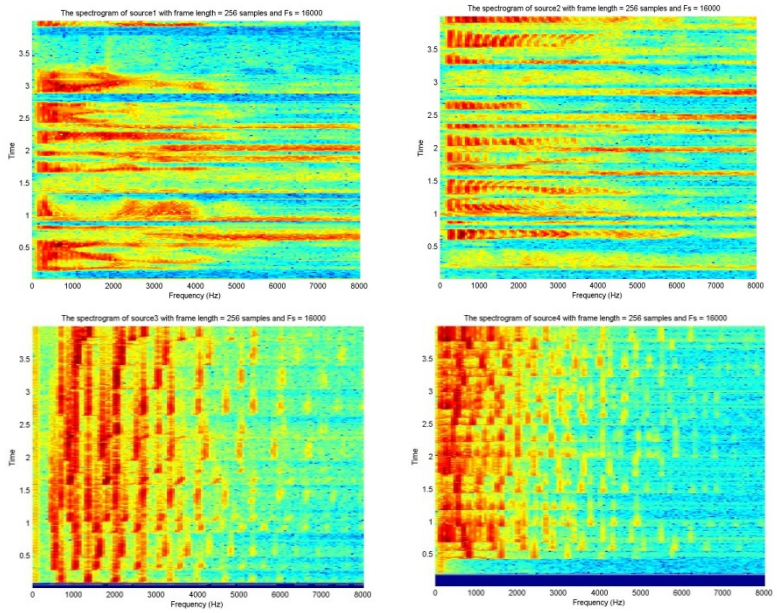


Figure 5.3 The spectrograms of different sources: source1: male speech (top left), source2: female speech (top right), source3: piano (bottom left), and source4: trumpet (bottom right)

Source Spectrograms

Audio signals' diversity in time-frequency characteristics can be illustrated by their spectrograms, which are the signals' amplitude values at particular time-frequency points. It's known that there is a tradeoff between time and frequency resolutions in spectrogram representation. That is, a high frequency resolution results in a low time resolution, and vice versa. Figure 5.3 shows the spectrograms of several source files with sampling rate 16KHz and frame length 256 samples, equal to 3.2 milliseconds. Source1 and source2 are respectively male and female speeches. Source3 and source4 are corresponding piano and trumpet music. They show typical time-frequency characteristics for each category. That is, speeches possess wide spectrum and short time duration, while music consists of harmonic frequencies.

Source Number Estimation

Eigenvalue based Method

Figure 5.4 shows the normalized eigenvalues of the correlation matrix at different frequencies for different source combinations at SNR = 30dB with frame length 256 samples. The plots are obtained by averaging the results over total 10000 runs. It is clear that for different sources, the whole trend of how eigenvalues changes with frequency is different. With SNR = 30 dB, the first two eigenvalues are much larger than the rest two eigenvalues for most of the frequencies. It's easy to set a subjective threshold to decide the source number. That is, when the signal power is dominant in the mixture, using eigenvalue analysis to estimate source number is suitable.

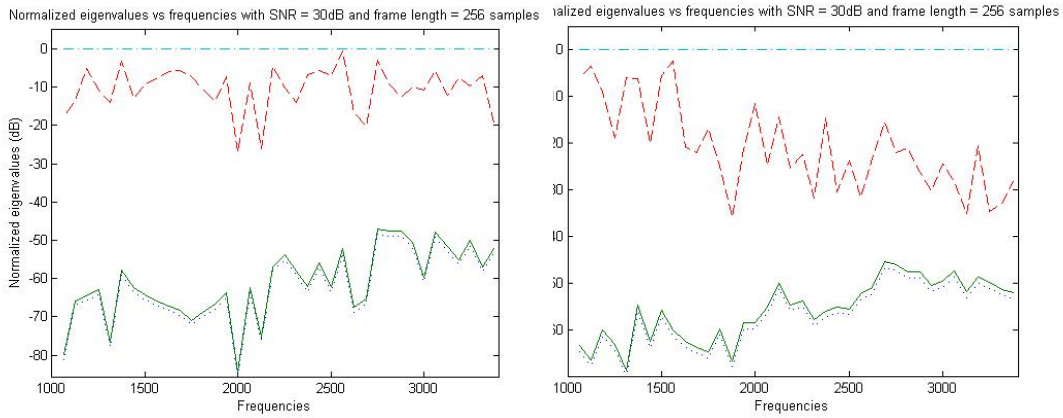


Figure 5.4 Normalized eigenvalues versus frequencies for different source combinations with SNR = 30 dB for mixture of source1 and source3 (left) and mixture of source2 and source4 (right)

Information Theoretical Criteria

Now we talked about information theoretical criteria, which include Akaike information criterion (AIC) and minimum description length (MDL). Figure 5.5 shows the percentage of correct source number estimates in total 10000 runs at SNR = 30dB using AIC and MDL. It is clear that the source estimation using MDL is 100 percent accurate and better than the estimation using AIC.

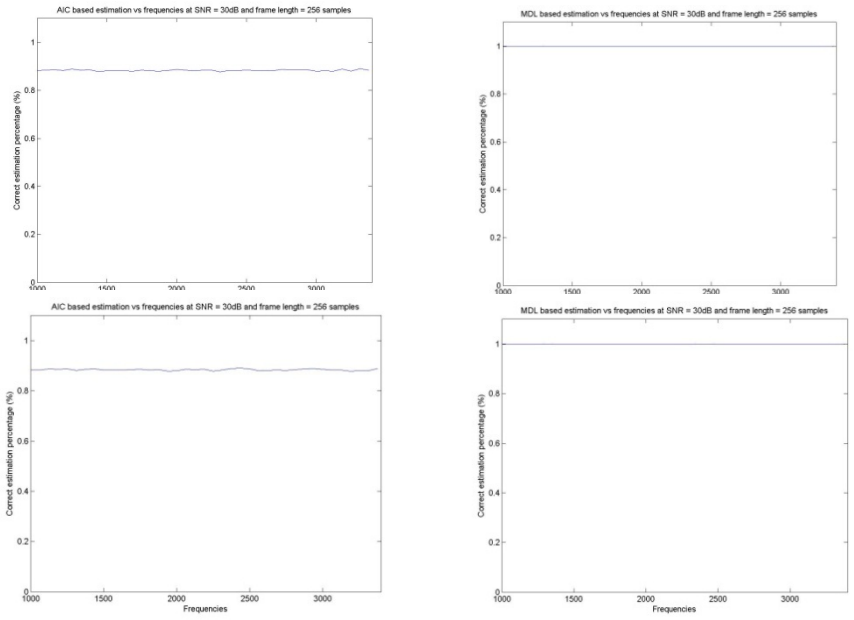


Figure 5.5 Correct estimation percentages versus frequencies with SNR = 30dB using AIC and MDL for mixture of source1 and source3 (top two) and mixture of source2 and source4 (bottom two)

Localization Performance

Figure 5.6 shows the MSE with respect to different frequencies using 10000 Monte Carlo runs with frame length 256 samples for different SNRs using mixture of source1 and source3. It is obvious that with higher SNRs, the MSE performance is much better. For different source files, the patterns of MSE versus frequency curves are very different. This is likely to be related to the SNR difference of different signals at different frequencies and at different time locations. In other words, it is because of the differences in time frequency characteristics for different source files.

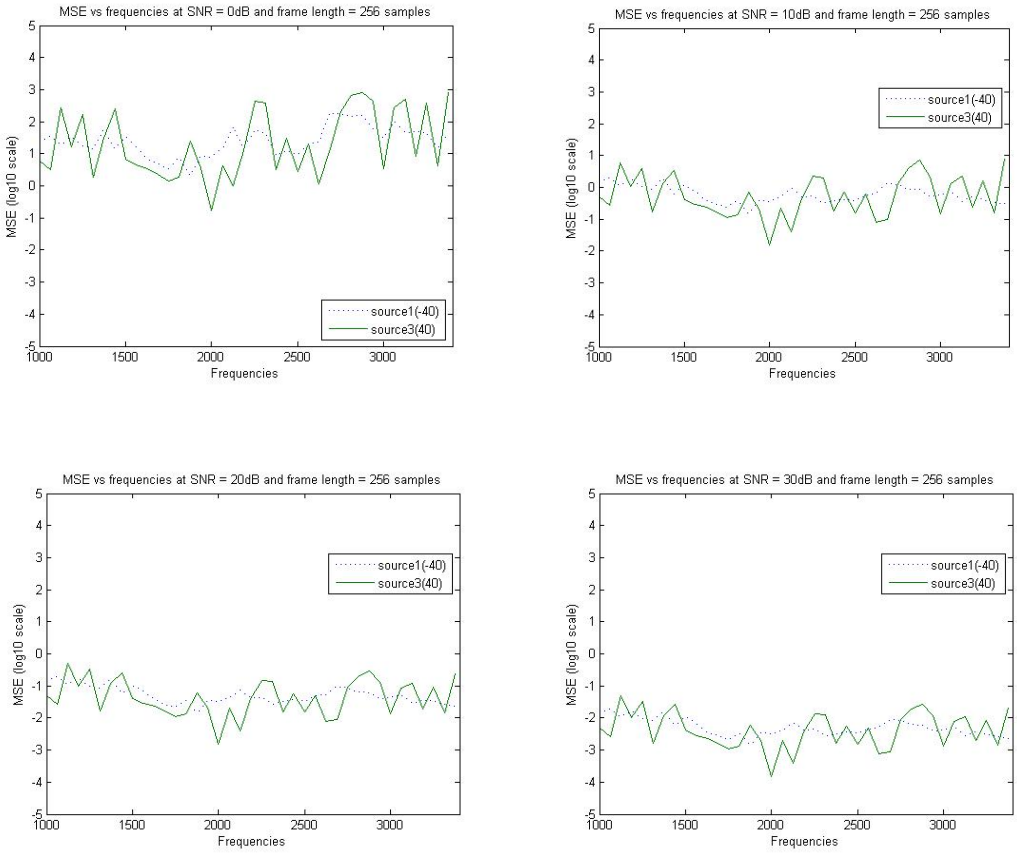


Figure 5.6 MSE versus frequencies for different SNRs with frame length 256 samples using mixture of source1 and source3

Frequency Bin Selection for DOA Estimation

Figures 5.7 and 5.8 show the MSE of DOA estimates versus SNRs using the average and weighted average of the DOA estimates at the frequencies with the largest 30% percent SSA. The results using two methods have little difference and the MSE monotonically decreases with an increasing SNR.

