This is to certify that the
dissertation entitled

SEGMENTATION, CLASSIFICATION, AND TRACKING OF HUMANS FOR SMART AIRBAG APPLICATIONS

presented by

Michael E. Farmer

has been accepted towards fulfillment
of the requirements for the

Ph.D.	degree in	The Department of Computer Science and Engineering

Major Professor's Signature

March 2, 2004

Date

SEGMENTATION, CLASSIFICATION, AND TRACKING OF HUMANS FOR SMART AIRBAG APPLICATIONS

VOLUME 1

By

Michael E. Farmer

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science & Engineering

2004

ABSTRACT

SEGMENTATION, CLASSIFICATION, AND TRACKING OF HUMANS FOR SMART AIRBAG APPLICATIONS

By

Michael E. Farmer

There has been considerable attention paid to developing 'smart' airbags that can determine, not only, if they should be deployed in a crash event, but also with what force they should be deployed. Information on the size and type of front passenger seat occupant is used to determine the safe level of force with which to deploy the airbag. To date vision systems have been successfully applied to relatively controlled environments, such as for manufacturing, or they have been used in uncontrolled environments, such as for surveillance, where there is a human in the loop to monitor the performance of the system.

In this thesis we have developed a computer vision-based approach to airbag suppression that attempts to provide the robustness required for an autonomous system fielded in a relatively uncontrolled environment. It addresses a very difficult real-world application, in which computer vision had not previously been applied, to simultaneously perform real-time human recognition and tracking. The specific contributions to pattern recognition include the development of a new filter-based feature selection algorithm based on robust statistical measures of discriminability and feature correlation. The algorithm performs as well as, or better than, other filter or wrapper methods, and is considerably faster.

We have also defined a contextual processing algorithm that uses a continuous stream of classifications and the theory of evidential reasoning. This stream of results is integrated using the Dempster-Shafer rules of belief revision, which allows us to classify the occupant to a level of abstraction commensurate with the available image information.

Another contribution is the development of a unique wrapper-based image segmentation algorithm. We have adopted a paradigm where an image is initially region labeled and, then, using proven feature selection methods, we group these regions based on our knowledge of the desired object being segmented. This algorithm is shown to provide segmentations as accurate as human hand segmentation in many cases.

There are also two contributions to the area of human motion tracking. The first is the definition of an information theoretic motion segmentation algorithm. This approach appears immune to illumination effects, and it dramatically changes the way we perceive image motion through information flow rather than optical flow.

Lastly, we have developed an integrated motion and shape tracking system based on interacting multiple models (IMM) Kalman filtering. The approach is superior to HMM-based tracking systems, since it intelligently blends the individual dynamics states. Also, the system has been shown to be able to react to high-speed motion events, such as a pre-crash braking event. As part of the tracking system, we have also defined a new mechanism for inferring the 3-dimensional pose of the occupant based on the gross changes in their shape during motion, called shape from deformation.

To My Children: Melissa, Bridget, and Patrick – *May they never lose their children's fascination with the world.*

ACKNOWLEDGEMENTS

The completion of this dissertation marks the fulfillment of a lifelong ambition. As such, there are many people and organizations to which I am extremely grateful. First and foremost, I would like to thank Professor Anil K. Jain for being my advisor. He has been a great teacher, mentor, and friend through the course of this degree program. I would like to thank all my other committee members, Professor Anthony Wojcik, Professor George Stockman, and Professor Sarat Dass for their valued contributions, wonderful encouragement, and thought-provoking questions throughout the development of this dissertation. Additionally, I would also like to thank the entire staff of the Michigan State University Computer Science and Engineering department for developing such a great PhD program.

There have been numerous "professors" in industry as well, who have provided me tremendous motivation throughout my career. Some of the wonderful people whom I have met through my work experience include: Dr. Marty Wolfson, Dr. William Rubin, Dr. Ken Matysik, Dr. Owen Davies, Dr. David Turner, and Dr. John Flatter. I greatly appreciate all the encouragement, technical guidance, and camaraderie they have provided over the years.

This degree would not have been possible without the incredible support of Eaton Corporation. They provided me not only the usual tuition assistance, but also allowed me to use my technical contributions on the Eagle project as the foundation of my dissertation work. Additionally, within Eaton, there were so many great people associated with the Eagle project whose assistance was invaluable towards developing

that system, especially the project manager Joel Hooper, and the systems, hardware, and test principals, Mark Dell'Eva, John Prainito, and Al Hutchenreuther. There were numerous other Eagle team members that provided help in implementing the algorithms described in this dissertation, including: Shweta Farmer, Xunchang Chen, Li Wen, Nikhil Hoskeri, and Pravene Chandrasekaria. Special thanks also to the teams of technicians, but especially Mark Beaupre and Dave Shock, for being willing to collect yet one more set of images for me in the freezing cold of winter or the sweltering heat of summer, and, of course, Ed Frank, for always managing to find me one more computer for my cluster. Additionally, this dissertation would never have been completed were it not for the endless café mochas at the West Bloomfield Barnes and Noble café.

Finally, I would like to thank my children, Melissa, Bridget, and Patrick, for the hours where, while I did homework or read papers, they kept me company by coloring or doing their own homework. With my deepest gratitude I would like to re-thank Shweta, now my beloved wife. She was such a source of constant support, encouragement, and motivation through this effort, initially as a classmate, then as a co-worker, and ultimately as a truly great wife. This dissertation would have been infinitely more difficult without her companionship and love. I would like to close by thanking my parents: my mother for instilling in me the importance of education, and my father for teaching me if your going to do something, then be sure do to it the best that you can.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1.
Introduction

The integration of airbags into passenger vehicles during the 1980's and 1990's has been particularly effective in reducing the number of highway fatalities in the United States. This is primarily due to the fact that automobile passengers in the United States are less likely to wear their seat belts while driving. The airbag is deployed when the vehicle experiences a crash greater than 14 miles per hour, and the airbags were designed to protect a 95^{th} percentile adult male during a 30 mph crash [12]. Clearly, the force required to protect such a large occupant is much greater than is required for children. There are also instances when the airbag may be inadvertently inflated if the crash detection module experiences a false detection. These cases can occur when there is an electronics failure in the module, or in other events such as hitting a parking block. In these cases, the airbag can be quite dangerous to the occupant, particularly if the occupant is a child. In these cases it would have been safer if the airbag had been disabled.

1.1 Smart Airbag Suppression

There has been considerable attention paid to developing 'smart' airbags that can determine not only if they should be deployed in a crash event, but also with what force they should be deployed [1][2][3][4][5][6][7][8][9][13]. The size and type of

automobile occupant is used to determine the safe level of force with which to deploy the airbag.

In May 2001 the U.S National Highway Transportation and Safety Administration (NHTSA) defined the Federal Motor Vehicle Safety Standard (FMVSS) 208 that mandates automatic airbag suppression when an occupant smaller than a 6 year-old child is in the passenger seat, while enabling the airbag when the occupant is a 5th percentile (by weight) adult female or larger [12]. This standard was proposed because a number of small children and infants have been killed by airbags. Between 1986 and 2001, 19 infants and 85 children were killed by airbags [13]. Figure 1.1 shows the effects of an airbag on a RFIS during an airbag deployment [15]. During the same timeframe (1986-2001), seven adult passengers were also killed by airbags, due to being too close to the airbag at the time of deployment [13].

(a) (b) (c)

Figure 1.1: Effect of airbag deployment on rear facing infant seats (RFIS). (a) RFIS prior to deployment, (b) RFIS during deployment, and (c) RFIS after deployment [11]

Despite the publicity regarding the dangers of airbags, there are still a large number of people that place children in the front set. Figure 1.2 shows the results from a 1998 NHTSA survey of drivers, where over 50% of the respondents had

placed their children in the front seat of their vehicle within the last 30 days prior to the date of the survey [14]. Clearly, the public awareness of the danger of airbags on children has not led to a dramatic change in behavior, and, consequently, there is a need for a system that will automatically disable the airbag when the occupant is at risk of injury from the deployment of the airbag.

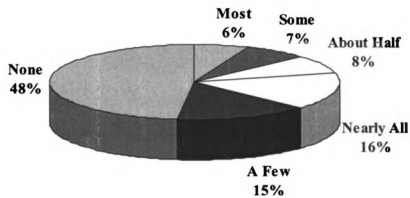


Figure 1.2: Relative frequency with which children are placed in the front seat in automobiles [14].

The goal of this thesis is to show the applicability of computer vision (using a single black-and-white camera) in solving the airbag suppression problem. We highlight some of the difficulties of the airbag suppression problem, propose a novel system framework, and show the resulting performance.

1.1.1 Overview of the Airbag Suppression Problem

The various types of occupants defined in the NHTSA specification include [12]:

1) infant (12 month olds in rear or forward facing infant seats),
2) child (< 56 lbs in weight & < 49 in. tall), includes children on convertible seats and booster seats,
3) adult (> 103 lbs in weight & > 55 in. tall), and
4) empty seat.

Figure 1.3 shows examples of each of the four classes, while the occupant sizes are provided in Table 1.1 [12].

Table 1.1 : NHTSA occupant size categories in FMVSS 208 [12].

Occupant Type	**Occupant Measurements**			
	Height (inches)		**Weight (lbs)**	
	Min	**Max**	**Min**	**Max**
3 year old	35	39	29.5	29.5
6 year old	45	49	46.5	56.5
5th percentile female	55	59	103	113
50th percentile female	69 inches nominal		171 lbs nominal	

Note there is a considerable gray zone between the child and the adult where there is no defined occupant size. This region is composed of children larger than six year olds but smaller than adults. The safety of the occupants in this range are not ensured by simple suppression based on occupant type and are much more safely handled by a dynamic suppression capability. Consequently, there are two types of

suppression defined by NHTSA: (i) *static suppression*, and (ii) *dynamic suppression* [12].

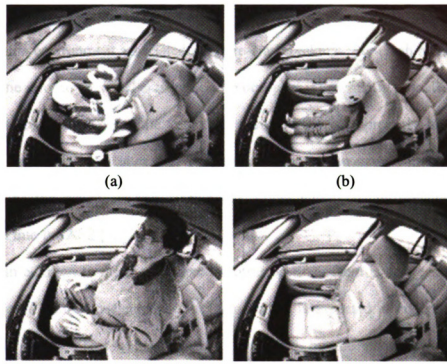


Figure 1.3: Examples of each of the four classes of occupants in a vehicle, (a) infant, (b) child, (c) adult, and (d) empty seat.

Static suppression refers to disabling the airbag based on recognizing the occupant type (i.e., infant, child, adult, empty seat). In static suppression the system is disabled if an occupant is too small to safely withstand the forces of the airbag. For the infant occupant, it is never safe to deploy the airbag. However, for children, and clearly for adults, there are cases when it may be safe to fire the airbag assuming the occupant is not too close to the airbag.

Dynamic suppression refers to the situation where the occupant is seated too close to the airbag for it to be safely fired. The region within which the occupant may

be injured by the airbag is called the automatic suppression zone (ASZ). It is defined as a vertical plane that is roughly 8 to 10 inches from where the airbag door is located in the instrument panel. Figure 1.4 shows the safe zone for a small stature adult female. The NHSTA specification for dynamic suppression addresses the acceleration environment associated with pre-crash braking. Pre-crash braking refers to the time prior to an actual crash when the occupant is potentially propelled into the ASZ due to the braking of the vehicle if the occupant is improperly belted. The NHTSA specification dictates that the occupant suppression system must detect the position of the occupant, and disable the airbag within 20 milliseconds of the intrusion into the ASZ [12].

In addition to the requirement of intrusion time, there is also a requirement regarding the actual accelerations that the occupant is expected to undergo during a pre-crash event. This value is usually specified by the particular automotive manufacturer, and is therefore proprietary, however, the typical range of specified accelerations have varied between 0.85 g and 1.2 g. These values can be confirmed in Figure 1.5, where actual high-speed sled tests of crash test dummies were used to synthesize pre-crash and crash events. In this figure the decelerations due to the pre-crash braking converge to roughly 1.0 g.

Figure 1.4: Safe distance for a small stature adult occupant in relation to the airbag [11].

The dynamic suppression system must be capable of not only detecting if the occupant is in the ASZ, but also determining if the occupant is leaning towards the driver or towards the window. These leaning positions will place the occupant in such a way that his head is no longer directly in front of the airbag, in which case he is no longer at the risk of injury and the airbag should be deployed. This ensures that the airbag is deployed as often as possible, given a safe occupant position. It is important to note that the goal of the system is to not only minimize the risk to the occupant from the airbag, but also to minimize the risk to the occupant during a collision. This places strict requirements not only on detection of intrusions, but also on the minimization of false alarms due to inadvertently detected intrusions.

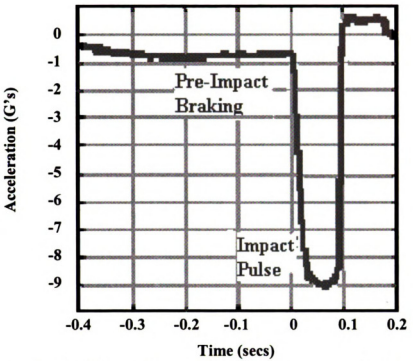


Figure 1.5: Definition of dynamics typical for a pre-crash braking event followed by an actual crash event [16].

There are multiple combinations of dynamic and static suppression schemes that are allowed by NHTSA to afford protection to all the occupant classes [12]. Figure 1.6 shows the breakdown of the possible uses of each of the suppression methods. We investigated two of the possible combinations: (i) suppress the airbag on all infant and children while enabling the airbag on all adults, and (ii) suppress the airbag only on infants and enable on all other occupants.

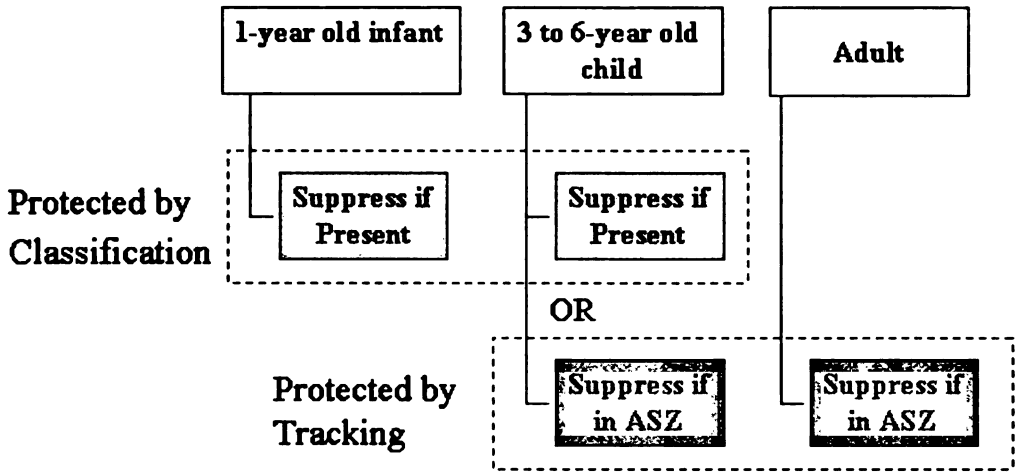


Figure 1.6: Overview of the possible mechanisms for suppressing the airbag depending on the occupant type.

1.1.2 Alternate Technologies for Airbag Suppression

The NHTSA specification is written around the use of a seat weight sensor, which measures the weight of the occupant and infers whether the occupant is of sufficient weight to be an adult. Consequently, weight sensors only provide static suppression. In addition to weight sensors, there have been numerous other technologies proposed for occupant sensing. Most are designed either for static or dynamic suppression, but are not capable of both [1][2][3][4][5][6][7][8]. The systems that claim capability for both types of suppression require either multiple sensors for estimating 3-D positional information [1][2][9], or else require high-speed distance measuring devices [1].

A search of the United States Patent Office website for patents related to the occupant position, weight, and classification sensing resulted in over 600 patents in this area, with the number growing due to the interest generated by the NHTSA

regulations [10]. We will discuss three of the more popular technologies defined in these patents. These technologies are:

1) Capacitive sensors,

2) Pressure and weight sensors, and

3) Time of flight sensors (ultrasound and infra-red beams).

A Capacitive Sensors

There are a number of possible implementations of capacitive sensors depending on the sensing objective [4]. If the goal is static suppression then the sensor can be mounted either under the seat bottom and seat back or on the roof-liner over the occupant. The sensors mounted in the roof liner are basically adult detectors and look for a capacitive effect some distance above the seat. Since an adult is taller than a child, the system detects adults and no other occupant. In the seat-mounted implementation the sensor detects the presence of the occupant in the seat, and then determines the size of the occupant (for instance if it is an infant) based on the strength of the signal from the seat back [4]. As an adult occupant leans forward, the signal from the seat back will be erroneous, and the occupant may be mis-classified as an infant.

For dynamic suppression, the capacitive sensor is mounted in the instrument panel in front of the occupant. As the occupant leans forward, he creates a signal in the capacitive sensor. By sampling the signal at a fast enough data rate, the system is able to detect intrusions into the ASZ. Consequently, for capacitive sensors to provide both static and dynamic protection, multiple units are required in the vehicle.

The capacitors function by detecting the change in voltage caused by an occupant or an object on the seat. Since the human body is mostly made of water, it has a dramatic effect on the capacitance compared to inanimate objects. The same effect that allows the sensor to detect humans also causes failure modes. Since the sensor detects the moisture in the volume above the seat, the sensor may be fooled by a wet towel or spilled juice on the seat, which could make a child appear as an adult.

B Pressure and Weight Sensors

Pressure and weight sensors are placed in the seat to detect the relative force that the object exerts on the seat [5][6]. The force is simply the mass times gravity. Likewise, the pressure sensor is simply force per unit of area. An infant seat or a child will apply less force to the seat bottom and, therefore, allow the sensor to distinguish an adult from a child by simply thresholding the weight measurement. Figure 1.7 shows how a typical weight sensor is integrated into the seat of a vehicle.

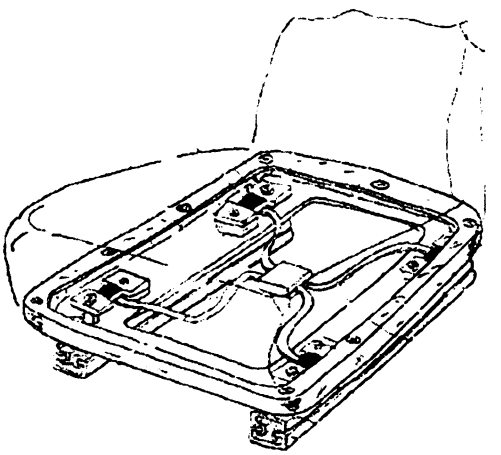


Figure 1.7: Example of weight sensors placed in the seat bottom for static classification [6].

There are many possible implementations of these types of sensors. Two of the most common include (i) strain gauges, and (ii) gel-filled bladders with pressure transducers. In the strain gauge approach a set of gauges are placed under the seat, where the seat springs connect to the seat base (called the seat pan). By measuring the average strain exerted on the four corners of the seat, it is possible to infer the weight on the seat. The pressure sensor with the bladder works like squeezing a balloon. As the weight of the person is increased, the force with which he presses on the bladder increases. There is a tube from the bladder to the pressure transducer, and the weight of the occupant is inferred from the pressure experienced at the transducer.

C Time-of-Flight Sensors

Time-of-flight sensors are most useful for dynamic suppression. However, by building arrays of these sensors it is also possible to create a system that can be used for static suppression as well. The basic approach is to transmit a signal and compute the time it takes for the signal to return to the sensor. In the simplest implementation there is a single sensor that measures the distance between the occupant and the ASZ. If the occupant is too close to the instrument panel then the airbag is disabled. This is clearly a dynamic suppression sensor only. The sensors are usually placed in the instrument panel or along the roof liner by the windshield as shown in Figure 1.8.

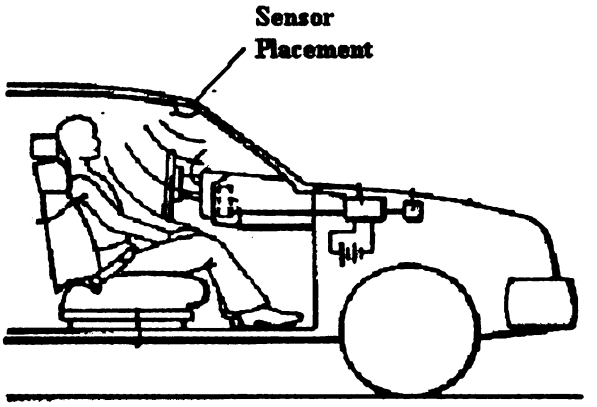


Figure 1.8: Typical installation location for time of flight sensors for occupant sensing [2].

In a more complex implementation there is an array of sensors each pointing to a different location on the seat. By measuring the distance to multiple points on the occupant, a range profile can be generated for that occupant. Infants and adults would have a significantly different range profile, and, therefore, the system can now also be used as a static suppression system. Typically, a large number of such sensors are required to generate a high-resolution range profile needed to adequately classify all of the occupant types. There are three possible mechanisms for time of flight sensors:

1) ultrasound,

2) infrared, and

3) radar.

Ultrasound systems provide a low cost solution to the ranging problem. They are currently in mass-production in the automotive industry for back-up warning sensors, which means they are already designed for the harsh automotive environment. Ultrasound is also considered safe for humans since they are widely

used in medicine. Infrared systems use a high power LED that is typically pulsed to reduce the effects of sunlight on the receiver. The receiver is timed to collect data over a specified time window. Infrared systems still have difficulty in some sunlight conditions due to the high level of background clutter (sunlight) that the system experiences. The infrared sensors must be designed with the human eye safety in mind, which limits the allowable peak power and therefore limits the performance against sunlight. While some experiments have been performed with radar, it is considered the least favorable technology due to unknown effects of long-term exposure to humans, particularly to infants.

1.2 Occupant Protection Using Computer Vision

As mentioned in the previous sections, there are two aspects to the automotive airbag suppression problem, static suppression and dynamic suppression. The problem of static occupant protection is a classic application of vision-based pattern recognition, while the problem of dynamic occupant protection using computer vision very closely parallels the work done in recognizing human behavior in video sequences. For our application, the critical human behavior is the possible intrusion by the occupant into the ASZ.

The design of a pattern recognition system requires three basic components [27][21]:

1) data acquisition and pre-processing,

2) data representation, and

3) decision making.

The data acquisition stage in this system consists of the imager module in the hardware subsystem, which will be defined in Section 1.2.1. The remaining components, including pre-processing, data representation, and decision-making are all elements of the software component of the system and will be discussed in section 1.2.2.

1.2.1 Vision System Hardware Architecture

The system hardware architecture for the airbag suppression system is provided in Figure 1.9, and is physically composed of a single monochrome digital CMOS camera, a wide field-of-view lens, a bank of LED illuminators, a digital signal processor (DSP), and a control microprocessor. The DSP performs the image processing functions in real-time. The microprocessor is responsible for system diagnostics and communications with the other vehicle subsystems via the vehicle bus, and for providing the airbag suppression signal to the vehicle airbag control module.

The camera is a semi-customized camera. It is a VGA resolution camera (640x480) that we down-sample to 400x320 pixels to reduce the processing burden. The camera has an 8-bit output, and also has three unique features: (i) extended dynamic range, (ii) automatic gain adjustment, and (iii) 10-bit internal histogram equalization. The extended dynamic range is a pseudo-logarithmic response as is shown in Figure 1.10 [169]. This dynamic range allows the system to operate in the

extreme outdoor illumination environment that has roughly 106 dB of dynamic range [169]. Clearly, the optimal camera for an automotive application would have an 18-bit linear output (6 dB per bit). There are imager chips under development with 16-bit linear responses, but the current state-of-the-art camera has only an 8-bit response, providing only 48 dB of dynamic range [169]. The logarithmic response compresses the higher amplitude pixels to ensure the lower amplitude pixels are still visible, thereby allowing the camera to support a significantly greater dynamic range. Another feature of the camera is the automatic gain adjustment that minimizes the number of saturated pixels.

The final camera feature is the internal histogram equalization. It can be argued that histogram equalization as a post processing technique is really only useful for human image display. However, in this camera, the imager device maintains a 10-bit internal representation of the image and performs the histogram equalization on this representation. The camera then drops the lower two bits and outputs an 8-bit image. This provides the camera with additional ability to extract lower amplitude features as shown in Figure 1.11.

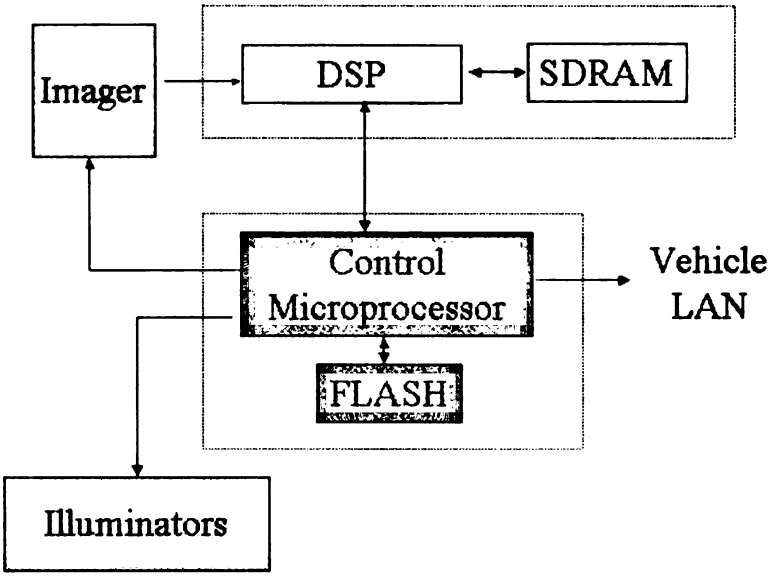


Figure 1.9: System hardware architecture of the computer vision based smart airbag suppression system.

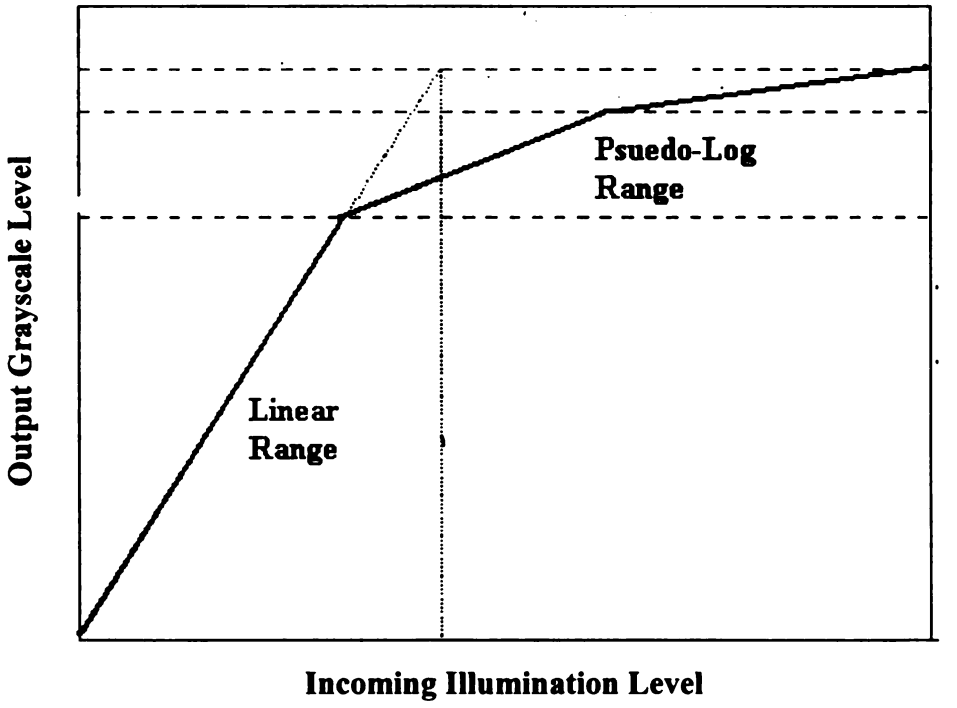


Figure 1.10: Typical amplitude response of an imager with a pseudo-logarithmic amplitude response for outdoor operation.

The camera is also manufactured without the standard IR filter to allow the camera to use the supplemental IR illumination in dark conditions. The illuminators consist of an array of 880 nanometer LEDs, geometrically configured to provide roughly a uniform illumination over the passenger area of the vehicle. The illuminators also contain a diffuser to ensure eye safety and to ensure uniform light levels across the occupant.

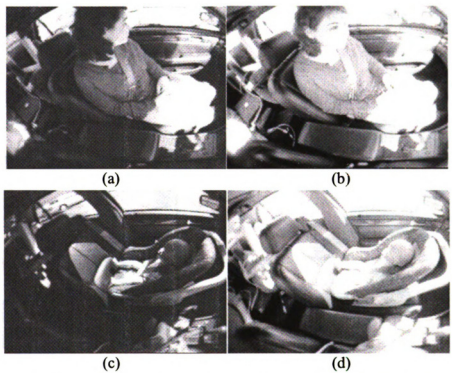


Figure 1.11: Effects of internal histogram equalization, (a) adult in outdoor illumination without equalization, (b) image in (a) with equalization, (c) infant in outdoor illumination without equalization, and (d) image in (c) with equalization.

The system is located in the roof liner of the vehicle, along the vehicle centerline, and near the edge of the windshield, as shown in Figure 1.12. This location provides nearly a profile view of the occupant in the passenger seat. This viewing angle is ideal for detecting the intrusion of the occupant into the ASZ, and

also aids in the classification of the occupant. This location also reduces the likelihood of the occupant blocking the sensor and makes aesthetically pleasing styling of the sensor into the vehicle possible.

The typical field-of-view (FOV) required for most passenger vehicles is roughly 100 degrees in the vertical direction and 120-130 degrees in the horizontal direction. This FOV ensures complete coverage of the occupant's head from when the occupant is leaning forward towards the instrument panel to when the occupant is seated in the rear-most seating position with the seat fully reclined as shown in Figure 1.13.

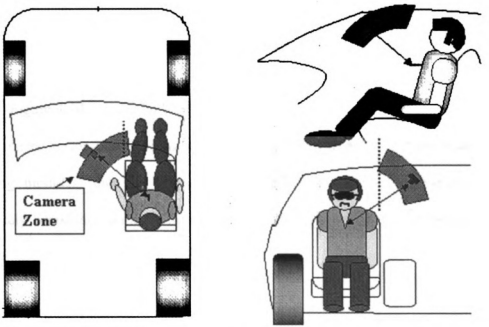


Figure 1.12: Installation of the camera system within the vehicle showing plan, profile, and forward viewing angles.

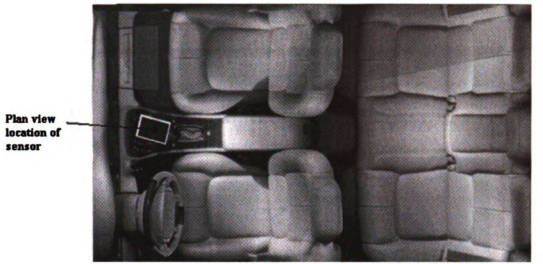


Figure 1.13: Resultant field of view within vehicle cabin is shown shaded, and the plan view location of the sensor is shown as well.

1.2.2 Vision System Software Architecture

Figure 1.14 shows the block diagram of the software algorithms required for the proposed vision based airbag suppression system. In both the classification and tracking modules, there is pre-processing, followed by data representation and ultimately decision-making. The mechanisms through which these three stages are performed are outlined in Table 1.2.

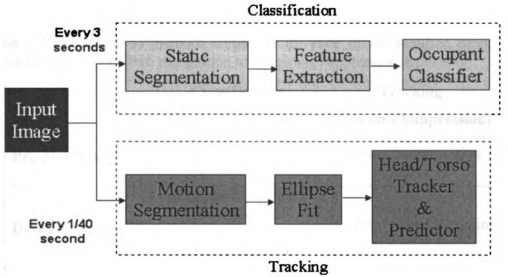


Figure 1.14: Software architecture for the computer vision based smart airbag suppression system.

Operationally, the software initially performs a classification of the incoming imagery. If the image is of the proper pattern class then the dynamic suppression processing is enabled, and the occupant's position is tracked. The classification processing is executed every 3-5 seconds to meet the NHTSA specification [12]. The dynamic suppression processing is executed at the frame rate of the camera sensor, but provides higher speed outputs to the airbag control module to ensure reporting the intrusion into the ASZ within 20 milliseconds of the actual intrusion event [12]. The goal of our system will be to detect the intrusion within 10 milliseconds to provide a significant margin of error.

Table 1.2: Summary of methods employed for each of the required processing stages for applying pattern recognition to the airbag suppression application.

	Classification (Static Suppression)	**Tracking (Dynamic Suppression)**
Pre-processing	Static Segmentation (Chapter 2)	Motion Segmentation (Chapter 7)
Data representation	Feature Extraction (Chapter 3) Feature Selection (Chapter 4)	Occupant Shape Modeling (Chapter 8)
Decision-making	Occupant Classifier (Chapter 5) Integrated Classifier/Segmenter (Chapter 6)	Occupant Tracking and ASZ Intrusion Prediction (Chapter 9)

A Static Occupant Protection Using Image Classification

According to the NHTSA specification, the system must determine the classification of the occupant within 10 seconds of a change of occupant state (i.e., from empty to adult or infant to empty, etc.) [12]. In addition to this time constraint, the NHTSA specification also requires 100 percent correct classification, for a subset of possible seating positions, and for all of the occupant types. Recall from Figure 1.6 that there are two classification approaches that must be considered. The first approach is where we disable the airbag on infants, children, or empty seats while we enable on adults. This produces is a four-class problem where the four classes are: (i) infant, (ii) child, (iii) adult and (iv) empty seat. The second approach is where we

disable on infants and enable on all other occupants. This produces a two-class problem where the two classes are: (i) infant and (ii) all other occupants. In the second scenario the children and the adults are protected by dynamic suppression. The occupant protection flow is demonstrated for these two suppression schemes in Figure 1.15. Likewise, examples of each of the occupant types for the two-class and the four-class problem are shown in Figure 1.16.

The underlying technology for the static suppression is image-based pattern recognition. There are four common approaches for designing a pattern recognition system, namely [27]:

1) template matching,
2) statistical classification,
3) syntactic or structural matching, and
4) neural networks.

It is arguable that neural networks are really a subclass of statistical pattern recognition, so they will not be considered as a separate methodology. Actually, it has been shown that "most of the neural network models are implicitly equivalent or similar to classical statistical pattern recognition methods" [27].

We will not consider template matching for this application since defining a suitable set of templates for all the occupant types is extremely difficult, when we consider all the possible sizes and postures of the occupants. This multitude of templates creates issues with the resultant storage of the required set of templates, as well as the resultant processing to test all of these templates.

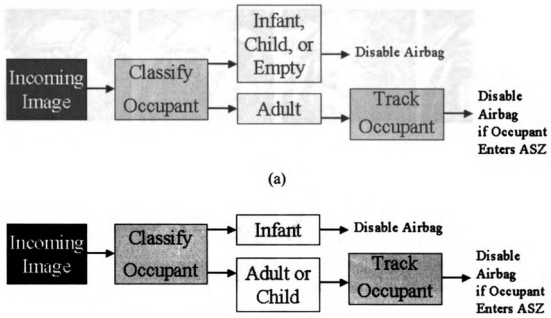
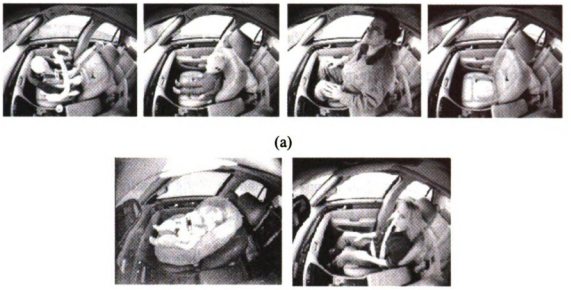


Figure 1.15: Decision flow for occupant classifier, (a) four-class problem, and (b) two-class problem.

We will also not consider structural methods. These methods rely on the successful identification of components that comprise the object of interest. In our application, it would require us to detect and isolate the head, the torso, the limbs, etc. and then develop a structural model with which to reassemble these components. We believe it would be extremely difficult to identify the components of the occupant in our application. For example, when an adult female occupant with long hair turns to look out the passenger window we would only see the hair and not any head feature. Consequently, in this thesis we will apply statistical pattern recognition, due to the difficulties associated with the other two methods.



(a)

(b)

Figure 1.16: Class decision boundaries for (a) four-class, and (b) two-class problems.

B Dynamic Occupant Protection Using Image Sequence Tracking

Dynamic occupant protection is required for the occupant categories that allow the airbag to be deployed to monitor the location of the occupant relative to the ASZ, and disable the airbag if the occupant is inside this region. Figure 1.4 shows a typical safe distance between an occupant and the airbag for a small stature adult (5^{th} percentile female) [15]. Recall that the NHTSA specification dictates that the occupant suppression system must detect the position of the occupant, and disable the airbag within 10 milliseconds of the intrusion into the ASZ [12]. This implies the system must estimate the occupant position at the rate of 100-200 Hz. Clearly, this is beyond the ability of most commercially available imaging chips. This update rate is

achievable if the tracking system can predict the positions of the occupant into the future at the rate of 10 milliseconds or higher.

The approach for computer vision-based dynamic occupant protection very closely parallels the work done in tracking humans in video sequences to infer their behavior. Applications of human motion tracking range from biomechanics analysis to wide-area surveillance applications. As shown in Figure 1.17, the key research areas in human motion analysis are [112]:

1) body structure analysis,
2) human motion tracking, and
3) human motion recognition.

Within this hierarchy, our dynamic suppression system addresses two of these elements: (i) human motion tracking, and (ii) human motion recognition. The classes of motions we will recognize are associated with the transition between a resting, or normally moving adult, and an adult being propelled into the ASZ due to sudden pre-crash braking. We model this range of motions by three distinct dynamics states [18]:

1) stationary,
2) human-like, and
3) pre-crash braking.

The stationary mode is when the person is sitting very still or is asleep. The human-like motion mode is when the occupant is moving about the seat in typical movements inside an automobile, such as opening the glove box, turning towards the driver or back seat, etc. The pre-crash braking mode represents the dynamics the

occupant would experience when the vehicle is being stopped at or near the full force of the brakes.

Figure 1.17: Breakdown of the underlying methodologies that support human motion analysis [112].

The ability of the tracking system to follow the subjects and accurately predict their future locations is critical to this application. This requirement is complicated by the fact that humans frequently transition between these typical motions [18]. The tracking system must address this requirement. Additionally, since the system must determine the position of the occupant in a 3-dimensional space, a pose estimation algorithm must be employed.

There are two general system architectures for performing human motion tracking as shown in Figure 1.18: (i) single view and (ii) multiple perspective systems [112]. Single view systems correspond to systems with a single camera, while multiple perspective views generally imply multiple cameras, but can imply a single camera that is moved over time. For system simplicity and cost reasons, only single camera systems will be addressed in this research.

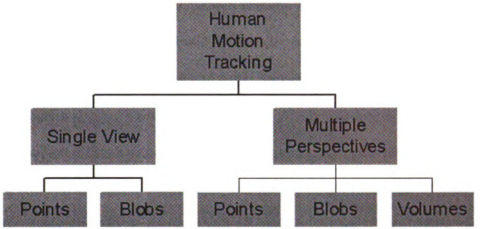


Figure 1.18: Breakdown of the methods of tracking humans in a video sequence [112].

1.2.3 Difficulties and Challenges in Vision based Occupant Protection

The following factors contribute to difficulties in applying computer vision to any application [21]:

1) non-uniform illumination,
2) poor image contrast,
3) shadows and highlights,
4) occlusions,
5) sensor noise, and
6) background clutter.

One of the primary reasons that the use of computer vision for the airbag suppression problem is very challenging is due to the extreme lighting variations typically encountered inside an automobile. Lighting can vary from bright sunlight, causing image saturation, to complete darkness, requiring supplemental illumination. In very

bright sunlight the image may require a camera with considerable dynamic range. The simultaneous existence of shadows near the occupant's legs and bright patches due to direct sunlight on the head and torso impacts the camera's ability to properly image the scene.

Another difficult situation is indoor lighting, for example in above ground parking structures, where there is some ambient lighting. Examples of images from each of these lighting conditions are shown in Figure 1.19. This operating environment requires the camera's internal amplifiers to be operated at full gain, resulting in increased image noise. The image noise can even be severe in outdoor environments due to the extreme temperatures experienced in a vehicle. Most commercial camera devices operate best at what is commonly referred to as room temperature (25 deg C), while vehicle temperatures can easily reach 85 deg C. This increased temperature results in a dramatic increase in the noise floor of imaging device.

Partial occlusion is often a common problem since the occupant is often holding items such as newspapers, boxes, or a bouquet of flowers. The existence of background clutter is directly impacted by the fact that the vehicle is moving. This causes not only moving and stationary shadows within the image, but also contributes to a varied and complex background that is visible outside the window of the automobile [19].

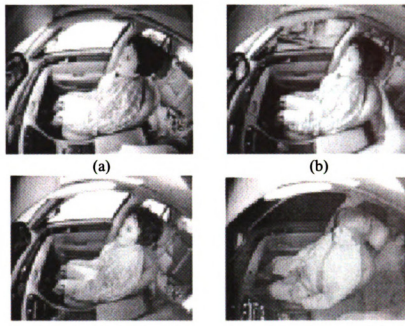


Figure 1.19: Effects of external illumination on occupant imagery, (a) bright sunlight with banding on legs, (b) bright sunlight across chest, (c) indoor lighting, and (d) night-dark lighting.

It has also been discussed that the complexity of a computer vision task can be defined by the following three factors [21]:

1) number of distinct objects to be recognized,
2) scene complexity, and
3) object complexity.

Object complexity is elaborated upon by Jain and Dorai, when they state, "the success of a vision system depends on the degree to which the designer can control the imaging environment and inter-class and intra-class object variabilities" [21]. In the airbag suppression problem, the designer has no control over the imaging environment. The airbag suppression application also suffers from the complication of very large intra-class variability.

For the child and infant classes, there are a number of seat types, as well as seating positions, that must be recognized, and the similarity between them is often not very high, as shown in Figure 1.20. One further complication is that infant and booster seats may be covered with blankets to protect the child from sunlight and cold, as shown in Figure 1.21 [12].