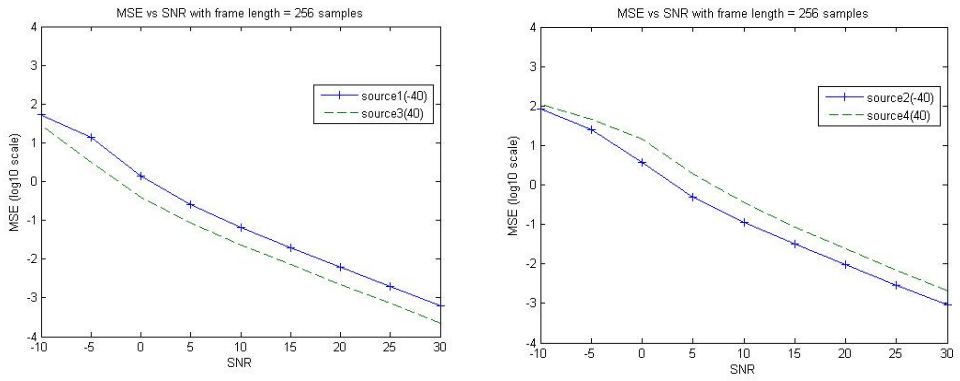


Figure 5.7 MSE versus SNRs with frame length 256 samples using average DOA estimates for mixture of source1 and source3 (left) and mixture of source2 and source4 (right)

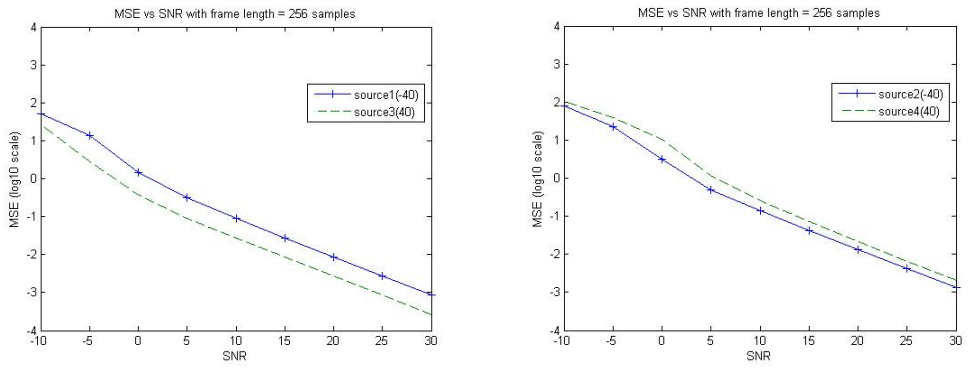


Figure 5.8 MSE versus SNRs with frame length 256 samples using weighted average DOA estimates for mixture of source1 and source3 (left) and mixture of source2 and source4 (right)

5.2 Fusion of Multiple Microphone Arrays

In this section, we consider separation and localization of multiple sources using multiple microphone arrays. The multisource condition further complicates the problem, which necessitates the separation step. Instead of performing BSS and DOA estimation at each array independently, and then fusing DOAs for localization, the proposed scheme handles all of the above processing simultaneously. The estimation problem is formulated as a constrained optimization problem, which can be solved in a distributed manner using the alternating direction method of multipliers. Each array exchanges only tentative DOA information with others in the iterative algorithm. Due to the mutual constraints, multiple arrays can collaboratively solve for

the unmixing matrices, DOAs and locations while avoiding the traditional permutation and scaling issues.

The whole algorithm contains the following steps:

(1) Initialization: Each array is given initial values of all variables: $\left\{\mathbf{W}_l^{(0)}(f)\right\}, \left\{\theta_{sl}^{(0)}\right\}$, and $\left\{u_s^{(0)}, v_s^{(0)}\right\}, s = 1, \ldots, S, l = 1, \ldots, L, f = 1, \ldots, K.$

(2) In-array processing: for iteration i and array l, update estimates $\left\{\mathbf{W}_l^{(i)}(f)\right\}, \left\{\theta_{sl}^{(i)}\right\}$, and $\left\{u_s^{(i)}, v_s^{(i)}\right\}$ according to (10), (11) and (13), $s = 1, \ldots, S, f = 1, \ldots, K$.

(3) Inter-array communication: each array broadcast its $\left\{\theta_{sl}^{(i)}\right\}_{s=1}^{S}$ to all the other array. If the stopping criteria (i.e., the difference between the estimates of the variables from two consecutive iterations are smaller than a predefined number) are satisfied, final estimates are achieved. Otherwise, i:=i+1 and go back to step (2).

Experiments are conducted in order to evaluate the performance of the proposed algorithm. We consider a simple setup with 2 sound sources, 3 microphone arrays, and each array equipped with 2 microphone elements. 3 seconds of audio data are obtained at each microphone. The sampling rate is 8 KHz. The frame length is 125 ms and the frame shift is 2 ms. The distance between the two microphone elements in an array is d = 0.04m, which satisfies the condition of being smaller than half of the minimum wavelength (0.085/2 m) in order to avoid the spatial aliasing effect. The coordinates of the three arrays are (3,4), (5,5), and (7,4), and the orientations are 10°, 20°, and 30°, respectively. The two sources are located at (4,7) and (6,6), respectively. A conventional method is implemented for comparison, which consists of two steps. First, for each array, the ICA method with the directivity pattern is used to estimate unmixing matrices and DOAs. Then, based on all DOAs, the source locations are estimated using the triangulation technique.

It is generally difficult to find a direct measure to evaluate the estimates of unmixing matrices. Since they are closely related to DOAs, to evaluate the performance of the signal separation, we examine the DOA estimation performance. Figure 5.9 shows the DOA estimation error for both the proposed algorithm and the conventional two-step algorithm. We can clearly see that due to the information exchange between the arrays, after a few iterations our proposed

algorithm converges to DOA estimates that are much closer to the ground truth than the conventional two-step method. To evaluate the performance of the source localization, we calculate the source location estimation error as shown in Figure 5.10. The proposed algorithm gives much more accurate source location estimates than the conventional triangulation method.

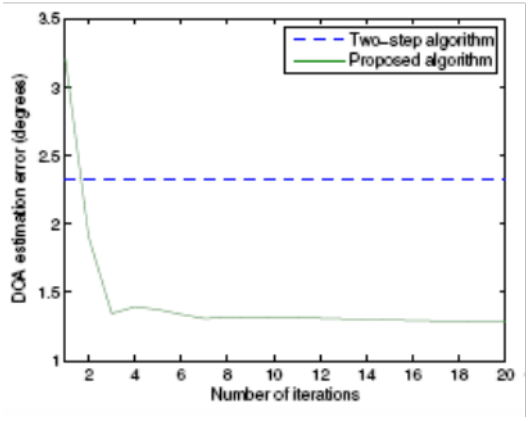


Figure 5.9 DOA estimation error vs. the number of iterations

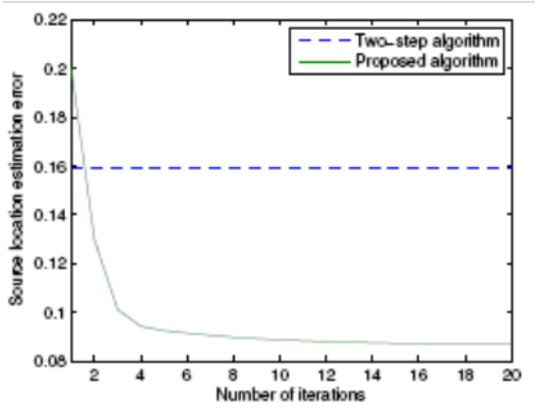


Figure 5.10 Source location estimation error vs. the number of iterations

5.3 Collision Sound Detection

The collision detection problem could be modeled as a binary hypothesis testing problem based on the received audio signals. Let H_0 stand for the case where there is no collision and H_1 stand for a collision occurred in the surrounding environment. Certain measurements are needed to distinguish the collision sound from some other environmental sound sources. These measurements can be referred as the features extracted from audio signals. There are many features like power, frequency, etc., that can be used to distinguish between the collision case

and the non-collision case. The most intuitive feature is the audio signal power P_a at any microphone, since the power is generally larger in the presence of a collision compared to the case where there is only normal traffic. However, this may not be a good feature sometimes, because it is possible that other sound sources have high power. Thus, more distinct features are needed.

5.3.1 Audio Feature Extraction

Collision sounds have some important characteristics based on which our human ears could easily distinguish them from non-crash sounds. It is found that the Mel Frequency Cepstral Coefficients (MFCC) obtained through the MEL Frequency Cepstral Transform (MCT) provides the most distinguishable features needed for collision sound identification. The frequency bands in MCT are equally spaced on the non-linear Mel frequency scale which approximates the behavior of human ear auditory system. The whole process of how to calculate MFCCs is shown in Figure 5.11.

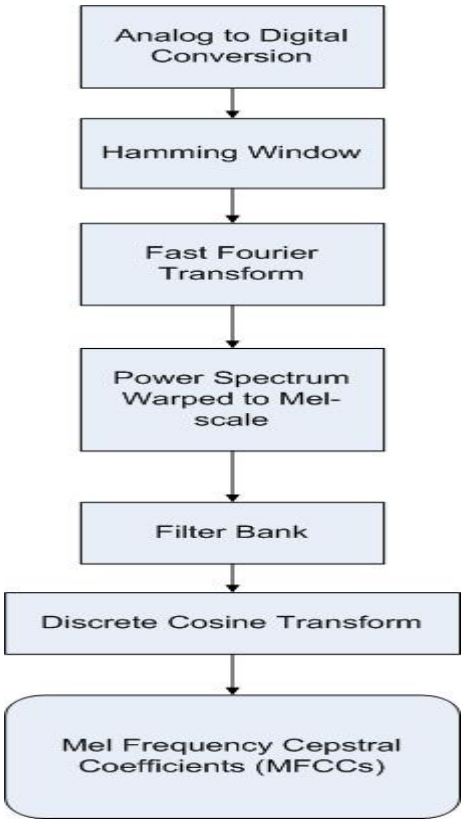


Figure 5.11 Block diagram of calculating MFCCs

The first step is to digitize received audio signals at the microphone array. Every time frame of the signal using the short time Fourier transform and a hamming window of 100 milliseconds and 50 milliseconds overlapping is processed in the experiment. This hamming window can be expressed as,

$$w(n) = 0.54 - 0.46\cos(2\pi n / (N-1))$$

All signals are sampled at 8 KHz at the microphone array and MFCCs are obtained by calculating the FFT of the sampled signals: $x(t)$ is $X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N}$.

The filter bank with P triangular filters ($p = 1, \cdots, P$) is given by

$$H_P(k) = \begin{cases} 0 & k < f(p-1) \\ \dfrac{k - f(p-1)}{f(p) - f(p-1)} & f(p-1) \le k \le f(p) \\ \dfrac{f(p-1) - k}{f(p+1) - f(p)} & f(p) \le k \le f(p+1) \\ 0 & k > f(p+1) \end{cases}$$

Condition $\sum_{p=1}^{P} H_p(k) = 1$ is satisfied. The lowest and highest frequencies of this triangular filter bank is f_l and f_h. Fs is the sampling frequency which is 8000 Hz. N is the value of the FFT number. $f(p)$ is evenly distributed in the Mel scale as follows,

$$f(p) = \frac{N}{Fs} A^{-1}[A_{f_l} + p\frac{A_{f_h} - A_{f_l}}{P+1}]$$

where the Mel scale A is given by $A = 2595 * \log_{10}(1 + f / 700)$, and thus $A^{-1}(a) = 700(e^{(a/2595)} - 1)$. The logarithm power at the output of each triangular filter is

$$L(p) = \log_{10}\left(\sum_{k=0}^{N-1} |X_k|^2 H_p(k)\right), \quad 0 < p \le P$$

MFCCs are the discrete cosine transform of the above P triangular filter outputs,

$$s(n) = \sum_{p=0}^{P} L(p)\cos(\pi n(p - 1/2)P), \; 0<\text{n} \le \text{P}$$

5.3.2 MFCC Based Neural Network Classification

The extracted MFCCs are to be used to distinguish the collision sound from other background traffic sounds. This process can be achieved by classifying the signals into collision and non-collision sounds by using a suitable classifier. The class for the background sounds, including cars and trucks passing by, cars skidding and braking, sounds from industrial and construction sites in the vicinity, etc., is denoted by H_0 and the class for the collision sounds is denoted by H_1. Various recognition methods are available for the classification purpose. Due to its popularity and efficiency, a back propagation (BP) neural network is adopted to identify the collision sounds from the non-collision sounds in this study.

Since the Mel Frequency Cepstrum contains important information only in the first few components, the remaining components can be neglected for audio processing in this case. Specifically, the first 13 cepstrum coefficients are used for detection. They are fed into the neural network for classification.

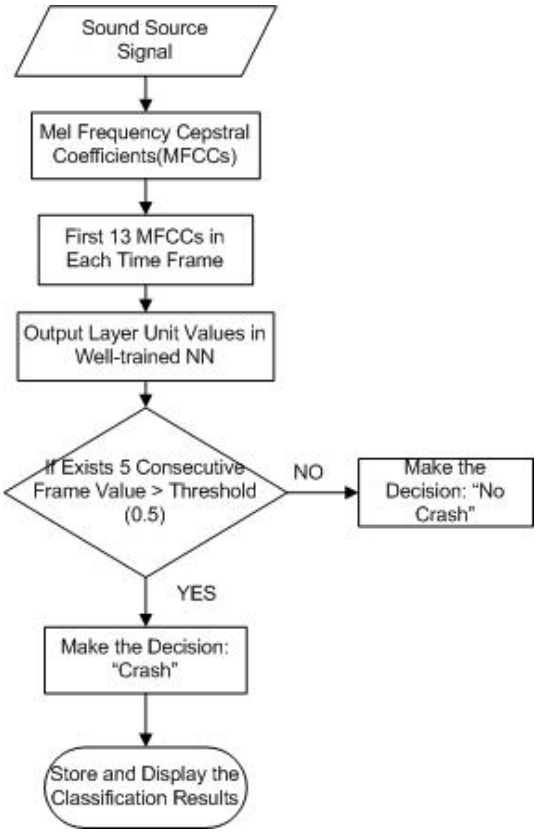


Figure 5.12 Block diagram of MFCCs based neural network classification

In the output layer of neural network, the output varies between 0 and 1. If the value is less than the threshold 0.5, it indicates the selected frame is not a collision sound. Otherwise, it is classified as collision. Since the collision sound may last a few consecutive time frames. If only one frame is determined as the collision sound among a number of frames, it can be asserted that the decision may be wrong. The identification of a collision can be obtained by observing whether there are at least a number of consecutive frames that provide the same detection results. If yes, then the detection decision can be made. The number of consecutive frames is set to be 5 in this research. The whole process of how to make a decision is shown in Figure 5.12. Based on the output of the classifier, once a collision is detected, estimation of the location of the collision is triggered.

5.3.3 Collision Detection Experiments

In the first set of experiments, the performance of the collision detection method is evaluated. A neural network is trained by using 20 non-collision sounds and 10 collision sounds with some specific collision features. The total number of MFCC feature vectors in the training files is 6310. They are divided into three parts: 4416 MFCC feature vectors (70% of data) are applied for learning and training, 947 feature vectors (15% of data) are utilized for validation and the remaining 947 MFCC feature vectors (15% of data) are applied for testing. A target class '0' corresponds to a background sound and '1' indicates a collision. The performance matrix of this BP neural network is shown in Figure 5.13. The overall accuracy of this neural network given is around 96%. The false alarm rate listed in all confusion matrices is around 1.0%, and the missed detection rate is around 2.5%.

The BP neural network is tested with 40 various audio samples in which 20 are pure traffic background sounds and the other 20 have collision sound in them. A collision decision is made detected if the target class is '1' for five consecutive time frames. It is noted that 19 out of 20 collision samples are correctly detected and all 20 background samples are accurately classified. This gives us a 5% missed detection rate and a 0% probability of false alarm, as a whole system renders an error rate of 2.5%. The test result is shown in Table 5.2.

Table 5.2 Test results analysis

Samples	Background	Collision	Missed detection	False alarm
40	20	20	5%	0%

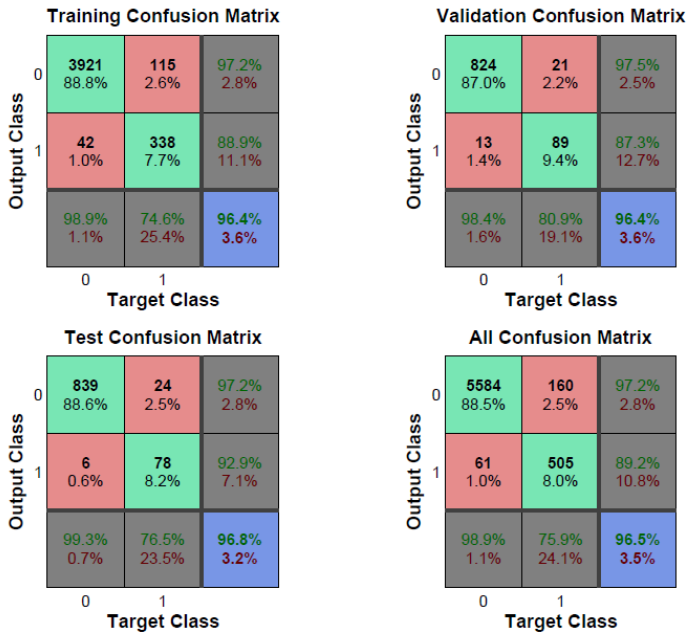


Figure 5.13 Neural network confusion matrix

5.4 Audio/Video Fusion for Decision-Making

The accuracy of video-based accident detection is usually affected by occlusion, change in illumination and limitations of segmentation and identification algorithms. On the other hand, in audio-based accident detection, a loud noise may be due to a large cargo truck passing over a bump or a near-accident that was eventually avoided after sudden application on the brakes. These ambiguities cannot be eliminated by using one sensing modality alone. We propose to take advantage of multiple, disparate sensing modalities wherein the fallacies of one sensor can be compensated by the strengths of a different sensor. By fusing data from audio and video, we expect to increase accuracy through collective reasoning.