Figure 1.20: A collage of infant seats and child positions showing the intra-class variability for the RFIS and child classes.

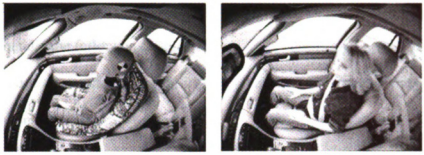
(a) (b)

Figure 1.21: Added intra-class variability for the infant class due to blankets, (a) RFIS, and (b) same RFIS under a blanket.

The adult class also has a large amount of intra-class variability, as shown in Figure 1.23, due to the following three factors [19]:

1) variability between the 5th percentile female and the 95th percentile male is 10 inches and 75 pounds,
2) variability in adult appearance due to hair and clothing variations, and
3) seasonal variability as clothing changes from only shirts in summer to down parkas and hats in winter.

In addition to the large intra-class variations, our problem is also complicated by very little inter-class separability for some of the classes. For example, the six year-old child on a booster seat is very similar in appearance to the 5th percentile adult (see Figure 1.22). The loss function for the pattern recognition system for this type of error is very high, since either the adult is not protected because the bag is disabled, or the 6 year-old is at risk. Constructing the decision boundary between the infant and the child class can also be difficult because the infants and the three-year olds share some common forward facing child seats.



(a) (b)

Figure 1.22: Low inter-class separation between the child class and the adult class, (a) 6 year-old child on booster, and (b) 5th percentile adult female.

Figure 1.23: A collage of adult images showing the large intra-class variability for the adult class.

To summarize, a vision-based system for airbag suppression must be robust enough to handle the following conditions:

1) large intra-class variability of the various occupant types under consideration,
2) camouflaged classes (e.g., blanketed infants),
3) large variation in light levels (day to night),
4) large lighting variations within an image (shadows to bright direct sunlight),
5) severe automotive environmental conditions,
6) low cost, and
7) extremely high reliability and performance.

1.3 Thesis Contributions

This thesis makes contributions at a number of levels in the area of pattern recognition and computer vision. It presents a detailed design for an occupant protection system that can dramatically improve the safety of children in automobiles. It also describes a very difficult real-world application for which monocular computer vision that is capable of simultaneously performing occupant recognition and occupant tracking has not previously been applied. The airbag suppression system developed in this thesis has been extensively evaluated on several thousand images and also during multiple field trials of vehicles.

The specific contributions in the field of pattern recognition are in the four key areas. First, we define a variant of the k-nearest neighbor and the nearest mean decision rules, where the k-nearest samples for each class are used to compute the mean of the k-nearest neighbors from each class. This differs from the commonly used nearest mean decision rule that computes the mean from the entire set of training samples. It is a dynamic formulism of the nearest mean algorithm, where the mean is computed from only the k-nearest local training samples rather than the entire dataset.

The second contribution in pattern recognition is the use of the Mann-Whitney statistic to define a robust and statistically well-founded means for performing feature selection. The method is compared to a number of filter-based and wrapper-based methods, and is shown to perform as well or superior to all of them, with sometimes significantly reduced processing. It also provides a robust statistic for defining the quality of discrimination of the selected features, rather than simply using the probability of classification error as the sole criterion function.

The third contribution in pattern recognition is the definition of a contextual processing mechanism that uses the continuous stream of classifications from a video image sequence to improve the overall classification accuracy. The method is based on evidential reasoning and Dempster-Shafer statistics. It utilizes the important concept of "ignorance" to allow the system to deal with low quality classifications due to illumination changes and the varying postures a human occupant may assume while seated in a moving vehicle.

The fourth contribution is the development of a new paradigm for image segmentation. In this approach segmentation and classification are not viewed as

sequential procedures where the segmenter is like an intelligent filter for the image, but rather they are viewed as an integrated and iterative processes. The model for this approach to segmentation is similar to the wrapper approach in feature selection, where the classification result plays an integral role in choosing the features. In this wrapper implementation of segmentation, the quality of the classification results is used to define the final segmentation.

The thesis also makes contributions in human motion tracking to support the dynamic airbag suppression functionality. The first contribution is the demonstration of how to use mutual information to provide an extremely robust motion segmentation algorithm that is highly immune to illumination effects. The approach is based on a fundamentally different way of viewing optical flow. It views optical flow as a flow of information through an image sequence, rather than simply as a flow of grayscale amplitudes.

The second contribution in the area of human motion tracking is in the formulation of the interacting multiple models (IMM) Kalman filter as a robust means of tracking humans in video sequences. Previous work in human motion tracking has been primarily focused on using simple Hidden Markov Models (HMMs) for the sequence analysis of humans in video. HMMs, however, do not provide the means to adequately deal with non-instantaneous transitions between states. The video sequence in our application is captured at a rate that far exceeds the motion rates of humans and, therefore, it contains large time intervals of transitional motions. IMMs provide a natural means of handling transitional motions by not requiring the system

to select a single state, but rather to probabilistically combine multiple states to describe these transitional events.

The third contribution in the area of human motion tracking is a new mechanism for inferring 3-dimensional pose from a sequence of 2-dimensional images through structure from deformation. While it is related to structure from motion, it benefits from not requiring precise correspondences of feature points, but rather infers the pose from global effects of the object's projection into the image plane. It also uses the concept of interacting multiple models (IMM) and extended Kalman filtering to provide the mathematical engine for providing the pose estimation.

Chapter 2.
Segmentation for Occupant Classification

We adopt the traditional image classification methodology for our airbag suppression application, where the first stage of processing is image segmentation, which is followed by object recognition [82]. Image segmentation has been extensively studied in computer vision, and there are "hundreds of segmentation techniques in the literature, but there is no single method which can be considered good for all images" [82]. Additionally, "semantics and prior information about the type of images are critical to the solution of the segmentation problem" [82]. Unfortunately, we have no obvious common attributes for our occupant classes that we can use for segmenting the occupants from the background.

The large intra-class variability of our occupant classes makes defining a set of common characteristics for segmentation extremely difficult. We are, however, in a fixed and structured environment, namely inside of a passenger car. In light of this, we will utilize all the available information regarding the interior of the vehicle in the segmentation process. Therefore, rather than trying to segment the occupant from the background, we will try to subtract all of the known background information from the input image. The remaining pixels will then belong to the occupant. This is the well-known approach, called background removal.

Since the background subtraction is never perfect due to the illumination effects, the output image from the background subtraction is further processed by

additional image enhancement techniques. Both the background removal and the image enhancement techniques will be addressed.

2.1 Background Removal

In background removal, an image without the presence of an occupant, or even the passenger seat, is required. The incoming images are compared to this known "empty" or "reference" image, and occupants are detected by detecting the difference. An empty reference image is collected for each of the three typical lighting conditions (daylight, indoor light, and night). These images are carefully collected to ensure there are no bright highlights or shadows in the image. The same camera system that is used for the airbag suppression is also used to collect the reference images to ensure identical camera characteristics. Examples of the empty reference images for each of the three lighting conditions are shown in Figure 2.1. Note that the vehicle seat has been removed in these images. This is due to the fact that the seat can be located in a variety of positions (e.g., fully forward and upright to full rearward and reclined) and, therefore, cannot easily be fully removed from the scene during background subtraction.

There are a host of methods defined for background subtraction [82][76][115][116]. One uses image-to-image correlation and finds the regions of the image that appear different from the reference image. Another method uses the concept of eigen-images to subtract a representation of the empty image from the incoming image to reveal the new objects in the image [116]. Finally, another

common method uses statistical modeling to develop a model of each of the background pixels [115][79]. It then compares each pixel in the incoming image with pixels in the model and determines which pixels belong to the foreground object. Of these common methods, we will investigate image correlation and eigen-images.

(a) (b) (c)

Figure 2.1: Empty reference images for each of the three lighting conditions, (a) indoor, (b) night, and (c) outdoor.

2.1.1 Background Correlation

The correlation approach computes the relative correlation between the incoming image and the reference image. Both the incoming image and the reference image are initially converted to gradient, or edge images to reduce the effects of variable illumination. The processing flow for the correlation-based segmentation is provided in Figure 2.2. The gradient images are computed along the two principal axes, which can then be used either as a gradient vector, or the root sum of squares (RSS) can be computed to use only the gradient amplitudes.

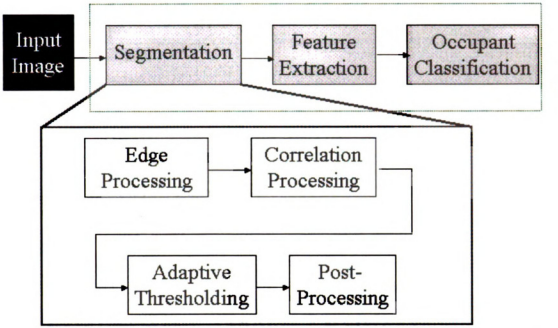


Figure 2.2: Processing flow for occupant classification, highlighting the correlation-based approach to background subtraction for segmentation.

If the edge amplitudes are used, then the correlation of an *NxN* patch from each of the two images is computed using:

$$C = \frac{\sum_A \sum_B g_1(x,y) \cdot g_2(x,y)}{\sqrt{\sum_A g_1(x,y)^2 \cdot \sum_B g_2(x,y)^2}}, \qquad (2.1)$$

where g_1 is the reference image and g_2 is the incoming image and *A* and *B* are the *NxN* regions to be correlated in the two images, as shown in Figure 2.3. For our application, we use a *10x10* window for computing the correlation.

If the edge directional components are used then the gradient information along each principal direction is preserved and a vector is produced for each pixel consisting of:

$$\mathbf{g}(i,j) = \left[grad_x, grad_y \right]^T . \qquad (2.2)$$

The correlation for the vector representation is written as:

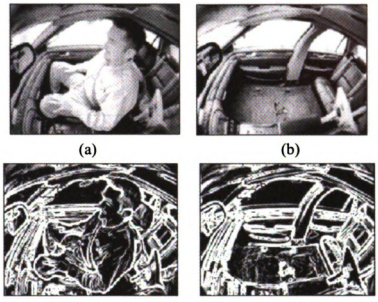
$$C = \frac{\sum_A \sum_B \mathbf{g}(x,y) \cdot \mathbf{g}(x,y)}{\sqrt{\sum_A |\mathbf{g}(x,y)|^2 \cdot \sum_B |\mathbf{g}(x,y)|^2}} . \qquad (2.3)$$

Regions of high correlation mean there is no change between the incoming image and the reference image in that region. Regions of low correlation are kept for further processing, since they differ from the reference image. Once the correlation value for each region is determined, an adaptive threshold is applied, and any region that falls below the threshold (recall a low correlation means a change from the reference image) is considered a part of the occupant. Figure 2.4 shows the incoming image, the correlation image after adaptive threshold, and the final correlation image after the largest component has been extracted.

2.1.2 Eigen-image Technique

The eigen-image technique builds an eigenspace representation of the background image, and then uses this representation to detect regions where objects of interest may be present, as shown in Figure 2.6 [116][76][78]. By including images in the training set that are from slightly different lighting conditions, it is possible to make the system somewhat robust to environmental conditions (small

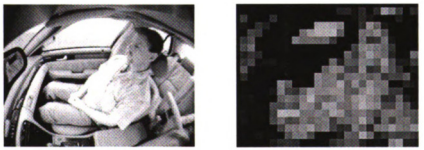
shadows, etc.) [116]. The general processing flow for the eigen-image based subtraction is provided in Figure 2.5.



(a) (b)

(c) (d)

Figure 2.3: Example of preprocessing for background correlation processing, (a) incoming image, (b) reference image, (c) edge map from incoming image, and (d) edge map from reference image.



(a) (b)

Figure 2.4: Example results for background correlation processing, (a) incoming image, and (b) after correlation processing (using edge vector).

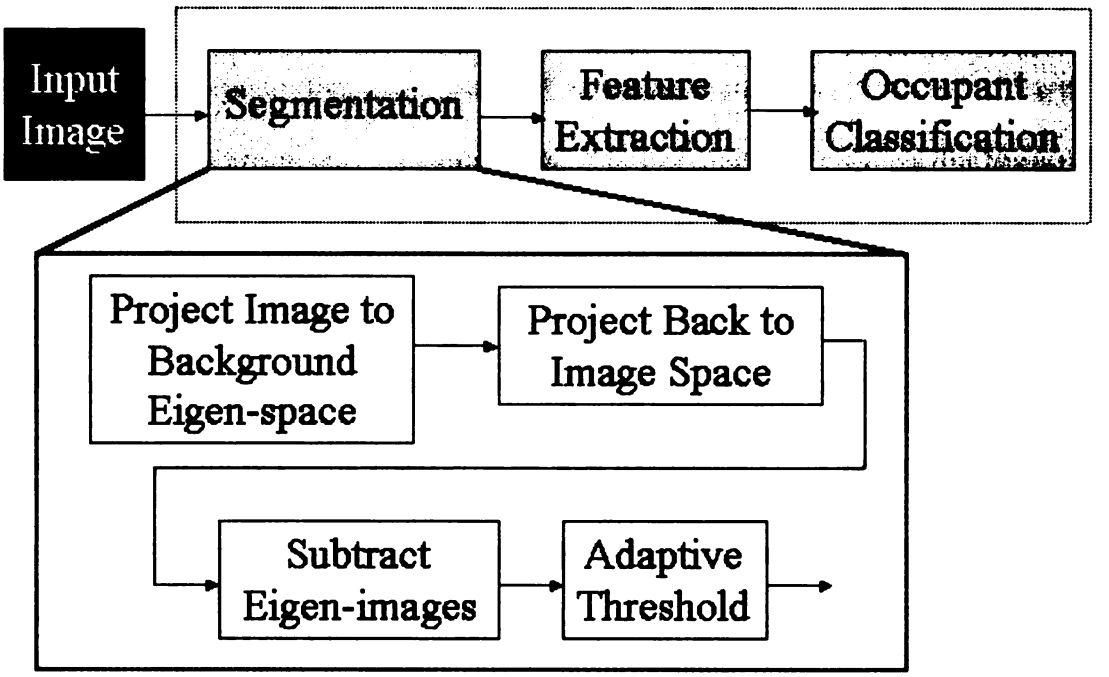


Figure 2.5: Processing flow for eigen-image background subtraction.

The eigen-image approach begins with a set of N training images which are rasterized and converted into image vectors, $\mathbf{I}_i$, $i=1,...,N$. The mean image $\boldsymbol{\mu}$ and its covariance matrix, $\mathbf{C}$, are computed from this set of images according to [116][76]:

$$\boldsymbol{\mu} = \sum_{i=1}^{N} \mathbf{I}_i, \text{ and}$$

$$\mathbf{C} = \sum_{i=1}^{N} (\mathbf{I}_i - \boldsymbol{\mu}) * (\mathbf{I}_i - \boldsymbol{\mu})^T. \qquad (2.4)$$

This covariance matrix is then diagonalized into the form:

$$\mathbf{L} = \boldsymbol{\Phi}\mathbf{C}\boldsymbol{\Phi}^T, \qquad (2.5)$$

where $\boldsymbol{L}$ is the diagonal matrix of eigenvalues, and $\boldsymbol{\Phi}$ is the matrix of eigenvectors [116][76]. The system then uses principal component analysis (PCA) to reduce this high dimensional space into a more manageable size, by only keeping the largest M eigen-vectors, where $M<N$ [116][76]. The underlying concept is that the largest eigenvalues contain the majority of the information in the background image. A background basis set that models the critical components of the background is generated from these top M eigen-images, and labeled $\boldsymbol{\Phi}_{Mb}$. Each of the M images used in this training set is then projected onto this basis according to [76]:

$$\mathbf{p}_j = \boldsymbol{\Phi}_{Mb}^T(\mathbf{I}_j - \boldsymbol{\mu}), j = 1,...,M. \qquad (2.6)$$

Thus each of these projections contains a compact representation of the incoming image, $\boldsymbol{I}$, where the size of the image representation is an $Mx1$ vector. Now any new image can be represented in this new basis by using equation (2.6) to generate an $Mx1$ representation of the image. This is accomplished via the following equation [116][76]]:

$$\mathbf{p}_{new} = \boldsymbol{\Phi}_{Mb}^T(\mathbf{I}_{new} - \mu_{new}). \qquad (2.7)$$

In order to perform the background subtraction, this projected image must then be expanded back into a full image by the inverse transform, namely [76]:

$$\mathbf{I}_{background} = \boldsymbol{\Phi}_{Mb}\mathbf{p}_{new} + \mu_{new}. \qquad (2.8)$$

In this process we take advantage of the fact that some information has been lost in the transformation, and the information that has been lost is the foreground objects (objects of interest), and only the background remains. Now the difference between the incoming image and its background projected image is compared on a pixel-by-pixel basis, and, if the difference exceeds a threshold, then a pixel is assumed to belong to a foreground object of interest when [116]:

$$|\mathbf{I}_{new} - \mathbf{I}_{background}| > threshold\ . \qquad (2.9)$$

(a) (b) (c)

Figure 2.6: Example image for eigen-background subtraction, (a) mean background image, (b) resultant moving objects, and (c) original image with bounding boxes around moving objects [116].

Olivier et al. state it is easy to modify this algorithm into an adaptive algorithm to compensate for changing shadows [116]. To do so, the system would monitor the outputs from this thresholding stage. If a large percentage of the image is deemed foreground, then it is assumed the background model is no longer valid, and a new background would be computed. Another approach could be to continuously update and replace the background model. While this would allow for a more gracefully degrading system, it would greatly increase the processing burden, since the PCA computation would be required for every new image. For our airbag

suppre

more

with e

perfor

image

image

proces

to rem

is a pc

that th

minim

the ou

region

seen ir

region

mathen

morpho

suppression application, we cannot guarantee the vehicle seat is empty. It is even more complicated by the fact that our preferred reference image is an empty vehicle with even the seat removed, which is clearly impossible. Thus we will investigate the performance of the eigen-image subtraction using a large and varied set of training images that attempts to capture all of the possible variations due to illumination.

2.2 Image Enhancement Methods

The term image enhancement used in our context is different from traditional image enhancement techniques. Traditional image enhancement techniques are pre-processing methods to facilitate image segmentation, or to enhance a human interface to remove noise and other artifacts in the image [83]. Image enhancement used here is a post processing stage that operates on the background subtraction output, such that the segmented image contains as little of the background as possible, while minimizing the erosion of the occupant region. There are two problems that arise in the output of the background subtraction segmentations: (i) holes in the occupant regions, and (ii) extraneous background regions. Examples of these effects can be seen in Figure 2.7 (c) where holes in the occupant are visible and there are extraneous regions in the area of the window and the rear seat.

The two methods investigated here for post processing are binary mathematical morphology for filling the image holes and grayscale mathematical morphology via the watershed algorithm for removing extraneous background pixels.

(a) (b) (c)

Figure 2.7: Demonstration of artifacts to be reduced by post-processing, (a) original image, (b) desired segmentation, and (c) output from the background subtraction-based segmentation.

2.2.1 Mathematical Morphology

Mathematical morphology is a very popular method for removing extraneous regions in images or for filling holes and gaps in images. The morphology algorithms we will be using are based on binary image morphology. The most fundamental approach for performing binary morphology is based on erosion and dilation operations. Other algorithms in morphology are derived from cascading these basic building blocks in a particular order. The definitions for erosion and dilation are [86]:

- Erosion of X by $B(x)$ is the set of all points x such that *all* points within $B(x)$ are included in X.
- Dilation of X by $B(x)$ is the set of all points x such that *any* point within $B(x)$ are included in X.

Dilation and erosion are represented by the mathematical terms [86]:

- *Dilation* = $A \oplus B$, and (2.10)

- $Erosion = A \Theta B$.

The operations called opening and closing can now be defined using these basic building blocks. They are defined to be [86]:

$$\text{- } Opening = \left(A \Theta \hat{B}\right) \oplus B \text{ and} \qquad (2.11)$$

$$\text{-}Closing = \left(A \oplus \hat{B}\right) \Theta B ,$$

where $\hat{B}$ is the transpose of B. The opening and closing operations are "among the most powerful of mathematical morphology" [86]. The operation we are interested in is the closing, which fills in small holes and gaps within the segmented object and fills dents in the contours of the object.

2.2.2 Watershed Technique

It is interesting to note that the watershed algorithm is traditionally considered a stand-alone segmentation technique [84]. For our application, where the occupant can be wearing any type of clothing and the image is relatively cluttered, the watershed method would not be able to completely isolate the occupant from the background. The watershed algorithm, however, is useful, for removing some of the remaining background pixels.

The watershed algorithm is a region-based segmentation approach based on grayscale morphology. The algorithm considers the image to be a landscape, and when this landscape is flooded with water, the watersheds define lines that divide

regions of flow into different catchment basins [84]. A watershed line is defined at that location, where the water from two different basins meets. This line serves as a region separator, as shown in Figure 2.8 [85].

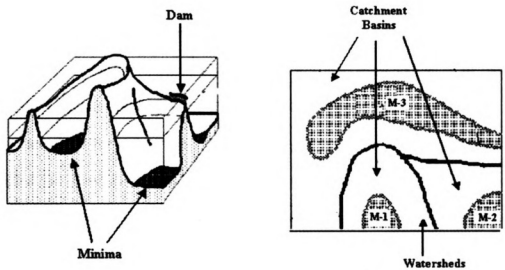
There are many algorithmic implementations of the watershed concept, for example [84]:

- watershed by immersion,
- watershed by hill climbing, and
- watershed by Dijkstra-Moore shortest path algorithm.

There are also two classes of watershed algorithms: (i) blind, and (ii) with markers. In blind watershed the image is divided into catchment basins with no other information. In the marker approach, catchment basins in the area of a marker are combined together. In the airbag suppression application we have used watershed by markers, since we want to use as much contextual information as possible for the segmentation [85]. The markers allow us to define the likely regions for the occupant and the vehicle background. We define a foreground marker and a background marker, and then begin the flow from these regions, while also merging catchment basins that fall within these marker regions.

For the airbag suppression application, we define the markers based on modeling the occupant by its bounding ellipse. This model will be defined in considerably more detail in Chapter 8, where it plays a key role in the occupant tracking algorithms. Initially, the bounding ellipse of the segmentation is defined as shown in Figure 2.9 (b). From this ellipse we then define an inner and outer ellipse based on the eccentricity and size of the initial ellipse. We know for example, that if

the initial ellipse is very circular as in Figure 2.9 (b), then we should force the marker regions to be more eccentric, since occupants tend to appear relatively elliptical. Likewise, if the ellipse is very large, we need to reduce the inner and outer ellipses, since we have reasonable a priori knowledge of the possible sizes of the occupant.



(a) (b)

Figure 2.8: Demonstration of the watershed algorithm, (a) dam building between catchment basins, and (b) graphic showing minima regions and resulting watersheds [85].

(a) (b)

Figure 2.9: Marker regions for watershed processing for the occupant shown in Figure 2.7, (a) hand segmented image, and (b) inner and outer marker regions.

2.3 Data Collection for Occupant Segmentation and Classification

An extensive data collection effort was undertaken to ensure an adequate number of images were available for the static suppression system development. We have also performed an extensive collection for dynamic suppression, which will be described in Chapter 7 when we begin defining our approach to dynamic suppression. Since the images are used for training of the final system, the training images are collected in the target vehicles, using the camera placement expected in the final production system. Likewise, the camera module used for this data collection is the same camera as for production, with the 640x480 hardware resolution, that is sub-sampled and stored as 400x320 images in 8-bit grayscale.

For the child class, images of 3 year-old and 6 year-old occupants on the vehicle seat and in the appropriate child seats (see Table 2.1) are all included in the 600 images. Since the 4-class classifier treats RFIS as a separate class, images with RFIS are collected separately. The data collections for the infant and child seats are based on the list of approved car seats in the NHTSA FMVSS-208 documentation [12]. The current list of the approved seats is provided in Table 2.1 [12].

To ensure proper collection of the imagery, we developed a user's manual to demonstrate the proper installation of the various infant and child seats, and to provide samples of the image from the camera system. An example of one of the entries in this user's manual is provided in Figure 2.10, where the installation of the infant seat is shown from the installer's perspective, as well as from the camera's perspective [17].

Table 2.1: List of NHTSA approved infant and child car seats.

	Description	Model Number
	Car Bed	
N01	Cosco Dream Ride	02-719
	Rear Facing Child Seat	
N02	Britax Handle with Care	191
N03	Century Assura	4553
N04	Century Smart Fit	4543
N05	Cosco Arriva	02-727
N06	Cosco Opus	35 02603
N07	Evenflo Discovery	212
N08	Evenflo First Choice	204
N09	Evenflo On My Way Position Right V	282
N10	Century Avanta SE	41530
N11	Graco Infant	8457
	Convertible	
N12	Britax Roundabout	161
N13	Century Encore	4612
N14	Cosco Touriva	02519
N15	Evenflo Horizon V	425
N16	Evenflo Medallion	254
N17	Century STE 1000	4416
N18	Cosco Olympian	02803
	Booster	
N19	Britax Roadster	9004
N20	Century Next Step	4920
N21	Cosco High Back Booster	02-442
N22	Evenflo Right Fit	245

Recall that since the system is an automotive application, it must be capable of operating in the following lighting conditions:

1) indoor light - conditions such as inside garages and parking structures,
2) outdoor light - normal outdoor driving conditions in bright sun, as well as other daytime weather conditions, and
3) night - night-time driving conditions.

The current training imagery was collected in indoor and night lighting conditions only. We have adopted a strategy for training the system where lighting variations should be treated as a source of noise in the image. We then attempt to develop algorithms that possess some level of immunity to illumination, rather than attempting to train the system for the illumination variations due to the near infinite variability of the illumination.

(a) (b)

Figure 2.10: Demonstration of proper installation of a child and infant seat with a 1-year old dummy, (a) installer's view of the infant seat, and (b) camera system view of the infant seat.

The training datasets must also capture the occupant variability in the population to provide generalizable results. The infant subjects in the RFIS and forward facing infant seats were realistic looking dolls used as the subjects since real infants are too difficult to work with. The three year-old and six year-old subjects were a combination of real children and crash test dummies. The adult subjects were also a combination of the 5th percentile female and the 50th percentile male crash test dummies, as well as a number of actual human subjects, ranging from the 5th percentile females to the 95th percentile males. The positions that these occupant types may assume in the image collection process are summarized in Table 2.2.

Table 2.2: Typical Occupant Positions for Data Collection.

Scenarios	3YO	6YO	3YO on child seat	6YO on booster seat	5'''%ile adult female
1. Sitting Upright	X	X	X	X	X
2. Sitting on Boosters Kept over 1" Blanket			X	X	
3. Normal Back against seat back	X	X	X	X	X
4. Back against reclined seat		X			X
5. Back not against seat back	X	X			X
6. Sitting on the seat edge	X	X			
7. Standing on the seat facing forward	X				
8. Kneeling on the seat facing forward	X				
9. Kneeling on the seat facing rearward	X				
10. Lying on the seat (If applicable)	X				
11. Leaning on the door	X	X			
12. Rotate 30 Degrees CW			X	X	
13. Rotate 30 Degrees CCW			X	X	
14. Inboard	X	X			X
15. Outboard	X	X			X
16. Legs In					X
17. Sitting on a fully reclined seat		X			X

Since the images are needed for both the training and testing of the classifier, we follow the basic rule of 10 training samples per feature per class [27]. For the child and adult classes, there are about 60 features that will be used for the pattern classification (after feature reduction). Therefore, about 600 images are targeted for

each class. The total number of images that are typically used for training and testing data sets for each occupant type are provided in Table 2.3.

Table 2.3: Summary of the number of images per occupant class for the training dataset.

Occupant Type	Classification	Number of Images
RFIS	Infant	1485
FFIS	Infant	1112
Carbed	Infant	60
Child	Child	620
5th Percentile adult	Adult	378
50-95th percentile adults	Adult	605
Empty Seat	Empty	72
Total number of images:		4332

There are three methods we use for testing the algorithms in the system: (i) cross-validation, (ii) independent validation test dataset, and (iii) live testing in vehicles. For the feature selection algorithms we use the training dataset defined in Table 2.3, and we perform a 50/50 cross validation with 10 iterations using randomized divisions of the data. For classification testing (both the traditional classifier and the integrated segmenter/classifier) we used an additional validation test dataset. This test dataset was collected both in indoor and in outdoor conditions, and the summary of the number of images collected for each occupant type is provided in Table 2.4. For the live vehicle testing we place the vehicles both indoors and outdoors, and perform both stationary testing (indoor and outdoor) as well as drive

testing (outdoor only). For the sake of managing the complexity of this thesis, we have not presented night image testing.

Table 2.4: Summary of the number of images per occupant class for the validation testing dataset.

Occupant Type	Classification	Number of Images
Infant (RFIS+FFIS)	Infant	1807
Child	Child	236
Adult	Adult	210
Empty Seat	Empty	8
Total number of images:		2261

2.4 Results of Occupant Segmentation

In this section we demonstrate the effectiveness of the two background subtraction-based methods for segmenting the occupant from the vehicle background. Recall, the only background subtraction was considered feasible for our application, since more general methods for segmentation do not allow for the integration of the contextual information about the vehicle. Since the occupant can be wearing any type of clothing, and since the system must operate in all lighting conditions, only methods that explicitly use the contextual knowledge of the empty vehicle were considered.

2.4.1 Results of Background Subtraction

Two methods of background subtraction were compared: (i) the edge-based image correlation, and (ii) the eigen-image subtraction. We will compare the outputs of each of these methods on a common data set of indoor and outdoor images containing adults, infants, and children to determine which method is more effective.

A Correlation

The background correlation method was tested in outdoor and indoor lighting conditions. An empty reference image was collected and stored for each of these lighting conditions. The algorithm was tested using only the gradient image correlation, since it provides additional robustness to lighting variations over grayscale correlations. Example images for these lighting conditions were processed using both the amplitude edge information and the vector edge information. The results are presented in Figure 2.11 through Figure 2.22.

In these results we see that the gradient vector correlation outperforms the gradient amplitude correlation. This is primarily due to the fact that the vector correlation provides better differentiation between foreground and background, as can be seen in the histograms and cumulative distribution functions (CDFs) shown for each of the example images. In all cases, the histograms for the edge vector correlations show relatively clear regions where the foreground and background may be segmented. In other words, for the edge vector correlation image the resultant histogram is bi-modal, while for the edge amplitude correlation image the histogram

is uni-modal. The consistency of the results for the edge vector correlation shows that it is the preferred approach for the airbag suppression application.

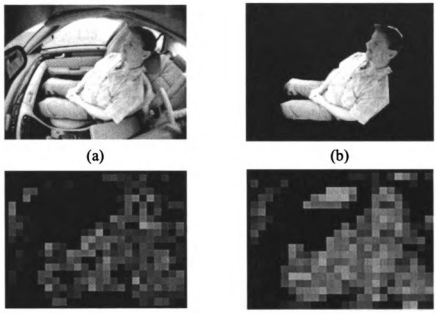


Figure 2.11: Results of correlation based background removal for an outdoor adult image, (a) original image, (b) hand segmented image, (c) edge amplitude image, and (d) edge vector image.

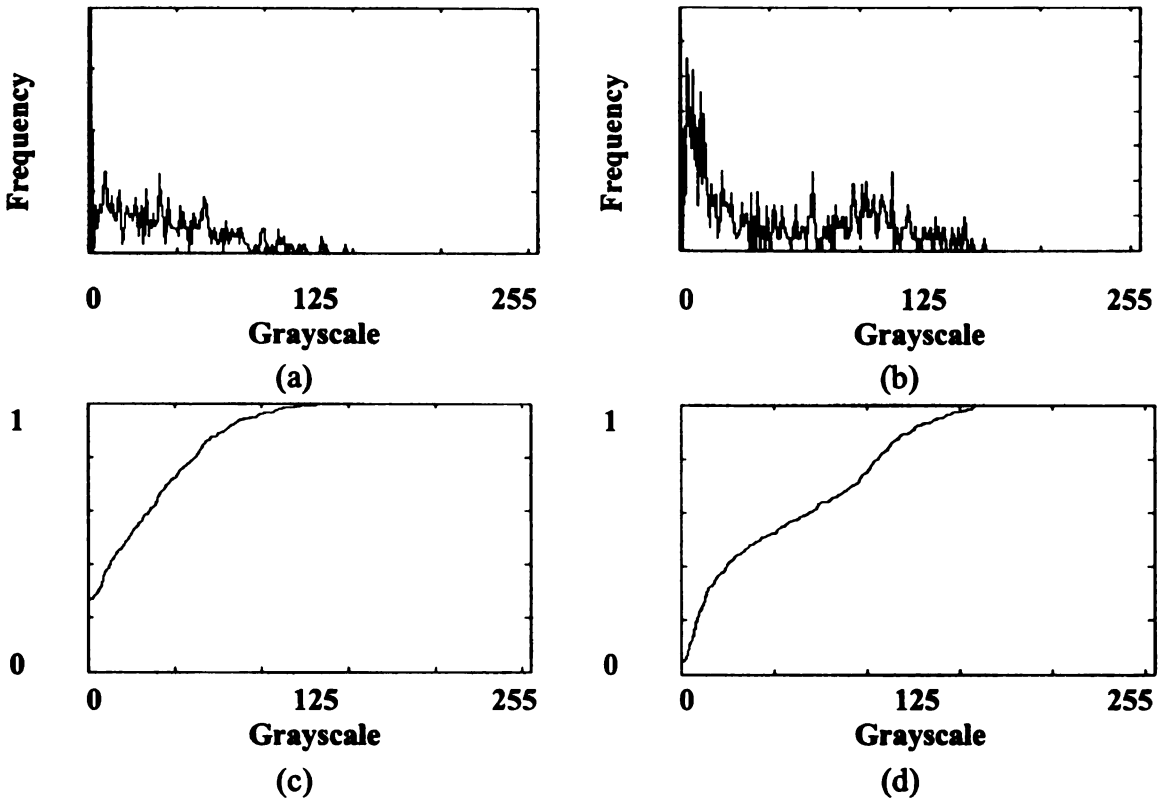


Figure 2.12: Resultant edge histogram and CDF for Figure 2.11, (a) histogram of edge amplitude image, (b) histogram of edge vector image, (c) CDF of edge amplitude image, and (d) CDF of edge vector image.

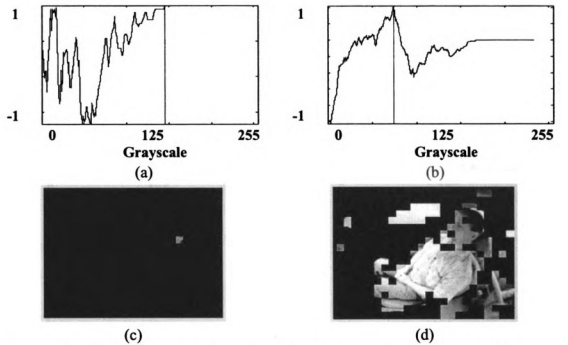


Figure 2.13: Threshold calculation and resultant segmentation for Figure 2.11, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

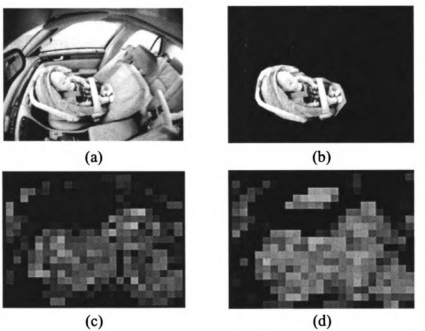


Figure 2.14: Results of correlation based background removal for an outdoor infant image, (a) original image, (b) hand segmented image, (c) edge amplitude image, and (d) edge vector image.

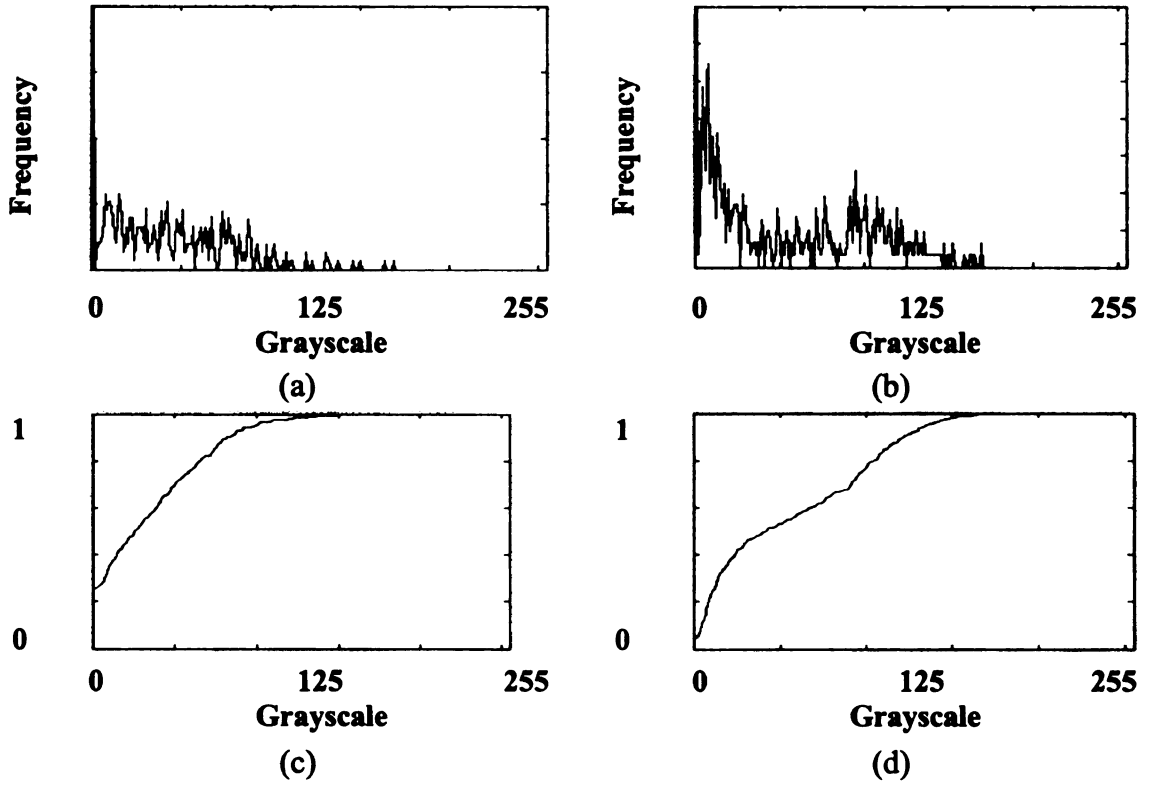


Figure 2.15: Resultant edge histogram and CDF for Figure 2.14, (a) histogram of edge amplitude image, (b) histogram of edge vector image, (c) CDF of edge amplitude image, and (d) CDF of edge vector image.

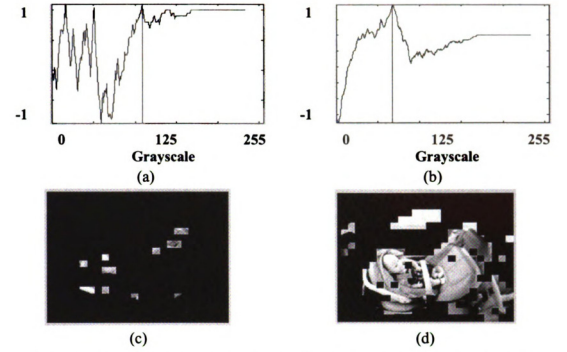


Figure 2.16: Threshold calculation and resultant segmentation for Figure 2.14, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

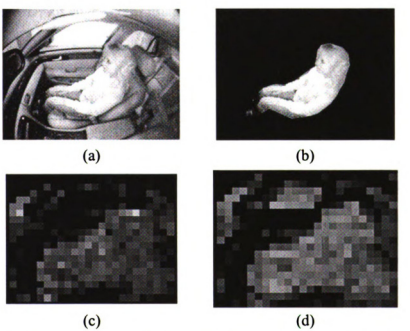


Figure 2.17: Results of correlation based background removal for an indoor adult image, (a) original image, (b) hand segmented image, (c) edge amplitude image, and (d) edge vector image.

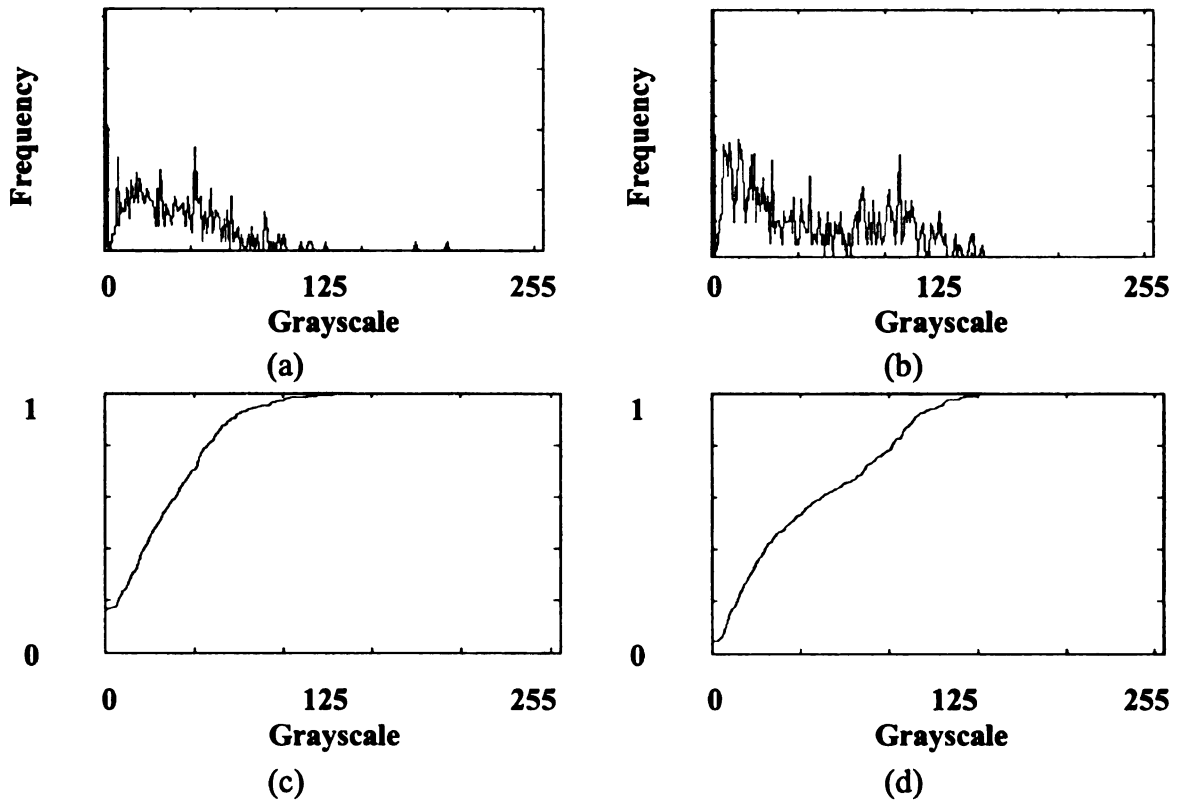


Figure 2.18: Resultant edge histogram and CDF for Figure 2.17, (a) histogram of edge amplitude image, (b) histogram of edge vector image, (c) CDF of edge amplitude image, and (d) CDF of edge vector image.