Fusion of auditory and visual information will take place within each sensor node. Given that a true accident occurs at a location $[\theta,z]$ where θ is the direction of the accident in the sensor-node coordinate system and z is the height of the event in the image, the probability density functions (pdf) of audio and video measurements can be represented by $p_a(rm(t)|[\theta,z])$, m=1,…,M and $p_v(\boldsymbol{X}(t)|[\theta,z])$, respectively. However, the pdfs are generally difficult to obtain

because they also depend on the environment, background noise and algorithms used for detection in each modality.

The posterior probability of an accident at [θ,z] based on all observed audio and video data can be represented as $p([\theta, z]|r1(t), \dots, rM(t), \boldsymbol{X}(t))$. The larger the probability value, the more probable the accident at [θ,z] in the scene. If we have no prior knowledge of accident location distributions, the prior probability $p([\theta,z])$ can be assumed to be uniform. Using Bayes' rule, it

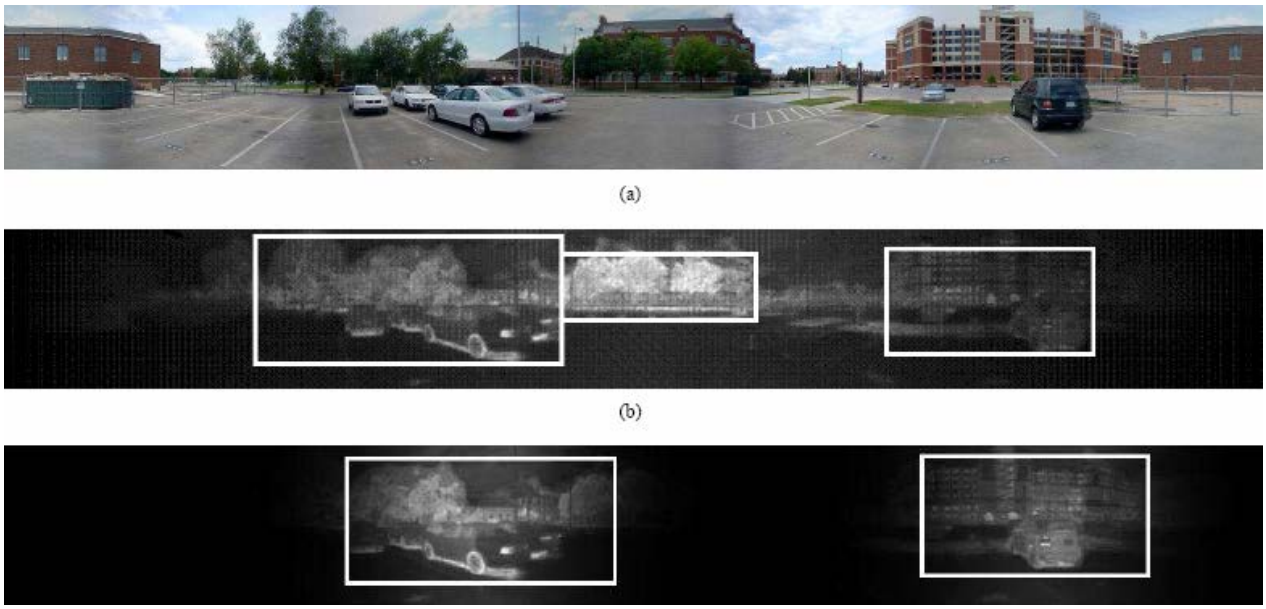


Figure 5.14 Saliency detection. (a) Original image. (b) Detected regions of interest based on image data. (c) Detected regions of interest from audio and video fusion

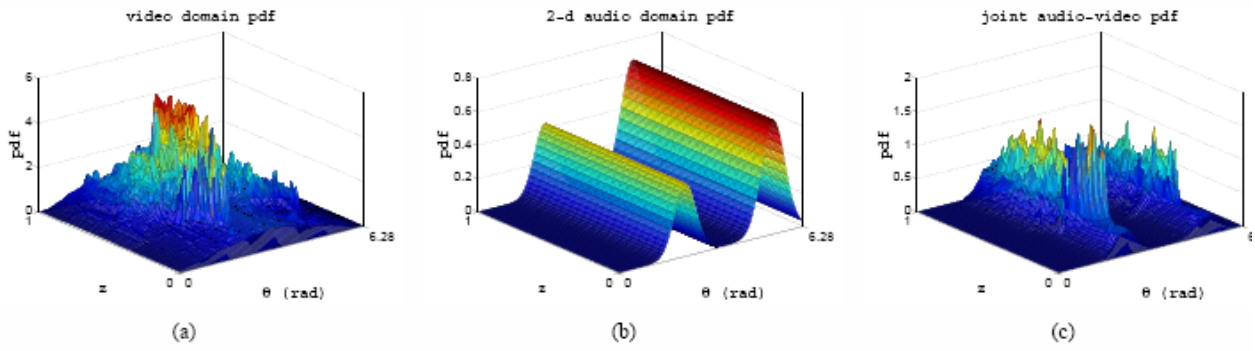


Figure 5.15 (a) Video domain pdf (b) Audio domain pdf (c) pdf after video/audio fusion

can be shown that

$$p([\theta, z]|r1(t), \dots, rM(t), \boldsymbol{X}(t)\ ) \propto p\_a\ ([\theta, z]|r1(t), \dots, rM(t))\ p\_v\ ([\theta, z]|\boldsymbol{X}(t))$$

due to the conditional independence between audio and video data because they are from different sensing mechanisms. This indicates that the final probability distribution of accidents is proportional to the product of its distribution based on audio and video data respectively. The pdf $p_a([\theta, z]|r1(t), \ldots, rM(t))$ can be obtained from the audio saliency map $R(\theta)$ through proper normalization, i.e., $p_a([\theta, z]|r1(t), \ldots, rM(t)) = R(\theta)/(D \times \int R(\theta)d\theta)$, where D is the height of the image. The pdf $p_v([\theta, z]|\boldsymbol{X}(t))$ can be obtained in a similar way by normalizing the video saliency map. Our results are shown in Figures 5.14 and 5.15, which depict how fusing auditory and visual information can lead to refined saliency maps.

5.5 Outdoor Experiments

We perform the experiments in an open area between two buildings. A rectangular wooden frame is used to support four microphones forming an array. The inter-element spacing of the array is 0.05m.

We use an NI CDAQ 9171 USB chassis, shown in Figure 5.16, to simultaneously connect four microphones with a laptop. The microphones are omnidirectional and the highest supported sampling rate is 51.2 KHz. Two USB-powered loud speakers are placed at one side of the microphone array as the audio sources. After collecting certain length (e.g., 4 seconds) of signals, our algorithm is used to estimate the DOAs of the sources. Figure 5.17 illustrates one example of the experiment setup.

Figure 5.16 a NI cDAQ 9171 USB chassis and four microphones

Figure 5.17 One example of the experimental setup

The main noise source in outdoor environments is wind, which significantly affects the performance of our algorithm. The parameter setting for experiments is summarized in Table 5.3.

Table 5.3 Parameter setting for outdoor experiments

Experimental Parameters	Values
Number of Sources M	2
Source Categories	Speech & Music
Source Length	4 Seconds
Number of Microphones N	4
Array Spacing d	0.05
Sampling Rate F_s	51.2 KHz
Frame Length	1024
Frame Shift	1024
FFT Window	Rectangular

As in the simulations, we set lower cutoff frequency and upper frequency, in which we perform subspace method. Moreover, environmental noise occupies mainly low frequencies, as shown in Figure 5.18. It is the spectrogram of the background noise during an experiment. It is clear that most power is distributed below 1 KHz.

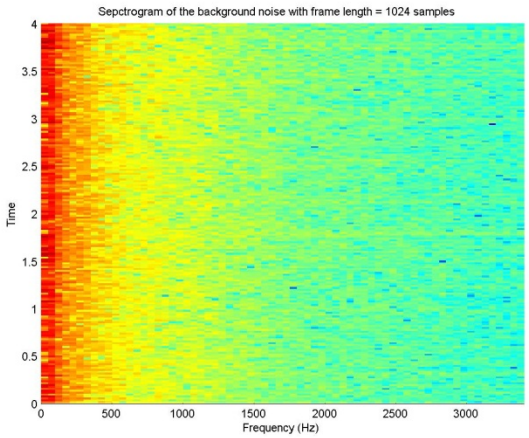


Figure 5.18 The spectrogram of the background noise

In outdoor environments, when signals consist mostly of low frequency components, the algorithm fails to provide good estimates because of the low SNR and also the existence of the cutoff frequency. When signals contain significant amount of high frequency components, the estimates at these frequencies are good. Figure 5.19 shows the relations between DOA estimation accuracy and original source spectrogram. Except for the low frequency part, the estimated DOAs are more accurate at frequencies with high power and less accurate at low frequency. This is clear in Figure 5.19 (left). The performance difference is because of the SNR difference at different frequencies.

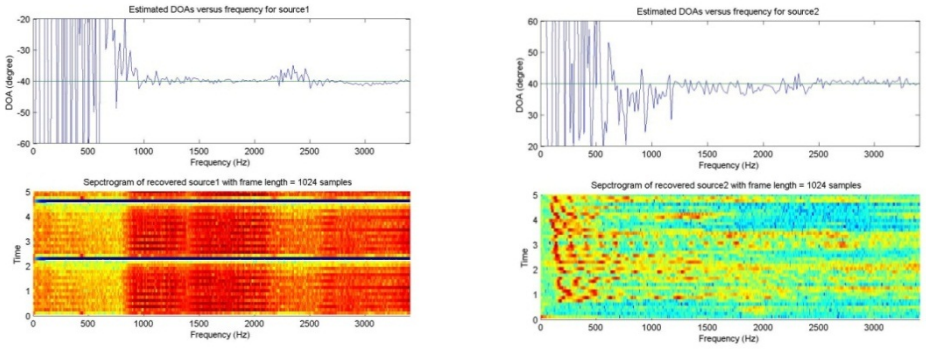


Figure 5.19 Estimated DOAs and original spectrograms for source1 (left) and source2 (right)

THIS PAGE IS INTENTIONALLY BLANK

CHAPTER 6

DEVELOPMENT OF A SMALL-SCALE TESTBED

As an important part of this testbed, it is desirable to develop autonomous driving RC cars that can follow predefined trajectories so that different traffic scenarios can be realized. However, controlling non-holonomic RC cars with low-accuracy steering angles and non-smooth velocity is a challenging problem. We propose an efficient feedback control algorithm based on virtual vehicles to allow the RC cars to track predefined trajectories. In Section I we present the hardware setup of the testbed. Section II shows the control hardware design of the automated RC car. Section III describes the RC car kinematic model, the tracking control algorithm, and the control of multiple RC cars, respectively. Section IV shows the experimental results.

6.1 Hardware Setup of the Testbed

6.1.1 Overview

The scale-down testbed we developed has four main parts:

- An arena,
- An indoor localization system,
- Automated radio controlled (RC) cars,
- Roadside monitoring facilities.

To mimic typical traffic environments we build an arena with a wooden floor, mock buildings and streets. An indoor localization system built from an optical motion capture system is developed. Automated radio controlled cars with both autonomous driving and human driving capability are developed. For the roadside monitoring facilities, an overhead fish-eye camera is used and the associated advanced video processing algorithms are developed which include image segmentation, object identification and tracking. The overall testbed is shown in Figure 6.1.

Figure 6.1 The developed testbed for experimental validation

6.1.2 Arena

We built an arena with a dimension of 16 feet by 12 feet, which can be used to create various mock environments. The arena is based on a wooden floor on which streets, roads and intersections can be set up. A carpet on top of the wooden floor is used to mimic concrete or asphalt road surfaces. We can also place mock buildings, trees, and other decorations to make the scene more realistic.

6.1.3 Indoor localization system

An indoor localization system is built up to localize RC cars in the simulated traffic environment. The purpose of this system is to provide location feedback of the cars in order to control them to move along predefined tracks. This indoor localization system can mimic the function of the GPS system in the real world. This system is developed from an optical motion capture system (Opti-Track) from NaturalPoint, Inc (http://www.naturalpoint.com/optitrack). There are 12 cameras mounted on tripods to cover the whole arena. The Opti-Track system has the capability

of capturing 100 frames per second, so that the location and orientation information can be obtained in real time and with high accuracy (above 95 percent). The Opti-Track system tracks each RC car via the markers (see Figure 6.2) mounted on top of the RC car. To simulate real world GPS signals, we can down sample the data and even inject noise into the RC car location and orientation estimates.

6.1.4 Automated RC cars

We have developed both autonomous driving and human driving RC cars. They are based on the design of an automated RC car which will be explained in Section 6.2.

Figure 6.2 Two automated RC cars: (Top) Autonomous driving RC car (Bottom) Human driving RC car

For the autonomous driving RC car (see Figure 6.2-Top), four markers are mounted on top of an automated RC car to build a rigid body so that the location and orientation of the car can be tracked. A speaker is mounted on the back of the car for the purpose of mimicking collision sound, which can be used in automated collision detection research. The tracking control algorithm that allows the RC car to track predefined trajectories is developed in the computer, and the control commands are sent to the RC car via the Xbee wireless communication.

For the human driving RC car (Figure 6.2-Bottom), a miniature wireless camera is mounted on the hood of the RC car to provide visual inputs. It is used to observe the environment in front

of the car and send the video stream through wireless communication to the PC. The human driver sits in front of a wheel stand and drives the RC car while he/she observes the video stream on the monitor. We developed a program using the software development kit (SDK) of the wheel stand to read the data from the steering wheel which include the wheel turning angle, brake, gas pedal and gear shift status. Based on that, we send control commands, such as "move forward", "backward", "turn left", "turn right", "speed up", or "slow down", through the Xbee wireless communication to the automated RC car. Such a human driving setup can partially mimic the human driving experience. The whole setup of this human driving system is shown in Figure 6.3. The human driving RC car is also equipped with speakers to generate collision or other sounds to mimic real traffic.

Figure 6.3 The setup for manually driving RC cars

6.1.5 Roadside monitoring facilities

A Mobotix Q24 fish-eye camera as shown in Figure 6.1 is mounted over the arena to serve as a roadside monitoring facility. This camera is capable of providing different views simultaneously, including a full 360 degree all-round view, hence it can cover the whole arena to monitor the traffic underneath it. This camera uses an IP-based interface. The stream of live images from the camera is obtained through a socket connection. The features of the camera (including

resolutions, frame rates, etc) can be easily modified by sending a web request. The zooming and panning of the camera lenses can be controlled by virtual PTZ (Pan, Tilt, Zoom) functions. The camera provides a highest resolution of 3M pixels and the color images are scalable from 160×120 to 2048×1536. The highest frame rate is 30fps. This camera can be used in research projects involving traffic monitoring, such as automated collision detection or anomaly detection through visual surveillance.

6.2 Hardware Design of Automated RC Car

We used commercial off-the shelf RC cars with a scale of 1:14. The RC car comes with two DC motors: a front DC motor for steering control and a rear DC motor for speed control. After testing the front DC motor we find that it has very poor steering performance and cannot be used in our project. Therefore the front DC motor is replaced by a servo motor which is mounted in the RC car as shown in Figure 6.4. The overall hardware design of the automated RC car control is shown in Figure 6.5. There are two major parts in the hardware design: an XBee wireless module and a control board embedded in the RC car body.

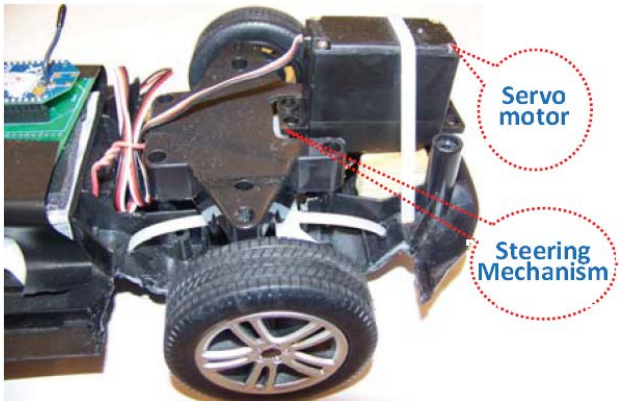


Figure 6.4 The servo motor is mounted in the RC car

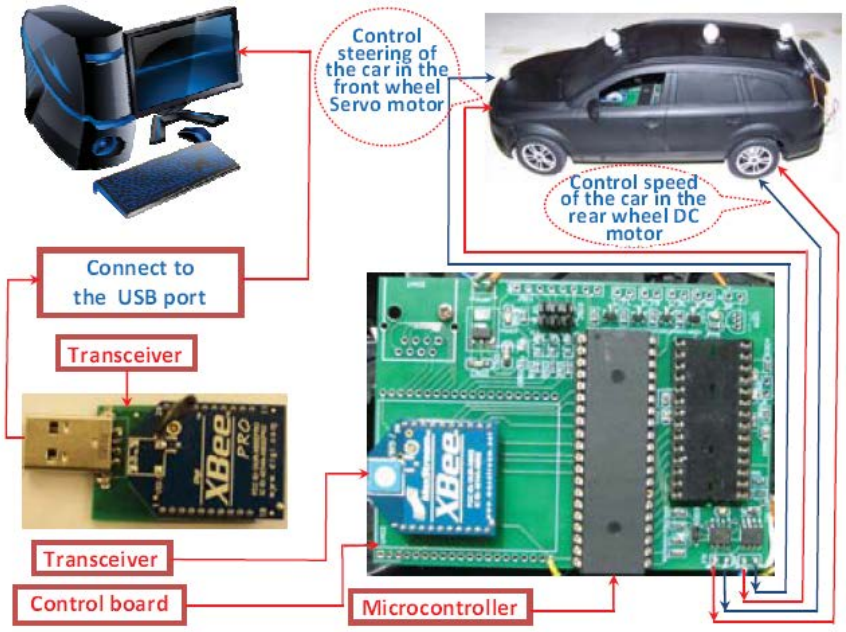


Figure 6.5 The hardware setup for the RC car control

XBee wireless module: This module is using small, low power radio based on the IEEE 802.15.4 standard and originally targeted for wireless personal area networks (WPANs). Due to the limited size of our testbed, we find it is a good solution to wireless communication in our testbed. This XBee wireless module has a data rate up to 250Kbps and can serve as the communication channel between RC cars and between RC cars and roadside infrastructures. For the purpose of automated RC car control, one XBee module is connected to the PC and another is mounted on the control board as shown in Figure 6.5, enabling the wireless communication between the PC and automated RC cars.