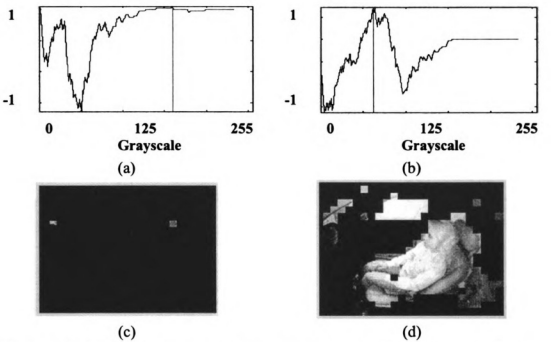


Figure 2.19: Threshold calculation and resultant segmentation for Figure 2.17, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

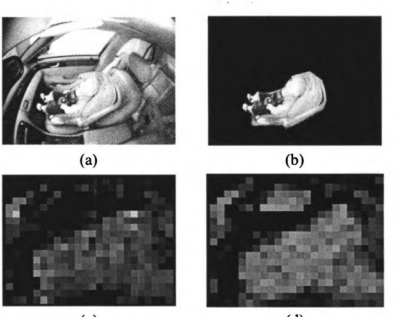


Figure 2.20: Results of correlation based background removal for an indoor infant image, (a) original image, (b) hand segmented image, (c) edge amplitude image, and (d) edge vector image.

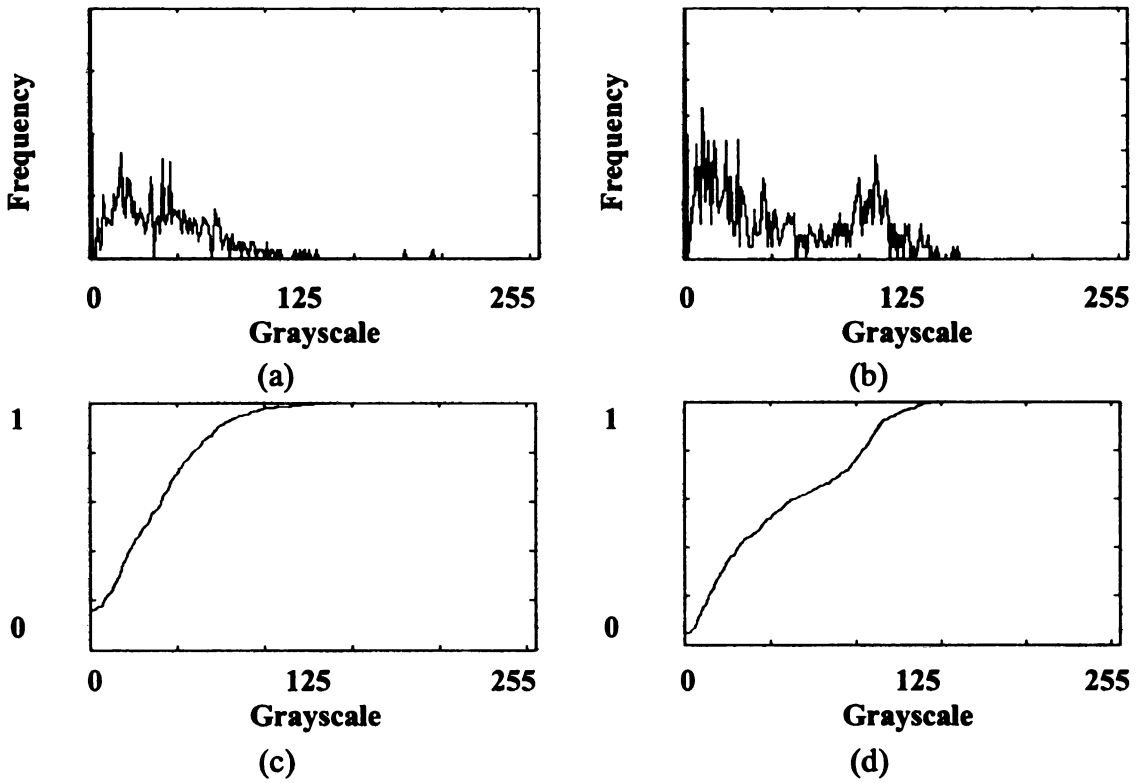


Figure 2.21: Resultant edge histogram and CDF for Figure 2.20, (a) histogram of edge amplitude image, (b) histogram of edge vector image, (c) CDF of edge amplitude image, and (d) CDF of edge vector image.

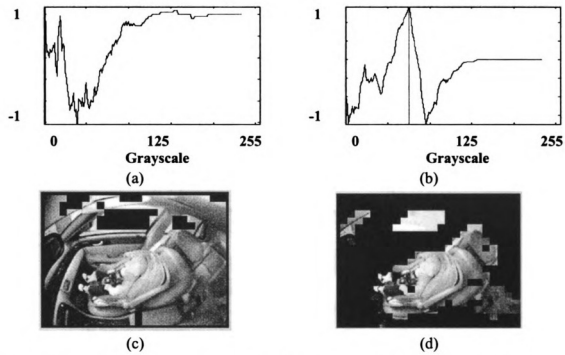


Figure 2.22: Threshold calculation and resultant segmentation for Figure 2.17, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

B Eigen-image

For the eigen-background processing, we generate the background eigen image using 772 images collected in the vehicle with the seat removed, while the vehicle was being driven outside around a cluttered parking lot. This driving route provided us with images where other vehicles and background clutter were visible through the window, as well as images where there was no background clutter. Also the training data was taken on a very bright sunny day to ensure we captured adequate illumination variation across the images. Some examples of these images are shown in Figure 2.23. Using this training set, the eigenvalues for the background image transform can be found in Figure 2.24.

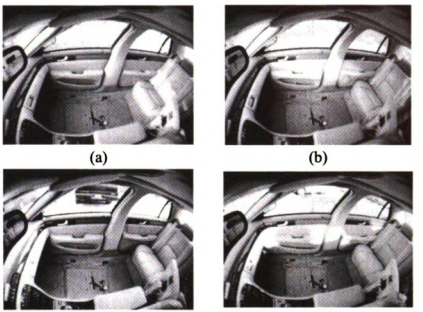


Figure 2.23: Samples from the image sequence used for eigen-image training, (a) bright illumination across rear seat, (b) bright illumination on door and trees in background, (c) bright illumination on dash and vehicle outside window, and (d) bright illumination on dash and parts of door and vehicle outside window.

In Figure 2.25 we demonstrate the application of the eigen-image background subtraction on one of the input training images. Note the difference image shows the illumination highlight on the door handle. Figure 2.26 through Figure 2.29 show the results for the eigen-image subtraction on the same example images used for the correlation processing testing. The results shown in these figures demonstrate some of the issues with using the eigen-image background subtraction.

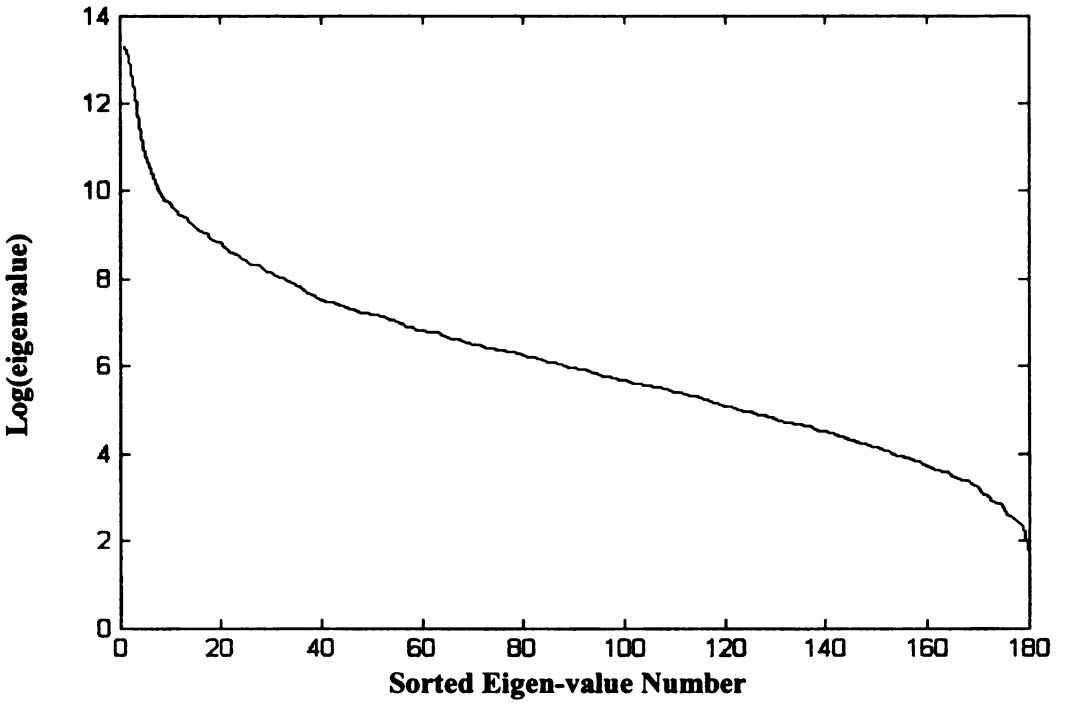


Figure 2.24: Plot of the log-amplitude versus eigenvalue number for the top 180 eigenvalues.

The first artifact in the eigen-image subtraction that we note is that in all the adult images, the heads of the occupants generally are not retained in the segmentations. This is probably due to the fact that the amplitudes of the occupants' heads and hair were too close in amplitude to the amplitude of the interior of the vehicle. It may also have been due to the fact that the occupant's head appears in the same area as the window, and this region experiences the greatest variations in amplitude due to the variable background clutter. Also note that in all the output images, a large amount of the roofline is preserved.

Notice in the indoor images, another artifact is that the roof-liner is retained. Recall, that we only used an outdoor training set since it is difficult to get a variable background sequence indoors. In the indoor images, the illuminators are playing a large role in providing the light in the image. There is considerable illumination of

the roof-liner due to the placement of the system in the vehicle. In the outdoor images, the natural sunlight far exceeds the levels of the illuminators and therefore the roof-liner is darker relative to the rest of the vehicle cabin. This region is easy to remove from the images, and therefore is not as significant concern as the issue regarding the occupant's head being consistently removed.

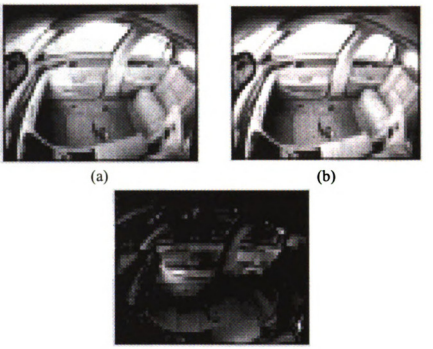


Figure 2.25: Example of a training image being processed through eigen-image background removal, (a) incoming image reduced to 50x70, (b) transformed into background domain, and (c) resultant difference image.

Another artifact in the eigen-image segmentation is that regions around the instrument panel and the A and B-pillars of the vehicle are also regularly captured in the segmented image. This is possibly because these regions are very susceptible to illumination change, and the training set did not completely capture the changes even with the thorough data collection that was performed.

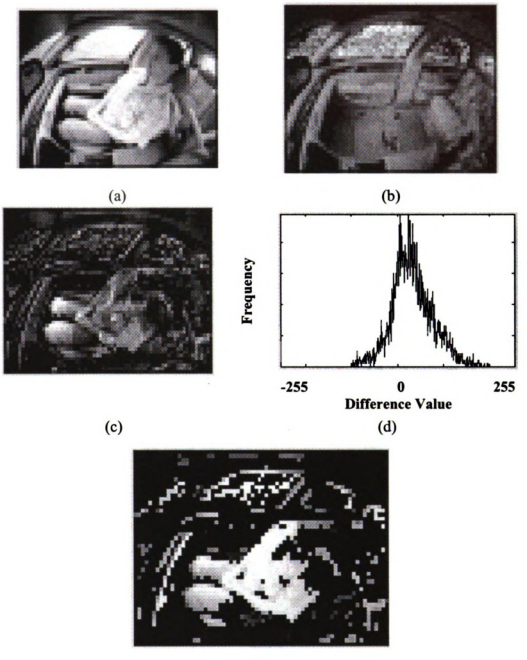


Figure 2.26: Eigen-image results for outdoor adult image in Figure 2.11, (a) input image reduced to 50x70, (b) transformed image, (c) difference image, (d) histogram of difference image, and (e) thresholded output image.

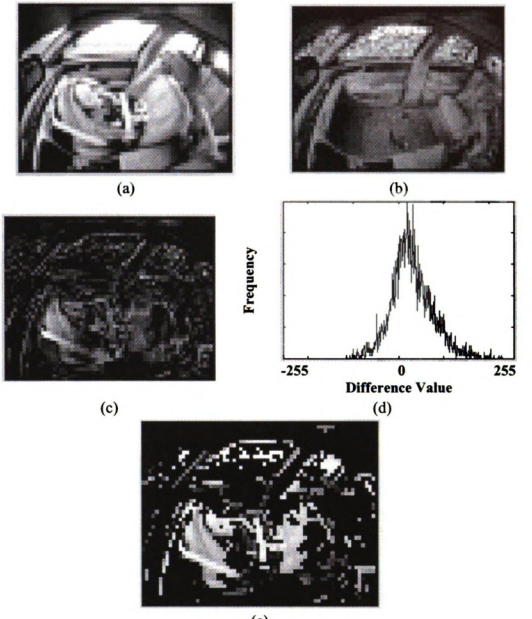


Figure 2.27: Eigen-image results for outdoor infant image in Figure 2.14, (a) input image reduced to 50x70, (b) transformed image, (c) difference image, (d) histogram of difference image, and (e) thresholded output image.

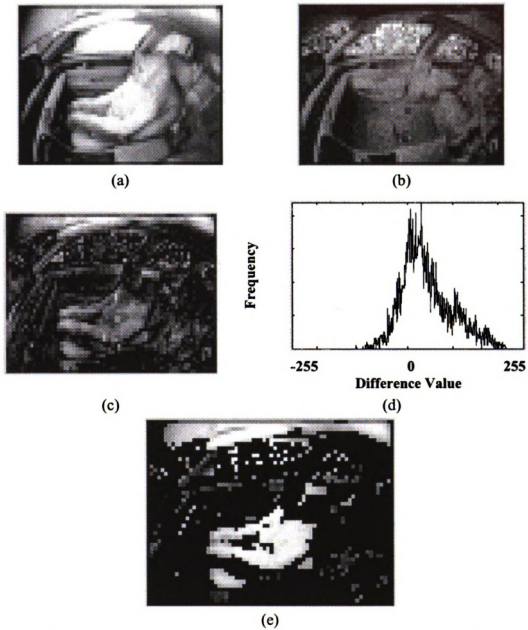


Figure 2.28: Eigen-image results for indoor adult image in Figure 2.17, (a) input image reduced to 50x70, (b) transformed image, (c) difference image, (d) histogram of difference image, and (e) thresholded output image.

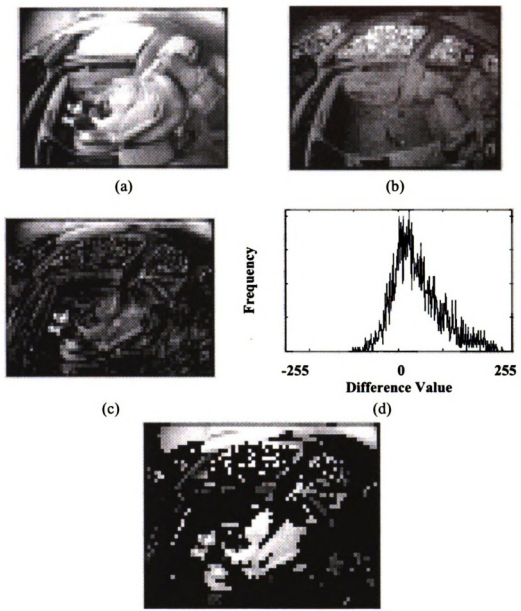


Figure 2.29: Eigen-image results for indoor infant image in Figure 2.20, (a) input image reduced to 50x70, (b) transformed image, (c) difference image, (d) histogram of difference image, and (e) thresholded output image.

Possible error sources for each of the aforementioned artifacts have been identified. One possible systemic source of error in the eigen-image segmentation may be due to the low image resolution required to support the real-time eigen-background processing. The images must be reduced to 50x70 pixels resolution from the incoming 400x320 in order to be processed due to memory limitations. This is because the correlation matrix that is used to generate the eigenvalues is *(NxM)x(NxM)*, where *N* and *M* are the number of rows and columns in the image, respectively. This dramatic reduction may be the source of the stair-stepping effects seen along the dashboard and the A and B-pillars.

Another systemic source of error in using the eigen-image background subtraction is the difficulty in establishing a threshold for the difference images. As can be seen in sub-frame (c) in Figure 2.26 through Figure 2.28, the histogram is two-sided since the difference image can vary in amplitude from –255 to +255. This creates a complicated threshold decision. We first process the difference image to center the mode of the histogram about zero-amplitude, then we take the absolute value of the image. This changes the histogram as shown in Figure 2.30, which results in a fairly smooth CDF as shown in Figure 2.30 (c). This is the same problem we experienced in the correlation processing when only the edge amplitude information was used in the correlation based processing, where adequate information was not available to differentiate the background from the foreground.

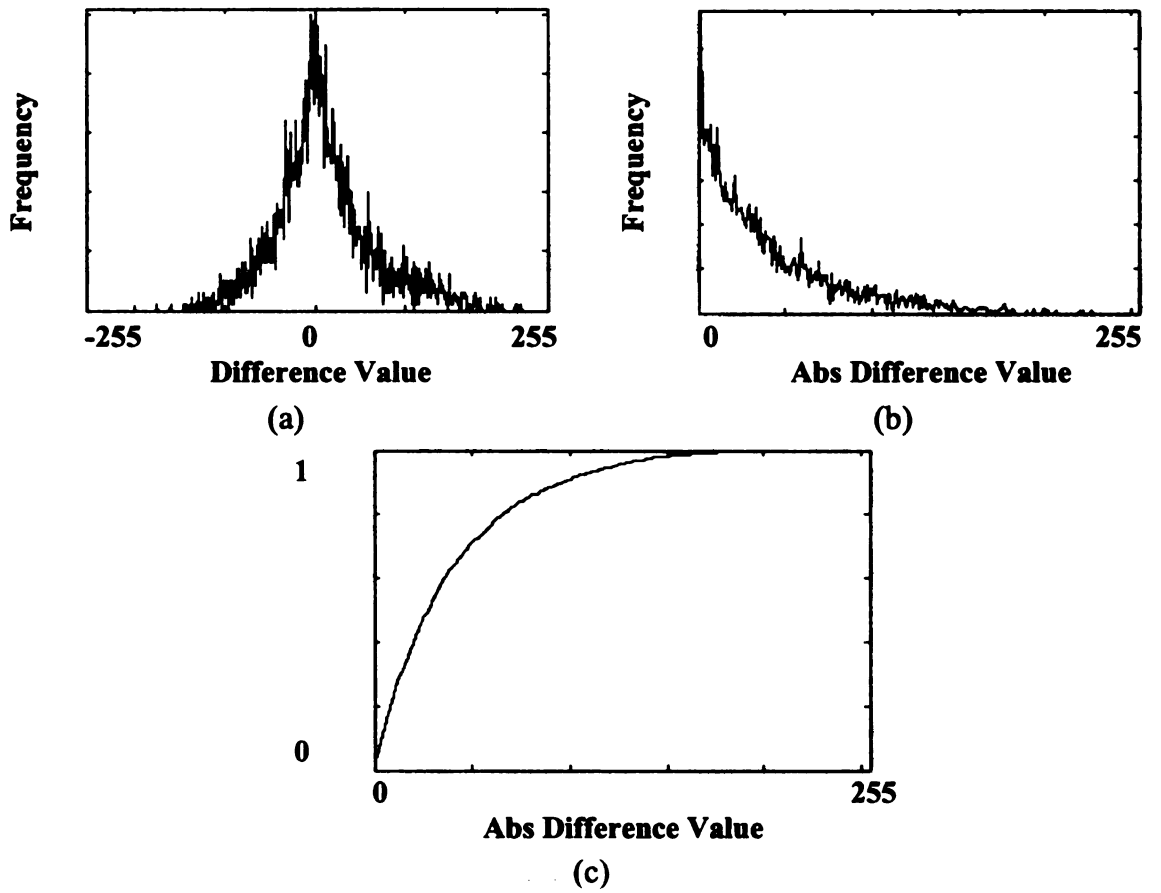


Figure 2.30: Demonstration of modification of histogram to form a single sided decision function, (a) incoming histogram for outdoor infant image in Figure 2.27, (b) resultant 1-sided histogram after shift and absolute value of the image grayscales, and (c) CDF of the 1-sided histogram.

2.4.2 Results of Image Enhancement

In the following figures, the outputs from the edge correlation processing are used to demonstrate the performance of the image enhancement processing. The images are processed using mathematical morphology and the watershed algorithm to demonstrate the relative merits of each approach. The hand-segmented images are provided to allow the reader to quickly see the performance of the algorithms.

Recall there were two aspects of image enhancement processing: (i) hole-filling using morphology, and (ii) additional background removal using watershed processing.

A Mathematical Morphology

The results of the morphology based hole filling can be seen in Figure 2.31 through Figure 2.34. In each of these figures, the input to the morphology processing is shown in sub-figures (a) and (b) which provide the binary and grayscale outputs from the edge-vector correlation processing. The output of the morphology processing is likewise shown in sub-figures (c) and (d) which provide the binary and grayscale resultant images.

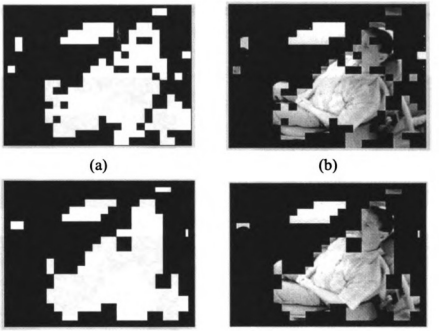


Figure 2.31: Application of morphology for hole filling for image from Figure 2.11, (a) binary segmentation after correlation processing, (b) grayscale segmentation after correlation processing, (c) binary segmentation after hole filling, and (d) grayscale segmentation after hole filling.

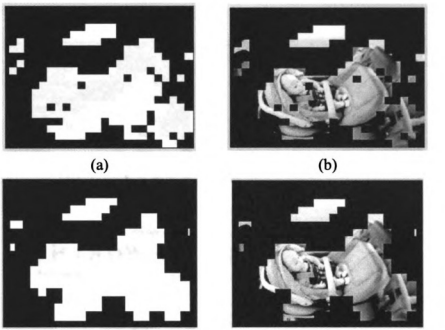


Figure 2.32: Application of morphology for hole filling for image from Figure 2.14, (a) binary segmentation after correlation processing, (b) grayscale segmentation after correlation processing, (c) binary segmentation after hole filling, and (d) grayscale segmentation after hole filling.

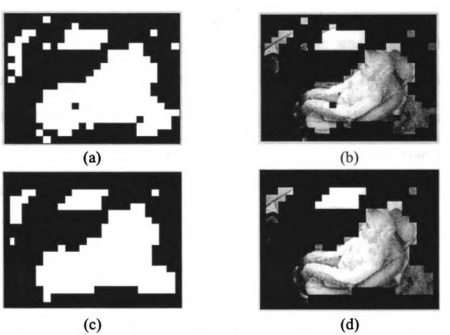


Figure 2.33: Application of morphology for hole filling for image from Figure 2.17, (a) binary segmentation after correlation processing, (b) grayscale segmentation after correlation processing, (c) binary segmentation after hole filling, and (d) grayscale segmentation after hole filling.

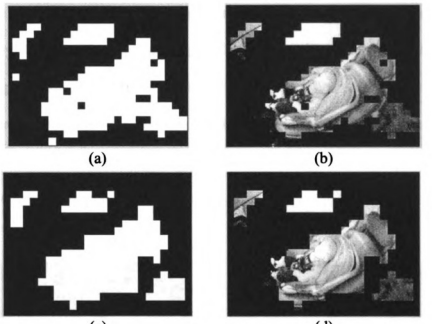


Figure 2.34: Application of morphology for hole filling for image from Figure 2.20, (a) binary segmentation after correlation processing, (b) grayscale segmentation after correlation processing, (c) binary segmentation after hole filling, and (d) grayscale segmentation after hole filling.

Note in all the output images shown above, the small internal holes are successfully filled, and, in many of the images, small irregular edge regions are also smoothed. Additionally, there is no unwanted joining of smaller regions, and there are no resultant images where the output segmentation is worse than the input segmentation.

B Watershed

The results of the watershed processing can be seen in Figure 2.35 through Figure 2.38. In these figures, sub-figure (a) shows the desired output of the segmentation, namely the hand-segmented image. Sub-figure (b) shows the input to

the watershed processing, which is the output of the morphology-based hole-filling algorithm. Sub-figure (c) shows the generation of the inner and outer markers that are used by the watershed processing, and sub-figure (d) shows the output of the watershed processing. Recall, the objective of the watershed algorithm is to remove as many extraneous background pixels as possible prior to feature extraction. Consequently, the output of the algorithm in sub-figure (d) should closely resemble the hand-segmented image in sub-figure (a).

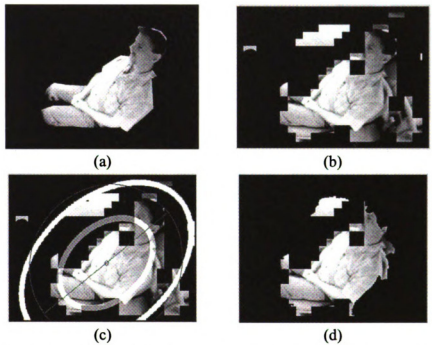


Figure 2.35: Threshold calculation and resultant segmentation for Figure 2.11, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

There are a few noticeable anomalous results that the reader can discern in Figure 2.35 (d) through Figure 2.38 (d). Notice that in the adult figures, the top of the head is often removed in the processing. This is because the hair is often relatively

dark in comparison to the rest of the occupant. This causes the hair to be very close in amplitude to the background, and consequently causes the hair to be removed. This is not a problem for lighter haired people as shown in Figure 2.37. But this is a concern overall since the system must work for all occupant types. A final artifact of the watershed is that the seat back is generally preserved in the infant seat cases. This is because the seat back is usually brighter than the remaining background.

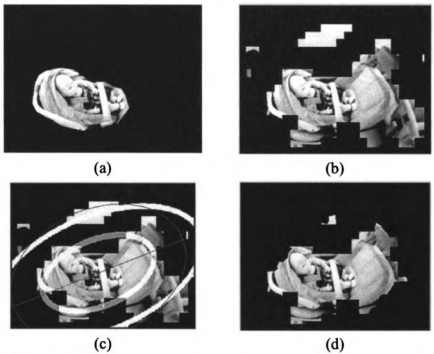
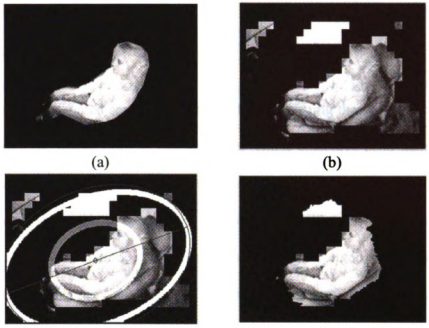


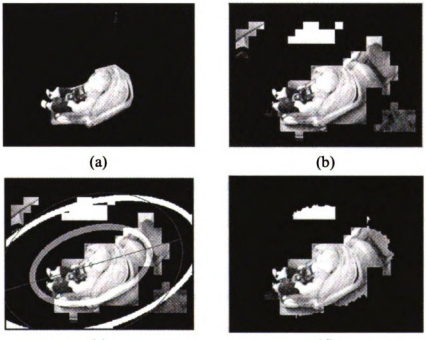
Figure 2.36: Threshold calculation and resultant segmentation for Figure 2.14, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.



(a) (b)

(c) (d)

Figure 2.37: Threshold calculation and resultant segmentation for Figure 2.17, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.



(a) (b)

(c) (d)

Figure 2.38: Threshold calculation and resultant segmentation for Figure 2.20, (a) edge amplitude threshold calculation, (b) edge vector threshold calculation, (c) edge amplitude segmentation, and (d) edge vector segmentation.

2.5 Summary

We have presented several approaches for performing the segmentation of the occupant from the image prior to classification. Since extensive prior knowledge of the interior of the vehicle is available, background subtraction was used for segmentation. Two methods were tested as potential candidates: (i) a local method of background cross-correlation, and (ii) a global method of eigen-image subtraction. We have shown the results of each of these methods, and demonstrated that the background cross-correlation method has superior performance. We believe this is due to the fact that the most challenging image-to-image variations for our automotive application involve local intensity variations due to illumination highlights and shadows. Since the background cross-correlation method relies on local correlation comparisons over small window sizes, it is inherently more immune to these effects than the global eigen-image subtraction.

As was shown in the example output images, the eigen-image subtraction performed well on images collected at the same time as the reference background, but performed poorly on imagery collected at considerably different times, and under slightly different lighting conditions. In order for the eigen-image method to perform well, the background reference must be updated regularly, but the processing complexity of real-time eigen-image updating makes the approach infeasible for our embedded airbag suppression application. Even with frequent updating, the method is still limited due to its global nature versus the local nature of many of the illumination effects.

In addition to the basic background processing, we have also demonstrated the effectiveness of the additional post processing algorithms designed to fill any holes in the segmented occupant and to reduce any residual background imagery in the output segmentation. While it is relatively easy to fill in the holes in the segmented occupant, it is very difficult to apply a context independent algorithm, such as the watershed, to remove any remaining background pixels. We were able to utilize a modest amount of contextual information based simply on known problem areas, such as the window region, the instrument panel in the night images, and the rear seat region. We then use this information to generate a reasonable set of inner and outer markers for the watershed algorithm.

In many cases the head region of the adult occupants are removed due to their hair color matching the interior color of the vehicle. This can only be corrected by applying even more robust contextual information. Chapter 6 defines an approach to segmentation that integrates the classification and segmentation and therefore can utilize this additional contextual information.

Chapter 3.
Feature Extraction for Occupant Classification

The second stage of the static suppression processing is feature extraction. In many classification problems the key to success is to find a proper representational space within which to interpret the incoming data stream. Devijer and Kittler [22] define feature extraction as "extracting from the raw data the information which is most relevant for classification purposes". Feature extraction involves defining the proper feature space that will provide the best discrimination between the various pattern classes [22]. Devijer and Kittler further state that the "feature extraction method is probably the single most important factor in achieving high recognition performance" [22]. We will investigate two approaches for extracting features for the occupant classification problem [22] [48]:

1) traditional feature-based classification, where numeric features are computed from the incoming image, and
2) feature-less classification, where the raw image data is fed directly to the classifier.

3.1 Candidate Feature Spaces

For feature-based classification, there are four basic feature spaces that can be utilized as shown in Figure 3.1 [22][25]:

1) color or grayscale,

2) texture,

3) shape, and

4) spatial locations.

For our airbag suppression application, there is often no clear separation between the occupant classes based on grayscale and texture. This is because an infant, child, or adult occupant can all be wearing materials (clothing and/or infant seat cover) of similar color and texture as can be seen in Figure 3.2, where both images have very similar grayscale and texture features. The infant, however, is clearly of a different shape than the adult.

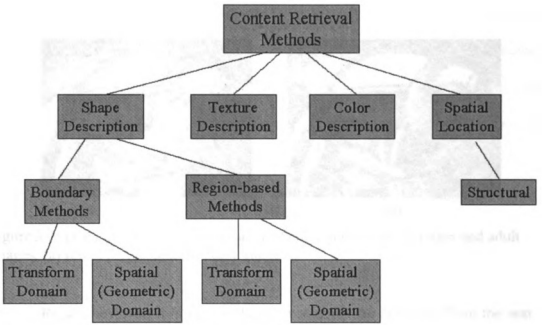


Figure 3.1: Taxonomy of feature-based description techniques for image classification [25].

Spatial or structural representations have been shown to be very effective in some human recognition applications [113][115][116][117][118]. Unfortunately,

most of these applications involved humans walking in relatively constant clutter backgrounds with uniform illumination across the human. In these situations, then it is relatively easy to segment the entire human subject from the background. Then since the people are walking or performing other regular motions, such as dance or Tai Chi, it is straightforward to segment the head, limbs, and torso from the background. Recall from Chapter 2 how difficult it is for us to even reliably segment the entire human in our application. This means it would be extremely difficult for us to reliably segment components of the human and then build the entire occupant from the components. Therefore spatial representation is not considered a valid approach so we will use shape for distinguishing the occupant classes.

(a) (b)

Figure 3.2: Demonstration of commonality of color and texture in infant and adult images, (a) infant image, and (b) adult image.

Recall, however, it is very difficult to segment the occupant from the seat in the imagery. The approach taken in the background subtraction processing was to segment the occupant and the seat together. Since the small child occupant is completely engulfed by the seat, as shown in Figure 3.3, recognizing the occupant on the seat requires some internal features to be used.

Figure 3.3: Example image showing that the occupant can be completely engulfed within the boundary of the vehicle seat.

The most common feature spaces for representing the shape of an object are the silhouette and the edge spaces, since the shape can be represented either as a solid or as a contour. One additional space that may be useful for shape representation is the wavelet transform space since it is closely related to the edge space (particularly for Haar Wavelets), but has the added advantage of providing an inherent multi-resolution capability. This may be attractive since various studies have shown that human visual perception is performed at several scales of resolution [35][24][23].

3.1.1 Edge Feature Space

The edge feature space consists of both the internal and external (boundary) edges of the occupant/seat combination. This may allow the system to distinguish a small occupant on the seat from the empty seat based on the edge information within the image. We are, therefore, interested in the characteristics of the internal edges to

try to classify one occupant type from another. Figure 3.4 demonstrates this representation space for an adult edge image and an infant edge image. Note the difference in the structure of the edges in each of the two images. In this example a simple gradient operator was applied, and then the image was thresholded to keep the dominant edge pixels. A threshold value of 65% of the cumulative distribution function was selected after testing a range of threshold values from 50% to 90%.

An alternative approach to compute the edges in an image is to use a two-stage process. The first stage processes the image with a simple gradient operator, generating the x and y directional gradient values at each pixel. The edge magnitude is then computed from these values and used for subsequent processing. The second stage uses a constant false alarm rate (CFAR) based detector, rather than a simple thresholding operation. The idea behind the CFAR detector is that it provides a consistent probability of detecting an edge to ensure that weak amplitude edges are still detected. The CFAR method used here is the cell-averaging CFAR, where the average edge amplitude in the background window is computed and compared to the current edge image, as shown in Figure 3.5 [44]. Only non-zero pixels are used to compute the background window statistics. The guard region is simply a separating region between the pixel of interest and the background.

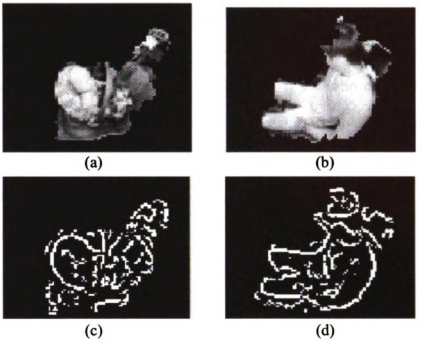


Figure 3.4: Edge maps of infant and adult segmented images, (a) segmented RFIS, (b) segmented adult, (c) edge image of RFIS, and (d) edge image of adult.

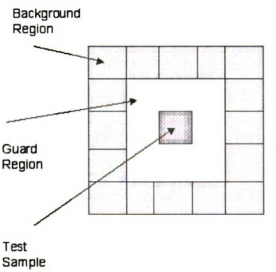


Figure 3.5: CFAR edge detector window.

Figure 3.6 shows the resultant edge map for the traditional edge detector and for the CFAR edge detector [44]. In these examples, a 5x5 CFAR kernel (see Figure 3.5) is used. A pixel is marked as an edge pixel if the ratio of the test sample amplitude to the background region statistic exceeds a threshold set by the desired false alarm rate. Note from the figure that CFAR results in a large number of small edge components. Therefore, it is advantageous to apply a connected components algorithm on the output of the CFAR that can then discard any edges that are of inadequate length. Figure 3.7 shows the result of this post- processing.

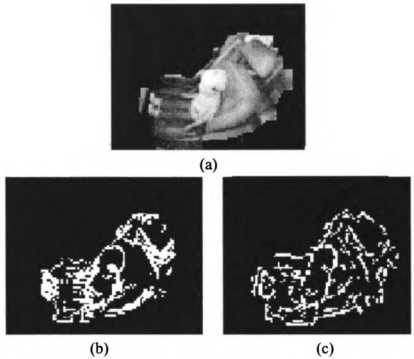


Figure 3.6: Edge detection comparison, (a) segmented image, (b) adaptive threshold edge detection, and (c) CFAR edge detection.

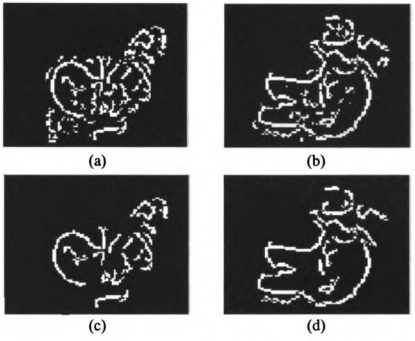


Figure 3.7: Connected components in CFAR edge map, (a) original CFAR edge image of RFIS, (b) original CFAR edge image of adult, (c) connected components for a RFIS, and (d) connected components for an adult RFIS.

3.1.2 Silhouette Feature Space

The silhouette feature space is obtained by simply thresholding the final segmented image. In the silhouette space all internal structures are ignored, and the combined shape of the occupant and the seat is represented. The benefit of this feature space is the simplicity of the representation. Figure 3.8 shows segmented images, and their resultant binary silhouette representations for an empty passenger seat and a child on a seat. This figure shows that the two silhouettes are nearly identical! Consequently, for the traditional implementation of segmentation,

followed by feature extraction, and ultimately classification, the silhouette space will not prove useful.

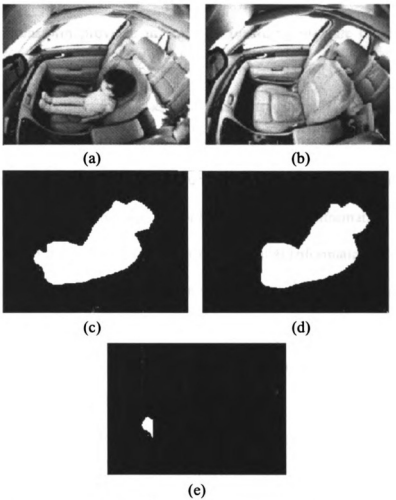


Figure 3.8: Binary silhouettes do not provide adequate separation, (a) child on seat (b) empty passenger seat, (c) segmented and binarized child, (d) segmented and binarized empty seat, and (e) the difference between image (c) and image (d).

3.1.3 Wavelet Feature Space

Wavelets provide an alternative transformation that can be applied to the imagery for the purpose of developing representations related to frequency domain. The wavelet transform provides information regarding the rate of change of the information at varying resolutions across the image [170][171]. The difference between the wavelet transform and the Fourier transform is that in the Fourier transform all positional information is lost when the frequency information is provided (the positional information is buried within the phase information for each sine wave, but is usually removed, and only the amplitude information is preserved) [170][171][173]. In the wavelet transform, the spatial information of the signal is preserved along with the frequency information. This may prove useful in image feature representation since the characteristics of the edge information and the location of this information can both be provided by the wavelet transform.

There are many methods for developing wavelet transforms, including Haar and Daubechies transforms. The Haar method is the most popular due to its simplicity of implementation. It involves two operators, a summation operation and a delta operation, as shown in Figure 3.9, and defined as [170][173]:

$$\varphi[0,1)(r) = \begin{Bmatrix} 1 & \text{if } 0 \le r < 1 \\ 0 & \text{otherwise} \end{Bmatrix}, \tag{3.1}$$

and

$$\psi[0,1)(r) = \begin{Bmatrix} 1 & \text{if } 0 \le r < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \le r < 1 \\ 0 & \text{otherwise} \end{Bmatrix}, \tag{3.2}$$

where the notation $\varphi[0,1)$ means the point 0 is included and the point 1 is excluded from the range of values.

The Haar wavelet algorithm is extremely simple and fast, and it involves taking sums and differences of adjacent pixel values in a pyramid style of processing. The processing continues by recording all of the sum and delta values until the algorithm completes $\log_2(N)$ cycles and is down to the last two values [170][173]. All of the delta values are re-ordered and stored as a replacement representation, and then the final summation is included, which represents the average amplitude of the signal.