Control board: An embedded control board is developed to replace the original circuit board inside the RC car. Its design is illustrated in Figure 6.6. The MCU (ATmega 162) in the middle of Figure 6.6 functions as the control unit for the automated RC car. A speaker is used for playing recorded sounds to mimic real world traffic sounds while a microphone is used to record environmental sounds. Both the speaker and the microphone are driven by a voice record IC (ISD 1700). The PWM output from the MCU is used to drive the front servo motor and the rear DC motor so that the orientation and the velocity of the RC car can be controlled, respectively.

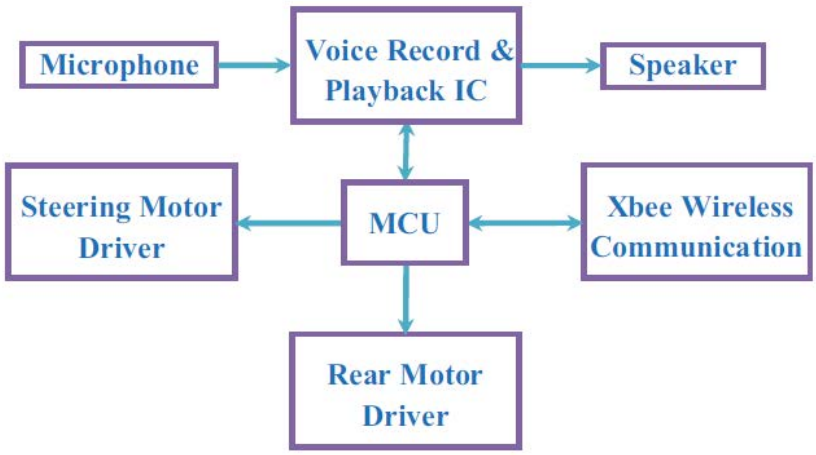


Figure 6.6 The function blocks of the control board

6.3 Autonomous RC Car Control

To create various traffic scenarios, we need control the autonomous RC cars so that they can follow predefined trajectories. In this way, we can develop scripts that describe the desired movement of the vehicles in the arena. For example, we can create a scenario of heads-on collision between two cars exactly the way we want.

To control multiple RC cars simultaneously, the control algorithm is implemented using multi-thread programming. The PC connects to a USB wireless adapter (XBee) for communicating with the RC cars. The architecture of the multi-car control program is shown in Figure 6.7. The controller for each RC car is implemented independently in a separate thread. These threads can also realize inter-vehicle communication.

The problem of controlling a non-holonomic vehicle is well studied [25-28], but controlling a nonholonomic vehicle with low accuracy and non-smooth velocity is a challenging problem. In this section, we first build a model of the RC car, then focus on developing efficient control algorithms for the RC car to track a predefined trajectory.

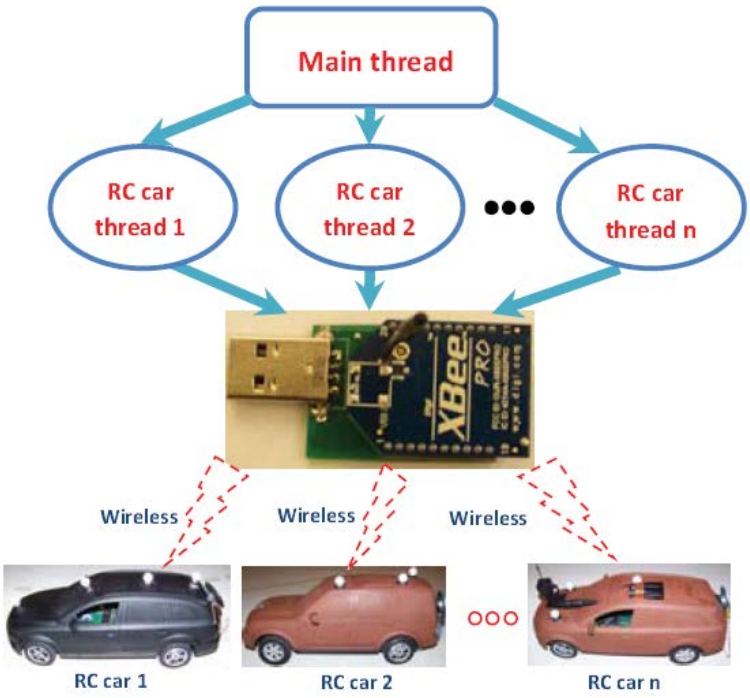


Figure 6.7 The architecture of the multi-car control program

6.3.1 RC Car Model

As we know most existing models of non-holonomic vehicles [25-27] are usually described as:

$$\begin{cases} \dot{x} = v\cos(\Psi) \\ \dot{y} = v\sin(\Psi) \\ \dot{\Psi} = \omega \end{cases}$$

where Ψ is the orientation angle of the vehicle (see Figure 6.8), and ω is the angular velocity. The model stated above is simple and does not consider the actual constraints on the range of steering angle and the sliding angle which reflects the sliding errors between the center point of the car and the center point of the front axial.

Since the RC car has low accuracy on steering we model it as:

$$\begin{cases} \dot{x} &= v\cos(\Psi + \theta + \beta) \\ \dot{y} &= v\sin(\Psi + \theta + \beta) \\ \dot{\Psi} &= \omega \end{cases}$$

where θ is the steering angle of the front wheels (see Figure 6.8), and β is the sliding angle that is obtained based on the center point of the car and the velocity vector v. The β angle is computed as $\beta = \Psi_c - \Psi$, here Ψ_c is the heading of the vehicle at the center point.

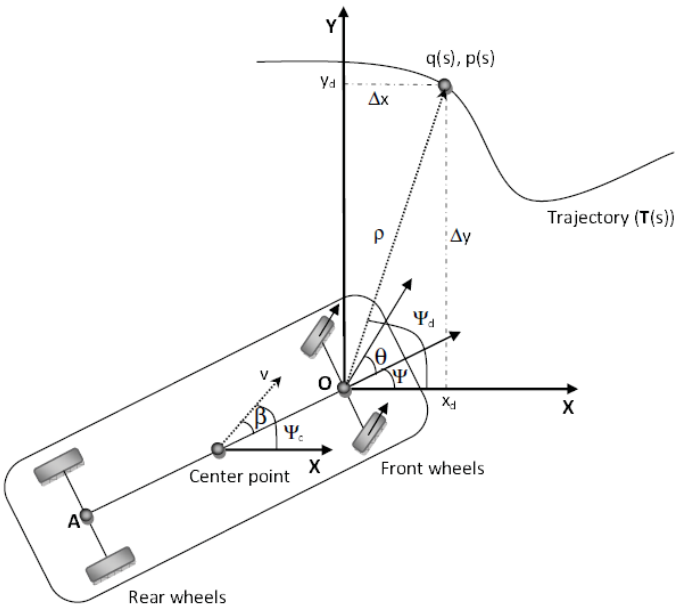


Figure 6.8 Illustration of RC car tracking the virtual vehicle moving in a predefined trajectory

6.3.2 RC car control algorithm

In order to let the RC car track a predefined trajectory we use a virtual vehicle based approach. The virtual vehicle is a reference point that is moving on the path we want the RC car to follow. The virtual vehicle, s(t), is designed to move along the path with xd = p(s) and yd = q(s). In order to track the virtual vehicle, two constraints are considered in the following two inequalities, which are related to the difference between the actual heading and the desired heading of the RC car, and the distance between the actual and the virtual vehicle, respectively.

$$lim(|\Psi(t) - \Psi_d(t)|)_{t\to\infty} \leq d_\Psi$$

here Ψ d is the desired angle, dΨ is a small angle threshold.

$$lim\big(\rho(t)\big)_{t\to\infty} \leq d_\rho$$

here dρ is a small distance threshold. ρ(t) is the Euclidean distance between the RC car and the virtual vehicle (see Figure 6.8). It is computed as

$$\rho(t) = \sqrt{\Delta x^2 + \Delta y^2}$$

here Δ x = xd − x and Δ y = yd − y.

In order to handle the first equality, the steering angle control for the RC car is based on the proportional- derivative control (PD control) as follows:

$$\theta(t) = -k_p[\Psi(t) - \Psi_d(t)] - k_d\big[\dot{\Psi}(t) - \dot{\Psi}_d(t)\big]$$

here kp and kd are positive constants.

As mentioned before, although we replaced the front DC motor by a servo motor to obtain a controllable orientation and wider steering angles (20 degree), the left and right steering angles are not the same because the mechanical steering part is not rigid. Additionally, the velocity of the RC car is not smooth. The low-accuracy steering angle of the RC car is illustrated in Figure 6.9. In this model, since the front wheels mounted on the car are not stable, the left steering angle range (Figure 6.9, Up) is different from the right one (Figure 6.9, Down). Specifically, in our experiment we use a RC car which has θ_L^{max} between 22 degree and 27 degree and θ_R^{max} between 15 degree and 20 degree. This difference on the left and right steering angle is one of the reasons that make the car not able to track the predefined trajectories when applying any traditional tracking control algorithm.

Based on the above analysis, in order to handle the second inequality the parameter γ is introduced [29, 30]

$$\dot{\rho} - \dot{d}_\rho = -\gamma\left(\rho - d_\rho\right)$$

here γ is a positive constant. From the relation between ρ(t) with Δ x and Δ y, we can obtain:

$$\begin{aligned} \dot{\rho} &= \frac{1}{\sqrt{\Delta x^2 + \Delta y^2}}(\Delta x \Delta \dot{x} + \Delta y \Delta \dot{y}) \\ &= \frac{1}{\rho}[\Delta x(\dot{x}_d - \dot{x}) - \Delta y(\dot{y}_d - \dot{y})] \end{aligned}$$

Here $\dot{x}_d = \dot{p}(s)\dot{s}$ and $\dot{y}_d = \dot{q}(s)\dot{s}$.