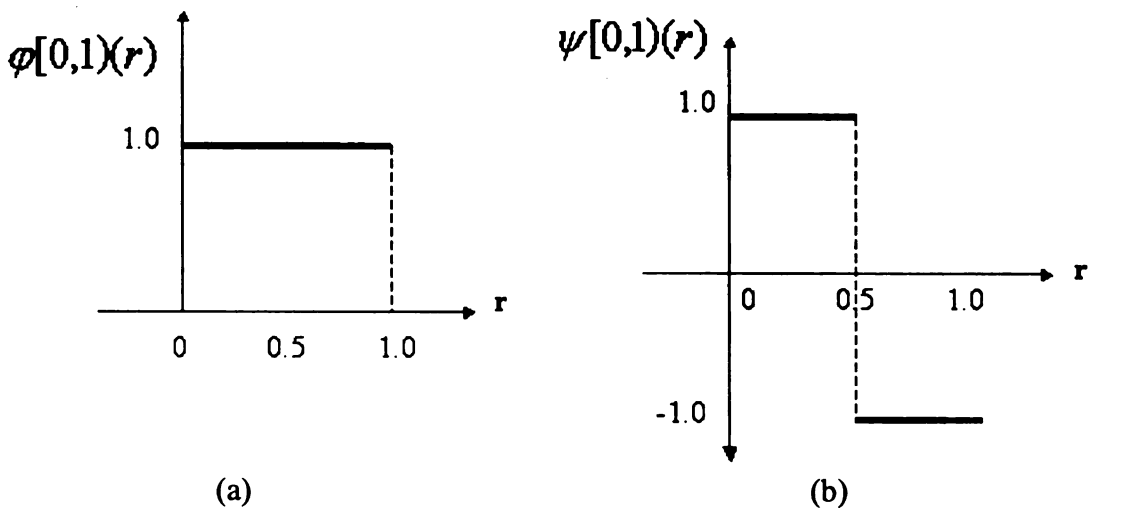


Figure 3.9, Haar wavelet basis functions, (a) the summation function $\varphi[0,1)(r)$, and (b) the delta function $\psi[0,1)(r)$.

In two-dimensional Haar transforms, the 1-dimensional step functions are replaced with 2-dimensional step functions. These two-dimensional operators are

generated through the tensor products of the two 1-dimensional operators as follows [170][173]:

$$\Phi_{0,0}(x,y) = \varphi[0,1)(x) \otimes \varphi[0,1)(y) = \begin{Bmatrix} 1 & \text{if } 0 \le x < 1 \text{ and } 0 \le y < 1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad (3.3)$$

$$\Psi^{h}_{0,0}(x,y) = \varphi[0,1)(x) \otimes \psi[0,1)(y) = \begin{Bmatrix} 1 & \text{if } 0 \le x < 1 \text{ and } 0 \le y < \frac{1}{2} \\ -1 & \text{if } 0 \le x < 1 \text{ and } \frac{1}{2} \le y < 1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad (3.4)$$

$$\Psi^{v}_{0,0}(x,y) = \varphi[0,1)(x) \otimes \psi[0,1)(y) = \begin{Bmatrix} 1 & \text{if } 0 \le x < \frac{1}{2} \text{ and } 0 \le y < 1 \\ -1 & \text{if } \frac{1}{2} \le x < 1 \text{ and } 0 \le y < 1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad (3.5)$$

$$\Psi^{d}_{0,0}(x,y) = \varphi[0,1)(x) \otimes \psi[0,1)(y) = \begin{Bmatrix} 1 & \text{if } 0 \le x < \frac{1}{2} \text{ and } 0 \le y < \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} \le x < 1 \text{ and } \frac{1}{2} \le y < 1 \\ -1 & \text{if } \frac{1}{2} \le x < 1 \text{ and } 0 \le y < \frac{1}{2} \\ -1 & \text{if } 0 \le x < \frac{1}{2} \text{ and } \frac{1}{2} \le y < 1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad (3.6)$$

Notice that there are now four Haar operators for 2-dimensional processing: (i) a summation operator, (ii) a vertical edge operator, (iii) a horizontal edge operator, and (iv) a diagonal edge operator, as shown in Figure 3.10. A segmented image of an infant seat is shown in Figure 3.11, and the resultant multi-resolution wavelet transform is shown in Figure 3.12. In Figure 3.12 the results of each of the Haar operators can be seen. As can be clearly seen in Figure 3.12, the Haar wavelet is an image transform that converts the incoming image into an alternate representational

space corresponding to an edge representation. The multi-resolution edge detection capability of the Haar transform can be seen in Figure 3.13 where images (a), (b), and (c) show the full resolution, half resolution, and quarter resolution edge detection images, respectively.

In order for the wavelets to be used as features for classification, they must either be further processed by feature generation algorithms, such as moment analysis, or used for appearance-based processing.

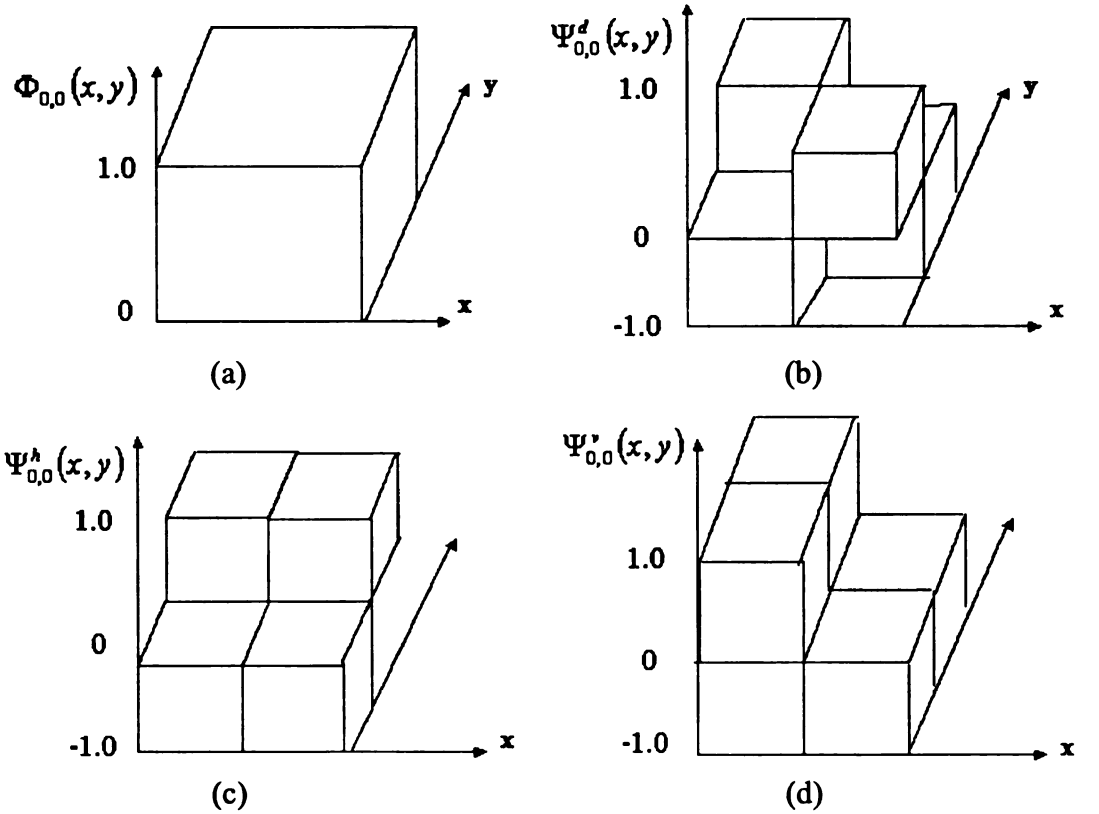


Figure 3.10: Haar wavelet basis functions for 2-dimensional processing, (a) summation operator $\Phi_{0,0}(x,y)$, (b) diagonal edge operator $\Psi^{d}_{0,0}(x,y)$, (c) horizontal edge operator $\Psi^{h}_{0,0}(x,y)$, and (d) vertical edge operator $\Psi^{v}_{0,0}(x,y)$.

Figure 3.11: Hand segmented image of an infant seat.

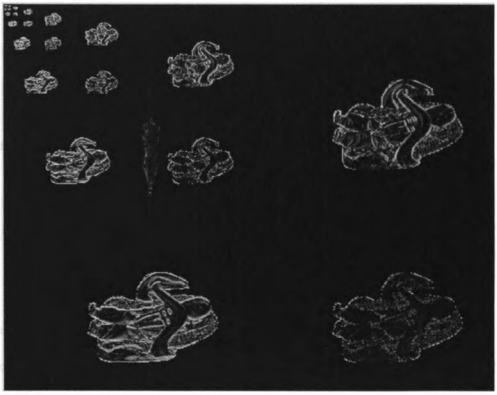


Figure 3.12: Output of Haar wavelet transform of the image shown in Figure 3.11.

(a) (b) (c)

Figure 3.13: Wavelet transform results for the vertical edge detector, (a) at full resolution, (b) at 1/2 resolution, and (c) at 1/4 resolution.

3.2 Feature Computation

In order to compute features to represent the shape of an object, there are three general characteristics of the representation that must be addressed, namely [23]:

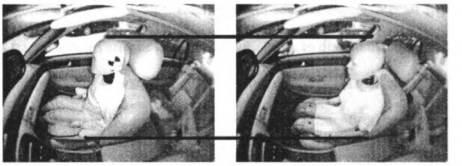
1) boundary points versus internal data,
2) numeric versus non-numeric, and
3) information preserving versus non-information preserving (whether or not representation supports image reconstruction).

To address the first characteristic, we should recall that in our background subtraction approach to segmentation, the seat and occupant are segmented together. As shown in Figure 3.8, the segmented boundary alone is not a reliable discriminator. Likewise, only using internal edge features can be confusing since they may be more dependent on the type of clothing the occupant is wearing than the boundary of the occupant. Therefore, algorithms that exploit both the internal edge and boundary edge features will be investigated.

To address the second characteristic, we must consider the type of classifier that will be used. Since our final objective is to use a classifier such as a nearest neighbor or support vector machine, rather than a rule-based classifier, we will only use the numeric representation. To address the third characteristic, recall that the airbag suppression application has no clear requirement for information preservation, in terms of the ability to use the object features to perform image reconstruction. A reconstruction capability can, however, be helpful for algorithm development, to better understand the performance of the system when mis-classifications do occur.

In most pattern classification applications it is desirable for the features to be invariant to certain transformations. The most common desired invariances are with respect to (i) translation, (ii) rotation, (iii) scale, and, (iv) skew [22]. The occupant classification problem is somewhat unique in that it is not desirable to have invariant features. For example, the primary difference between a six year-old and an adult is the size, which implies that size invariance is not desirable for the occupant classification problem. Due to the placement of the camera and the perspective effects of the wide-angle lens, translational invariance is also not desirable, since an adult in the rear-most seat position generates the same size object as a child in the forward-most seat position as shown in Figure 3.14. Rotational and skew invariances are also inherently not required for this problem.

Since all of the representational spaces defined in the earlier sections of this chapter operate either on the internal characteristics or on the boundary characteristics, the following two feature sets have been widely used to represent these characteristics, namely moments and Fourier descriptors [22][23][24].



(a) (b)

Figure 3.14: Perspective effect on perceived occupant size, (a) 6 year-old child on booster in front-most seat position, and (b) 5th percentile adult in rear-most seat position (Note 6 year-old is larger in the image due to perspective).

3.2.1 Moment Features

A common set of features for shape recognition is the geometric moments of an object [22][23][24]. The geometric moment of order $(m+n)$ for an MxN image is defined as:

$$\mu_{mn} = \sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j) \cdot x(i)^m \cdot y(j)^n, \qquad (3.7)$$

where $x(i) \in [-1,1]$ and $y(j) \in [-1,1]$. Image moments have been used for a variety of applications from aircraft recognition to Chinese character recognition [22][24][29][35][36]. We were particularly motivated by the use of moments in Chinese character recognition described in [36], where we believe the ability of the moments to distinguish the fine variations in character strokes may also prove useful in discriminating the edge images of adults from children in our application.

There are a number of alternatives to the basic geometric moments that provide invariance to translations, rotations, and scale changes. Recall from Figure 3.14, that translation invariance can impact our ability to distinguish the 6 year-old on a booster from a 5th percentile adult female due to the perspective effects of the camera viewing angle within the passenger compartment of the vehicle [29]. Consequently, we will not utilize any moments with invariant properties for this application.

One disadvantage of the geometric moments is that they do not form an orthogonal basis set. This means that there is redundant information in each of the moment values. Since in the subsequent feature selection task we will try to minimize the number of features used, it is prudent to investigate more optimal representations. First, let us generalize equation (3.8) to the following [34]:

$$\mu_{mn} = \sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j) \cdot h_{mn}(x(i), y(j)), \qquad (3.9)$$

where for geometric moments the kernel function h is simply the monomial:

$$h_{mn}(x(i), y(j)) = x(i)^m \cdot y(j)^n. \qquad (3.10)$$

Now we can generate a set of moments of an image, where we can replace the function $h(x,y)$ with a function generated from an orthogonal basis which will provide an "optimal" representation of the image (by the definition of orthogonality, there is no overlap in the information content between the moments). There are sets of moments that are inspired by orthogonal function theory, which states that the most

efficient representation of a general function will be through the use of an orthogonal basis. There are a number of orthogonal basis functions, including [29] [30]:

- Legendre polynomials,
- Chebyshev polynomials, and
- Zernike functions.

The Chebyshev and Legendre representations are actually intimately related through a generating function called the Gegenbauer polynomial. The generating function for the Gegenbauer polynomials is [32]:

$$\frac{2^m}{\left(1-2xt+t^2\right)^{m+1/2}} = \frac{\sqrt{\pi}}{\left(m-\frac{1}{2}\right)!} \cdot \sum_{n=0}^{\infty} T_n^m(x) \cdot t^n , \qquad (3.11)$$

where the coefficients $T_n^m(x)$ are the Gegenbauer polynomials, and $x \in [-1,1]$. When $m=0$, this function produces the generating function for the Legendre polynomials, and when $m = \frac{1}{2}$ the Type 1 Chebyshev polynomials (the type commonly used for pattern recognition) are produced [32].

Legendre moments have been shown to be more effective for recognition of Chinese characters than either central or geometric moments [36]. A number of researchers have compared the behavior of various moment types in the presence of noise [31]. Legendre moments were compared with Chebyshev moments in terms of reconstruction accuracy of noisy images, and it was found that Chebyshev moments had a slight advantage [30]. Teh and Chen also compared geometric, Legendre, and Zernike moments by measuring the signal to noise ratio of the reconstructed images

at various moment orders. These authors found that Zernike moments were more robust to noise than either geometric or Legendre. In all these noise tests, however, the images were corrupted with only Gaussian noise, while we believe the more interesting effects are due to segmentation errors, which are clearly not Gausssian. Therefore, rather than make a decision on which moment family to use from these studies, we will test all of them for our airbag suppression application.

A Legendre Moments

The Legendre moments are based on the Legendre polynomial representation of a function. The recurrence relation for the one-dimensional Legendre polynomials is [29] [30]:

$$p(0) = 1;$$

$$p(1) = x; \qquad (3.12)$$

$$p(n) = 1/n * ((2 * n - 1) * x * p(n-1) - (n-1) * p(n-2)).$$

where n is the order of the Legendre polynomial. From these polynomials, a set of orthogonal moments can be generated, based on the traditional geometric moments. While the above relation is for one dimension, the two-dimensional representation is simply a product of the two individual dimensions. The generating function for the Legendre moments of order $(m+n)$ from the geometric moments is [29] [30]:

$$L_{mn} = \frac{(2m+1)(2n+1)}{4} \sum_{j=0}^{m} \sum_{k=0}^{n} C_{mj} C_{nk} M_{jk}, \qquad (3.13)$$

where C_{ij} are the coefficients that result from executing the polynomial generating function (3.3) and M_{ij} are the traditional geometric moments of order *(i+ j)*. The first few Legendre moments are listed below [29] [30]:

$$L_{00} = \frac{1}{4} \cdot M_{00}, \qquad L_{20} = \frac{5}{4} \cdot [\frac{3}{2} M_{20} - \frac{1}{2} M_{00}],$$

$$L_{10} = \frac{5}{4} \cdot M_{10}, \qquad L_{30} = \frac{7}{4} \cdot [\frac{5}{2} M_{30} - \frac{1}{2} M_{10}], \tag{3.14}$$

$$L_{11} = \frac{9}{4} \cdot M_{11}, \qquad L_{21} = \frac{15}{4} \cdot [\frac{3}{2} M_{21} - \frac{1}{2} M_{01}].$$

B Chebyshev Moments

The Chebyshev moments are based on the Chebyshev polynomial representation of a function. The recurrence relation for the one-dimensional Chebyshev polynomial is [29] [30]:

$$t(0) = 1,$$

$$t(1) = x, \tag{3.15}$$

$$t(n) = 2 * x * t(n-1) - t(n-2).$$

where n is the order of the Chebyshev polynomial. From these polynomials, a set of orthogonal moments can be generated based on the traditional geometric moments.

The generating function for the Chebyshev moments of order *(m+n)* from the geometric moments is computed by [30]:

$$T_{mn} = \frac{1}{scale} \cdot \sum_{j=0}^{m} \sum_{k=0}^{n} C_{mj} C_{nk} M_{jk}, \qquad (3.16)$$

where C_{ij} are the coefficients from the Chebyshev polynomial generating function (3.16), M_{ij} are the traditional geometric moments of order *(i+ j),* and the scaling parameters ensure the Chebyshev polynomials are orthonormal. This scale function is not a closed-form expression, but can be calculated via numeric integration to provide the correct scaling factors. The scale factors up to order 6 are provided in Table 3.1.

Table 3.1: Summary of Chebyshev scale factors for up to order 6.

Polynomial Order	Scale factor
0	2
1	2/3
2	14/15
3	34/35
5	62/63
5	98/99
6	142/143

The first few Chebyshev moments are listed below [29] [30]:

$$T_{00} = \frac{1}{4} \cdot M_{00}, \qquad T_{20} = \frac{15}{28} \cdot [2 \cdot M_{20} - M_{00}],$$

$$T_{10} = \frac{3}{4} \cdot M_{10}, \qquad T_{30} = \frac{35}{68} \cdot [4 \cdot M_{30} - 3 \cdot M_{10}], \qquad (3.17)$$

$$T_{11} = \frac{9}{4} \cdot M_{11}, \qquad T_{21} = \frac{45}{28} \cdot [2 \cdot M_{21} - M_{01}].$$

Other authors have developed more complex representations of the Chebyshev moments by representing the image over the range [0, N] rather than the more traditional range of [-1, 1] that we adopted [30]. This modified and scaled representation of the Chebyshev moments leads to additional cross terms that are not present in the canonical representation shown above. Notice that our forms of the Legendre and Chebyshev polynomials are identical, except for scale factors and the multiplicative coefficients applied to each geometric moment, and therefore should perform in a similar manner to each other.

There is another very interesting property of the Chebyshev polynomials that can be seen in the representation above. If we let $x = \cos(\theta)$ then we can also write $T_n(x) = \cos(n\theta) = \cos\left(n\cos^{-1}(x)\right)$, which then implies that an even function that is expanded using these $T_n(x)$ is equivalent to the Fourier cosine approximation of the function. Thus the moment transform using the Chebyshev polynomials with this change of variables is equivalent to the discrete cosine transform of the image. While few images are actually even functions, it still allows us to place different feature

extraction algorithms such as moments, Fourier transforms, and even the wavelet transforms on an equivalent footing. This is clear when we recall from Equation (3.18), that the moments transform is a projection onto the basis defined by the kernel *h(x,y)*, with the goal being to define an orthonormal basis set, which is exactly what polynomial, Fourier and wavelet transforms all seek to accomplish.

Perhaps it is now clear why there is no 'best' feature set for representing an image in all applications, since they are all derived from a common mathematical foundation of projections of the image into some basis set. Therefore, a pattern recognition researcher must search for the particular transform that may be best for a specific application.

C Zernike Moments

As with the other orthogonal moments, the Zernike moments can be generated from the traditional geometric moments. While the geometric, Legendre, and Chebyshev moments are defined on the unit square, $x(i) \in [-1,1]$ and $y(j) \in [-1,1]$, Zernike moments are defined on the unit circle $r \in [0,1]$. The equations for generating the Zernike moments from the geometric moments are defined by [29][30][33]:

$$Z_{nl} = \frac{n+1}{\pi} \cdot \sum_{\substack{k=|l| \\ (n-k)=\text{even}}}^{n} \sum_{j=0}^{q} \sum_{m=0}^{|l|} w^{m} \cdot \binom{q}{j} \binom{|l|}{m} \cdot B_{n|l|k} \cdot M_{k-2j-m,2j+m} \qquad (3.19)$$

where we additionally define:

$$w = \begin{cases} -i & l > 0 \\ +i & l \le 0 \end{cases}, \quad q = \frac{1}{2} \cdot \left(k - |l|\right), \quad i = \sqrt{-1},$$

and lastly:

$$B_{n|l|k} = (-1)^{\left(\frac{n-k}{2}\right)} \cdot \frac{\left(\frac{n+k}{2}\right)!}{\left(\frac{n-k}{2}\right)! \cdot \left(\frac{k+|l|}{2}\right)! \cdot \left(\frac{k-|l|}{2}\right)!}. \tag{3.20}$$

The first few Zernike moments are listed below [29] [30] [33]:

$$Z_{00} = \frac{1}{\pi} \cdot M_{00},$$

$$Z_{10} = 0,$$

$$Z_{11} = \frac{2}{\pi} \cdot \left(M_{10} - i \cdot M_{01}\right), \tag{3.21}$$

$$Z_{20} = \frac{3}{\pi} \cdot [2 \cdot M_{20} + 2 \cdot M_{02} - M_{00}],$$

$$Z_{21} = 0,$$

$$Z_{22} = \frac{3}{\pi} \cdot \left(M_{20} - M_{02} - 2 \cdot i \cdot M_{11} \right),$$

$$Z_{30} = 0,$$

$$Z_{31} = \frac{4}{\pi} \cdot \left(-2 \cdot M_{10} + 2 \cdot i \cdot M_{01} + 3 \cdot M_{30} - 3 \cdot i \cdot M_{21} + 3 \cdot M_{12} - 3 \cdot i \cdot M_{03} \right), \quad (3.22)$$

$$Z_{32} = 0,$$

$$Z_{33} = \frac{4}{\pi} \cdot \left(M_{30} - 3 \cdot i \cdot M_{21} - 3 \cdot M_{12} + 3 \cdot i \cdot M_{03} \right)$$

Note that the Zernike moments are complex-valued, and have the following interesting characteristics [33]:

$$\begin{aligned} &Z_{nl} = 0 \text{ if } (n-l) \text{ is odd,} \\ &Z_{nl} = Z^{*}_{n,-l}, \text{ and} \\ &\phi_{nl} = \left| Z_{nl} \right|^2 \text{ is rotationally invariant.} \end{aligned} \quad (3.23)$$

Therefore, we will not use $\left| Z_{nl} \right|^2$ as features for the airbag suppression problem because of their invariance property.

3.2.2 Fourier Descriptors

Boundary features are related to the silhouette feature space defined in Section 3.1.2. Boundary features are obtained by retaining the boundary of a binary-

segmented region. All internal structure is ignored, and the space merely represents the overall shape of the occupant on the seat. The benefit of this feature space is the simplicity of the representation. The object of interest is merely represented by the boundary of the segmented region. Boundaries have been consistently represented by Fourier descriptors [83]. The original Fourier descriptors measure the change in the angle between successive line segments along a closed contour, and generate the following descriptor values [22]:

$$a_n = -\frac{1}{n\pi} \cdot \sum_{k=1}^{N} \Delta\varphi_k \cdot \sin\frac{2\pi n t_k}{T},$$

and (3.24)

$$b_n = -\frac{1}{n\pi} \cdot \sum_{k=1}^{N} \Delta\varphi_k \cdot \cos\frac{2\pi n t_k}{T},$$

where N is the total number of line segments, T is the total length of the contour such that $T = \sum_{k=1}^{M} t_k$, and $\Delta\varphi_k$ is the change in angle between line segment *k-1* and line segment *k*. For our application, where the occupant and the seat are segmented as a single entity, Fourier descriptors are not practical.

3.3 Feature-less Representations

It has also been recognized that in many applications feature extraction is difficult since there is no clear feature space for efficiently representing the data [48].

In these instances, forcing the system to extract features may result in a loss of classification accuracy. In such situations, it may be better to use the raw image or the segmented image directly as the input to the classifier [48]. There are two general approaches to featureless representation that are explored in this research, namely template matching and image sub-sampling [22][48]. Template matching is, in itself, a classification mechanism, where the 'best' matching template is deemed to be the class of the object [27]. The image sub-sampling is a featureless classifier, also called the appearance-based classifier.

3.3.1 Template Matching

There are many approaches to template matching based on the characteristics of the template. A complete taxonomy of these include:

1) solid 2-D templates,
2) line templates, and
3) point-set templates.

A Solid 2-D Templates

In solid 2-D templates, a two-dimensional image of the desired object is stored, and is either convolved over the entire image or over a region of the input image. The typical measures for the goodness of the match between the template and the input image are: (i) sum of absolute difference (SAD), (ii) mean square distance

(MSD), and (iii) correlation (Corr). They are each defined for an *MxN* image to be [22]:

$$SAD = \sum_{i=1}^{M} \sum_{j=1}^{N} |I(i,j) - T(i,j)|, \tag{3.25}$$

$$MSD = \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - T(i,j))^2, \tag{3.26}$$

$$Corr = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j) \cdot T(i,j)}{\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j)^2 \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} T(i,j)^2}}, \tag{3.27}$$

where I is the input image, and T is the template. When the correlation measure is used, the 2-D template matching is also called a correlator detector, since the operation can be applied to an entire image and detection occurs if the correlation exceeds a threshold [44].

B Line Templates

The line templates are most often used as representations for Hough transforms. The basic Hough transform detects lines in imagery while the generalized Hough transform can detect any reasonably defined shape [159]. This approach is not considered for the airbag suppression application due to the complexity of the shape of the human occupant, and the resulting difficulty in defining the occupant shape in an algebraic form.

C Point Set Templates

There are two basic methods for matching point sets: (i) point-to-point correspondence and (ii) set-level point matching. The deformable template (or snake) is an example of the point-to-point correspondence methods, and the Hausdorff distance is an example of the set-level point matching [23] [159].

Deformable templates are also called deformable contours. active contours, or snakes. They fit a known closed curve to a contour within an input image. The contour is actually a chain of discrete points rather than an actual continuous contour, so it is considered point set matching. The algorithm assigns an energy function to each possible contour shape in an image, and then the contour with the lowest energy is considered the best match with the incoming template [159]. In the discrete form, the energy functional is defined over a set of N points to be [159]:

$$E = \sum_{i=1}^{N} \left(\alpha_i \cdot E_{cont}(p_i) + \beta_i \cdot E_{curv}(p_i) + \gamma_i \cdot E_{image}(p_i) \right), \qquad (3.28)$$

where the terms α_i, β_i, and γ_i are weights to set the relative importance of each term in the energy functional.

Each of the three terms in equation (3.28) controls a different characteristic of the contour. $E_{cont}(p_i)$ controls the continuity of the curve and is defined to be [159]:

$$E_{cont}(p_i) = \left(\bar{d} - \left\| p_i - p_{i-1} \right\| \right)^2, \qquad (3.29)$$

where $\bar{d}$ is the average distance between all the neighboring points. This term promotes equal spacing between the points. The next term, $E_{curv}(p_i)$, controls the smoothness of the contour. It is the second derivative of the contour and is defined to be [159]:

$$E_{curv}(p_i) = \left\| p_{i-1} - 2 \cdot p_i + p_{i+1} \right\|^2 . \qquad (3.30)$$

The last term is actually the only term that involves the image values. It is a measure of the gradient of the intensity image at each point on the contour. It is defined to be [159]:

$$E_{image}(p_i) = -\|\nabla I\| . \qquad (3.31)$$

The deformable template algorithm is designed to iterate multiple times over the image, until a pre-defined percentage of points on the contour reaches a minimum. The iteration can be controlled by a number of search algorithms. One of the common algorithms is the greedy search algorithm [159]. This algorithm consists of two steps, the first step searches in a small neighborhood of each point and finds the new point that provides a minimal energy contribution. The next step looks for any sharp corners that may have resulted and removes them by setting $\beta_i = 0$ which effectively ignores the curvature (the curvature is maximum at a corner).

Note that the deformable template algorithm performs a 1-to-1 matching of every point on the known contour with a point in the input image. If there are

occlusions or gaps in the image contour, due to illumination effects or other problems in the image, it will fail to find proper point matches and may not converge properly.

The Hausdorff distance metric does not require the same number of points to exist in the template as in the input image. The input image must be reduced to a binary edge image through edge processing and thresholding, and then these edges are treated as a point set and compared with the template point set. The Hausdorff distance, *H(A, B)*, between two point sets, $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_m\}$, is [43]:

$$H(A,B) = \max(h(A,B), h(B,A)), \tag{3.32}$$

where

$$h(A,B) = \max_{a \in A} [\min_{b \in B} \|a - b\|]. \tag{3.33}$$

The above definition is more clearly understood when Equation (3.32) is substituted into Equation (3.31), and the equation for Hausdorff distance is re-written as [43]:

$$H(A,B) = \max\left[\max_{a \in A} [\min_{b \in B} \|a - b\|], \max_{b \in B} [\min_{a \in A} \|b - a\|] \right]. \tag{3.34}$$

The Hausdorff distance identifies the point $a \in A$ that is at the farthest distance from any of the points in set *B*, and also measures the distance from that point $a \in A$ to its closest neighboring point in the set *B*. The Hausdorff distance metric effectively defines a ranking of each point in set *A*, based on the distance between it and its

nearest neighbor in the point set B. This is accomplished through the $min||a-b||$ operation in equation (3.16). It then uses the largest ranking of each of these points (the max operation in the equation). Unlike the previous point set matching algorithm, the Hausdorff algorithm does not attempt to find a specific correspondence between points in A with points in B.

This implementation of the Hausdorff distance is defined for a fixed set of points in set A relative to a fixed set of points in set B. The Hausdorff distance can also be used to find the best matching set of points by defining a group G of transformations (using the theoretical definition of a group from abstract algebra) to be [43]:

$$H_G(A,B) = \min_{g \in G} H(A, g \circ B), \qquad (3.35)$$

where G is a group of transformations. The group G is the set of all translations if $\|\bullet\|$ in equation (3.34) is any norm, and the group G is the set of all rigid motions if $\|\bullet\|$ is the Euclidean norm. In equation (3.35) A would be the set of points from the input image, and B would be the set of reference template points.

3.3.2 Image Sub-sampling

Image sub-sampling is a simple appearance-based feature vector mechanism. In image sub-sampling a feature vector is generated by performing a raster scan of the image, keeping every *1*-of-*N* pixels, and storing them in a vector, as shown in Figure 3.15. It is a very simple method that is typically preceded by an image translation

operation to properly align the feature vectors. In appearance-based processing, the actual pixel value at each of the sampled regions can be included in the feature vector, or the mean or median value of a window about the sampled pixel can be included. The wavelet representation is a natural representation to be used for appearance-based recognition [9]. Recall the Haar wavelets provide directional edge information regarding an object at varying resolutions. The appearance vector can be generated using different resolutions at different areas within the image.

(a) (b)

(c)

Figure 3.15: Generation of feature vector for image sub-sampling for an adult image, (a) original image, (b) 10:1 sub-sampled image, and (c) rasterized sub-sampled image.

(a)

(b)

(c)

(d)

Figure 3.16: Appearance-based feature comparison, (a) sub-sampled empty image, (b) sub-sampled adult image, (c) rasterized empty image, and (d) rasterized adult image.

3.4 Results of Feature Extraction

Recall from the above discussions that for our airbag suppression application, we believe the edge-based representations have the highest likelihood of providing the discrimination power needed for both the two-class and our four-class problems. Since it is clear from the imagery that the Haar wavelet transform is simply an edge detector at varying resolutions, we will concentrate simply on the edge-based features for this stage of the research, since there does not appear to be any benefit to be gained from the wavelets. We observed from the silhouette images that we are unable to differentiate a child from an empty seat, and therefore we will also not consider

these. Figure 3.17 shows the segmentation and the edge images for a infant and an adult.

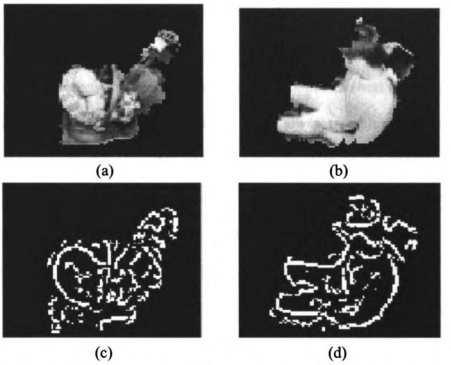


Figure 3.17: Original segmentations and edge images for RFIS and adult classes, (a) RFIS segmentation, (b) adult segmentation, (c) RFIS edge image, and (d) adult edge image.

For feature sets we also looked at the appearance-based methods and the moments methods for the generation of the actual feature vectors. Since we are unable to ensure where in the vehicle the occupant is located, we believe that appearance-based methods will be inferior to moments. Also, with moments, the task of feature selection is more understandable, since the classification of the occupants will be dependent on certain structural characteristics that can be captured by varying the order of moments. For appearance-based feature vectors, however, feature selection will consist of simply ignoring information from specific pixels and not at a

global structural level. In order to understand the capabilities of the moments, we must address two key points (i) what moment order to use from their reconstructions of the edge imagery, and (ii) what specific moment family to use.

We have evaluated the effect of moment order on classification accuracy. For simplicity, we have only used the Legendre moments to explore the effects of moment order on reconstruction, since our goal is to define the best order number for representing the edge-based images. Figure 3.18 shows the resultant reconstructions using moments up to the 25^{th}, 35^{th}, and 45^{th} order for both of the images shown in Figure 3.17. Clearly, there is a significant improvement in the appearance of the reconstruction as the moment order is increased.

The next issue that must be addressed is which family of moments provides the best discrimination ability. For this task we have compared the geometric, Legendre, and Zernike moments. We have tested them for 25^{th}, 35^{th}, and 45^{th} order moments. We compared the moments for the two-class problem of infants versus adults. In order to test the separability of the various moment families, we compared the Z-statistic from the Mann-Whitney test, which tests the inherent separability between two distributions. See Chapter 4 for a detailed description of this test. We computed the Z-statistic for all of the moments, and then sorted the list of moments from smallest to the largest Z-statistic. The results of these tests are provided in Figure 3.19 through Figure 3.21.

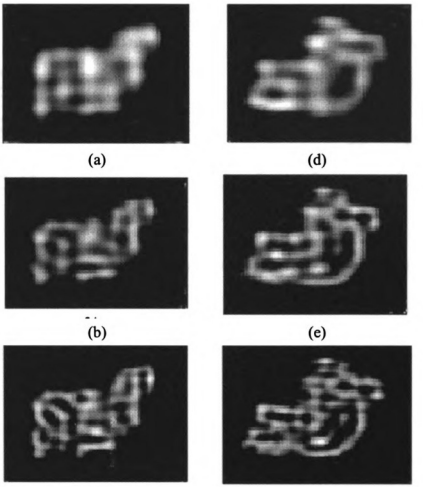


Figure 3.18: Reconstructions of RFIS and adult images, (a) RFIS at 25th order, (b) RFIS at 35th order, and (c) RFIS at 45th order, (d) adult at 25th order, (e) adult at 35th order, and (f) adult at 45th order for Figure 3.17.

Note that for all the tests, the Zernike moments were considerably worse than the geometric or the Legendre moments for discriminating the two classes. Across all of the moments, the geometric moments provided greater separability, however, the Legendre moments with the highest Z-statistic were slightly better than the corresponding geometric moments.

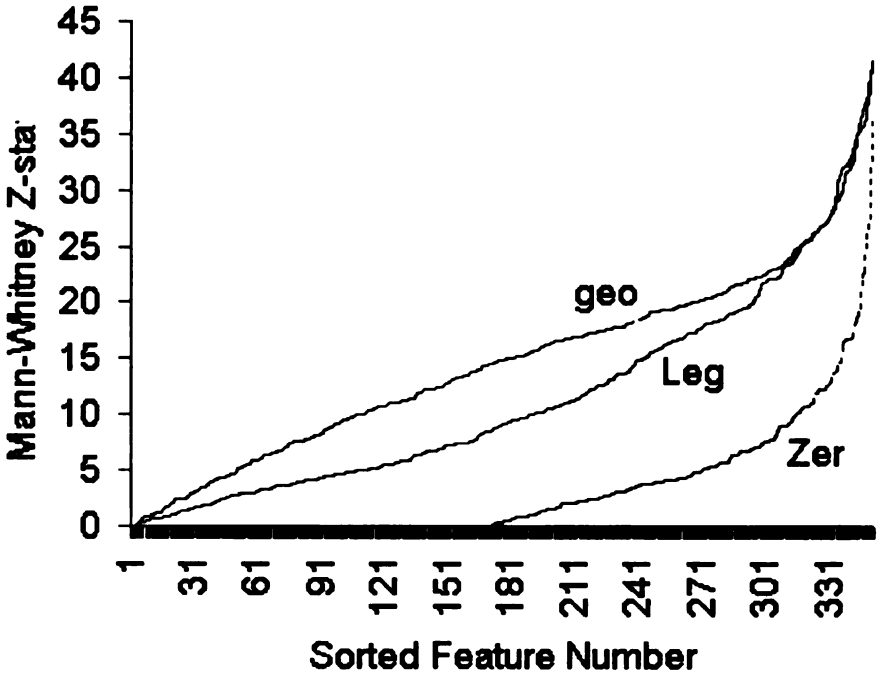


Figure 3.19: Sorted Mann-Whitney Z-statistic for 25th order moments.

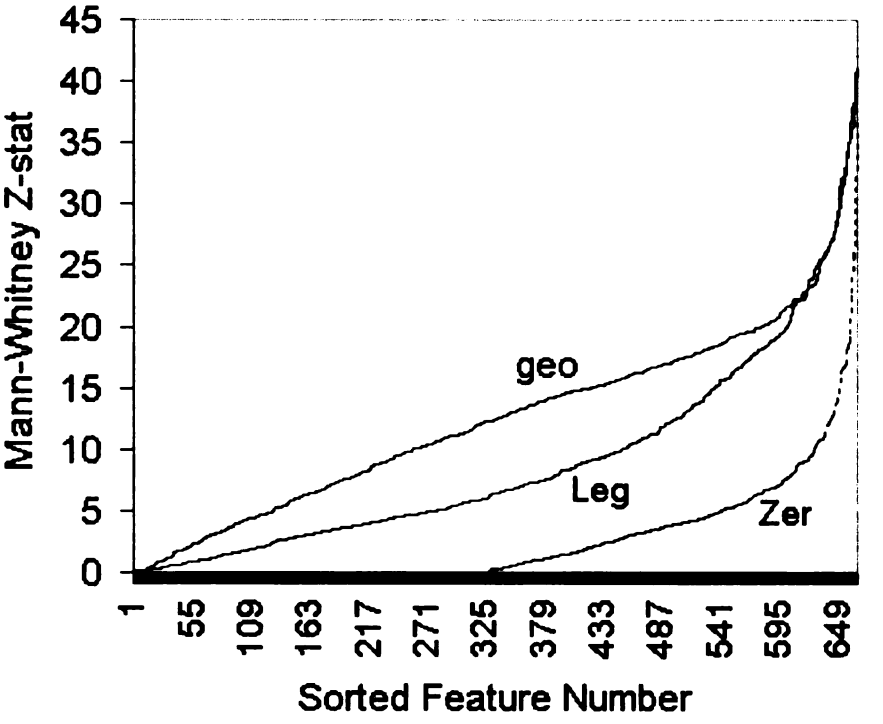


Figure 3.20: Sorted Mann-Whitney Z-statistic for 35th order moments.

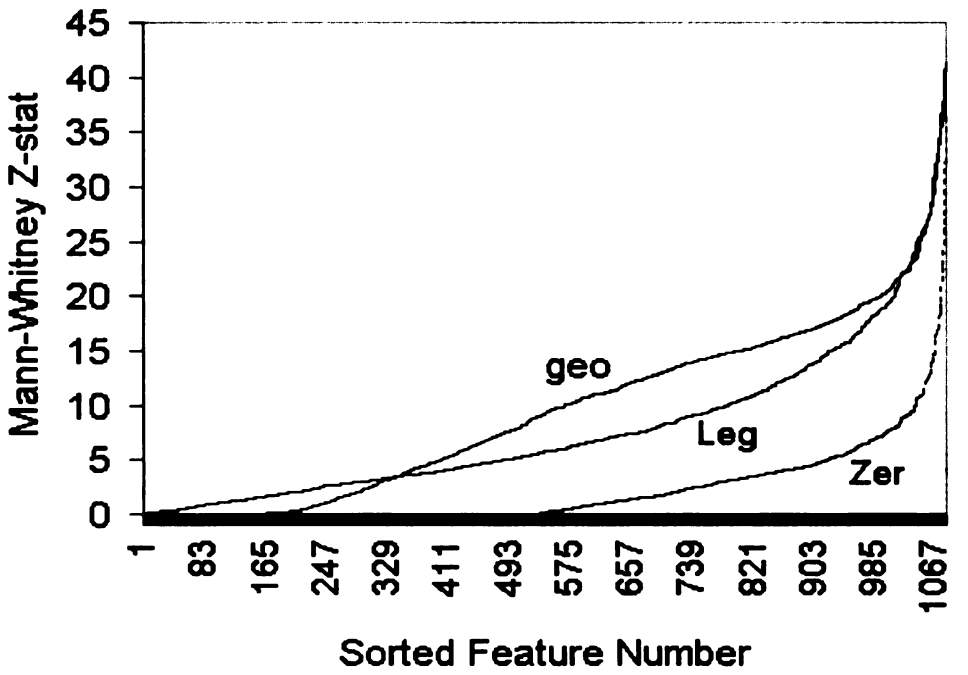


Figure 3.21: Sorted Mann-Whitney Z-statistic for 45th order moments.

3.5 Summary

For the task of feature extraction, we have presented a number of possible data representations, including: grayscale, silhouette, edge, and wavelets. We determined that for our airbag suppression application, the edge representation was the most suitable. We also presented two alternative methods for developing a feature vector from the edge images, namely moment analysis and appearance-based. For the airbag application, the moment analysis appeared the most suitable since we are trying to develop features that describe the underlying shape of the occupant at a global level, while the appearance methods describe the occupant at the pixel level.

We demonstrated that the higher moment orders provide the best reconstruction accuracy for these edge images, and we consequently focused on the 45^{th} order moments. Moments higher than 45^{th} order require arithmetic precision that exceeds the IEEE double precision floating-point arithmetic, and, due to the real-time nature of our system, we decided to limit ourselves to double precision.

We performed an analysis of which moment family provides the best separability between classes for our airbag suppression application. We compared geometric, Legendre, and Zernike moments and found that the results with Zernike moments were far worse than the other two moment types. Also we found that, while the geometric moments provided generally better separability for all moment orders, the Legendre were slightly better for the features with the highest discrimination ability. Also, since the Legendre moments form an orthogonal basis within which the image is represented, we believe this ultimately will allow us to require fewer moments.

Chapter 4.
Feature Selection Methods

Feature selection is a critical component of many pattern recognition applications. Feature selection is very simply defined as: "given a set of d features, select a subset of size m features that leads to the smallest classification error" [27]. There are three primary goals in feature selection, including [72]:

1) reduce the processing time to extract features,
2) improve the classification accuracy, and
3) improve the reliability of the system performance estimates (i.e., system generalizability).

It is easy, with the current state-of-the-art processors, to extract many features from an image in real-time. Also, some of the feature extraction methods mentioned in the previous sections can produce a large number of features. For example, computing the moments of an image up to the 45^{th} order generates 1,081 features. Appearance-based methods can also generate a large number of potential features. For example, sub-sampling our 400x320 pixel image at every 10^{th} pixel generates a feature vector with 1280 components.

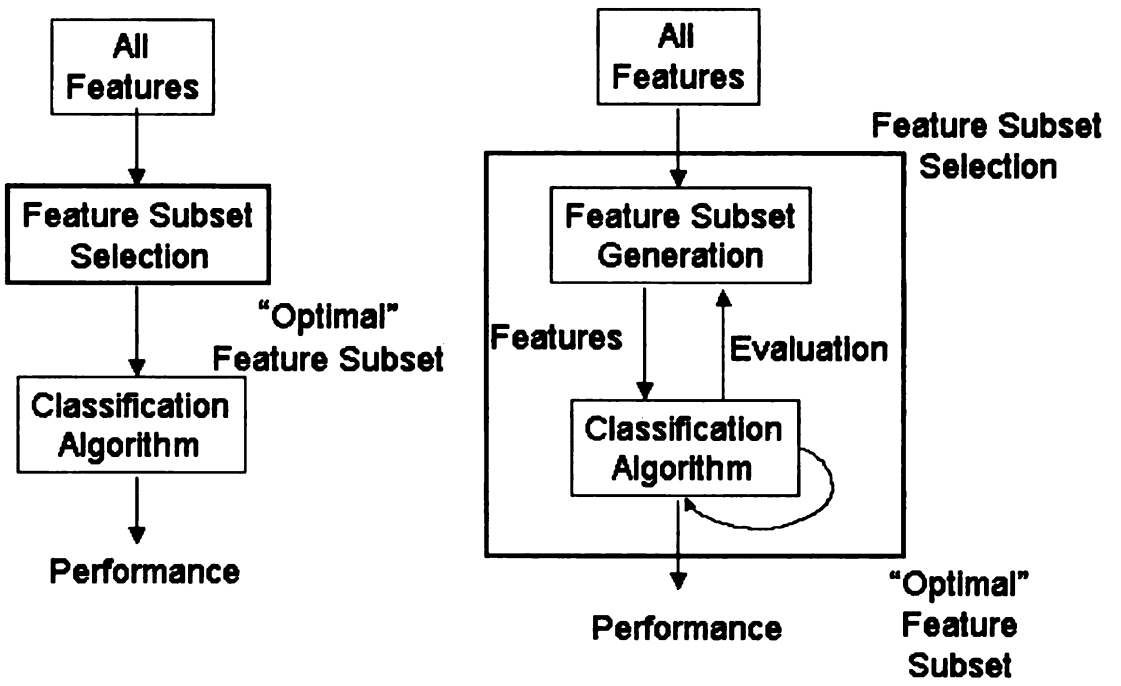
The classification performance of the optimal Bayes rule improves monotonically with the number of features. This implies that adding more features can never reduce classification performance [66]. Unfortunately, since we are rarely able to attain this optimal rule due to lack of information regarding underlying

distributions, the 'curse of dimensionality' comes into play. This states that the performance of the classifier, in the presence of a finite number of training samples, may not actually improve as the number of features increases. In many applications the performance actually degrades as the number of features increases beyond some optimal number [27][45]. Other authors have found, however, that, in most real-world applications, if a reasonable number of training samples have been collected then every feature provides some amount of discrimination ability, though it may be very small [67]. We will investigate whether the curse of dimensionality is a factor in our application. Aside from the classifier performance, since the airbag suppression system is designed to be an embedded real-time application, there is clearly a computational complexity limit within which the processing hardware must operate.

The most straightforward means of feature selection would be to try all possible combinations of subsets of size m from the initial set of d features, but this leads to a combinatorial explosion, since the number of subsets increases exponentially. For the airbag suppression application, we use moments up to the 45^{th} order, producing 1081 features, while to support our real-time requirement we would like to limit the number of features to no greater than 50 features. This would result in 5.1×10^{86} possible subsets of size 50 from this initial set of 1081. If subsets of all possible sizes were considered, there would be greater than 2.6×10^{325} possible combinations. It is also well known that any non-exhaustive search method cannot be guaranteed to find the optimal feature set [27].

Two common and distinct mechanisms for feature selection have been devised to make the problem computationally manageable, the wrapper method and the filter

method [57][64][65]. Wrapper methods use a specific classifier, and then use the resultant probability of error from the classifier, to select the feature subsets. The feature selection algorithm is wrapped inside the classifier. Filter methods analyze features independently of the classifier and use a 'goodness' metric to decide which features should be kept. Figure 4.1 demonstrates the difference between the filter and the wrapper methods [57].



(a) (b)

Figure 4.1 : Methods of feature selection, (a) filter approach, and (b) wrapper approach [24].

One key issue in choosing a feature selection method is the choice of the selection criterion [27]. For wrapper methods the obvious choice of a criterion is the classification error. For these methods we will choose only those features that result

in a net reduction in the probability of classification error. Of course, the use of a different classifier in the wrapper can lead to the selection of different features [27]. In the filter methods the choice of a criterion function is more complicated since the probability of error is not directly computed as part of the feature selection. Here, other methods, such as measures of statistical separability and uniqueness of information content of the features, are used. The simplest filter method processes each feature individually, but it is often advantageous to analyze the interdependencies of features (e.g., correlation among features), as well.

The taxonomy for feature selection methods, shown in Figure 4.2, divides these methods into three classes: (i) complete, (ii) heuristic, and (iii) random. [57]. While the literature has shown no clear superiority of any particular feature selection method, some are more suitable for large-scale applications than others. For example, Jain and Zongker [56], and Kudo and Sklansky [72] agree that simple "individually best" algorithms tend to perform worse than other methods, but they are much faster. These authors also agree that the floating search methods perform well in most applications, however, Kudo and Sklansky found that they are inferior to genetic algorithms in very large-scale problems. On the other hand, Jain and Zongker found that genetic algorithms tend to perform worse on large-scale problems, while Kudo and Sklansky found that they are the superior algorithms for large-scale problems [56][72]. Clearly, there is still no consensus as to which method is the best, since there is such a strong dependence of the performance of the algorithm on the data sets being analyzed.

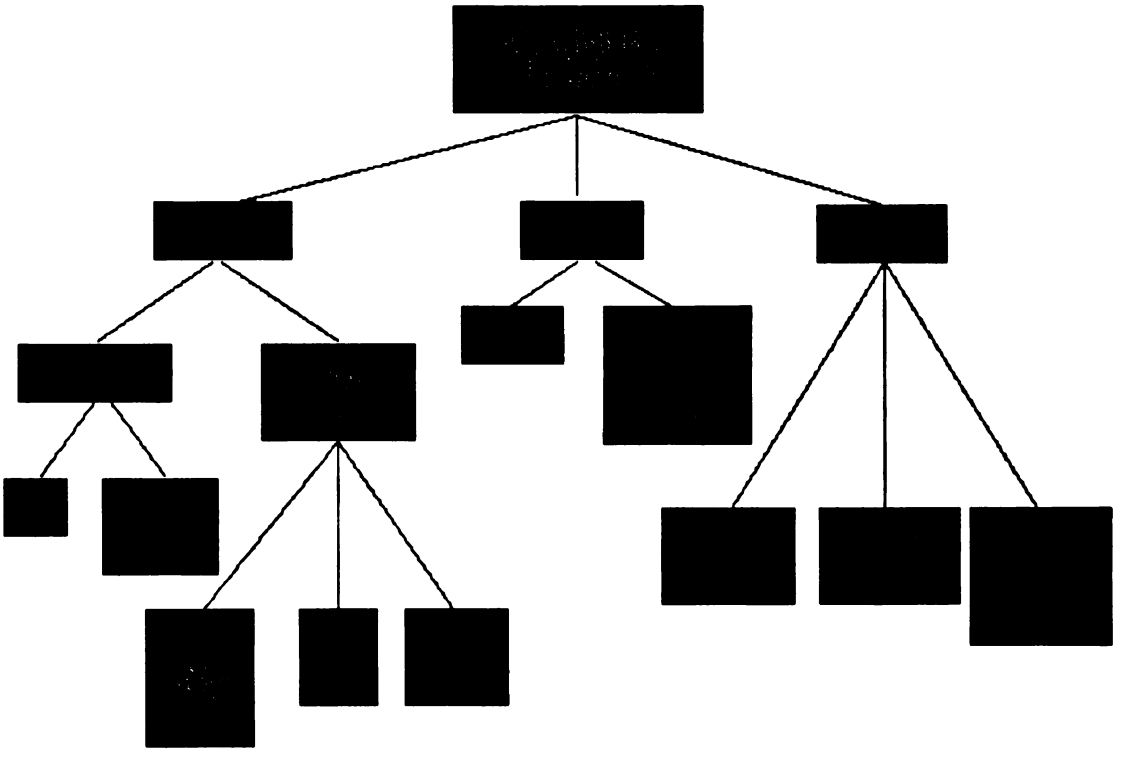


Figure 4.2 : Taxonomy of feature selection methods [57].

Table 4.1 provides a summary of the feature selection methods we have tested for our airbag suppression application and their categorization relative to the above taxonomy. Note that the set of feature selection algorithms we have investigated for our application span the tree in Figure 4.2. For our large-scale feature selection problem, the complete-exhaustive methods were deemed computationally too intensive to be considered. Note that we have at least one algorithm for both the filter and wrapper implementations, and one from the categories of complete, heuristic, and random. In addition to these methods, we will investigate if one can improve the performance of the filter methods by removing correlated features.

Table 4.1: Summary of Feature Selection Methods Compared for the Airbag Suppression Application

	Feature Selection Approach			
		Complete	**Heuristic**	**Random**
Feature Selection Implementation	**Wrapper**	-	Forward Sequential Search	Random Mutation Hill Climbing
	Filter	Mann-Whitney, Mutual Information	-	-

4.1 Wrapper Methods

In wrapper methods the objective is to find an optimal feature subset that maximizes the accuracy of the given classifier [66]. Since the classification engine is an integral part of the selection process, the method for estimating its performance must be defined. The most common method is an n-fold cross-validation process, where the data set is randomly divided into n subsets, and $n-1$ subsets are used for training, and the remaining one for testing. Since these subsets are determined at random, this partitioning can be repeated multiple times to derive better estimates of the classification accuracy and its variance. To ensure generalizability, it may be better to select a feature set with slightly lower classification accuracy, if its variance is significantly lower than for the best feature set. Also, it is important to note that any change in the classification algorithm, for example a change in the value of k in

the k-nearest neighbor decision rule, or a change from the k- nearest neighbor algorithm to the SVM algorithm, can lead to a different feature subset.

4.1.1 Forward Sequential Selection

In the forward sequential selection method the system begins with the empty set, and then iteratively adds features until a criterion function is optimized [62]. The forward sequential selection method can be generalized (plus-*l* take-away-*r*) to allow for the addition or removal of subsets of features at a time rather than single features. The plus-*l* take-away-*r* algorithm begins with an initial set of features, X_k (this set can be the empty set or any randomly chosen subset of the features), and proceeds as defined in the algorithm given in Table 4.2 [62]. Since it is a wrapper-based algorithm, it uses the classification accuracy derived from the subsets of features to determine the best set.

Table 4.2: Algorithm for plus-*l* take-away-*r* forward sequential search feature selection.

1) Inclusion: Add the most significant feature that remains in the unused set to the selected features so that $X_{k+1} = X_k + x_{k+1}$, where lower case x_{k+1} is the feature selected on pass $k+1$.
2) Exclusion: Test all of the features in the selected feature set and, if the removal of any feature, x_r, improves the classification accuracy, $P_{class}(X_{k+1} - x_r) \geq P_{class}(X_k)$, then discard that feature. If this decision is true then it means that feature x_r is deteriorating the performance of the system. Step (2) continues until all the features in the subset are tested.
3) Return to step (1) unless the last feature has been processed.

4.1.2 Random Mutation Hill Climbing

Random Mutation Hill Climbing is a member of the family of random search optimization tools that include methods such as simulated annealing and genetic algorithm [59][60]. Random search algorithms derive their power from the ability to search the optimization space in a random manner, which makes them more robust against local minima. For the random mutation algorithm, the complete set of features is represented as a string of binary values, where a bit in the string is set to *'1'* if the associated feature is to be kept in the final subset, and set to *'0'* if the feature is to be discarded [59].

The key free parameter to set when using this algorithm is the number of bits that are allowed to randomly change at a given iteration of the algorithm. The most

conservative approach is to only allow a single bit to change per pass [59]. We have adopted a method that is motivated by simulating annealing, where initially a larger number of bits are allowed to mutate, and then, as the algorithm progresses, this value is reduced until in the final iterations only a single bit is allowed to change. This is similar to the cooling strategies used in simulated annealing [60]. In this approach the number of bits to mutate begins at some number *M*, and then decays to only one mutation per iteration over the course of execution. The random mutation algorithm operates as shown in Table 4.3.

The definition of the fitness function is also required for random mutation hill climbing. Since this is a wrapper algorithm, the fitness function should be a function of the classification accuracy. Note the fitness function must also be a function of the number of features remaining, or else there will be no incentive to reduce the number of features.

Consequently, our fitness function is a weighted average of the classification accuracy and the number of remaining features, according to [55]:

$$F(S) = \alpha \cdot P_{class}(S) - (1-\alpha) \cdot \frac{Cardinality(S)}{N}, \qquad (4.1)$$

where N is the original number of features, S is the current set of features, and $0 \leq \alpha \leq 1$. The parameter α is set based on how aggressively the algorithm reduces the number of features. A smaller α encourages final solution to have a lower number of features, since it reduces the importance of the classification accuracy in favor of the cardinality of the remaining feature set.

Table 4.3: Algorithm for random mutation hill climbing feature selection.

1) Set M, the number of bits to mutate per iteration, to its maximum value; then randomly select M features for the initial feature vector.
1) Test the sample for fitness: Compute $P_{class}(\{X_k\}|C)$, where the set $\{X_k\}$ is the current feature subset. If $P_{class}(\{X_k\}|C) \geq P_{class}(\{X_{k-1}\}|C)$, keep this feature set. If not then keep this feature set anyway with a pre-determined probability P_{mutate}.
2) The fitness test used for evaluating the population is defined in Equation (4.1).
3) Compute M, the number of bits to mutate per iteration, and then randomly mutate M bits in the current feature vector.
4) Return to step (2) and continue until either the fitness measure has converged or the maximum number of iterations is reached.

4.2 Filter Methods

Filter methods for feature selection do not consider the classification algorithm in the actual selection process, but rather define the optimal feature subset by some other criterion. The filter methods are generally considered inferior to the wrapper methods for the following three reasons [66]:

1) inability to remove a feature in a symmetric feature space (all features equally separable),
2) inability to include irrelevant features that may help in classification, and

3) inability to remove correlated features.

It is important to note that reasons (1) and (2) are generally applicable for fabricated data sets, and are of no consequence in the real world (See Kohavi [66] for descriptions of these data sets). Also item (3) is not necessarily true for filter methods. Any filter method can be modified to include correlation among features as an element of the criterion function. For example, joint mutual information inherently considers the correlation between features. Therefore, we contend that there is no longer a clear distinction between the wrapper and filter methods in terms of performance. What is most important in feature selection is to develop a criterion function that provides the most insight into the discrimination abilities of the features and the various subsets of these features.

4.2.1 Mann-Whitney Test

Since we are interested in finding the subset of features that maximizes the discrimination between the classes, it is possible to frame the problem in a purely statistical manner. If each labeled feature value is considered to be from a distinct distribution, then a good feature selector would be the one that chooses the features based on maximizing the distances between these distributions.

Non-parametric statistics provides a number of tests to determine whether multiple data sets are from the same, or from different, underlying distributions [73]. We are interested in non-parametric tests, since we do not have any knowledge of the

underlying class-conditional distributions. There are two specific measures that are applicable to this problem, the Mann-Whitney test and the Wilcoxon signed-rank test.

The Wilcoxon signed-rank test is used when the samples being used for the comparison may be correlated [73]. Since our training data is a collection of images of numerous occupants, collected in a variety of vehicles, and over many days, there is very little concern that the data is correlated over time. Therefore, we will use the Mann-Whitney test. Note that the Mann-Whitney test is a two-sample (or two-class) distribution test. It can be extended to the four-class problem by either running the feature selection pair-wise on the data and collecting the best features for each class pair, or it can be replaced with the related Kruskal-Wallis multiple sample-set method, which is also a non-parametric test.

The assumptions that must be met to use either the Mann-Whitney or the Kruskal-Wallis tests are [73]:

1) sample sets are randomly and independently drawn,

2) dependent variable is intrinsically continuous, and

3) measures within the samples have an ordinal scale (<, >, = have meaning).

The mechanics of the Mann-Whitney test are as follows. All of the class labels are removed, and the patterns are ranked from the smallest to the largest for each feature. The labels are then re-associated with the data values, and the sum of the ranks is computed for each of the two classes, labeled A and B. The sum of these ranks is then compared to the sum of the ranks expected if the two data sets were from the same underlying distribution. This expected rank sum, and the corresponding variance, is computed by [73]:

$$\mu_A = \frac{n_A(N+1)}{2}, \text{ and}$$
$$\sigma_{AB} = \frac{n_A n_B (N+1)}{12}. \qquad (4.2)$$

where n_A and n_B are the number of samples from each of the two classes A and B, respectively. The value μ_A is then compared with the actual sum of the ranks for label A, namely S_A. A z-ratio test is used since the underlying distribution of the rank data is normal, based on the weak law of large numbers [73]:

$$z = \frac{(S_A - \mu_A) \pm 0.5}{\sigma_{AB}} \quad . \qquad (4.3)$$

The results of the Mann-Whitney test can be used in three ways:

1) each feature is ranked based on its z-score, and the top M of the original N features can be kept,
2) only features with significance above a pre-defined z-ratio score are included in the final list, and
3) all of the features are ranked based on their Mann-Whitney score, and then the features can be post-processed in the order of decreasing z-ratio score to remove correlated features.

We will use method (3), and use post-processing to remove correlated features. By applying the Mann-Whitney test in this manner, we will use the discrimination ability of each feature merely to order the available features. All of the features are then

analyzed for potential correlations, and only the features that are uncorrelated (at some user-defined confidence level) are retained. The benefit of maintaining all of the features prior to post-processing is that features that may not have the strongest discrimination ability may still be useful when feature correlation is considered. Figure 4.3 shows a typical decay in the Mann-Whitney z-statistic across our complete feature set of the 45th order moments (1081 features). Notice there is a gradual decay in the discrimination ability, and there is no clear point at which the features no longer provide discrimination. This supports Kudo and Sklansky's [67] comment that reducing the dimension of the feature space always results in some loss in discrimination ability in real-world applications.

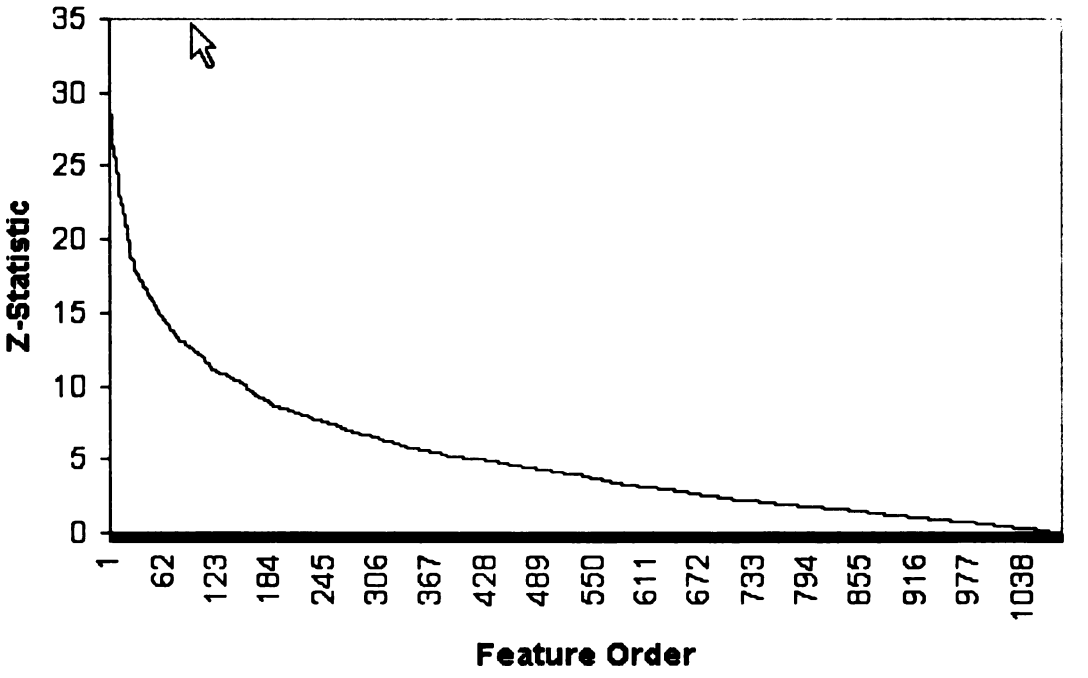


Figure 4.3: Mann-Whitney statistic versus feature ranking for 45th order Legendre moments feature vector.

4.2.2 Mutual Information

Mutual information is a measure between two data sets that determines what can be learned from one data set given knowledge about the other data set [71]. It can also be used to measure the relative independence between two variables. Mutual information can be defined in one of the following three ways [145]:

$$\begin{aligned} &1)\ I(A;B) = H(B) - H(B|A) = H(A) - H(A|B), \\ &2)\ I(A;B) = H(A) + H(B) - H(A,B), \text{ and} \\ &3)\ I(A;B) = \sum_{a,b} p(a,b) \cdot \log\left(\frac{p(a,b)}{p(a)p(b)}\right), \end{aligned} \tag{4.4}$$

where $p(a)$ and $p(b)$ are the individual distributions of data sets A and B, and $p(a,b)$ is the joint distribution of data sets A and B, and a is a value in dataset A and b is a value in dataset B.. Also, $H(A)$ is the entropy of data set A, and $H(A,B)$ and $H(A|B)$ are joint and conditional entropies, respectively. They are defined as [143][145]:

$$\begin{aligned} H(A) &= -\sum_{i \in A} p(i) \cdot \log[p(i)], \\ H(A,B) &= -\sum_{i \in A,\, j \in B} p(i,j) \cdot \log[p(i,j)], \text{ and} \\ H(A \mid B) &= -\sum_{i \in A,\, j \in B} p(i \mid j) \cdot \log[p(i \mid j)], \end{aligned} \tag{4.5}$$

where $p(i)$, $p(i, j)$, and $p(i \mid j)$ are the individual, joint, and conditional density functions, respectively, and the summations are over all elements in the datasets A and B.

Mutual information $I(A;B)$ measures how the uncertainty in A is reduced by knowing B. If A and B are independent, then the mutual information $I(A;B) = 0$, since in the second definition for $I(A;B)$ in equation (4.4), $H(A,B) = H(A) + H(B)$ for independent variables [145]. Features can be selected based on their individual mutual information by considering variable A to be the feature and variable B to be the class label, and then the third form of equation (4.4) is used. Features with the highest mutual information are retained.

Mutual information can be extended to define the joint mutual information between a set of N variables according to the equation [145]:

$$I(X_1, X_2, \ldots, X_N; Y) = \sum_{i=1}^{N} I(X_i; Y \mid X_{i-1}, X_{i-2}, \ldots, X_1). \qquad (4.6)$$

Based on individual mutual information, it is possible that some of the features are redundant, but joint mutual information can simultaneously provide the subset of features that provides the most discrimination and the least redundancy. Clearly, an exhaustive search is not possible due to the number of possible combinations. A typical implementation of joint mutual information is to initially rank all the features, based on their individual mutual information relative to the classification labels. Then these features are iteratively tested to perform the following optimization on a candidate feature, X_j, according to [145]:

$$\text{Maximize}: I(X_j;Y) \quad \text{while}$$
$$\text{Minimize}: \sum_{k=1}^{M} I(X_k, X_j), \tag{4.7}$$

where M is the current number of features being retained, Y are the class labels, and X_j are the individual features. This rule then maximizes the discrimination ability of each feature while simultaneously minimizing the mutual information between all the features.

Since the mutual information is being calculated using the joint probability density functions (PDF) of the two images, it is critical to discuss the calculation of this joint PDF. Since the joint PDF is estimated from the joint histogram, the selection of the number of bins used to compute this histogram is important [143]. Rather than use a fixed number of bins, it has been proposed to select the number of bins based on the non- Gaussianity of the data distribution [145]. The number of bins is computed, according to [145]:

$$N_{bins} = \log_2 N + 1 + \log_2\left(1 + \kappa\sqrt{N/6}\right), \tag{4.8}$$

where $\kappa = \dfrac{1}{\sqrt{24N}\cdot\sigma^4} \cdot \sum_{i=1}^{N}\left(x_i - \bar{x}\right)^4 - \sqrt{3N/8}$.

Here σ is the standard deviation of the image data, κ is the normalized kurtosis of the distribution, and N is the number of pixels. Once the number of bins is selected, the distribution of the bins across the dataset must also be determined. The simplest

method

maximu

bins too

range [

method distributes the bins across the spread of the data in the image, from its maximum value to its minimum value, but this method can be prone to making the bins too coarse due to outliers. An alternative method distributes the bins across the range $[\mu - 2\sigma, \mu + 2\sigma]$, and this greatly reduces the impact of outliers [145].

4.2.3 Feature Correlation Post-processing

While joint mutual information will tell us the relative amount of information provided by one feature, given another feature, it is also possible to directly compute the relative correlation between each feature as well using statistical measures. For the statistical-based approach, we again use the results of non-parametric statistics, and compute the Spearman-R correlation coefficient between all of the features over the training dataset. This value is computed in a manner similar to the traditional correlation coefficient, where the actual values are replaced by their ranks. Thus once again, while no assumptions can be made regarding the distributions of the data values, the ranks of the values can be assumed to be Gaussian [73][74]. The first step in the Spearman-R statistic calculation is to rank the values of each feature individually. Now, the Spearman-R correlation coefficient is defined identically to the traditional correlation coefficient [74]:

$$\rho(A,B) = \frac{Cov(A,B))}{\sqrt{\sigma^2(A)\cdot\sigma^2(B)}}, \qquad (4.9)$$

where *Cov(A, B)* is the covariance of *ranks* of feature *A* with respect to the *ranks* of feature *B*, and $\sigma^2(A)$ is the variance of *ranks* of feature A over all of the training samples.

Given *N* features, this generates an *NxN* correlation coefficient matrix, which can now be thresholded based on the statistical significance of these correlation values. We may use the Student-t test, since, as mentioned earlier, the underlying distributions of the ranks are now Gaussian. In most real-world data sets there is often some level of correlation between all of the features, as shown when we generate a histogram of the correlation values from zero to one, which is provided in Figure 4.4. Therefore, deciding if features are correlated is not a simple binary decision, but rather a decision based on the level of significance of the correlation we are willing to accept in our final feature set. It is this fact that limits the ability of wrapper methods to ensure that final features are not correlated, except in artificially constructed data sets.

The correlation significance test takes the form of [74]:

$$\left| \sqrt{(n-2)} \cdot \frac{\rho(X,Y)}{\sqrt{1-\rho(X,Y)^2}} \right| \geq t_{n-2}, \qquad (4.10)$$

where t_{n-2} is the Student-t test of degree *n-2*, and *n* is the number of training samples. This thresholding process creates an *NxN* binary feature significance matrix where, a *1* (white) indicates a correlated feature, and a *0* (black) indicates an uncorrelated feature. An example of this binary matrix is shown in Figure 4.5. Note

that all the diagonal elements are *1* (white), since each feature is correlated with itself. The algorithm for the feature correlation analysis is defined in Table 4.4.

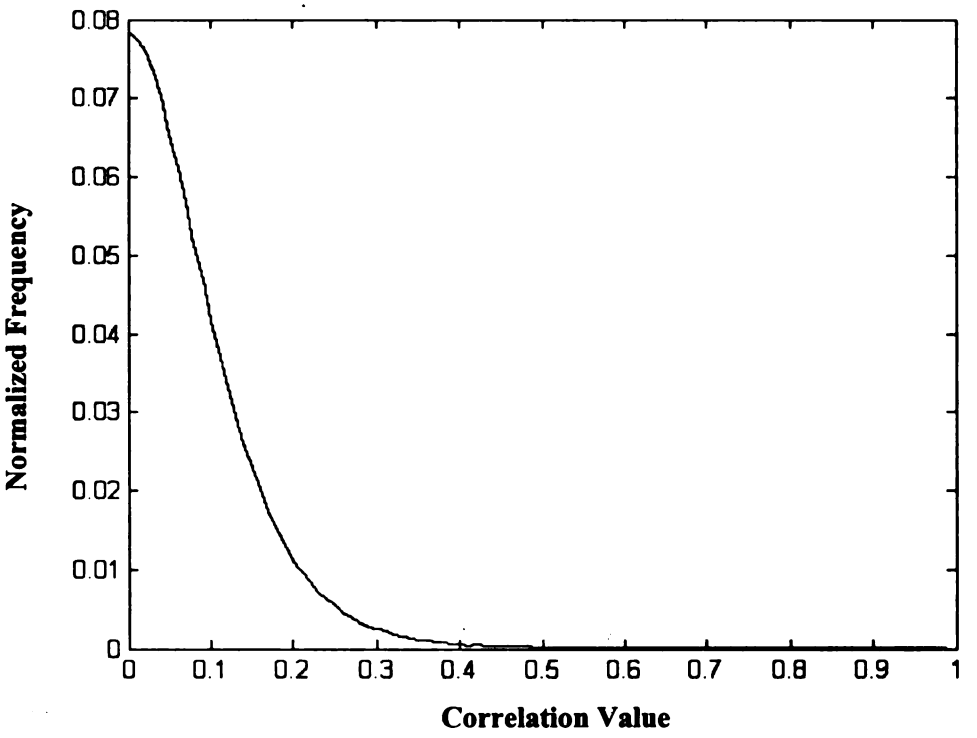


Figure 4.4: Typical histogram of correlation coefficient values for our 1081 element Legendre moments feature vector; horizontal axis is the correlation value and the vertical axis is the frequency of occurrence for each of those values in the dataset.

The intermediate *NxN* correlation matrix, *CM,* defined in step 1) in Table 4.4 is shown in Figure 4.6 (a). The final *NxN* correlation matrix, *CM,* is shown in Figure 4.6 (b). All $CM(j,1) = 0$ (black) signify that the feature is a member of the final feature set. These features comprise the subset of mutually uncorrelated features with the best available discriminating ability.

Table 4.4: Definition of the algorithm for correlation post-processing for feature selection.

1) Create the NxN correlation coefficient matrix, *CM(-,-)*.
2) Threshold *CM* based on the t-test of the coefficients to create a binary version of *CM* as shown in Figure 4.6 (a).
3) Retain the first feature since it has the best discrimination (i.e., make *CM(1,1) = 0* (black) or uncorrelated).
4) For every row in the first column, make row *j* and column *j* all ones (white) if *CM(j,1) = 1* (white). This creates the matrix shown in Figure 4.6 (b).
5) For every row in the first column where *CM(j,1) = 0* (black), test all *i>j* if any *CM (i,1) = 0*, it implies feature *i* is correlated with feature *j*. Make the row and the column for feature *i* all ones (white).
6) Repeat step 5 for all the features remaining in the matrix.

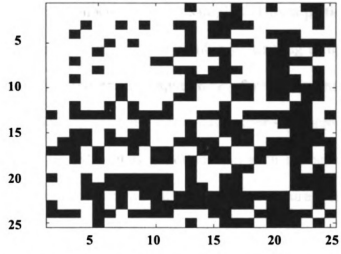


Figure 4.5: Initial binary correlation map during feature correlation processing after t-test thresholding (black squares denote uncorrelated features).

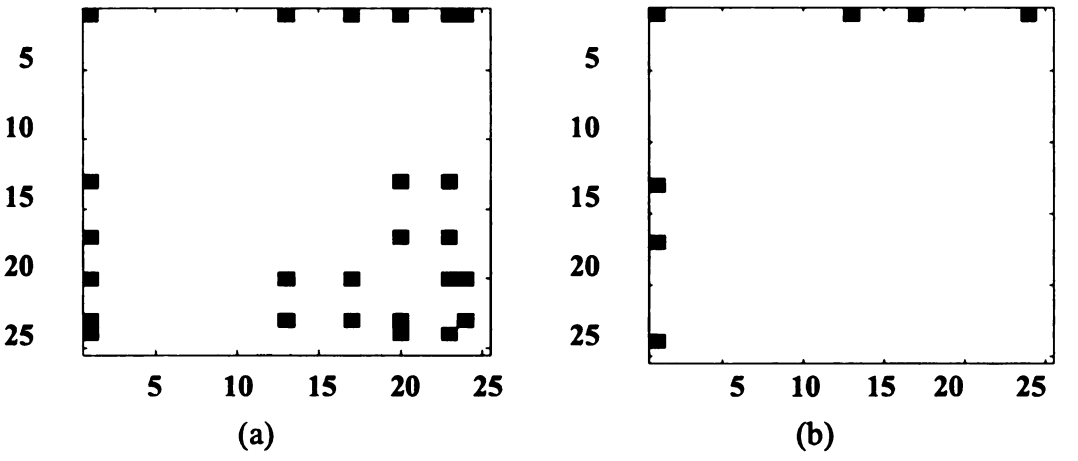


Figure 4.6: Binary correlation map during the feature correlation processing, (black squares denote uncorrelated features), (a) binary correlation matrix after step (4) of the algorithm, and (b) final binary correlation matrix where only 4 of the original 25 features are retained.

4.3 Pruning Methods for Training Set Reduction

Aside from reducing the number of features used by a classifier, it may also be important to limit the number of training samples. This is particularly true if a k-nearest neighbor classifier is used. The number of training samples used directly impacts the memory and processing requirements for the system. Recall that the recommended guideline is to use ten times as many training samples per class as the number of features [27]. Also, it is important that each of these training samples provide as much independent information as possible to ensure that the feature space is adequately sampled. This will maximize the likelihood that performance in the laboratory is matched in the real-world applications, since all possible conditions are covered. The use of redundant training samples can unnecessarily increase the

processing required during training, as well as for in-field operation if they are not removed.

For nearest neighbor classifiers, there are two schemes for training set pruning that solve the problem of reducing the number of training samples in opposite ways. The first method only keeps the training samples that lie along the classification boundaries of the classes [45]. Any training samples that are surrounded only by samples of the same class are discarded, since they are assumed redundant. The second method increases the separation between classes by discarding the samples that would be mis-classified (i.e., lie within the other class or directly along the class boundary) [167]. This method effectively increases the margin between the classes. The final number of training samples can further be reduced by randomly discarding additional samples once the mis-classified samples are removed.

Figure 4.7 shows an example of the two pruning methods on a 2-class problem with two Gaussian data sets. The first method (shown in Figure 4.7 (b)) removes all samples that are correctly classified with a k-nearest neighbor rule with $k=5$. The second method (shown in Figure 4.7 (c)) removes all the samples that are mis-classified. Note the first method leaves all the samples along the decision boundary, while the second method removes samples along the boundary.

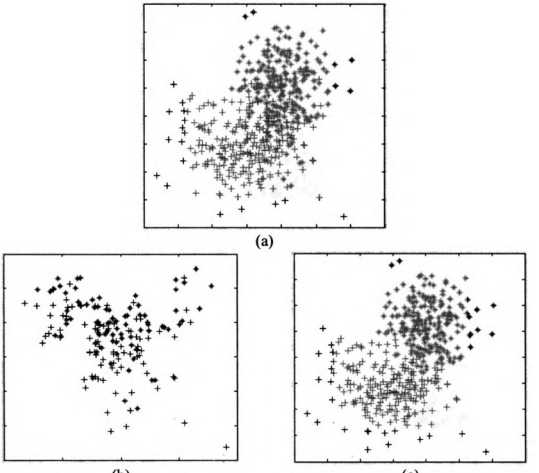


Figure 4.7: Example of pruning on a two-class Gaussian dataset of 200 samples per class, (a) original scatter plot, (b) pruning by removing redundant samples, and (c) pruning by removing mis-classified samples.

Figure 4.8 shows an example of the two pruning methods on a 2-class problem with two Gaussian datasets, labeled *A* and *B*, where one of the two data sets, *B*, is corrupted by:

$$data(B) = 1/3 * data(A) + 2/3 * data(B). \quad (4.11)$$

This data manipulation is designed to simulate our application, where segmentation or other pre-processing errors for one class tend to make the samples look more like the

other class. This can occur when segmentation errors cause a child to look like an empty seat as shown in Figure 4.9.

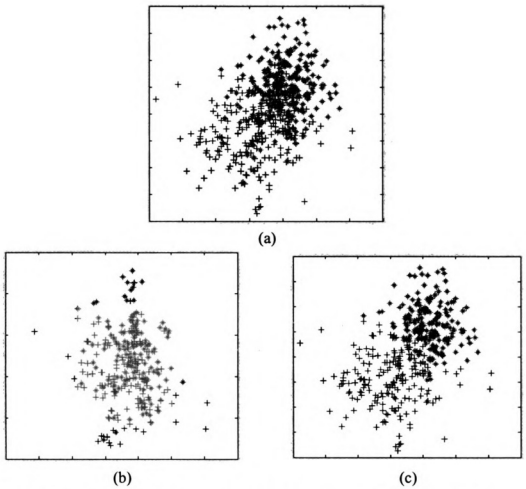
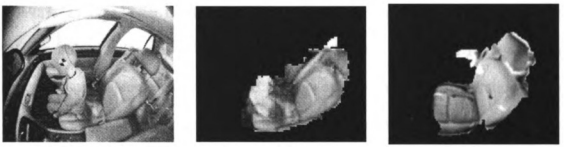


Figure 4.8: Example of pruning on a two-class Gaussian dataset with one class corrupted due to errors such as segmentation error, (a) original scatter plot, (b) pruning by removing redundant samples, and (c) pruning by removing misclassified samples.

The first method (shown in Figure 4.8 (b)) removes all the samples that are correctly classified with a k-nearest neighbor rule with $k=5$. Note now that the decision boundary covers a significant portion of the feature space compared to the results in Figure 4.7. The second method (shown in Figure 4.8 (c)) removes all the

samples that are mis-classified, and therefore, cleans the decision boundary. Since the proper segmentation of the occupants is so difficult in our application, we will use the second pruning method to try to maximize the margin between the classes. Thus we will assume that a sample that is surrounded by sampled from another class is most likely the result of a bad segmentation rather than a boundary-defining data point.



(a) (b) (c)

Figure 4.9: Example of poor segmentation causing confusion between a child and an empty seat, (a) original child image, (b) segmentation of child image appearing to be an empty seat, and (c) an actual empty seat segmentation.

4.4 Results for Feature Selection

We performed classification testing of the feature selection algorithms using a 50/50 cross validation, where we divide our existing database randomly into training and test sets. The database of images we use is defined in Table 2.3. We perform this exercise 10 times, and we generate an average performance result. We use the k-nearest neighbor classifier with $k=9$. We also perform the testing on both the 2-class problem, and the 4-class problem. Recall in the 2-class problem, we test infants

versus adults only. The children are not in the tests since they are protected by either the classifier or by the tracker. In the 4-class problem we classify the occupants as infant, child, adult, or empty seat. All of the tests are performed on the 45^{th} order Legendre moments of the edge representations of the segmented images. The 45^{th} order moments yield 1081 features that need to be reduced as much as possible using each of the feature selection methods. For the two-class problem, without any feature selection, the results are shown in Table 4.5. The overall classification accuracy is 99.98%, averaged over the 10 iterations of the 50/50 cross validation. For the four-class problem, without any feature selection the results are shown in Table 4.6. The overall classification accuracy is 98.93%, again averaged over the 10 iterations of the 50/50 cross validation. These results will serve as the baseline for comparing all of the feature selection methods.

Table 4.5 : Two-class confusion matrix with all the 1081 features used.

	Classified as Infant	**Classified as Adult**
True Infant	1328	0
True Adult	0	492

Table 4.6 : Four-class confusion matrix with all the 1081 features used.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1327	1	0	0
True Child	20	289	1	0
True Adult	1	0	490	0
True Empty	1	0	0	35

4.4.1 Results for Wrapper Methods

The results for the wrapper methods include confusion matrices for both the two-class and four-class problems. For the random mutation algorithm we experiment with varying degrees of the free parameter α. For the forward sequential selection we fix l and r to be both equal to 1.

A Random Mutation Hill Climbing

The random mutation hill climbing algorithm was run for four different weighting parameters of the probability of error, 0.2, 0.4, 0.6, 0.8, and 1.0. The results in terms of the confusion matrix and the actual features selected are provided below. The confusion matrix for the probability of error weighting of $\alpha = 0.2$ is provided in Table 4.7 for the two-class problem. For this α value, the algorithm converged to this solution in 66 iterations. The classification accuracy was 98.57%, and the total number of features retained is 27.