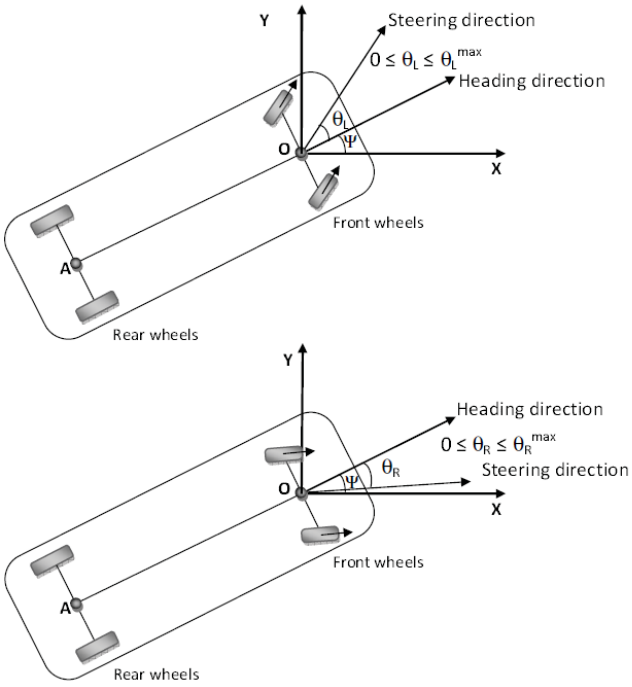


Figure 6.9 Low accuracy steering angle of the RC car. The left steering angle range is different from the right one

Notice that dρ is a constant, $\dot{d}_\rho = 0$, we have

$$\frac{1}{\rho}\dot{s}\left(\Delta x\dot{p}(s) + \Delta y\dot{q}(s)\right) = \frac{1}{\rho}(\Delta x\dot{x} + \Delta y\dot{y}) - \gamma\left(\rho - d_\rho\right)$$

or,

$$\dot{s} = \frac{1}{\Delta x\dot{p}(s) + \Delta y\dot{q}(s)}\left[(\Delta x\dot{x} + \Delta y\dot{y}) - \gamma\left(\rho - d_\rho\right)\right]$$

From the above equation, it is easy to see that if $\Delta x\dot{p}(s) + \Delta y\dot{q}(s) = 0$ then $\dot{s} \to \infty$, or the car cannot track the virtual vehicle. In order to have $\Delta x\dot{p}(s) + \Delta y\dot{q}(s) \neq 0$, the car should stay close to and behind the virtual vehicle. From this analysis we should make the virtual vehicle move with constant velocity at initial time when the car is far from it. This means that we need to have s(t) = s(t − 1) + c (c is a positive constant) if $t<t_{threshold}$. The whole tracking control algorithm is shown in the Algorithm 1.

Algorithm 1: The virtual vehicle tracking algorithm

Initialization phase:

- Create a trajectory of the virtual vehicle that the RC car wants to track.
- Initialize parameters: v, k_p, k_d, $d_\rho, \gamma, s, \dot{s}, \Delta_t$.

Implementation phase:

If $t<t_{threshold}$ **then**

Let the virtual vehicle move with constant velocity

(to relax the assumption $\Delta x\dot{p}(s) + \Delta y\dot{q}(s) = 0$)

$$s(t)=s(t\text{-}1)+c,$$

here, c is a positive constant.

else

- Compute the velocity of the virtual vehicle:

$$\dot{s} = \frac{1}{\Delta x\dot{p}(s) + \Delta y\dot{q}(s)}\left[(\Delta x\dot{x} + \Delta y\dot{y}) - \gamma\left(\rho - d_\rho\right)\right]$$

- Compute the steering angle for the RC car:

$$\theta(t) = -k_p[\Psi(t) - \Psi_d(t)] - k_d\left[\dot{\Psi}(t) - \dot{\Psi}_d(t)\right]$$

- Update the position of the virtual vehicle based on its velocity $\dot{s}$:

$$s(t) = s(t-1) + \dot{s}\Delta_t$$

end

6.4 Experimental Results

In this section we test our proposed control algorithm for a single RC car, then we extend the test to three RC cars based on multi-thread programming.

The parameters for RC control algorithm (Algorithm 1) are as follows: the desired distance between the RC car and the virtual vehicle dρ is 300mm; the initial velocity of the virtual vehicle is 0; the constants for the PD steering controller kp = 1, kd = 0.8; the constant γ for computing the velocity of the virtual vehicle is 2; and other parameters are v = 67 and Δt = 0.00056. The parameters of the virtual vehicle moving in a circle are as follows: [x, y] = [Rcos(s), Rsin(s)] with its radius R = 1500mm.

The tracking results of Algorithm 1 are shown in Figure 6.10. Namely, Figure 6.10 (left) shows the RC car tracking the virtual vehicle which moves in the circle trajectory. Figure 6.10

(middle) shows the evaluation of the distance between the RC car and the virtual vehicle, and we can see that this distance converges to the predefined value of dρ = 300mm. Hence, this result meets our control goal on page 80. In addition, we evaluate the difference between the actual heading of the RC car and the desired one as shown in Figure 6.10 (right). This result also satisfies the requirement mentioned previously.

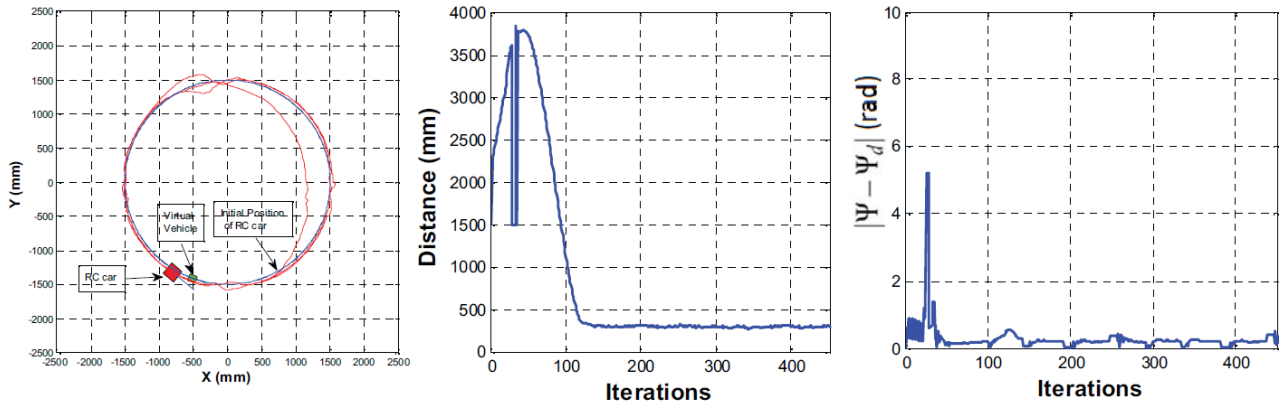


Figure 6.10 Trajectories of the RC car tracking the virtual vehicle moving in circle (Left), the distance between the RC car and the virtual vehicle (Middle), and the difference between the actual heading of the RC car and the desired one (Right)

We also evaluate Algorithm 1 for 3 RC cars running through an intersection as shown in Figure 6.11. In this test we design the trajectories which are more complicated than the circle trajectory. These trajectories have sharp changing points at the transition from circle trajectory to line trajectory. Figure 6.11 (left) shows the three RC cars tracking the three virtual vehicles which move in the mock streets with an intersection. Figure 6.11 (middle) shows the evaluations of the distance between the RC cars and the virtual vehicles. Figure 6.11 (right) shows the difference between the actual heading of the RC cars and the desired ones, respectively. Observing these figures we can see that at these sharp turning points on the trajectories, the tracking performance deteriorates.

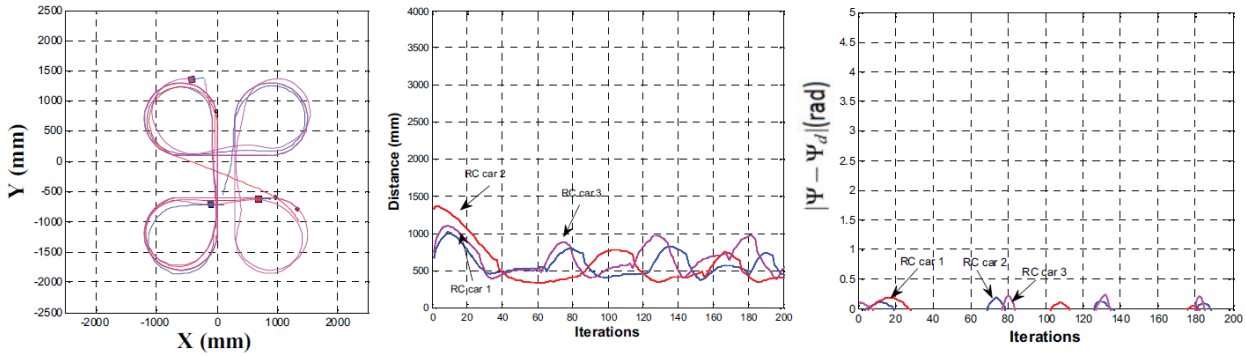


Figure 6.11 Trajectories of the 3 RC cars tracking the virtual vehicles moving in desired trajectories (Left), the distance between the RC car and the virtual vehicle (Middle), and the difference between the actual heading of the RC cars and the desired ones (Right)

CHAPTER 7
CONCLUSIONS AND RECOMMENDATIONS

7.1 Summary

The overarching goals of OKCARS are (1) to reduce the time required for proper assistance (not just first responders) to arrive at the scene, and (2) to mitigate further accidents by forewarning oncoming traffic. In this project, we investigated the limitations of current automatic incident detection systems and developed an intelligent collision analysis and response system through audio and video information collection from a traffic scene. We developed a prototype of OKCARS by carefully selecting cost-effective hardware components and software platform. Advanced audio/video-based collision detection algorithms and multisensory/multimodal fusion algorithms were developed and integrated into the prototype. We validated the developed system through comprehensive experiments on a small-scale testbed.