Table 4.7 : Two-class confusion matrix for random mutation hill climbing ($\alpha = 0.2$).

	Classified as Infant	**Classified as Adult**
True Infant	1313	6
True Adult	20	481

The confusion matrix for the probability of error weighting of $\alpha = 0.6$ is provided in Table 4.8 for the two-class problem. For this α value, the algorithm stopped at the maximum of 100 iterations. The classification accuracy was 98.57%, and the total number of features retained is 59.

Table 4.8 : Two-class confusion matrix for random mutation hill climbing with $\alpha = 0.6$.

	Classified as Infant	**Classified as Adult**
True Infant	1319	0
True Adult	14	475

The confusion matrix for the probability of error weighting of $\alpha = 0.8$ is provided in Table 4.9 for the two-class problem. For this α value, the algorithm converged in 71 iterations. The classification accuracy was 99.95%, and the total number of features retained is 67.

Table 4.9 : Two-class confusion matrix for random mutation hill climbing with $\alpha = 0.8$.

	Classified as Infant	**Classified as Adult**
True Infant	1319	0
True Adult	1	500

The confusion matrix for the probability of error weighting of $\alpha = 1.0$ is provided in Table 4.10 for the two-class problem. For this α value, the algorithm converged in only 26 iterations. The classification accuracy was 99.95%, and the total number of features retained is 120.

Table 4.10 : Two-class confusion matrix for random mutation hill climbing with $\alpha = 1.0$.

	Classified as Infant	**Classified as Adult**
True Infant	1319	0
True Adult	1	500

Figure 4.10 summarizes the resultant performance of the classification system versus the weighting parameter, α. Notice that even for $\alpha = 0.2$, the performance of the system is still better than 98% correct, while the number of features used has fallen by a factor of five. Clearly, by adding in our weighting factor to the fitness function, the random mutation algorithm is able to more optimally select the features with very little loss in system performance.

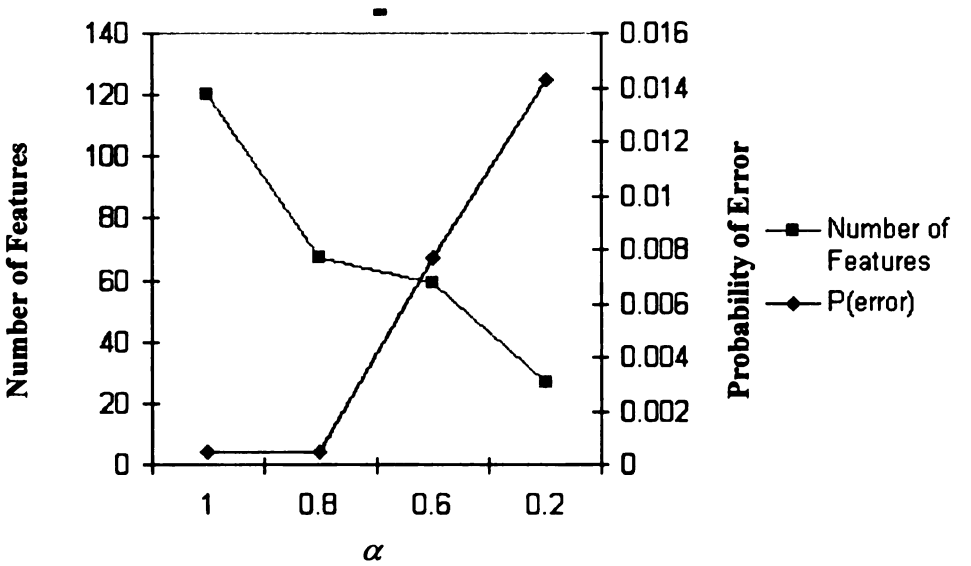


Figure 4.10: Plots of number of features and probability of error versus weighting for random mutation hill climbing.

Figure 4.11 shows the resultant improvement in convergence time of the system by allowing the number of features mutated, for each iteration, to be cooled over time. The improvement in convergence time is nearly a factor of 10 for a system that initially randomly selects 8 features per iteration, and then eventually cools to one feature per iteration. This is in stark contrast to an algorithm such as floating forward sequential search where the execution time dramatically increases with the number of features selected during each iteration, and all combinations of these must be processed, which results in a combinatorial growth in the processing time.

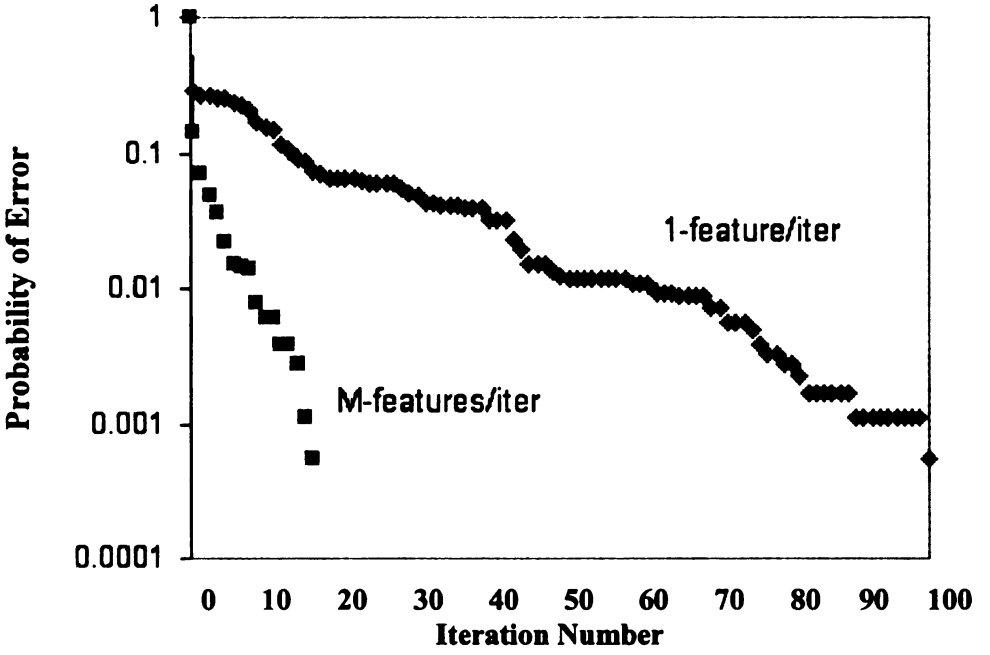


Figure 4.11: Comparison of convergence rate when cooling is added to random mutation hill climbing.

We also perform the random mutation hill climbing on the 4-class problem for the $\alpha = 0.2$ and the $\alpha = 0.8$ cases. The results in terms of the confusion matrix and the actual features selected are provided below. The confusion matrix for the probability of error weighting of $\alpha = 0.2$ is provided in Table 4.11. For this case, the algorithm failed to converge after 100 iterations. The classification accuracy is 96.87%, and the total number of features remaining is 41. Likewise, the confusion matrix for the probability of error weighting of $\alpha = 0.8$ is provided in Table 4.12. Again, the algorithm failed to converge after 100 iterations. The classification accuracy is 98.97%, and the total number of features remaining is 104.

Table 4.11 : Four-class confusion matrix for random mutation hill climbing ($\alpha = 0.2$).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1322	0	0	0
True Child	42	259	5	0
True Adult	15	4	455	0
True Empty	0	1	0	35

Table 4.12 : Four-class confusion matrix for random mutation hill climbing ($\alpha = 0.8$).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1322	0	0	0
True Child	17	289	0	0
True Adult	4	0	470	0
True Empty	0	1	0	35

B Forward Sequential Search

For the forward sequential selection we fix l and r to be both equal to 1. The results for the forward sequential search in terms of the confusion matrix and the actual features selected are provided below. The confusion matrix for the two-class problem is provided in Table 4.13. The classification accuracy is 99.56%, and the average total number of features remaining is 9.

Table 4.13 : Two-class confusion matrix for forward sequential search.

	Classified as Infant	**Classified as Adult**
True Infant	1326	2
True Adult	6	485

We also test the forward sequential search for the 4-class problem, and the confusion matrix is provided below. The confusion matrix for 4-class problem provided is in Table 4.14. The classification accuracy is 98.2%, and the total number of features remaining is 13.

Table 4.14 : Four-class confusion matrix for forward sequential search.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1325	2	1	0
True Child	25	278	6	1
True Adult	2	1	489	0
True Empty	0	0	1	35

4.4.2 Results for Filter Methods

For the filter methods we will provide the results for the Man-Whitney feature selector both with and without correlation post-processing. We will also provide the results for mutual information-based feature selection. As for the wrapper-based

methods, we test the feature selection algorithms for the two-class and the four-class problems. Recall the testing is performed on 10 iterations of the 50/50 cross-validation. Also, the incoming feature vector contains 1081 elements for the 45^{th} order Legendre moments.

A Mann-Whitney

For the Mann-Whitney tests we performed the testing both with and without correlation post-processing. We initially perform the Mann-Whitney testing using the correlation post-processing function, since this is our preferred method of operation. For the testing with correlation post–processing, we use Mann-Whitney as a ranking tool, and the correlation post-processing determines the final number of features. Then as a comparison, we generate the same number of features without the correlation post-processing by simply ranking and thresholding the Mann-Whitney results to the appropriate matching number of features. The confusion matrices, corresponding to the various levels of Mann-Whitney thresholding, are provided.

When we perform the correlation post-processing on all the 1081 features, after ranking the features with Mann-Whitney. The average number of remaining uncorrelated features is 81 for the two-class problem, and the classification accuracy is 99.4%. The confusion matrix is provided in Table 4.15.

Table 4.15 : Two-class confusion matrix using Mann-Whitney (with correlation post-processing performed on all the 1081 features, we retain 81 features).

	Classified as Infant	**Classified as Adult**
True Infant	1328	0
True Adult	9	482

When we use the Mann-Whitney to threshold the features, and we retain only the features with the highest 200 Z-statistics scores, and then perform the correlation post-processing, the average number of remaining features is 16 for the two-class problem. The classification accuracy is 99.0%, and the confusion matrix is provided in Table 4.16.

Table 4.16 : Two-class confusion matrix using Mann-Whitney (with correlation post-processing performed on only the top 200 features, we retain 16 features).

	Classified as Infant	**Classified as Adult**
True Infant	1324	5
True Adult	12	479

When we use the Mann-Whitney as a threshold, and we retain only the features with the highest 120 Z-statistics scores, and then perform the correlation post-processing, the average number of remaining features is only 6 for the two-class problem. The classification accuracy has reduced to 98.6%, and the confusion matrix is provided in Table 4.17.

Table 4.17 : Two-class confusion matrix for Mann-Whitney (With correlation post-processing performed on onlythe top 120 features, we retain 6 features).

	Classified as Infant	**Classified as Adult**
True Infant	1318	11
True Adult	14	477

The summary of the number of features retained for various Mann-Whitney thresholds followed by correlation post-processing for the two-class problem is summarized in Table 4.18. To compare the effectiveness of the correlation post-processing, we now will retain only this same number of resultant features (81, 16, and 6), but without performing correlation post-processing, and compare the resultant classification performance.

Table 4.18: Summary of number of features retained after using Mann-Whitney and correlation post-processing.

Number of Features Retained from Sorted Mann-Whitney	**Resultant Number of Features After Correlation Processing**
1085	81
200	16
120	6

When we use only the Mann-Whitney as a threshold, and we retain only the features with the highest 81 Z-statistics scores, then the classification accuracy is 99.9%. The confusion matrix for the two-class problem is provided in Table 4.19.

Table 4.19 : Two-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 81 features).

	Classified as Infant	**Classified as Adult**
True Infant	1328	1
True Adult	0	491

When we use only the Mann-Whitney as a threshold, and we retain only the features with the highest 16 Z-statistics scores, then the classification accuracy is 99.3%. The confusion matrix for the two-class problem is provided in Table 4.20.

Table 4.20 : Two-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 16 features).

	Classified as Infant	**Classified as Adult**
True Infant	1323	6
True Adult	7	484

Finally, when we use only the Mann-Whitney as a threshold, and we retain only the features with the highest 6 Z-statistics scores, then the classification accuracy is 96.9%. The confusion matrix for the two-class problem is provided in Table 4.21.

Table 4.21 : Two-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 6 features).

	Classified as Infant	**Classified as Adult**
True Infant	1310	19
True Adult	37	454

The summarization of the classification results for the Mann-Whitney feature selection algorithm, with and without correlation post-processing, is provided in Table 4.22. It is interesting to note that, in all but the most aggressive feature reduction, the use of correlation post-processing has no apparent effect, and may even reduce the classification accuracy slightly. For the case where only 6 features are retained, however, we do see there is an advantage to performing the correlation post-processing.

Table 4.22: Summary of classification accuracy using Mann-Whitney with and without correlation post-processing.

Number of Features	Classification Accuracy Using only Mann-Whitney	Classification Accuracy Using Mann-Whitney and Correlation Post- Processing
1081	99.98%	Not applicable
81	99.9%	99.4%
16	99.3%	99.0%
6	96.9%	98.6%

We will now show the results for the Mann-Whitney feature selection for the four-class problem. When we perform the correlation post-processing on all the 1081 features, after ranking the features with Mann-Whitney, the average number of remaining features for the four-class problem is 106. The classification accuracy is 98.5 %, and the confusion matrix is provided in Table 4.23.

Table 4.23 : Four-class confusion matrix using Mann-Whitney (with correlation post-processing performed on all the 1081 features, we retain 106).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1327	1	0	0
True Child	21	286	3	0
True Adult	7	0	484	0
True Empty	0	1	1	35

When we use the Mann-Whitney as a threshold, and we retain only the features with the highest 200 Z-statistics scores, and then perform the correlation post-processing, the average number of remaining features for the four-class problem is 43. The classification accuracy is 98.0%, and the confusion matrix is provided in Table 4.24.

Table 4.24 : Four-class confusion matrix using Mann-Whitney (with correlation post-processing performed on only the top 200 features, we retain 43).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1326	2	1	0
True Child	27	277	6	0
True Adult	6	0	485	0
True Empty	0	0	1	35

When we use the Mann-Whitney as a threshold, and we retain only the features with the highest 120 Z-statistics scores, and then perform the correlation post-processing, the average number of remaining features for the four-class problem

is only 31. The classification accuracy has reduced to 97.4%, and the confusion matrix is provided in Table 4.25.

Table 4.25 : Four-class confusion matrix using Mann-Whitney (with correlation post-processing performed on only the top 120 features, we retain 31).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1323	4	1	0
True Child	31	270	9	0
True Adult	8	0	482	0
True Empty	0	0	1	35

The summary of the number of features used for the four-class problem, when the correlation testing is performed is summarized in Table 4.26. To compare the effectiveness of the correlation post-processing, we will now retain only this same number of resultant features (106, 43, and 31), but without performing correlation post-processing, and compare the resultant classification performance.

Table 4.26: Summary of number of features retained for the four-class problem after using Mann-Whitney and correlation post-processing.

Number of Features Retained from Sorted Mann-Whitney	**Resultant Number of Features After Correlation Processing**
1085	106
200	43
120	31

When we use only the Mann-Whitney as a threshold for the four-class problem, and we retain only the features with the highest 106 Z-statistics scores, then the classification accuracy is 98.7%. The confusion matrix is provided in Table 4.27.

Table 4.27 : Four-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 106 features).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1327	1	0	0
True Child	22	287	2	0
True Adult	1	0	490	0
True Empty	0	0	1	35

When we use only the Mann-Whitney as a threshold for the four-class problem, and we retain only the features with the highest 43 Z-statistics scores, then the classification accuracy is 97.8%. The confusion matrix is provided in Table 4.28.

Table 4.28 : Four-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 43 features).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1325	2	1	0
True Child	31	273	6	0
True Adult	3	0	488	0
True Empty	0	0	1	35

Lastly, when we use only the Mann-Whitney as a threshold for the four-class problem, and we retain only the features with the highest 31 Z-statistics scores, then the classification accuracy is 97.5%. The confusion matrix is provided in Table 4.29.

Table 4.29 : Four-class confusion matrix using Mann-Whitney (no correlation post-processing, but we retain only the top 31 features).

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1322	4	2	0
True Child	35	267	8	0
True Adult	4	1	486	0
True Empty	0	0	1	35

The summarization of the classification results for the four-class problem using the Mann-Whitney feature selection algorithm with and without correlation post-processing is provided in Table 4.30. In the four-class case, there appears to be no advantage to using correlation post-processing, while for the two-class problem, there only appeared to be an advantage when we aggressively reduced the number of features. Thus it appears that correlation between features is not a significant problem for our airbag suppression application when using the 45^{th} order Legendre moments.

Table 4.30: Summary of four-class classification accuracy using Mann-Whitney with and without correlation post-processing.

Number of Features	Classification Accuracy Using Only Mann-Whitney	Classification Accuracy Using Mann-Whitney and Correlation Post- Processing
1081	98.93%	Not applicable
106	98.73%	98.48%
43	97.98%	98.01%
31	97.47%	97.43%

B Mutual Information

For the mutual information tests, we performed the testing with a fixed bin count for the histogram generation. We tested the mutual information-based feature selection for the two-class problem. To compute the mutual information, we used 100 bins for the generation of the joint probability density functions. The features were ordered using the mutual information, and then, to compare them with the Mann-Whitney results, we performed cross-validation using both correlation post-processed on the entire feature vector, and also on the top 200, and the top 120. The resultant numbers of features retained were the same lengths as those for the Mann-Whitney tests, namely 81, 16, and 6 respectively. The results are provided in Table 4.31 through Table 4.33. As we can see, the resultant probabilities of correct classification were 99.51%, 92.7%, and 82.82%, respectively. Clearly, these are considerably worse than any of the previous results for the other feature selection

algorithms. Therefore, we will not consider using mutual information for the four-class problem, since it is more difficult than the two-class problem.

Table 4.31: Two-class confusion matrix for mutual information using the top 81 features (start with 1081 and then perform correlation post-processing).

	Classified as Infant	**Classified as Adult**
True Infant	13299	0
True Adult	9	482

Table 4.32: Two-class confusion matrix for mutual information using the top 16 features (start with top 200 features and then perform correlation post-processing).

	Classified as Infant	**Classified as Adult**
True Infant	1300	29
True Adult	104	387

Table 4.33: Two-class confusion matrix for mutual information using the top 6 features (start with top 120 features and then perform correlation post-processing).

	Classified as Infant	**Classified as Adult**
True Infant	123	91
True Adult	221	270

4.5 Summary

The summary of the performance results for all of the feature selection methods is provided in Table 4.34. Note since the mutual information method performed so poorly on the two-class problem, we did not test the algorithm on the four-class problem, so the results are marked not available (N/A). Overall, we have found very little performance difference between the wrapper and the filter methods for our application. We believe this is because in our dataset, it is difficult to rank the features based on their discrimination ability and correlation, as shown in Figure 4.3 and Figure 4.4. Most of the datasets used in the literature for feature selection, are contrived to provide features that are clearly discriminating or clearly correlated with other features.

Overall, the forward sequential search algorithm performed the best in terms of combined classification accuracy and number of features retained with 99.6% accuracy while using only 9 features for the two-class problem. The next best for the two-class problem is the Mann-Whitney, thresholded at 200, which achieved 99.0% classification accuracy while using only 16 features. Another interesting observation is that the Mann-Whitney thresholded at 120 achieved 98.6% classification accuracy while using 6 features. This is the same classification accuracy achieved by random mutation with 27 features.

Table 4.34: Summary of classification accuracy and the number of features retained for all the feature selection methods.

Feature Selection Method	Classification Accuracy		# of Features Retained	
	Two-class	Four-class	Two-class	Four-class
Random Mutation Hill Climbing ($\alpha = 0.2$)	98.6%	96.9%	27	41
Random Mutation Hill Climbing ($\alpha = 0.8$)	99.95%	98.97%	67	104
Forward Sequential Search	99.6%	98.24	9	13
Mann-Whitney w/ Correl. (all 1081)	99.4%	98.5%	81	106
Mann-Whitney w/ Correl. (retain top 200)	99.0%	98.0%	16	43
Mann-Whitney w/ Correl. (retain top 120)	98.6%	97.4%	6	31
Mutual Information w/ Correl. (all 1081)	99.5%	N/A	81	N/A
Mutual Information w/ Correl. (retain top 200)	92.7%	N/A	16	N/A
Mutual Information w/ Correl. (retain top 200)	82.8%	N/A	6	N/A

The worst classification performance was achieved with mutual information. Its relatively poor performance is particularly obvious when only 6 features are retained, and mutual information achieved only 82.8% accuracy. The Mann-Whitney outperformed mutual information by over 10%. A key strength of the Mann-Whitney is that there are no critical free parameters that need to be empirically set. In the random mutation, and in the mutual information there are critical parameters that greatly affect the performance.

One of the key issues when choosing a feature selection method is the required processing time. A summary of the processing time required for the Mann-Whitney filter method, the random mutation method, and FSS method is provided in Table 4.35. As can be seen, there is a dramatic variation in the processing time, with the FSS algorithm requiring nearly one day of processing per feature selected, for the two-class problem. These results were computed running MATLAB on a 2 GHz AMD Pentium process with 256 GBytes of RAM. These results are for a single run of the 10 runs in the 50/50 cross validation feature selection processing.

Table 4.35: Total processing time for the three best feature selection methods.

Method	Time
Mann-Whitney w/ Correlation	< 10 minutes
Random Mutation Hill Climbing	3.25 hours
Forward Sequential Search	~ 1 day per output feature (2-class) ~ 2 day per output feature (4-class)

For our application, where we have a large number of features and a relatively large number of training and test samples available, the use of the filter methods can allow us to rapidly experiment with varying the parameters in the segmentation and feature extraction algorithms. This allows us to optimize the overall performance of the system. Since there is no appreciable reduction in the classification performance, the use of the wrapper methods appears unnecessary given its considerable increase in processing time.

Chapter 5.
Occupant Classification

The occupant classifier takes as input the features generated by the feature extraction processing. Its output is an estimate of the pattern class that most closely resembles the input imaged occupant. It bases this estimate on training image data, which was collected from the vehicles for which the system will operate. Recall there are two possible classification problems in our application: (i) a 2-class problem that classifies infants versus all adults (ignoring all other classes), and (ii) a more complicated 4-class problem that classifies the occupant into:

1) Infant,
2) Child,
3) Adult, and
4) Empty seat.

In the two-class implementation, all of the child occupants older than one year-old are protected either by the airbag being suppressed (if the occupant is classified as an infant), or they will be protected by the tracking system if they are classified as an adult. This chapter will review a subset of the available classification methods and will summarize the reasons that led to our decision of which classifier to use. We will also define a post-processing stage, called contextual processing, which provides a history-based smoothing function to the classifier to reduce the number of erroneous classifications.

5.1 Classification Methods

Figure 5.1 provides a hierarchy of the various classification methods available to a pattern recognition practitioner [27]. The top level of the hierarchy separates various methods based on the completeness of the "knowledge" of the problem. Complete knowledge means the designer has access to the *a priori* probability distributions of the classes, as well as the functional definition of the class-conditional probability distributions. Since we have no way of knowing the *a priori* probabilities of the occupant types, nor the true knowledge of the class-conditional densities, we must resort to *incomplete* reasoning.

Within the *incomplete* reasoning branch, there is the concept of supervised versus unsupervised learning. Since for this application we have collected and labeled the occupant type for our training data, we do not need to deal with unsupervised learning. Below we will describe methods from each of the supervised learning branches, namely a parametric approach and two non-parametric approaches. There are clearly a large number of candidate classifiers that can be used in supervised learning, however, there is no evidence that there is one method that is superior to any other method for every application domain [45][46].

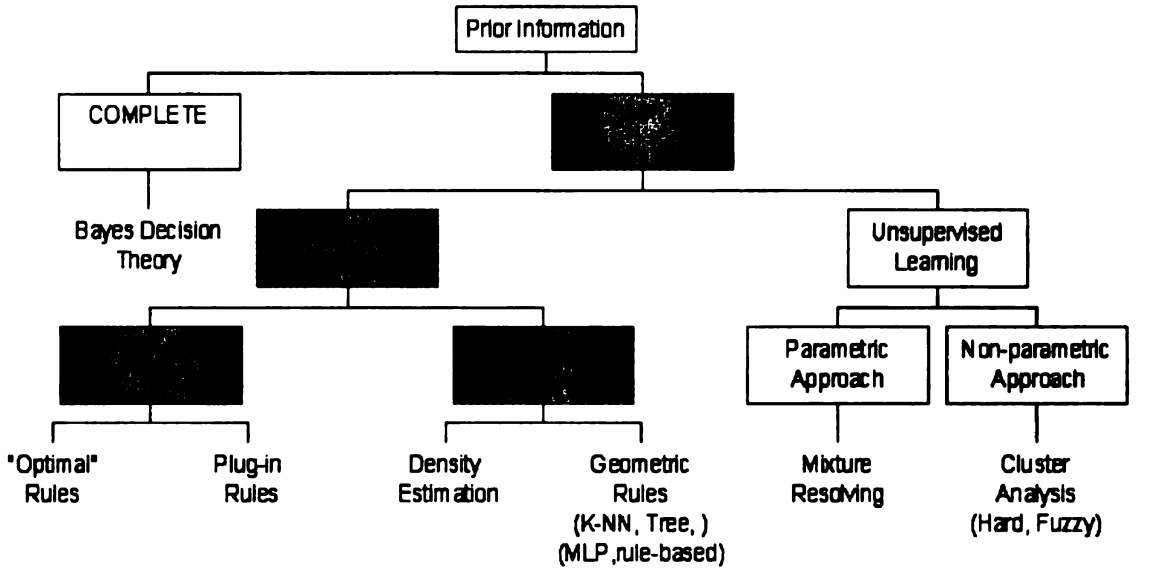


Figure 5.1: Hierarchy of the classification methods in statistical pattern recognition [28].

The primary non-parametric approach we will use is based on the nearest neighbor rule. We will propose a variant of the k-nearest neighbor (k-NN) rule that uses the aggregate distance to subsets of k nearest neighbors rather than a simple voting scheme. There are three primary reasons for choosing the k-nearest neighbor rule classifier family over other non-parametric methods, such as artificial neural networks, namely their ease of:

1) implementation,

2) training, and

3) explainability of the classification result.

The first reason was a key factor for our decision. Recall our ultimate objective is a system that can meet specific real-time requirements. The k-NN rule is a highly optimal algorithm to implement in a digital signal processor (DSP). A DSP device is relatively unique in that it can perform a multiply and an accumulate operation

(MAC) in a single clock cycle. Additionally, the DSP architectures typically have relatively deep instruction pipelines (5-11 instructions deep), which also enhances their efficiency in repetitive tasks. The k-NN rule can be implemented as a series of vector inner products that can be very efficiently implemented in these devices. This is in stark contrast to other decision rules, such as decision trees. Recall that trees use nested if-then-else statements. These are very inefficient to implement in a DSP due to the constant branching, which flushes the instruction pipeline, and the lack of MAC operations, which reduces the effectiveness of their super-scalar architectures. This third reason is also of great importance in a safety critical product, where it is desirable to fully understand, explain, and repair any of the failure modes of the system.

5.1.1 Bayes Classifier

The Bayes classifier is the optimal classification rule when all of the information is available for the decision-making, namely the class priors $P(\omega_i)$, and the class-conditional densities $p(\mathbf{x} \mid \omega_i)$. Recall Bayes rule computes the *a posteriori* probability for each of the M classes [45]:

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}, \tag{5.1}$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{M} p(\mathbf{x} \mid \omega_i)P(\omega_i). \tag{5.2}$$

A test sample is assigned to the class with the maximum *a posteriori* probability, $P(\omega_i \mid \mathbf{x})$. Since we do not know the class-conditional densities and the class prior probabilities for the airbag suppression application, the best we can do is to apply the standard plug-in rules, where the priors are not considered. If we assume the data to be normally distributed then the class-conditional density function has the form [45]:

$$p(\mathbf{x} \mid \omega_i) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_i\right|^{1/2}} \cdot \exp\left[-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_i\right)^T \cdot \boldsymbol{\Sigma}_i^{-1} \cdot \left(\mathbf{x}-\boldsymbol{\mu}_i\right)\right]. \qquad (5.3)$$

We now define the log likelihood, $g_i(\mathbf{x}) = \ln(p(\mathbf{x} \mid \omega_i))$, as the discriminant function, and it is computed by [45]:

$$g_i(\mathbf{x}) = -\frac{1}{2}\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_i\right)^T \cdot \hat{\boldsymbol{\Sigma}}_i^{-1} \cdot \left(\mathbf{x}-\hat{\boldsymbol{\mu}}_i\right) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln\left|\hat{\boldsymbol{\Sigma}}_i\right|, \qquad (5.4)$$

where $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\Sigma}}_i$ are the maximum likelihood estimates of $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, respectively. We then assign the test sample to the class with the maximum discriminant function. Note the inherent quadratic form of the classification boundary for this decision rule due to the quadratic term: $\left(\mathbf{x}-\hat{\boldsymbol{\mu}}_i\right)^T \cdot \hat{\boldsymbol{\Sigma}}_i^{-1} \cdot \left(\mathbf{x}-\hat{\boldsymbol{\mu}}_i\right)$.

Note that while the Bayes rule is mathematically simple, it still requires some care in the computation. Since we may be dealing with very large feature spaces, it is

possible that the matrix $\hat{\Sigma}_i$ is ill-conditioned, and the inverse $\hat{\Sigma}_i^{-1}$ cannot be calculated. In this case the pseudo-inverse must be used, which is defined as:

$$\textbf{psuedo}\ \hat{\Sigma}_i^{-1} = (\hat{\Sigma}_i^T \hat{\Sigma}_i)^{-1} \cdot \hat{\Sigma}_i^T . \qquad (5.5)$$

If we are unable to compute the inverse of $\hat{\Sigma}_i$, then the determinant will most likely be zero, in which case the term $\left|\hat{\Sigma}_i\right|$ will not be computable. We must then recall that the determinant of a matrix is the product of its eigenvalues. We can approximate the determinant by only computing the product of the non-zero eigenvalues of $\hat{\Sigma}_i$. With these two modifications, the Bayes rule can be used for nearly all applications.

5.1.2 Nearest Neighbor Classifier

We will investigate the use of two variants of the nearest-neighbor classification rule: (i) the traditional k-nearest neighbor classifier, and (ii) a new average distance-based k-nearest neighbor classifier. The traditional k-nearest neighbor classifier computes the distance between the input sample and each of the stored training samples, and then sorts these distances in ascending order. The class that is in the majority among the top-k neighbors is chosen as the correct class [45]. The distance-based k-nearest neighbor classifier computes the average distance of the test sample to the k-nearest training samples in each class as shown in Figure 5.2. For

example, it computes the mean of the top k infant training samples, the top k adult samples, etc. The final decision is to choose the class with the lowest average distance to its k-nearest neighbors.

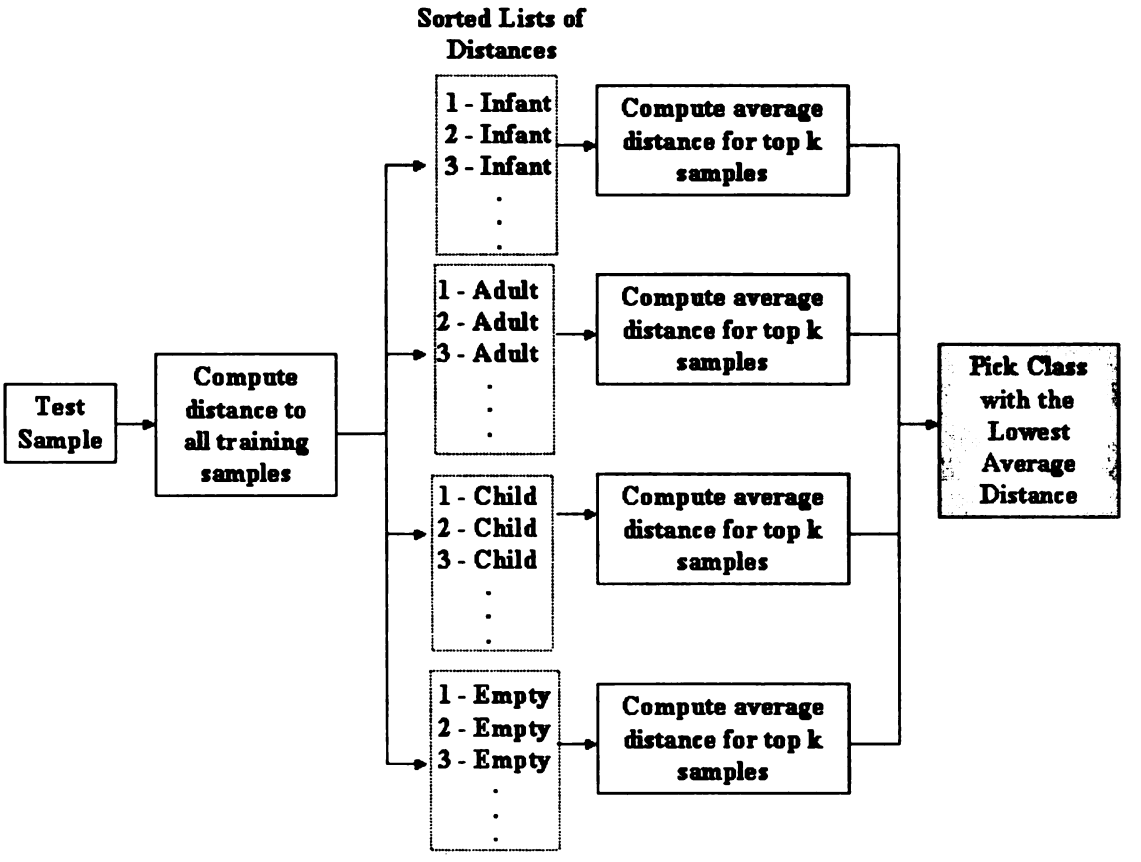


Figure 5.2: The distance-based k-nearest neighbor classifier.

This distance-based k-nearest neighbor classifier is related to the nearest mean classifier, where we compute a pruned mean for each class based on the sorted distances to all of the training samples. The strength of this classifier is derived from its use of local sample mean information rather than the global sample mean information. It is similar to the local linear embedding (LLE) methods for geodesic

distance estimation, where computing the total distance between the samples in a local piecewise manner allows distances across complex manifolds to be computed.

5.1.3 Support Vector Machines

A Support Vector Machine (SVM) is a method for classifying two class problems. It takes as input labeled data from the two classes, and outputs a model for classifying test data into one of the two classes. SVM can generate linear and non-linear models [1]. For the linear model, the algorithm finds a separating hyper-plane that maximizes the margin of separation between the two classes [49][50]. Each input point is assigned a weight, and the points with non-zero weights are called support vectors. Most of the input points have zero weights, and hence the number of support vectors is much less than the total number of training points. The separating hyper-plane is defined as a weighted sum of support vectors. The equation of the hyper-plane is given by [49][50][51]:

$$\mathbf{x}^T\mathbf{w} + b = 0, \tag{5.6}$$

where $\mathbf{x}$ is the p-dimensional data vector, and

$$\mathbf{w} = \sum_{i=1}^{s} (\alpha_i \cdot y_i)\mathbf{x}_i, \tag{5.7}$$

where s is the number of support vectors, y_i is the known class for the data point $\mathbf{x}_i$, and α_i are the support vector coefficients that maximize the margin of separation

between the two classes. A test sample can be classified using the decision rule [49][50][51]:

$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{x}^T\mathbf{w} + b) = \begin{Bmatrix} > 0 \text{ if class 1} \\ < 0 \text{ if class 2} \end{Bmatrix}. \qquad (5.8)$$

For the nonlinear model, the input data points are mapped to higher dimensional feature space using mapping function Φ, and then linear classification algorithm is applied in this feature space. Notice that in Equation (5.8), the training data only appears in the form of the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$. For this nonlinear model, the product $\mathbf{x}_i \cdot \mathbf{x}_j$ is replaced in the higher dimensional space by a product of kernel functions operating on $\mathbf{x}_i$, according to [49][50][51]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j). \qquad (5.9)$$

By computing the dot product directly using a kernel function, we avoid the mapping $\Phi(\mathbf{x})$, which is desirable since the dimension of this new space may be very large, which can make the computation of $\Phi(\mathbf{x})$ difficult. Note, however, this means that $\mathbf{w}$ cannot be pre-computed, which means the decision function will be replaced by [50][51]:

$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\sum_{i=1}^{s}(\alpha_i \cdot y_i)K(\mathbf{x}_i, \mathbf{x}) + b) = \begin{Bmatrix} > 0 \text{ if class 1} \\ < 0 \text{ if class 2} \end{Bmatrix}. \qquad (5.10)$$

Now, for every incoming test vector, the kernel function for each support vector needs to be recomputed. There are many possibilities for the kernel operator in the literature [49][50]. One of the more common kernels is the Gaussian radial basis function, which is what we will use. It is defined as [50][51]:

$$K(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|}{2\sigma^2}\right), \qquad (5.11)$$

where the free parameter σ defines the spatial extent of the kernel.

5.2 Contextual Processing

The image classification defined throughout this chapter, has been applied to a single image. Clearly, for our airbag suppression application, during the normal operation of the vehicle, the system may collect hundreds of images over time. During this time, the system may experience a variety of lighting conditions and passenger seating positions that may make the correct classification of the occupant difficult at any given time. However, there may also have been many images collected in more favorable conditions that will provide reliable classification results. Figure 5.3 shows the sequence of an adult seated normally, then leaning forward to tie his shoes, and finally returning to a normally seated position. Notice that in the leaning forward position, the adult will look like an infant in a RFIS. Without classification history, the algorithm would classify the occupant as an infant, while with some contextual processing, based on the history of previous classifications, the system may continue to correctly classify the occupant as an adult.

There will also be other conditions where it is impossible to properly classify the occupant, for example, when an occupant is lifting a sweater over her head. In cases like this, it is advisable to have a 'reject' option, when the object does not properly match any of the classes [21]. In such a situation, the classifier provides no useful information regarding the true object pattern class, and the system should be able to declare 'ignorance' regarding the current classification. Likewise, when a system begins operation, there is no knowledge of the classification, and the system should declare 'ignorance'. Other research efforts in image sequence analysis have also shown the importance of developing a context of the incoming classifications based on the previous classifications [164].

The ability of a classification system to work in environments of uncertainty and ignorance is integral to the field of evidential reasoning [160]. Specifically, evidential reasoning "deals with information that is to be uncertain, imprecise, and occasionally inaccurate", and it provides a framework for managing beliefs in this environment [165]. It has been stated, "belief change does not aim to produce the 'best' sequence of belief states but focuses on producing the 'best' current belief state" [161]. This is similar to Kalman filtering in time series filtering, where the goal is not to correct all past observations, but rather to derive the best estimate for the current state, based on the past observations. Since we are attempting to provide the best current classification from a stream of input classifications, the concepts of evidential reasoning should be useful for classification history processing.

In the field of evidential reasoning there are two mechanisms for managing the changes in the belief of a hypothesis, *belief revision* and *belief updating* [161].

Belief revision is the element of belief change that involves integration of new information on a static situation, while belief updating involves modifying beliefs about the world, when the state of the world is changing [161]. In our airbag suppression application, we are only interested in belief revision, since the occupant will not grow from a child into an adult during the duration of the drive. There are many methods for belief revision, such as Transferable Belief Model (TBM), Dempster-Shafer theory, etc. [160]. Most of these approaches are related to the Dempster-Shafer theory, so we will use Dempster-Shafer theory as the basis for our contextual processing approach.

5.2.1 Dempster-Shafer Theory

Dempster-Shafer theory assumes that there is a fixed and exhaustive set of mutually exclusive objects called the *environment,* defined as $\Theta = \{\theta_1, \theta_2, ..., \theta_N\}$ [160][162][165]. The environment is also sometimes referred to as the frame of discernment. The Dempster-Shafer theory is based on the aggregation of the atomic elements of the frame of discernment into subsets, and the complete collection of these subsets is called the power set, which is defined as:

$$P(\Theta) = \{\phi, \{A\}, \{B\}, \{C\}, \{A,B\}, \{A,C\}, \{B,C\}, \{A,B,C\}\}, \qquad (5.12)$$

where the frame of discernment is $\Theta = \{A, B, C\}$.

(a) (b)

Figure 5.3: Demonstration of need for history algorithm, (a) adult seated normally, and (b) adult leaning forward and appearing to system to be an infant in a RFIS.

The basic element of evidence in Dempster-Shafer is the mass or basic probability assignment, defined by m. It has the following two properties that make it similar to the Bayesian probability assignments [165][162]:

$$m(\phi) = 0 \quad \text{and} \quad \sum_{A \in P(\Theta)} m(A) = 1. \tag{5.13}$$

The key difference between Dempster-Shafer and Bayesian methods is that Dempster-Shafer does not force belief to be assigned *a priori* to all possible outcomes, but rather only to the subsets of outcomes for which evidence has directly been accumulated. Any belief that is not assigned to specific subsets $A \subset \Theta$ simply remains with the entire environment Θ, and this belief is referred to as the *ignorance* in the system [165]. For example, if $\Theta = \{A, B, C\}$, and the system is initially completely ignorant of the possible outcomes, then the Bayesian theory would assign each element probability 1/3. In Dempster-Shafer, however, no probability is assigned to the individual elements, but only the entire set such that $P(\Theta) = 1$.

Another key difference between these two theories is that the probability of an outcome is not a single value, but rather a range of values. The lower value is called the *belief* in the proposition, and the upper value is the *plausibility* in the proposition, thereby generating a range of confidence in a hypothesis: [*Belief, Plausibility*]. This concept of a range of probabilities is critical in a number of mathematical representations of evidential reasoning [160][162] [165]. Belief in a proposition is the total accumulation of evidence that directly supports that proposition, while plausibility in a proposition is the accumulation of evidence that does not directly refute that proposition. Mathematically, they are defined to be [165]:

$$Bel(X) = \sum_{A \subseteq X} m(A), \text{ and}$$

$$Pls(X) = 1 - Bel(\tilde{X}) = 1 - \sum_{A \subseteq \tilde{X}} m(A) \ . \qquad (5.14)$$

Additionally, ignorance in a particular proposition is defined to be [165]:

$$Igr(X) = Pls(X) - Bel(X). \qquad (5.15)$$

Since evidential reasoning is interested in accumulating evidence over time, it is important to define a mechanism for combining sequences of probability masses. In the Dempster-Shafer theory, this is accomplished through Dempster's rule of combination, which combines two probability masses m_1 and m_2 according to [165]:

$$m_1 \oplus m_2(Z) = \frac{\sum_{X \cap Y = Z} m_1(X) \cdot m_2(Y)}{1 - \kappa}, \qquad (5.16)$$

$$\text{where } \kappa = \sum_{X \cap Y = \phi} m_1(X) \cdot m_2(Y).$$

These definitions of belief and plausibility, as well as the definition of a mechanism for combining evidence, summarize the Dempster-Shafer theory. In the following section we will specifically show how the Dempster-Shafer framework can be used for classification history processing.

5.2.2 Contextual Processing Using Dempster-Shafer Theory

The continuous stream of classification outputs from the system over time can be framed as an evidential reasoning problem. The sequence of classification results is a stream of evidence being accumulated, regarding the true class of the occupant. The Dempster-Shafer framework suits this problem very well, where we have an environment of a fixed set of outcomes that we are constantly viewing over time. In our system, we have the following elements in our environment, $\Theta = \{\text{infant}, \text{child}, \text{adult}, \text{empty}\}$.

Clearly, this set meets the requirement of a Dempster-Shafer environment of mutual exclusivity. It also meets the exhaustive requirement if we consider the empty class to also include small objects on the seat. The processing flow for the contextual processing is provided in Figure 5.4, and each function is defined in more detail in the following subsections.

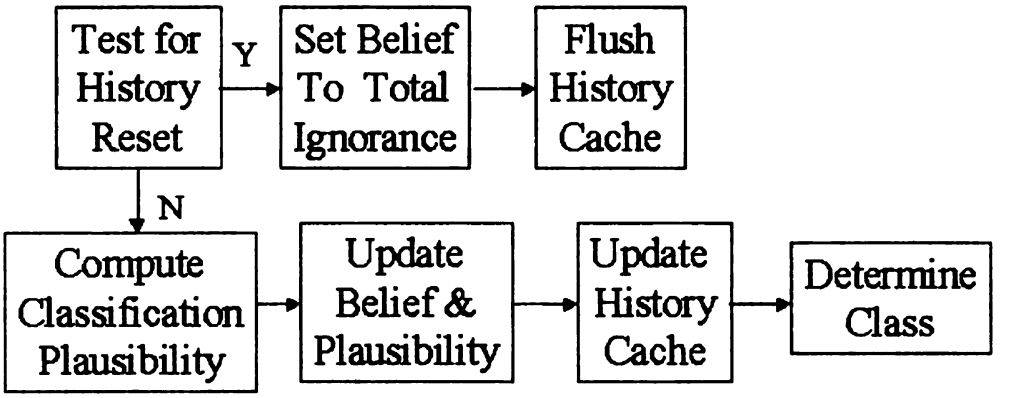


Figure 5.4: Contextual processing for the airbag suppression application.

A Test for History Reset

The process to test if the history function requires a reset is application dependent. It tests whether the current history must be reset to complete ignorance, or if the current stream of evidential inputs should continue to be integrated. For our airbag suppression application, the history must be reset if the vehicle passenger door has been opened, while the vehicle is stopped. If both of these conditions are true, then it is possible that the actual occupant has changed state. Consequently, the history of the current occupant must be reset to complete ignorance in anticipation of a new occupant entering the vehicle.

B Test Classification Plausibility

The next step of the processing is to test the plausibility of the incoming classification result, and adjust the incoming probability mass assignment

accordingly. This processing addresses the problem where the evidence to date has been of one class (adult), and now the evidence implies another class (infant). This processing stage verifies that the current classification is plausible, given the past historical classifications and any other available temporal information. There are two mechanisms by which the occupant classification can change over time: (i) an **assignable** mechanism and (ii) a **random** mechanism. We employ a two-stage process, one to address each of these sources of error in the classification.

In our airbag suppression application, assignable variations occur due to the motions of the occupant. Recall, for example, we showed in Figure 5.3 that when an adult occupant is leaning forward, he is easily confused with the infant class. This change in occupant class is due to the fact that the occupant moved to a position that makes correct classification difficult. This type of classification error that is the result of the movement of the occupant is considered assignable, since the outcome of the classification can be specifically assigned to the motion of the occupant.

The other type of classification error is the random error. The random errors are caused by the overlap amongst the classes in the decision space, and consequently there is a finite probability that the classifier will output the wrong classification at any given time. Randomly occurring variations in the incoming classification, clearly, cannot be valid transitions between the actual object classes (e.g., a child cannot transition into an adult during the duration of a vehicle drive). We address these errors by testing the current belief and plausibility in the incoming classification by computing the total change in belief that this new input would cause if it were integrated into the history, according to:

$$|\Delta Bel| = \sum_{P(\Theta)} \left| Bel_{current} - Bel_{history} \right|. \qquad (5.17)$$

If the value in Equation (5.26) exceeds a threshold, then we assume the system has experienced a transition that is too unlikely, and we do not integrate the current classification information. We must, however, allow for the fact that the history may have been wrong to this point due to irregular lighting, etc. Consequently, when the plausibility fails, we reset the current input to be complete ignorance. It is important to note that this is considerably different from simply discarding the current input. When the current classification is set to complete ignorance, it will then slowly degrade the overall belief in any particular classification, as these questionable results are integrated into the classifier history. At some point, the resultant belief in any particular class will be low enough that the incoming information is no longer considered contradictory, and it can be integrated into the history. The evidence for this new class can now begin to be accumulated, and its belief and plausibility will continue to grow if the classification outputs now remain consistent.

C Update Belief and Plausibility

The stage of processing for updating the belief and plausibility follows the definition of Dempster's rule of combination from equation (5.12). There is one change we have incorporated into our implementation that departs from the traditional theory. We found that if the confidence for each input pattern class is very

close to 1 for long periods of system operation, then the belief in that class will converge to 1.0. After the belief in a particular hypothesis has converged to 1.0, the system cannot properly integrate any contradictory information. The incoming contradictory information will be orthogonal to current belief, and will be ignored in the rule of combination. To fix this problem, we add some probability mass into the complete ignorance subset for every classifier input no matter how certain we are of that result. We then renormalize the sum of all masses to sum to one. At this point, we then perform the normal rule of combination. This is implemented according to:

$$m_{old}(\Theta) = \varepsilon \text{, and } m_{new}(B) = \frac{m_{old}(B)}{\sum_{A \in P(\Theta)} m_{old}(A)}, \quad \forall B \in P(\Theta), \qquad (5.18)$$

where we note that, due to the additional mass assigned to Θ, $\sum_{A \in P(\Theta)} m_{old}(A) \neq 1$, and $0 < \varepsilon < 1$ is the amount of mass empirically determined to be effective for preventing convergence. For our airbag suppression application we set $\varepsilon = 0.05$.

D Update History Cache

Once the newly arrived classification evidence is integrated into the current belief and plausibility values, the complete power set vector containing all of the beliefs and plausibilities are added into the history cache. This is a rolling buffer of the last N classifications. It provides an additional smoothing function. For our airbag suppression application, a buffer depth of 10 gives adequate performance. Recall the Belief and Plausibility form a confidence range for any given hypothesis A,

[*Bel(A), Pls(A)*]. To compute the final confidence in each of the atomic elements for the system we simply compute:

$$\begin{aligned} Conf(Infant) &= w_{Bel} \cdot Bel(Infant) + w_{Pls} \cdot Pls(Infant), \\ Conf(Child) &= w_{Bel} \cdot Bel(Child) + w_{Pls} \cdot Pls(Child), \\ Conf(Adult) &= w_{Bel} \cdot Bel(Adult) + w_{Pls} \cdot Pls(Adult), \\ Conf(Empty) &= w_{Bel} \cdot Bel(Empty) + w_{Pls} \cdot Pls(Empty), \end{aligned} \tag{5.19}$$

where w_{Bel} and w_{Pls} are the relative weighting between the Belief and the Plausibility, respectively, and $w_{Bel} + w_{Pls} = 1$. The setting of these values determines how conservatively the system computes the output confidence.

5.3 Results for Occupant Classification

The classification results are provided for the two-class and the four-class problems for the occupant suppression application. The dataset consists of the numbers of images defined in Table 2.3. We use two methods for verifying the performance of the classifiers: (i) 50/50 cross-validation testing using the training dataset, and (ii) testing on the independent test image database. The results for both of these test methods follow.

5.3.1 Results of Individual Classifiers

Recall the classification methods we compare are the Bayes, k-nearest neighbor, and support vector machine classifiers. The testing is performed using the

best set of features derived from the Mann-Whitney feature selection with the correlation post-processing.

A Bayes Classifier

The results for the Bayes classifier on the 50/50 cross validation database are provided in Table 5.1 and Table 5.3 for the two-class and four-class problems, respectively. The overall accuracy for the two-class problem using the cross validation test is 99.74%, and the overall accuracy for the four-class problem is 98.4% for this test.

Table 5.1: Two-class confusion matrix for the Bayes classifier on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Adult**
True Infant	1329	0
True Adult	4	487

Likewise, the results for the Bayes classifier on the independent test database are provided in Table 5.3 and Table 5.4 for the two-class and four-class problems, respectively. The overall accuracy for the two-class problem on the independent test dataset is 98.2%, and 88.4% for the four-class problem using this dataset.

Table 5.2: Two-class confusion matrix for the Bayes classifier on the independent test database.

	Classified as Infant	**Classified as Adult**
True Infant	1282	1
True Adult	25	106

Table 5.3: Four-class confusion matrix for the Bayes classifier on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1328	0	0	0
True Child	26	282	1	0
True Adult	1	0	490	0
True Empty	2	1	3	30

Table 5.4: Four-class confusion matrix for the Bayes classifier on the independent test database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1282	1	0	0
True Child	149	35	17	0
True Adult	14	0	117	0
True Empty	8	0	0	0

B Nearest Neighbor Classifiers

The results for the k-nearest neighbor classifier (with k=9) on the 50/50 cross validation database are provided in Table 5.5 and Table 5.6 for the two-class and four-class problems, respectively. The overall accuracy for the two-class problem for the cross validation test is 99.5%, and the overall accuracy for the four-class problem is 98.5% for this test.

Table 5.5: Two-class confusion matrix for k-nearest neighbor with k=9 on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Adult**
True Infant	1329	0
True Adult	9.3	481.7

Likewise, the results for the k-nearest neighbor classifier (with k=9) on the independent test database are provided in Table 5.7 and Table 5.8 for the two-class and four-class problems, respectively. The overall accuracy for the two-class problem for the independent dataset test is 96.7%, and the overall accuracy for the four-class problem is 88.2% for this test.

Table 5.6: Two-class confusion matrix for k-nearest neighbor with k=9 on the independent test database.

	Classified as Infant	**Classified as Adult**
True Infant	1270	13
True Adult	34	97

Table 5.7: Four-class confusion matrix for k-nearest neighbor with k=9 on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1327	1	0	0
True Child	21	286	3	0
True Adult	7	0	484	0
True Empty	0	1	1	35

Table 5.8: Four-class confusion matrix for k-nearest neighbor with k=9 on the independent test database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1279	1	3	0
True Child	144	41	16	0
True Adult	17	2	112	0
True Empty	5	3	0	0

C Support Vector Machines

The results for the support vector machine classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the 50/50 cross validation database are provided in Table 5.9 and Table 5.10 for the two-class and four-class problems, respectively. The software used was SVM-Light [52]. The overall accuracy for the two-class problem for the cross

validation test is 98.9%, and the overall accuracy for the four-class problem is 93.6% for this test.

Table 5.9: Two-class confusion matrix for the SVM classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Adult**
True Infant	1327	2
True Adult	18	473

Likewise, the results for the support vector machine classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the independent test database are provided in Table 5.11 and Table 5.12 for the two-class and four-class problems, respectively. The overall accuracy for the two-class problem for the independent dataset test is 96.8%, and the overall accuracy for the four-class problem is 87.8% for this test.

Table 5.10: Two-class confusion matrix for the SVM classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the independent test database.

	Classified as Infant	**Classified as Adult**
True Infant	1254	29
True Adult	16	115

Table 5.11: Four-class confusion matrix for the SVM classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the 50/50 cross-validation database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1322	4	2	0
True Child	53	249	9	0
True Adult	28	37	426	0
True Empty	0	5	1	30

Table 5.12: Four-class confusion matrix for the SVM classifier (using the RBF kernel with $\sigma^2 = 10.0$) on the independent test database.

	Classified as Infant	**Classified as Child**	**Classified as Adult**	**Classified as Empty**
True Infant	1281	0	1	0
True Child	128	39	34	0
True Adult	26	0	105	0
True Empty	3	3	2	0

The summaries of the classification accuracies for these three classification methods, using both the 50/50 cross-validation test and the independent dataset test, are provided in Table 5.13.

Table 5.13: Summary of two-class and four-class classification accuracies for the three classification methods.

	Two-class		Four-class	
Classifier Method	**50/50**	**Independent**	**50/50**	**Independent**
Bayes	99.7 %	98.2 %	98.4 %	88.4 %
k-nearest neighbor (k=9)	99.5 %	96.7 %	98.5 %	88.2 %
Support Vector Machine (RBF, σ^2 =10.0)	98.9 %	96.8 %	93.6 %	87.8 %.

5.3.2 Results of Contextual Processing

We have provided two examples of the results for the contextual processing based on evidential reasoning. The initial example sequence of images is shown in Figure 5.5. In this sequence, an adult is initially seated normally, slowly leans forward into a crouching position (where his appearance resembles that of a large infant seat), and then returns to a normally seated position. The sequence represents an image collected every 3 seconds for a period of roughly 1.5 minutes.

Figure 5.5: Sequence of images to demonstrate history processing.

The pattern class confidences generated by the classifier for each of the four classes is provided in Figure 5.6, and the resultant classification for each frame (taking the pattern class with the largest confidence) is provided in Figure 5.7. Note that the adult is classified as an infant during the portion of the sequence where he is leaning forward.

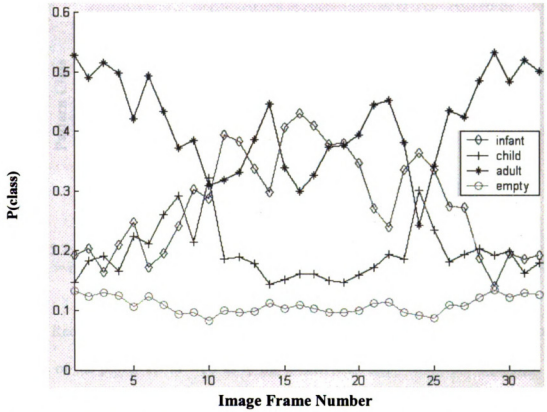


Figure 5.6: Input confidences with no history for each class for sequence in Figure 5.5.

Figure 5.8 shows the result resulting beliefs for the four atomic class set elements, after the Dempster-Shafer history processing. Notice the beliefs in the atomic element {adult} remains the preferred classification even during the brief sub-sequence where the {infant} set is more likely due to using the Dempster's rules of combination to integrate the sequence of incoming classifications results.

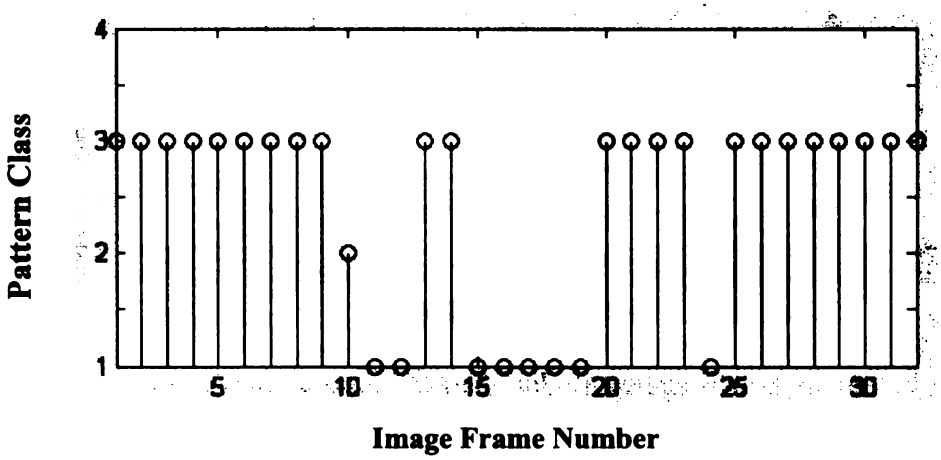


Figure 5.7: Original classifications with no history processing versus image frame (class 1 = infant, class 2 = child, class 3 = adult, and class 4 = empty).

Recall, the rules of combination distribute the probability mass to the various subsets of the power set as well as to the atomic elements. Since the difference in likelihoods between the infant class and the adult class is not great during the leaning subsequence, the probability mass is distributed into the {infant, adult} subset, as well as into the {adult} and {infant} atomic sets. The amount of mass distributed to the atomic sets is not adequate to cause the final confidence in the {infant} set to surpass that for the {adult} set.

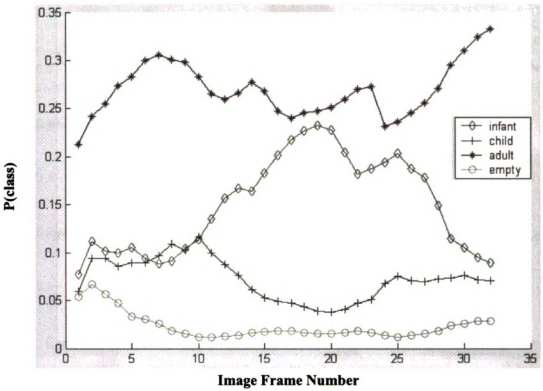


Figure 5.8: Dempster-Shafer atomic element beliefs versus image frame.

Figure 5.9 shows the final confidence in the {adult} set versus the input beliefs for the infant and adult classes. Recall, this final confidence is the average of the beliefs and the plausibilities for the {adult} set for each of the frame times, where we have set the weights w_{Bel} and w_{Pls} are equal. Likewise, Figure 5.10 provides the final classification results for this image sequence after the Dempster-Shafer evidential reasoning processing. It is important to note that the history cache has been set to be only 1 deep for this processing, so the improvements in the classification accuracy are due solely to the Dempster-Shafer processing.

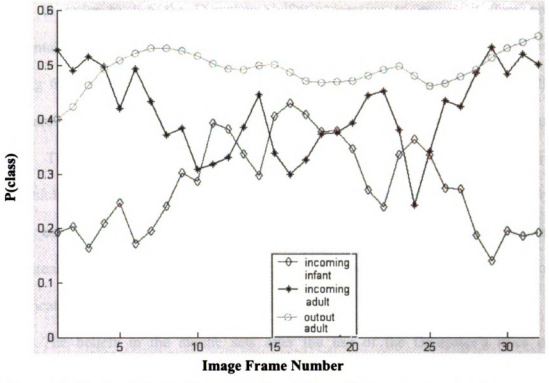


Figure 5.9: Final confidence for output class D-S confidence compared to input confidences.

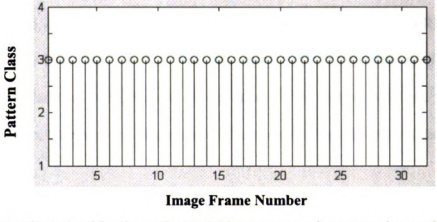


Figure 5.10: Final classifications after D-S history processing versus image frame (class 1 = infant, class 2 = child, class 3 = adult, and class 4 = empty).

In order to test the Dempster-Shafer processing to ensure it does not accidentally propagate mis-classifications, we modified the original sequence by moving frames 13-20 to the beginning of the sequence (see Figure 5.11). Recall, these are the frames where the adult is leaning forward, and mis-classified as an infant. This new scenario is designed to mimic the situation where the adult enters the vehicle and immediately proceeds to tie his shoes, and then moves to the normally seated condition. The raw input atomic beliefs from the image classification subsystem are provided in Figure 5.12, and the resultant sequence of output classifications are provided in Figure 5.13.

The beliefs in the atomic sets after the use of the Dempster's rules of combination for this sequence are provided in Figure 5.14. Notice that for frames 1 through 9, the belief in the {infant} set is the greatest since the incoming evidence suggests an infant. Note also that the belief for the {adult} set is not that much lower, since recall from Figure 5.12 that the confidences in the two classes are very close. This is because, even though the crouching adult appears somewhat like a RFIS, he is actually a little large for that class. For this sequence, the Dempster-Shafer theory initially mis-classifies the occupant as an infant, since that is what the evidence suggests. Note, however, that from frame 10 and on, the {adult} set has the greatest confidence.

Figure 5.11: Permuted sequence of images to demonstrate history processing when the initial classification is wrong.

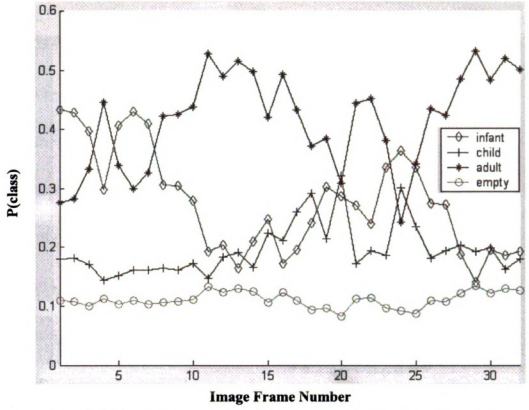


Figure 5.12: Input confidences with no history for each class for sequence in Figure 5.11 if we move frames 13-20 to the front of the sequence so the occupant initially looks like an infant.

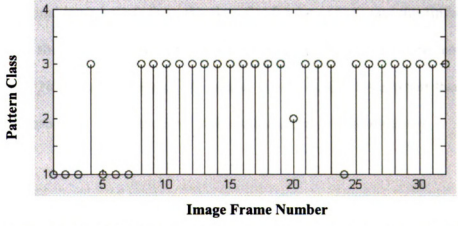


Figure 5.13: Original classifications of permuted image sequence with no history processing versus image frame (class 1 = infant, class 2 = child, class 3 = adult, and class 4 = empty).

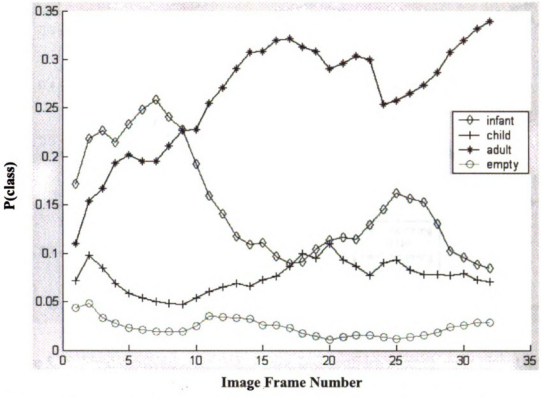


Figure 5.14: Dempster-Shafer atomic element beliefs versus image frame for permuted image sequence.

Figure 5.15 shows the output confidence for the {adult} set, and compares it with the original input beliefs from the classifier for the {adult} and the {infant} sets. Figure 5.16 shows the output classifications after the Dempster-Shafer contextual processing. Note the stability of the output classification, even when there are a couple of subsequent erroneous input classifications due to random errors.

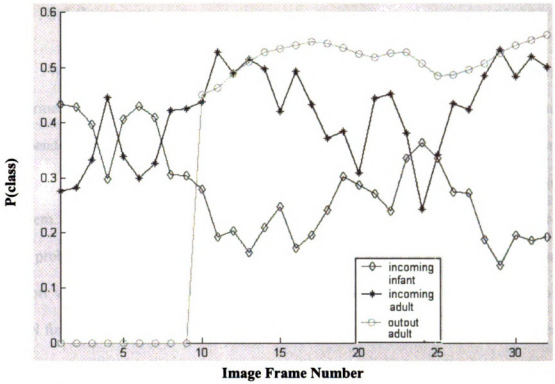


Figure 5.15: Final confidence for output class D-S confidence compared to input confidences for permuted sequence.

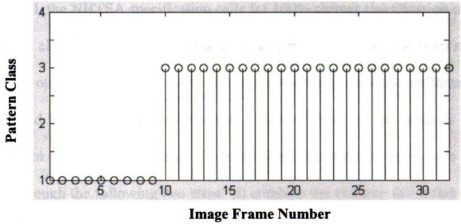


Figure 5.16: Final classifications after D-S history processing versus image frame for permuted sequence (class 1 = infant, class 2 = child, class 3 = adult, and class 4 = empty).

5.4 Summary

We see that, overall, the Bayes and the k-nearest neighbor classifiers performed similarly for the 50/50 cross-validation testing. When tested on the independent dataset, however, the Bayes classifier achieved 1.5% better classification accuracy for the two-class problem, while they performed equally for the four-class problem. Note both classifiers achieved only 88% correct classification for the four-class problem. The support vector machine performed worse in all of the tests. The support vector machine, however, has numerous tuning variables, and a variety of kernel functions, which if thoroughly experimented with, may have resulted in better performance. For applications such as our airbag suppression application, working with classifiers with minimal free parameters allowed us to concentrate on the many other difficult aspects of the problem, without extensive experimentation being required for this single subsystem.

Recall, the NHTSA specification calls for 100% correct classification on these test sets, and clearly, with the four-class problem approach, we are far from meeting that goal. Notice, that the areas with the greatest difficulty are the children-infant confusion and the empty seat classification. Both of these are not actually critical boundaries for the NHTSA testing. The overall performance of the system can be improved through the following two steps: (i) combine the children and infant classes (since they are both suppress conditions in the four-class approach), and (ii) ignore the empty seat class (assuming we will use another method for empty seat detection). We now see that the classification accuracy increases to 98.1%, which is approaching

the NHTSA requirement. For the two-class problem, we are also at 98.2% accuracy, which is also close to the specification.

While the performance of the Bayes and the k-NN are close in terms of classification accuracy, we decided to use the k-NN classifier for the implementation of the system. As we have mentioned earlier, the k-NN architecture is particularly well suited for the DSP hardware architecture. Also, by not performing the covariance matrix inversion, we are sure to maintain reasonable demands on the arithmetic precision of the operations. Lastly, we believe that the k-NN architecture has a significant advantage when fielding of the system. The system must be operational for 15 years, which means there will be many new models of child restraint produced during this timeframe. If some of these future models are considerably different in shape and appearance than any of the current models, then it would be very easy to field upgrade systems at the vehicle dealerships with these new models, simply by appending them to the existing set of store training samples.

These results are for stationary vehicle testing where the occupant is required to sit in pre-specified positions. We also demonstrated through using our contextual processing that we are able to manage the incoming stream of classification results using evidential reasoning to handle erroneous classification results. Classification results that are unreliable due to abnormal occupant seating positions are handled intelligently, and are not allowed to affect the final classification result. We demonstrated the performance of the contextual processing on a real-world situation of an adult seated normally, leaning far forward, and then returning to the upright position. In the input classification stream, 9 of the 32 images in the sequence were

mis-classified, while after the Dempster-Shafer-based contextual processing there were no errors. This is a dramatic improvement from only 71.8% correct classification to 100% correct classification.

We also showed the ability of our algorithm based on Dempster-Shafer theory of evidential reasoning to quickly return to the correct classification, when the image stream begins with an unusual image that causes the classification stream to begin mis-classified. In this example scenario, the system recovered within two image frames to provide the proper classification, and the system continued to provide the correct classification, even when some noise events caused two subsequent mis-classifications in the data stream.

DATE DUE	DATE DUE	DATE DUE

SEGMENTATION, CLASSIFICATION, AND TRACKING OF HUMANS FOR SMART AIRBAG APPLICATIONS

VOLUME 2

By

Michael E. Farmer

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science & Engineering

2004

Chapter 6.
Integrated Segmentation and Classification

The traditional processing flow of segmentation followed by classification assumes that segmentation is able to reliably extract the object of interest from the input scene. This is optimistic without any context as to what object is being extracted from the scene. Pal and Pal comment, "any mathematical algorithm usually should be supplemented by heuristics which involve semantic information about the class of images under consideration" [82]. Additionally, they comment, "the literature is very rich on the methods of segmentation, but not many attempts have been made for the objective evaluation of segmented outputs" [82].

We propose a method of segmentation that addresses both of these issues by using the classification subsystem as an integral part of the segmentation to provide:

1) "semantic information about the class of images under consideration" by focusing the segmenter on a given class at a time and trying to find the best segmentation for that class, and
2) "the objective evaluation of segmented outputs" by using the probability of correct classification as the fundamental metric to determine the segmentation quality.

Traditional segmentation can be viewed as a filter operating on the image that is independent of the subsequent classifier. This parallels the filter methods for feature selection, where the optimal subset of features is chosen independently of the

classification. In contrast to this, we propose a paradigm for segmentation that follows the wrapper methods of feature selection. Recall Figure 4.1, where the comparison of filter versus wrapper methods for feature selection was shown, and then consider Figure 6.10 where traditional image segmentation is compared to the proposed wrapper-based image segmentation. In the wrapper method of the feature selection, the classifier is an integral part of the selection process, and serves as the generator for the metric that decides the feature set. In the same way, we propose *wrapping* the segmentation and the classification together, and using the classifier to determine the best segmentation.

Other researchers have proposed using classification output as a metric for low-level image processing tasks such as image binarization [94]. In this particular work, the classification result was used to alleviate the problem of the absence of a clear metrics for evaluation, but they did not propose a framework for maintaining the classifier as an integral part of the low-level algorithms. Other approaches for integrating the segmentation and classification tasks have been directed at developing structural models of the desired object, and by using graph theoretic techniques to search for regions in the image that may correspond to these substructures [95]. This has been applied to segmenting humans in an image. The drawback of this method is that it defines a particular form for the classification problem, namely structural pattern recognition using graph-distances. Other researchers have shown that these methods are sensitive to errors during the low-level segmentation of the sub-regions, and that statistical pattern recognition methods are generally superior [27].

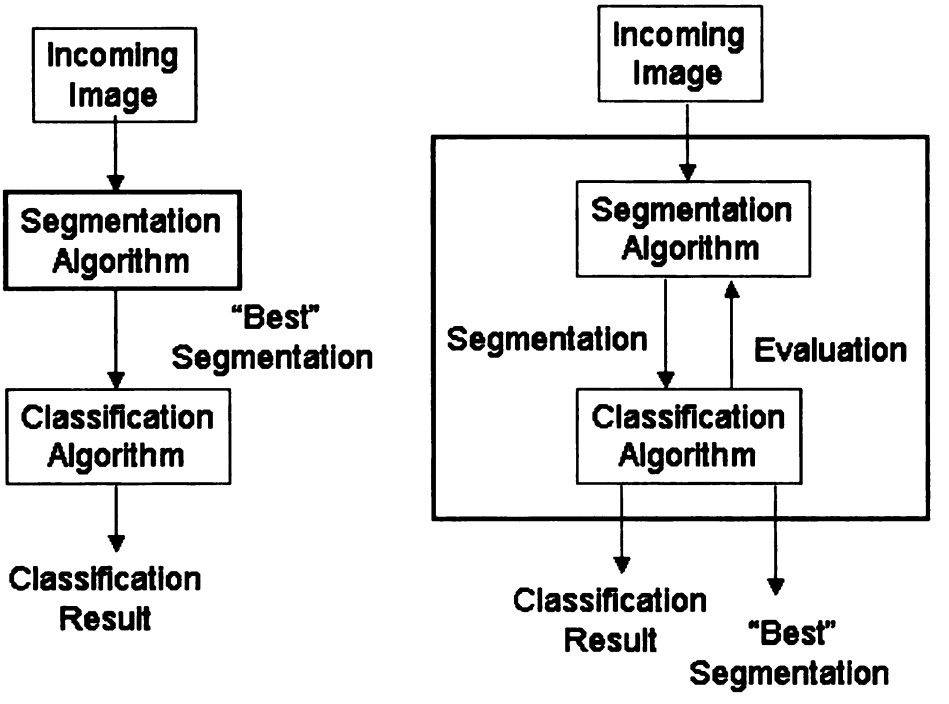


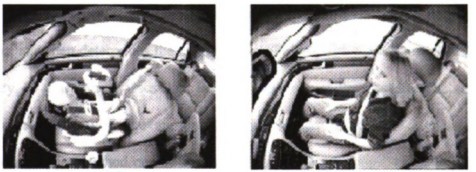
Figure 6.1: Representation of approaches to segmentation. (a) traditional segmentation represented as a filter approach, and (b) integrated segmentation represented as a wrapper-based approach.

Our approach of a wrapper-based segmenter has three significant advantages over the traditional segmentation methods followed by a classification algorithm:

1) Our method relaxes the requirements on the image-labeling algorithm. Other segmentation methods assume the pixels on the object share a common set of attributes, thereby allowing the object to be extracted as a single entity. Unfortunately, even extremely simple images may not satisfy this requirement, as is certainly true for our application, as is shown in Figure 6.2.

2) The second advantage of our wrapper-based segmentation approach is that it builds on the rich literature on feature selection, which provides many proven methods for addressing the selection of a subset of features from a larger set. We will show how the blob combining process can easily be mapped into this feature selection framework, thereby allowing these existing algorithms for feature selection to be readily adapted to blob combining.

3) The third advantage is our flexibility in the choice of the classifier used. Any classification algorithm may be used in this wrapper method. This allows the user to select the classifier that is best suited for their particular application, which is attractive, since there is no universally best classifier [46].



(a) (b)

Figure 6.2: Examples of simple objects that cannot be segmented based on a simple common attribute. (a) infant in child seat, and (b) adult on passenger seat.

There is an abundance of literature on image segmentation, and a number of review articles highlighting them [80][81][82]. The segmentation approach we will adapt for our wrapper-based segmenter is the region-based segmentation algorithm. In their traditional implementation, the region-based methods assume the object of interest and the background can be separated into two clearly differentiable regions. We will

relax this underlying assumption, and use the region-based algorithm to divide the image into a number of distinct regions based on some common characteristic, such as grayscale, color, texture, etc. We merely use these algorithms to divide the input image into a set of smaller image regions, which are often called "blobs", a term we will adopt here [89].

In our approach, we do not assume that the object of interest must be segmented from the background as a single entity. Our approach actually prefers that the image be over-segmented, in the sense that the object is divided into some number of smaller labels [87][89]. This greatly relaxes the performance requirement on the image-labeling algorithm, and it allows nearly any reasonable region-labeling algorithm to be used.

Once the image is fully labeled into blobs, the classifier collects combinations of blobs, and then determines the combination of blobs with the highest classification accuracy. Our approach uses the classifier to intelligently determine the subset of blobs corresponding to the appropriately segmented object. Consequently, our proposed wrapper method can be considered a framework within which any existing traditional image segmentation algorithms may be executed to improve its performance. The classifier provides the required contextual information to direct the segmentation. The five stages to this segmentation processing are shown in Figure 6.3.

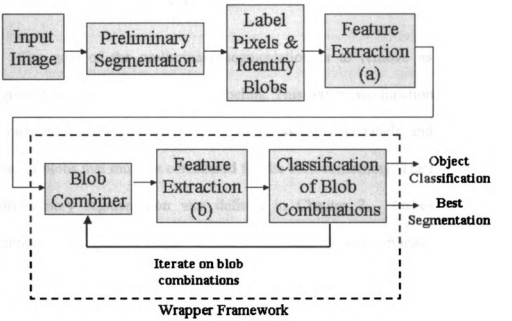


Figure 6.3: Wrapper-based image segmentation.

Notice that the feature extraction processing is shown in two possible locations, labeled (a) and (b). In traditional pattern recognition systems, feature extraction follows the segmentation, which places this function after the blob combiner (location (b)). This is a sub-optimal location for our wrapper-based segmentation processing, since a complete set of features must be calculated for every blob combination. Note, however, that many combinations of blobs share common blob subsets, which means some of the feature extraction calculations may be repeated many times. By moving the feature extraction processing to location (a) in the processing flow, we can pre-compute the features for each blob and then, as the blobs are combined, we merely arithmetically add the feature vectors together. This provides a significant speed up, and allows for the possibility of real-time operation.

6.1 Preliminary Segmentation

The goal of the preliminary segmentation is to remove as much of the background as possible, prior to blob labeling. This serves two functions: (i) remove a large number of pixels from the image prior to assigning the labels, and (ii) reduce the number of blobs that must be considered for the blob combining. The method we use for preliminary segmentation was defined in Chapter 2. An example of the preliminary segmentation output for the airbag suppression application is shown in Figure 6.4. Note that in other applications there may be no logical mechanism for performing this preliminary segmentation, in which case this stage may be skipped.

6.2 Pixel Labeling and Blob Identification

The purpose of the pixel labeling and blob identification module is to group together pixels of common grayscale, texture, etc. into contiguous regions, or blobs. There are many mechanisms proposed for defining these 'common characteristics', such as EM, normalized cuts (and other eigenvalue-based methods), relaxation methods, region growing methods, and split-and-merge methods [78][79][80][81][82][83][84][87][89]. There are two types of outputs from these algorithms. Methods, such as the region growing and the split and merge algorithms, output actual labeled regions, while algorithms such as EM merely assign each pixel individually to a class. These labeled pixels must then be aggregated into labeled regions using a grouping algorithm. Since there is no universally best algorithm for

region labeling, the selection of the algorithm should be based on the ease of use and its flexibility.

We will now define the three processing tasks that we must perform in this stage: (i) pixel labeling, (ii) specularity reduction, (iii) pixel-grouping and blob identification.

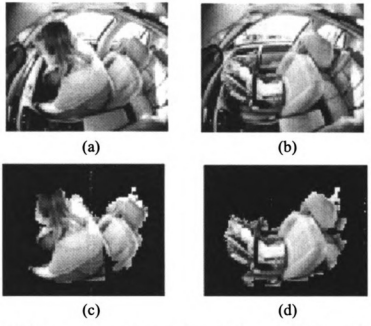


Figure 6.4: Preliminary segmentation results, (a) input adult image, (b) input RFIS image, (c) segmented adult, and (d) segmented RFIS.

6.2.1 Pixel Labeling

We will model this processing stage as an unsupervised clustering process, where the input pixel values are the observed data, and they are clustered into labeled entities as shown in Figure 6.5. We will use the EM algorithm, and model the input image grayscale values as a mixture of Gaussians [45][90]. The Expectation

Maximization (EM) algorithm has been used for fitting a mixture of Gaussian distributions to a dataset [90]. The dataset in our case is the histogram of the image grayscale values. While the EM algorithm is guaranteed to converge, it is not guaranteed to converge to a global minimum. We use EM as a coarse image-labeling algorithm. Finding the globally optimal mixture is not critical, since we will be performing additional grouping of the blobs as part of the subsequent blob combining.

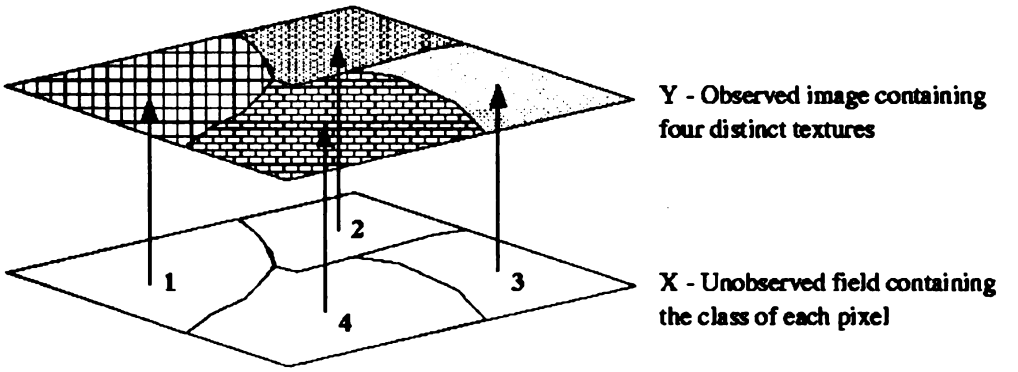


Figure 6.5: Unobserved labels for each region of observed pixel values [71].

For an assumed mixture of Gaussians, the EM processing is based on the idea that the image is comprised of a weighted-sum of Gaussian densities, according to [90]:

$$p(x \mid \Theta) = \sum_{i=1}^{M} \alpha_i \cdot p_i(x \mid \vartheta_i), \qquad (6.1)$$

where the α_i are the weighting factors, and ϑ_i are the mean and variance estimates for each of the underlying Gaussians. The weights, the means, and the variances for the Gaussians are estimated in an iterative manner, according to the following standard equations [90]:

$$\alpha_l^{new} = \frac{1}{N} \sum_{i=1}^{N} p(l \mid x_i, \Theta), \tag{6.2}$$

$$\mu_l^{new} = \frac{\sum_{i=1}^{N} x_i \cdot p(l \mid x_i, \Theta)}{\sum_{i=1}^{N} p(l \mid x_i, \Theta)}, \text{ and} \tag{6.3}$$

$$\Sigma_l^{new} = \frac{\sum_{i=1}^{N} p(l \mid x_i, \Theta) \cdot (x_i - \mu_l^{new}) \cdot (x_i - \mu_l^{new})^T}{\sum_{i=1}^{N} p(l \mid x_i, \Theta)}. \tag{6.4}$$

In these equations, the likelihood function, $p(y_i \mid x_i, \Theta)$, is computed according to [90]:

$$p(y_i \mid x_i, \Theta) = \frac{\alpha_{y_i} \cdot p_{y_i}(x_i \mid \theta_{y_i})}{\sum_{k=1}^{M} \alpha_k \cdot p_k(x_i \mid \theta_{y_i})}. \tag{6.5}$$

We have modeled the segmented images in our automotive airbag suppression system with one Gaussian set to zero mean and a very small variance to model all the previously removed background pixels. Additional Gaussian components are distributed across the histogram to model the actual data of interest. We have two

mechanisms for placing these Gaussian components based on the relative uniformity of the incoming histogram. If the number of pixels in the upper or lower one third of the grayscale range exceeds a threshold, we place more Gaussians in that third, otherwise we uniformly distribute the initial Gaussian components.

In addition to the placement of the Gaussian components across the input histogram, we must also determine the number of Gaussian components to use in the mixture model. An automated method for selecting the number of components in a mixture has been proposed, but this method is basically a greedy search method, which processes a broad range of mixtures, and then selects the number of components that provides the best fit to the original histogram [91]. This approach is computationally too intensive for our real-time application. Instead, we begin with a fixed number of components, and then we rely on the classification accuracy to determine if more mixture components are required.