7.2 Findings and contributions

In this project, we have developed four SAV sensor nodes. Each of them consists of an omnidirectional vision sensor, a microphone array and the associated data acquisition board. We have two versions of this omnidirectional sensor: catadioptric camera and fish-eye camera. We have developed the software to interface with the vision in both Windows and Linux OSes. The microphone array consists of multiple microphones and a plastic ring which is light and adjustable is radius. Two types of data acquisition boards have been tested. One is USB7202 and the other is NI 9234. We also developed the software platform for the networking of multiple audio/video sensor nodes. The software platform is based on the ROS (Robot Operating System), which is an open source software framework for robots and sensors.

The main finding on the video-analysis portion of this project is that it is possible to use very low-level features for detecting and tracking moving objects in video. To save computation, many of these low-level features can later be used for the accident-detection process. Our work provides an intuitive, effective, and computationally efficient alternative to statistical tracking techniques. The primary contribution is the development of an efficient algorithm for detecting and tracking vehicles which operates by using low-level features and HVS-based modeling.

Secondary contributions include a significantly accelerated version of an image similarity algorithm (MAD), and a feature-based extension, called F-MAD, which is more robust to motion blur and shadows.

We used collected audio information from microphone arrays to monitor traffic at intersections. Audio source separation and localization simultaneously using microphone arrays, and potential car accidents identification based on MFCC features and neural networks were developed and implemented. We combined source separation, localization, and collision sound detection into a unified system. Various aspects of the system were studied and tested through simulations and experiments. The main contribution lies in that we did not assume any knowledge of the signal mixing process, which reflects the situation encountered in practical applications more accurately.

We implemented a small-scale testbed (or platform) to conduct the experiments that can be used to validate our proposed collision detection algorithms. Our testbed has four main parts: an arena, an indoor localization system, automated radio controlled (RC) cars and roadside monitoring facilities. First, to mimic traffic environments we built an arena with a wooden floor, mock buildings and streets. Second, to facilitate feedback control for trajectory following, an indoor localization system was set up to track the RC cars. Third, both autonomous driving RC cars and human driving RC cars were developed, based on an automated RC car design. The automated RC cars can receive control signals from a computer through an Xbee RF module and control the front and rear wheels through motors. A new control algorithm was developed to allow the RC cars to track predefined trajectories. Finally, the roadside monitoring system is a collection of the smart audio visual (SAV) nodes which can collect both the audio and video data from the collision scenario.

7.3 Implementation

The results of this project will advance and expand the strategic plan of the OTC in the area of safety and security. One of the four key initiatives put forth in the OTC Strategic Plan is "Enhancing roadway traffic, transit and infrastructure safety and security through improvement of universal mobility, hardening assessment, emergency response preparedness, and development of decision support tools for risk assessment and management." We believe that the outcomes of this project have great potential to further push this initiative. In addition, the results

of this research will have implications for emergencies and congestion, two of the nine areas identified in the 2005 "Critical Issues in Transportation" report put forth by the U.S. Transportation Research Board. The underlying technology developed in this project has implications for other monitoring applications as well. Intelligent low-cost audio-visual sensors with wireless communication capabilities can be used for applications such as wildfire/crop/ livestock monitoring thereby further promoting economic growth.

7.4 Recommendation for Future Work

In an effort to improve the tracking accuracy, we investigated the use of a statistical tracking technique based on a Kalman filter. Although this technique increased the tracking accuracy, its major limitation was its extremely high computational complexity. Similarly, although F-MAD is more robust to motion blur and shadows, it is not yet suitable for real-time operation. Thus, our future work will focus on developing simplified versions of these two algorithms which can operate in real-time, yet still yield acceptable tracking performance.

For most of the audio analysis, we assumed overdetermined situations, i.e., the number of microphones used is more than the number of sound sources. In our future work, we intend to explore the more challenging underdertermined audio processing system, which is able to handle source separation and localization using fewer microphones. Moreover, we will continue to investigate how to relax the stringent requirements on the placement and sampling rate of microphones. To improve the system robustness, more studies are needed in harsher environments at roadway intersections.

THIS PAGE IS INTENTIONALLY BLANK

REFERENCES

[1] W. M. Evanco, "The potential impact of rural mayday systems on vehicular crash fatalities," *Accident Analysis & Prevention,* vol. 31, pp. 455-462, 1999.

[2] World Health Organization. (2007). *Fact sheet: the top ten causes of death*. Available: www.who.int/mediacentre/factsheets/fs310.pdf

[3] U.S. Department of Transportation, "Traffic safety facts 2006," National Highway Traffic Safety Administration, Ed., ed, 2006.

[4] C. L. Dudek, C. J. Messer, and N. B. Nuckles, "Incident detection on urban freeway," Transportation Research Board, Ed., ed, 1974, pp. 12-14.

[5] P. T. Martin and J. Perrin, "Automatic incident detection," University of Utah Traffic Laboratory Report,2000.

[6] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 1, pp. 108-118, 2000.

[7] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing,* vol. 21, pp. 359-381, 2003.

[8] W. Kun-feng, J. Xingwu, and T. Shuming, "A survey of vision-based automatic incident detection technology," in *Vehicular Electronics and Safety, 2005. IEEE International Conference on*, 2005, pp. 290-295.

[9] H. Ikeda, Y. Kaneko, T. Matsuo, and K. Tsuji, "Abnormal incident detection system employing image processing technology," in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, 1999, pp. 748-752.

[10] H. Veeraraghavan, P. Schrater, and N. Papanikolopoulos, "Switching Kalman Filter-based approach for tracking and event detection at traffic intersections," in *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, 2005, pp. 1167-1172.

[11] M. S. Shehata, J. Cai, W. M. Badawy, T. W. Burr, M. S. Pervez, R. J. Johannesson, and A. Radmanesh, "Video-Based Automatic Incident Detection for Smart Roads: The outdoor nnvironmental challenges regarding false alarms," *Trans. Intell. Transport. Sys.,* vol. 9, pp. 349-360, 2008.

[12] E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *Journal of Electronic Imaging,* vol. 19, pp. 011006-011006, 2010.

[13] K. Yong-Kul and L. Dong-Young, "A traffic accident recording and reporting model at intersections," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 8, pp. 188-194, 2007.

[14] C. T. Vu, T. D. Phan, and D. M. Chandler, "S3: A spectral and spatial measure of local perceived sharpness in natural images," *Image Processing, IEEE Transactions on,* vol. 21, pp. 934-945, 2012.

[15] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, "Vehicle detection and tracking in car video based on motion model," *Trans. Intell. Transport. Sys.,* vol. 12, pp. 583-595, 2011.

[16] H. Perez-Meana, *Advances in Audio and Speech Signal Processing: Technologies and Applications* vol. Hershey, PA, USA: IGI Global, 2007.

[17] F. Asano, S. Hayamizu, T. Yamada, and S. Nakamura, "Speech enhancement based on the subspace method," *Speech and Audio Processing, IEEE Transactions on,* vol. 8, pp. 497-507, 2000.

[18] P. Stoica and T. Soderstrom, "Statistical analysis of MUSIC and subspace rotation estimates of sinusoidal frequencies," *Trans. Sig. Proc.,* vol. 39, pp. 1836-1847, 1991.

[19] H. Akaike, "A new look at the statistical model identification," *Automatic Control, IEEE Transactions on,* vol. 19, pp. 716-723, 1974.

[20] J. Rissanen, "Modeling by shortest data description," *Automatica,* vol. 14, pp. 465-471, 1978.

[21] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics,* vol. 6, p. 4, 1978.

[22] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 33, pp. 387-392, 1985.

[23] S. Winter, H. Sawada, and S. Makino, "Geometrical understanding of the PCA subspace method for overdetermined blind source separation," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 2003, pp. 769-72.

[24] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on,* vol. 14, pp. 1462-1469, 2006.

[25] G. Indiveri, "Kinematic time-invariant control of a 2D nonholonomic vehicle," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, 1999, pp. 2112-2117 vol.3.

[26] C. C. de Wit and O. J. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," *Automatic Control, IEEE Transactions on,* vol. 37, pp. 1791-1797, 1992.

[27] Y. Zhu and U. Ozguner, "Constrained model predictive control for nonholonomic vehicle regulation," in *Proceedings of the 17th World Congress The International Federation of Automatic Control*, Seoul, Korea, 2008.

[28] J. Cochran and M. Krstic, "Nonholonomic source seeking with tuning of angular velocity," *Automatic Control, IEEE Transactions on,* vol. 54, pp. 717-731, 2009.

[29] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *Automatic Control, IEEE Transactions on,* vol. 46, pp. 1777-1782, 2001.

[30] S. V. Gusev and I. A. Makarov, "Stabilization of program motion of transport robot with track laying chassis," *Proceedings of LSU,* vol. 1, 1989.