If the classification distances for all of the possible classes are too high, then we assume an inadequate segmentation (parts of the occupant may still be connected to the background), and at this point we have two options: (i) output an unknown classification for the current image, and let the contextual processing derive the classification, or (ii) re-execute EM on the current image with a larger number of components. If, after modifying the EM parameters, the classification results still do not improve, or they become globally worse, then we assume the image does not contain any of the desired objects and output the unknown class. If the change does improve the classification, then we output this new classification result, and we continue to use this number of mixture components for processing future images.

While this method of component number selection is not a universally useful approach, it is well matched to our concept of developing a classifier-driven segmentation algorithm.

Figure 6.6 (a) shows a segmented image that is fed to the EM algorithm, and Figure 6.6 (b) shows its histogram. Figure 6.6 (c) shows the Gaussian mixture that best models this histogram. From the mixture distribution in Figure 6.6 (c), we generate the labels corresponding to each pixel in the image. The mixture component with the highest probability becomes the label for that pixel. The output of the EM algorithm is then itself an image, where the actual pixel values are replaced by the membership label indicating the component to which the pixel is most likely to belong, as shown in Figure 6.6 (d) [90].

6.2.2 Specularity Reduction

Additional processing may sometimes be needed to reduce the specularity of the image, because EM labels the individual pixels without any constraint on the spatial variation of the pixel labels. This may cause many small regions to be labeled in the image as can be seen in Figure 6.6 (d). Imposing a smoothness constraint on the output image will reduce the number of very small blobs that are generated, and reduce the complexity of the subsequent blob combining processing. A very common method for imposing a smoothness constraint is through the use of Markov Random Fields [93].

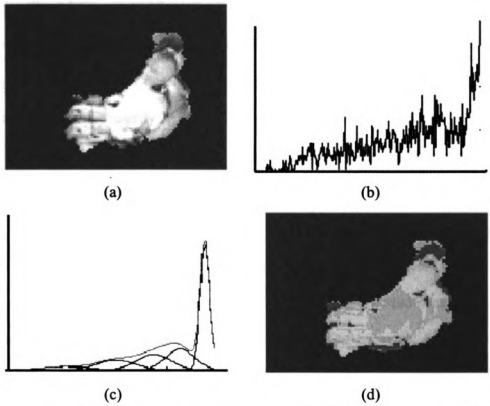


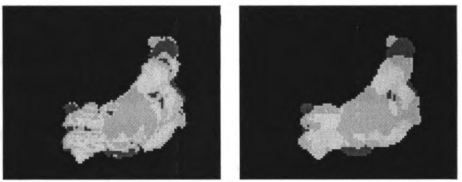
Figure 6.6: EM processing stages, (a) input pre-segmented image, (b) histogram of this image, (c) final Gaussian mixture, and (d) final labeled image.

Within the framework of MRFs, it is then common to use Maximum a priori Probability (MAP) estimates of the actual pixel labels [93]. Unfortunately, there are many difficulties with MRFs and MAP estimation for segmentation, namely [93]:

1) MRFs cannot explain large scale behaviors, since they provide constraints only on adjacent pixels,
2) estimation of MRF parameters is difficult,
3) many local minima occur in the MAP in the search space, and
4) MAP estimates the probability of correct label for each pixel, which is overly conservative since regions are composed of many pixels at once.

Additionally, MRF and MAP methods are computationally quite intensive. The purpose of the wrapper-based segmenter is to direct large-scale grouping of pixels through the classifier. Consequently, we are interested in a simple method for performing some specularity reduction at minimum processing expense. Histogram based blob smoothing has been used for a similar content-based retrieval system with reasonable results, so we will adopt a similar method here [89].

The histogram-based blob smoothing uses a simple mode filter with an *NxN* kernel [89]. The label of the pixel in the center of the kernel is replaced with the label that most often occurs in the kernel window. The kernel size we have found to be the most effective is 5x5. Figure 6.7 (b) shows the output of the mode filtering for the labeled image in Figure 6.7 (a). This mode filter is preferred over other filters (such as median or mean filtering), since the objects being filtered are discrete label values rather than grayscale values. The mode filter serves as a voting filter, where the most commonly occurring label in its neighborhood replaces a pixel's label.



(a) (b)

Figure 6.7: Output of specularity reduction, (a) original EM labeled image, and (b) mode filtered image.

6.2.3 Blob Identification

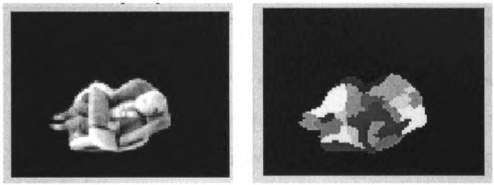
The blob identification stage generates a list of blobs, and their statistics, including, pixel count, maximum and minimum row and column locations, and an ID tag. The labeled pixels are converted into contiguous blobs using an 8-way connected components region-labeling algorithm [83]. The output image from the blob identification is an image where each contiguous blob in the image has its own unique ID value.

6.3 Blob Combining

Analogous to the wrapper methods for feature selection, each of the possible combinations of the blobs is evaluated based on the probability of correct classification of the segmentation with the latest blob added or removed. Since image segmentation is a situation where we do not have labeled samples, in contrast to the feature selection problem, the segmentation is assumed to be correct when a minimum distance between the test sample and a known pattern class is attained. The final classification decision is then the pattern class with the lowest overall classification distance, across all the a priori pattern classes.

There are a number of feature selection methods that can be adopted for blob combining, including sequential search, gradient descent, greedy search, genetic algorithms, etc. Specifically, we will show how the gradient descent, random mutation, and sequential search algorithms can be adapted to this application. For

initially demonstrating the mechanics of the algorithms, we will use the hand-segmented image of a 3 year-old ATD in a forward facing child restraint, and the resultant blob labeled image shown in Figure 6.8.



(a) (b)

Figure 6.8: Test image of a 3 year-old child in a forward facing child restraint for demonstrating the blob-combining algorithms, (a) hand segmented grayscale image, and (b) blob labeled image.

6.3.1 Gradient Descent

In the gradient descent algorithm, blobs are combined based on choosing the blob that provides the greatest improvement in the classification accuracy during each iteration. We extended the basic gradient descent algorithm to provide some ability to avoid local minima by changing the fitness test. Rather than using an elitist strategy, where the new blob set is chosen only if it is more fit than the last blob set, we modified the original algorithm to include the possibility of choosing a worse fit, as is also possible in simulated annealing [60]. Recall in simulated annealing, there are two cases for choosing a new state: (i) choose the state if it improves the overall

fitness of the system, or (ii) choose the new state with some randomization. The gradient descent blob-combining algorithm is shown in Table 6.1.

Table 6.1: Algorithm for gradient descent blob combining.

1) For a given class C, create an initial set of blobs $X_0 = \phi$, the empty set.
2) For each blob x_k, create the set $\{X_k\} = \{X_{k-1}, x_k\}$, and compute $P_{class}(\{X_k\} \mid C)$ for iteration k.
3) Include the blob x_k that provides the greatest improvement in the classification accuracy. If no blob improves it then, with probability P_{select}, select the blob with the least negative impact to the classification accuracy, and record $P_{class}(\{X_k\} \mid C)$.
4) Return to (2) until all blobs are used, or until no blobs are selected, and record $P_{class}(\{X_k\} \mid C)$ for each subsequent subset of blobs.

The gradient descent algorithm is performed for every class, C, and, at the completion of the processing of all the candidate classes, we select the class C that provides the highest $P_{class}(\{X_k\} \mid C)$. The set $\{X_k\}$ then defines the blob combination that comprises the best segmentation. Figure 6.9 and Figure 6.10 demonstrate the behavior of the classification distance as blobs are added. Notice that around the fifteenth iteration of the algorithm, an extremely large local minimum was encountered for the infant class rather than the correct child class. Without the ability to select the state with a slightly worse classification result, the final segmentation would not have been possible. Additionally, gradient descent only sequentially adds

blobs and cannot remove blobs once they are added. We will see that this will dramatically limit the ability of the algorithm to converge to a proper classification and segmentation.

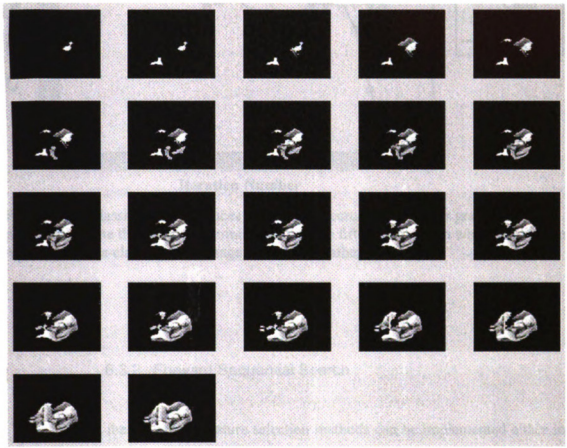


Figure 6.9: Sequence of blob combinations using the gradient descent algorithm.

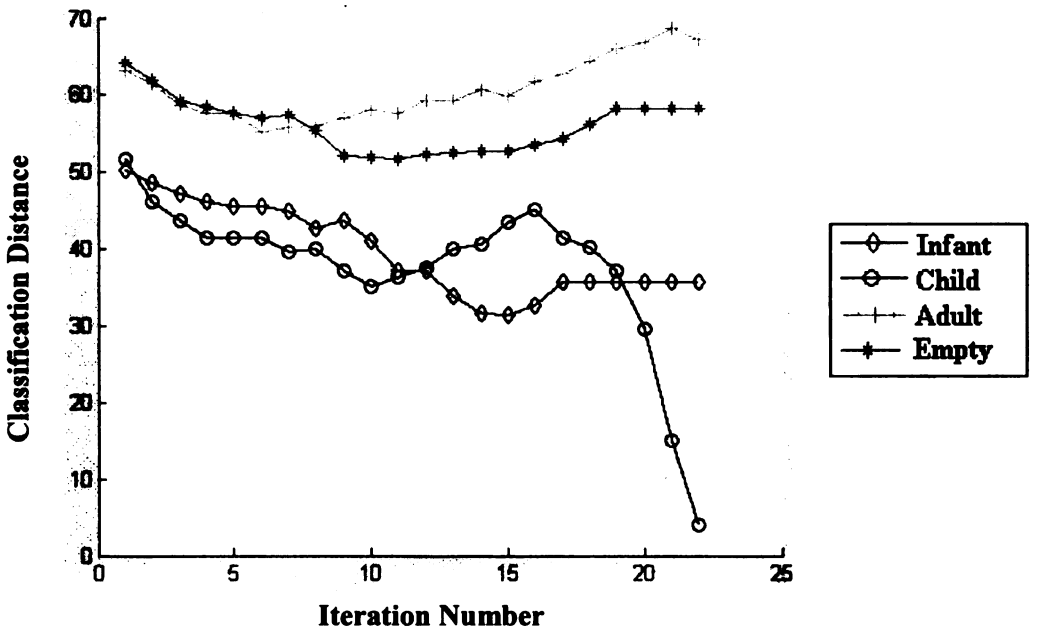


Figure 6.10: Classification distances versus blob accrual iteration for gradient descent processing. Note the deep local minima around the fifteenth iteration where the system would have mis-classified the image as an infant rather than a child.

6.3.2 Forward Sequential Search

Recall, the sequential feature selection methods can be implemented either in the forward selection mode or in the backward selection mode. For our blob combining application, the forward selection method was selected. There are numerous variations of the forward selection algorithm available, and we selected the implementation called the plus-L, minus-R algorithm. This algorithm begins with an initial set of blobs, $\{X_0\}$, then adds up to L blobs per iteration, and finally subtracts up to R blobs per iteration. The details of the algorithm are provided in Table 6.2 [62].

Table 6.2: Algorithm for using the plus-*L*-minus-*R* forward sequential search for blob combining.

1) For a given class C, create an initial set of blobs $X_0 = \phi$, the empty set.
2) Blob Addition: Test each blob of the unselected blobs, and add blob x_l if $P_{class}(\{X_k\} + x_l \mid C) \geq P_{class}(\{X_k\} \mid C)$, where $P_{class}(\{X_k\} \mid C)$ is the classification accuracy for the blob set $\{X_k\}$, given class C. The output of this stage is a new subset $\{X_{k+1}\} = \left\{\{X_k\}, x_{k+1}^{(1)}, x_{k+1}^{(2)} \ldots x_{k+1}^{(L)}\right\}$, where $x_{k+1}^{(i)}$ is the blob with the (i)th best improvement in classification accuracy, up to L blobs.
3) Blob Removal: Test each blob in the current selected blob set, $\{X_{k+1}\}$, and remove each blob x_r from the set if $P_{class}(\{X_{k+1}\} - x_r \mid C) \geq P_{class}(\{X_k\} \mid C)$, where $P_{class}(\{X_k\} \mid C)$ is the classification accuracy for the blob set $\{X_k\}$ given class C. Continue testing and removing blobs until all the blobs in the current subset $\{X_k\}$ are tested, or until R blobs have been removed.
4) Record the $P_{class}(\{X_k\} \mid C)$, and the corresponding subset of blobs $\{X_k\}$, and return to step (2), unless the last blob has been processed.

This algorithm is also performed for every class, C, and, at the completion of the processing of all the candidate classes, we select the class C that provides the highest $P_{class}(\{X_k\} \mid C)$. Again, the set $\{X_k\}$ defines the blob combination that comprises the best segmentation. The sequence of blob operations, when L and R

are both set to one, is provided in Figure 6.11, and the sequence of distance measures for each blob addition is provided in Figure 6.12. Notice that the deep local minima problems we saw in the gradient descent are dramatically reduced, and the convergence of the algorithm to the correct solution is more robust than for gradient descent.

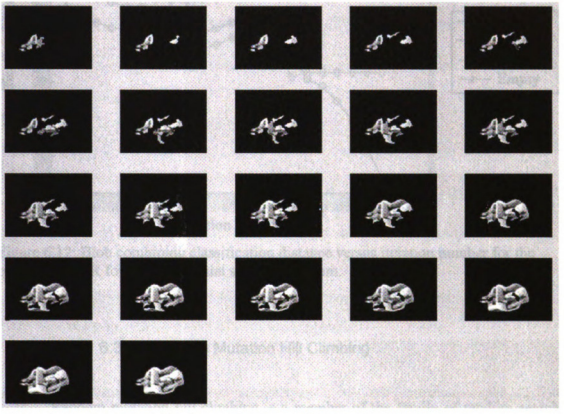


Figure 6.11: Sequence of blob combinations using the plus-L-minus-R forward sequential search algorithm.

Unfortunately, the forward sequential search algorithm is computationally very intensive, due to the thoroughness of its testing of blob combinations. This thoroughness is the source of its effectiveness, however, for a real-time application it is necessary to attempt to find a faster algorithm. The following algorithm is based

on random search, which can be fast if the search space is relatively well behaved (i.e., few local minima).

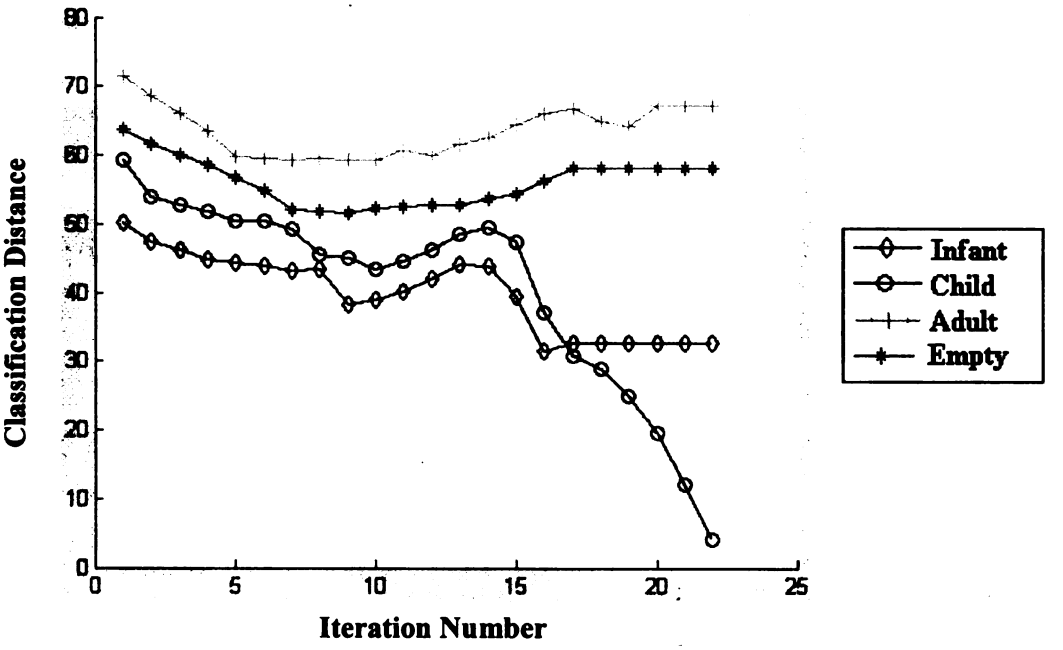


Figure 6.12: Blob combining classification distance versus iteration number for the plus-L-minus-R forward sequential search algorithm.

6.3.3 Random Mutation Hill Climbing

Random mutation hill climbing is a member of the family of random search optimization tools that includes methods such as simulated annealing [59][60]. It has proven useful in the area of feature selection. For the random mutation algorithm, the complete set of blobs is represented as a string of binary values, where a bit in the string is set to *'1'* if the blob is to be kept in the final subset, and set to *'0'* if the blob is to be discarded [59].

mut

mut

with

mut

Tab

com

prov

com

for

relat

We made the same modifications that we had made when applying random mutation hill climbing to the feature selection problem: (i) we added multiple blob mutation with cooling, and (ii) we allow the system to choose a blob combination with an inferior classification accuracy, with some probability, P_{mutate}. The random mutation algorithm for blob combining operates as shown in Table 6.3.

Table 6.3: Algorithm for using random mutation hill climbing for blob combining.

1) Set the initial value for M and create an initial set of blobs $X_0 = \phi$, the empty set.
2) Randomly mutate M blobs.
3) Test for fitness: Compute $P_{class}(\{X_k\}|C)$, where the set $\{X_k\}$ is the current blob set. If $P_{class}(\{X_k\}|C) \geq P_{class}(\{X_{k-1}\}|C)$ keep this blob set. If not then choose the new blob set anyway with a pre-determined probability P_{mutate}.
4) Return to step (2) and continue until either the fitness goal is reached or the maximum number of iterations is reached.

As before, this algorithm is performed for every class, C, and, at the completion of the processing of all the candidate classes, selects the class C that provides the highest $P_{class}(\{X_k\}|C)$, where the set $\{X_k\}$ defines the blobs that comprise the best segmentation. The behavior of the random mutation hill climbing for a mutate-1 strategy can be seen in Figure 6.13 and Figure 6.14. Note the relatively monotonic behavior of the performance versus blob selection.

Figure 6.13: Sequence of blob combinations using the random mutation hill climbing algorithm

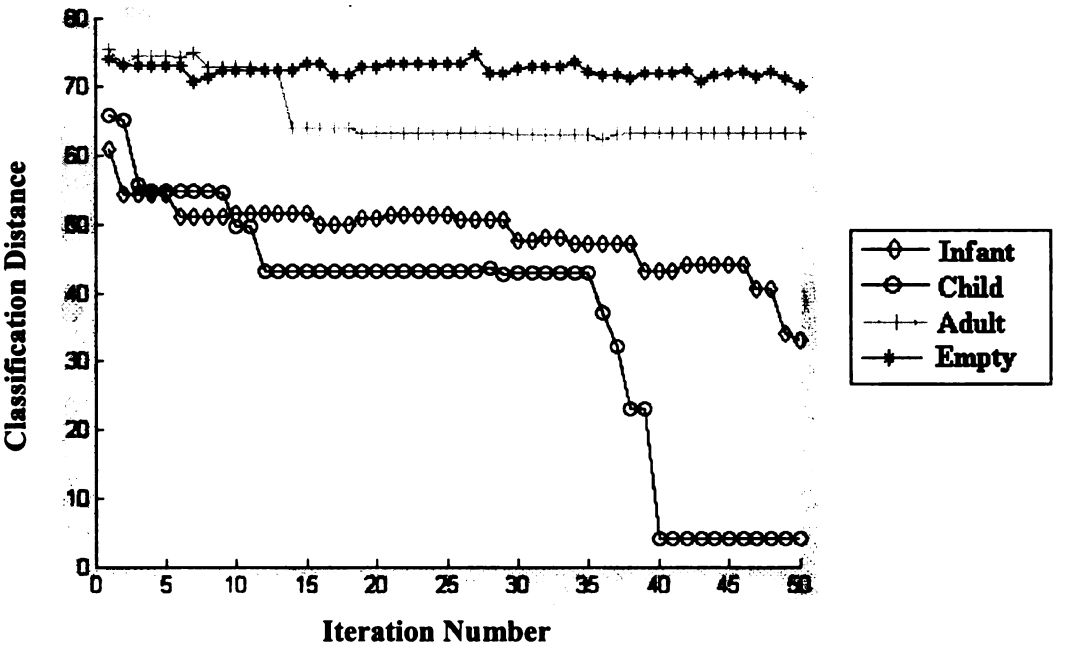


Figure 6.14: Blob combining classification distance versus iteration number for random mutation hill climbing.

6.4 Feature Extraction

Recall from Chapter 3 that there are numerous candidate features that can be used to represent object shapes [23][24]. The shape can be represented either as a contour or as a solid in this space. Two methods that have been widely used in a variety of shape-based feature extraction applications are (i) moments and (ii) Fourier descriptors [22][23][24]. Since Fourier descriptors are computed on the entire boundary, they cannot support the pre-computation of the features for each blob that moments can support, which is critical to achieving real-time performance.

Consequently, the feature set we chose is the moments of the binary combined blob images.

6.4.1 Moment Computation

The most fundamental geometric moment of order $(m+n)$ for an KxN image, $I(i, j)$, is defined by the equation:

$$M_{mn} = \sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j) \cdot i^m \cdot j^n, \qquad (6.6)$$

and the central moments are defined by:

$$\mu_{mn} = \sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j) \cdot (i - \bar{i})^m \cdot (j - \bar{j})^n, \qquad (6.7)$$

where

$$\bar{i} = M_{10} / M_{00} \quad \text{and} \quad \bar{j} = M_{01} / M_{00}. \qquad (6.8)$$

Recall that one of our objectives is to pre-compute the moments for each blob, and then mathematically add them together during blob combination to enhance the real-time performance. This means the central moments for each blob cannot be pre-computed, since the relative spatial information of each blob will be lost. Therefore, if the central moments are to be used, then the geometric moments for each blob can be pre-computed. These moments can then be centralized after the blob combination through the equation:

$$\mu_{mn} = \sum_{r=0}^{m} \sum_{s=0}^{n} \binom{j}{r} \cdot \binom{k}{s} \cdot (\bar{i})^{j-r} \cdot (\bar{j})^{k-s} \cdot M_{rs} . \qquad (6.9)$$

Note that as with the blob combining itself, this is merely an additive weighted combination of the moments rather than a complete re-computation. Recall we also showed the benefits of using orthogonal moments for shape analysis. The specific set of orthogonal moments we chose is the Legendre moment representation of the binary combined blob images. Like geometric moments, the Legendre moments can be pre-computed on each blob and then added together.

We found the central Legendre moments to be the very effective. To compute these, we first pre-compute the geometric moments. Then for each blob combination, we compute the central moments, and then convert them to the Legendre moments defined earlier. Since these conversions to central and then Legendre moments only depend on the number of moments, and do not depend on the number of pixels in the image, this algorithm is still amenable to real-time execution.

6.4.2 Speed-up Techniques for Moment Computation

The first speedup mechanism for feature extraction is related to the linearity of the moment processing. It was motivated by the real-time implementation of moment computations proposed by Spiliotis and Mertzios, where the moment computation is decomposed into a summation of the moment calculations over a set of non-overlapping rectangular homogenous blocks, as shown in Figure 6.15 (a)

[37][38]. We abstract this approach one step further and, rather than computing the moments over rectangles, we compute the moments over our arbitrarily shaped, non-overlapping blobs as shown in Figure 6.15 (b).

Computing the moments on the individual blobs allows us to pre-compute the moments for each blob, and then simply add the moments for each blob together, as we perform the blob combining. This is possible, since the input image is merely a summation of all of the blobs:

$$I(i,j) = \sum_{k=0}^{K} I^{(k)}(i,j), \tag{6.10}$$

where $I^{(k)}(i,j)$ is the portion of the image corresponding to blob k. From this we can now rewrite the moment equation:

$$\begin{aligned} M_{lk} &= \sum_{j=0}^{N}\sum_{i=0}^{M} I(i,j)\cdot i^{l}\cdot j^{k} = \sum_{j=0}^{N}\sum_{i=0}^{M}\sum_{k=0}^{K} I^{(k)}(i,j)\cdot i^{l}\cdot j^{k} \\ &= \sum_{k=0}^{K}\sum_{j=0}^{N}\sum_{i=0}^{M} I^{(k)}(i,j)\cdot i^{l}\cdot j^{k} = \sum_{k=0}^{K} M_{mn}^{(k)}, \end{aligned} \tag{6.11}$$

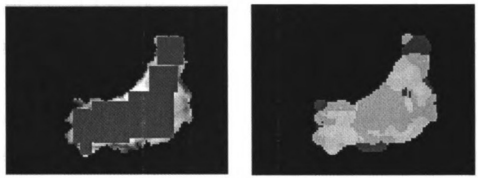
where we have changed the order of the summations, and defined $M_{mn}^{(k)}$ as the moment of order $(m+n)$ corresponding to the k^{th} blob. Thus, we can pre-compute the moments for each blob, and then we need to only add the feature vectors to compute the moments for any blob combination. This allows us to very rapidly try different blob combinations with a processing burden that is only linear in the number of blobs,

rather than linear in the number of pixels. . For an 80x100 image, if we assume there are 20 blobs, this produces a speed-up of 400:1 for each moment calculated.

The second speedup was also motivated by the work of Spiliotis and Mertzios [38]. We apply their results and compute the moments only over the bounding rectangle for each blob. Recall, we recorded the start and end pixels for the rows and column for each blob, and now we can use that information to compute the geometric moments for each blob according to:

$$M_{mn}^{(k)} = \sum_{j=r_1}^{r_2} \sum^{c_2} I^{(k)}(i,j) \cdot i^m \cdot j^n, \qquad (6.12)$$

where r_1 and r_2 are the start and stop rows and c_1 and c_2 are the start and stop columns for the bounding rectangle for blob k, and $I^{(k)}$ is the image corresponding to blob k.



(a) (b)

Figure 6.15: Comparison of fast moment calculation algorithms, (a) shape decomposition method, and (b) our EM-based labeled blobs.

Reducing the area of the image to process can considerably reduce the processing load required by the following ratio:

$$speedup = \frac{N \cdot M}{(r_2 - r_1) \cdot (c_2 - c_1)}, \qquad (6.13)$$

where the image is N rows by M columns. For a 10x10 region extracted from our 80x100 images, this produces a 80:1 speedup. Thus, by combining both speedup mechanisms, we attain nearly a $3 \cdot 10^4$ reduction in processing time for each moment calculated.

6.5 Classification of Blob Combinations

The blob combinations must be classified during all iterations of this wrapper method to create a metric for selecting the best segmentation. This is accomplished through classifying the current blob combination with respect to each class, and then recording the probability of correct classification. This metric is then used in the subsequent processing to determine the overall best segmentation and classification.

The alternative k-nearest neighbor classifier scheme that was defined in Section 4.1 is used for this application. The k-nearest neighbor classifier was chosen for the same reasons discussed earlier and due to its good performance on the edge-based features of our application. The modified k-nearest neighbor was chosen over the traditional k-nearest neighbor, since the distance metric that it produces is of a finer resolution than a m-of-k voting scheme, without making k too large. This distance metric is, therefore, more useful for ordering the possible blob combinations.

For each a priori class, we compute the average distance of the test sample to the k-nearest neighbors for each known pattern class. As the combination of blobs is modified, the classification distances are recomputed and stored. Figure 6.16 shows a plot of the classification distances versus blob combination for the two-class problem of adult versus infant, using the FSS algorithm with $L=3$ and $R=2$.

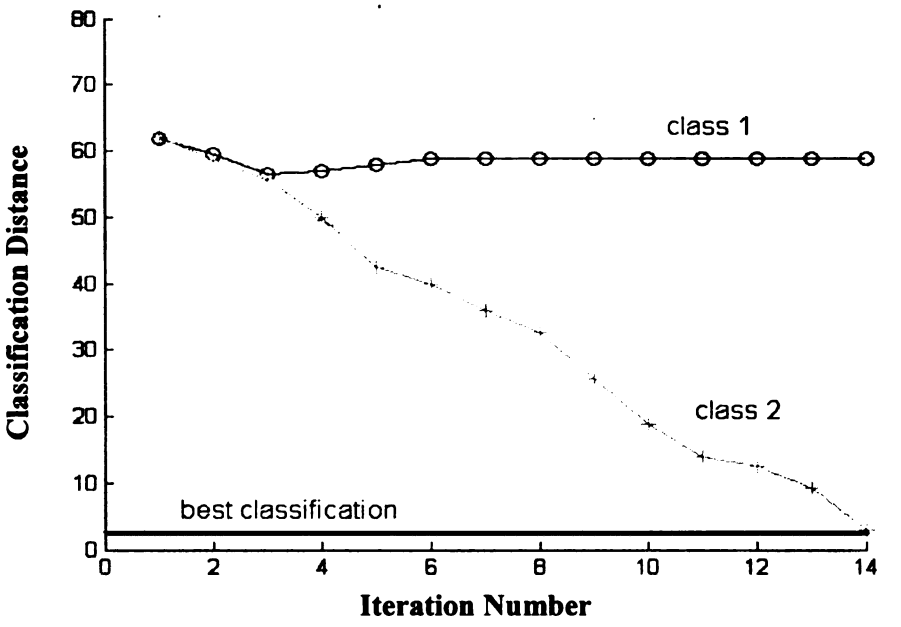


Figure 6.16: Classification distances for each of the a priori class assumptions; class 2 provides the best classification value, and the best segmentation corresponds to blob combination # 14.

The classification distance curve containing the minimum overall distance for all classes defines the output pattern class. The segmentation is then defined by the blob sequence ID number corresponding to that minimum distance. The bold line in Figure 6.16 shows the best classification on the y-axis, and the location of the occurrence of this minimum along the x-axis defines the index of the resultant best segmentation. The blob sequence ID is the identification of the iteration in the blob

combining process at which the best classification was attained. By maintaining a list of the specific blobs that were included during all the iterations in the processing, the best segmentation is reconstructed.

6.6 Results of Integrated Segmentation Classification System

To develop a training database, we hand-segmented nearly 400 images. Separate databases are used for training and testing, and the breakdown of training and test images can be found in Table 6.4. The features extracted are the central Legendre moments of up to the 25^{th} order. We selected lower order moments here than for the edge-based classification, since the silhouette image contains lower frequency information than the edge-based image, and therefore can be adequately represented with lower order moments. This will help us maintain a reasonable processing burden for classifying the blob combinations.

The classification is performed with a 1-nearest neighbor classifier. The classification results for each blob combination algorithm are provided as a confusion matrix and a corresponding probability of correct classification matrix. The classifier uses the raw moment features with no subsequent feature set reduction, since we are using the features to perform some level of image reconstruction as we test the blob combinations, rather than simply classification. Examples of the correct, as well as incorrect, segmentations, generated for the two classes, are provided for each algorithm.

Table 6.4: Breakdown of training and test datasets for testing the wrapper-based segmenter on the two-class suppression problem.

Pattern Class	Number of Images in Training Set	Number of Images in Test Set
Infant	182	1797
Adult	206	208

6.6.1 Results for Gradient Descent Blob Combining

The classification results using the gradient descent method of blob combining can be found in the form of a confusion matrix in Table 6.7 and the resultant probability of correct classification can be found in Table 6.8. Note the overall accuracy of the system is roughly 65%. Since this algorithm performed so poorly, we will not show example segmentation outputs. We must find an algorithm that performs better than 90 % in order for it to warrant any further consideration.

Table 6.5: Confusion matrix for the gradient descent blob combining on the two-class suppression problem.

	Classified as Infant	Classified as Adult
True Infant	1022	421
True Adult	79	129

Table 6.6: Percentage correct classification for the gradient descent blob combining on the two-class suppression problem.

	Classified as Infant	**Classified as Adult**
True Infant	70.8%	29.2%
True Adult	38.0%	62.0%

Clearly, the classification results are not encouraging for this method. Its inability to remove blobs after they are included clearly has a negative impact on the performance.

6.6.2 Results for Plus-L-Minus-R Forward Sequential Search Blob Combining

The classification results using the forward sequential search method of blob combining can be found in the form of a confusion matrix in Table 6.7 and the resultant probability of correct classification can be found in Table 6.8. Note the overall accuracy of the system is roughly 91%. Some examples of the correct segmentations performed by the FSS algorithm for the infant and adult classes are provided in Figure 6.17 through Figure 6.22. Likewise, some of the incorrect segmentations are provided in Figure 6.23 and Figure 6.24. These incorrect examples demonstrate the common failure modes of the system. Figure 6.23 shows the case where errors result because the initial segmentation removed too much of the occupant (in this case a large portion of the infant seat). Figure 6.24 shows another case where the occupant was moving forward and was not in the standard seating position. The results for this case can be improved in two ways: (i) through

additional training with occupants in a broader range of positions, and (ii) using the contextual processing defined earlier for processing the historical stream of occupant classifications.

Table 6.7: Confusion matrix for the plus-L-minus-R forward sequential search blob combining on the two-class suppression problem.

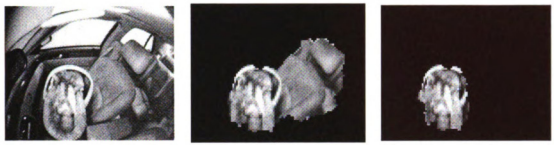
	Classified as Infant	**Classified as Adult**
True Infant	1631	166
True Adult	19	189

Table 6.8: Percentage correct classification for the plus-L-minus-R forward sequential search blob combining on the two-class suppression problem.

	Classified as Infant	**Classified as Adult**
True Infant	90.8%	9.2%
True Adult	9.1%	90.9%

(a) (b) (c)

Figure 6.17: Example of correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.



(a) (b) (c)

Figure 6.18: Example of correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.19: Example of correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.20: Example of correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.21: Example of correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.22: Example of correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.23: Example of incorrect segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

(a) (b) (c)

Figure 6.24: Example of incorrect segmentation, (a) raw image, (b) preliminary segmentation, and (c) final forward sequential search segmentation output.

6.6.3 Results for Random Mutation Blob Combining

The classification results using the random mutation method of blob combining can be found in the form of a confusion matrix in Table 6.9 and the resultant probability of correct classification can be found in Table 6.10. Note the overall accuracy of the system is roughly 75%. Some examples of the correct segmentations performed by the random mutation algorithm for the infant and adult classes are provided in Figure 6.25 through Figure 6.28. Likewise, some of the incorrect segmentations are provided in Figure 6.29 and Figure 6.30.

Table 6.9: Confusion matrix for random mutation blob combining on the two-class suppression problem.

	Classified as Infant	**Classified as Adult**
True Infant	1307	490
True Adult	49	159

Table 6.10: Percentage correct classification for random mutation blob combining on the two-class suppression problem.

	Classified as Infant	**Classified as Adult**
True Infant	72.7%	27.3%
True Adult	23.6%	76.4%

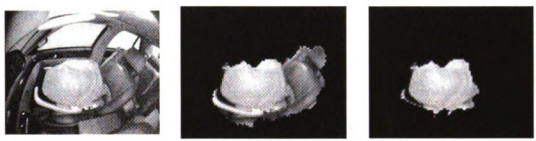
Clearly, the results for the random mutation algorithm are not as encouraging as the results for the FSS algorithm for blob combining. When we refer to the mis-classified images, we see that there are many highly random and disconnected results that were not present in the FSS results. While the random mutation worked nearly as well as the FSS for feature selection, we see a significant reduction in performance here.

We believe it is because in blob selection, the inclusion or exclusion of any individual blob can have a dramatic effect on the resultant classification. This makes blob combining more sensitive to random changes. In feature selection, on the other hand, inclusion of any single feature does not have a dramatic effect on the classification accuracy. Recall, Figure 4.3 showed the ranked Mann-Whitney Z-statistics. In this figure, there was a very gradual and monotonic decrease in feature discriminability versus feature rank, which reinforces this belief that the impact of any single feature is not dramatic.

The interaction between the blobs is more critical for blob combining than for the feature selection. In blob combining there is a strong correlation in the blobs due to their relative spatial location that may not be found with the random search method, while the more methodical FSS algorithm is able to discover the proper combination.

(a) (b) (c)

Figure 6.25: Example of correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.



(a) (b) (c)

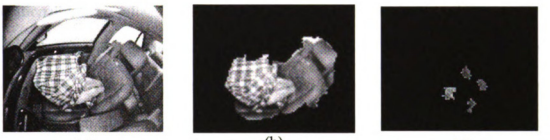
Figure 6.26: Example of correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.

(a) (b) (c)

Figure 6.27: Example of correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.

(a) (b) (c)

Figure 6.28: Example of correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.



(a) (b) (c)

Figure 6.29: Example of in-correct infant segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.

(a) (b) (c)

Figure 6.30: Example of in-correct adult segmentation, (a) raw image, (b) preliminary segmentation, and (c) final random mutation blob combining segmentation output.

6.7 Summary

We have developed a unique mechanism for overcoming the semantic gap present in performing unsupervised image segmentation. We have adapted the wrapper formalism from feature selection to image segmentation. Our new approach uniquely integrates the classification and the segmentation processes. Consequently, the particular object class being considered now guides the segmentation. Additionally, the resultant classification accuracy within that class provides us with a metric for determining the best segmentation for that class. Our approach provides two benefits to image segmentation: (i) it provides the context needed to support segmentation in complex background environments, and (ii) it provides a measure for the quality of the segmentation based on the resultant classification accuracy.

The results for our airbag suppression application show that the approach is very robust to background illumination and clutter, and also to the natural variations in the shape and grayscale distribution of the occupant. The algorithm provided greater than 90 % correct classification on the two-class suppression problem using a test database of nearly 2,000 images. Additionally, the resultant segmentations are of comparable quality to hand segmentations, in many instances. Our results demonstrate the power of combining the task of image segmentation with object classification. As part of our future work, we will test this algorithm on the four-class problem as well.

We showed that using the plus-L-minus-R forward sequential search method adapted to blob combining provided the best results. The random mutation was

second and the gradient descent performed the worst. There is a potential for future work in this area for improving the results of random methods of blob combination through the use of genetic algorithms.

Overall, our wrapper-based segmentation algorithm provides the user a considerable flexibility in implementing a solution that best matches his application, by providing:

1) the ability to use any reasonable image labeling algorithm,
2) a large number of existing feature selection algorithms that can be adapted to the blob combining application, and
3) the ability to use the classification algorithm that is best suited for the user's particular application.

Chapter 7.
Segmentation Methods for Image Sequences

The human motion tracking and prediction problem consists of three key processing steps shown in Figure 7.1. The segmentation stage extracts the human from the background scene. Segmentation for image sequences has two key differences from single frame segmentation for classification: (i) the motion of the object of interest can be used as an additional cue for segmentation that is not available in individual frames, and (ii) for many applications, such as the current airbag suppression application, the segmentation of image sequences must be accomplished at a considerably faster rate than for single frame segmentation. There are numerous approaches to motion-based segmentation, and two of the more common paradigms are: (i) template matching and (ii) optical flow [127] [128][129][141].

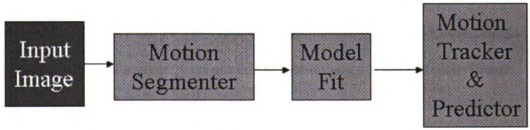


Figure 7.1: Occupant tracker processing.

Template-based segmentation does not rely on the actual motion in the image, but rather attempts to find a known template of the object being tracked in successive

image frames. After the detection of a template in a particular frame, it is common to update the shape of the template to provide some robustness to transformations of the object shape due to 3-D motions. There are two primary benefits to template matching: (i) when the object stops moving it is still detected in the image, and (ii) template matching provides very accurate estimates of the boundary of the object being tracked. There are also three drawbacks to template matching: (i) they are prone to error when the object is highly textured or the background is highly cluttered, (ii) the template must be initialized with a known or presumed initial template, and (iii) template matching has difficulty with large amounts of motion between frames, or large amounts of motion perpendicular to the image plane [141].

Optical flow methods specifically rely on the motion of the object relative to the background in order to segment the object. Consequently, optical flow techniques are able to estimate the motion of regions of the image quite accurately, but do not, necessarily, provide an accurate boundary of the object [141]. The key benefits of the optical flow techniques are: (i) they do not require any initialization, (ii) they are not sensitive to background clutter or object texture, and (iii) they can distinguish multiple motions simultaneously. The drawbacks of optical flow processing are: (i) they traditionally rely on the principle of image constancy; namely, any changes in the intensity level of a given pixel are due to motion and not illumination changes, (ii) if the object stops moving, or if the object is moving at the same relative speed as the background, then it becomes indistinguishable from the background [133][134][135].

While the image constancy constraint has been somewhat relaxed in recent implementations, illumination changes can still be a challenge [133][134][135]. For

the airbag suppression application, lack of relative motion is typically not a problem, particularly during a pre-crash braking event. In these events, the occupant is moving in the opposite direction from the objects outside the vehicle (the car is moving forward so the background is moving rearward), while the occupant is being accelerated forward. Since there is no clear advantage of one method over the other, we will investigate both template matching and optical flow.

7.1 Template-based Motion Segmentation

The general processing flow for a template-based motion segmentation algorithm is provided in Figure 7.2. We will address the three key processing stages for template-based motion segmentation, namely:

1) template initialization,
2) template matching, and
3) template updating.

The methods for template matching were defined in Chapter 3. A taxonomy of these methods can be defined to include:

1) 2-D template,
2) line template, and
3) point-set template.

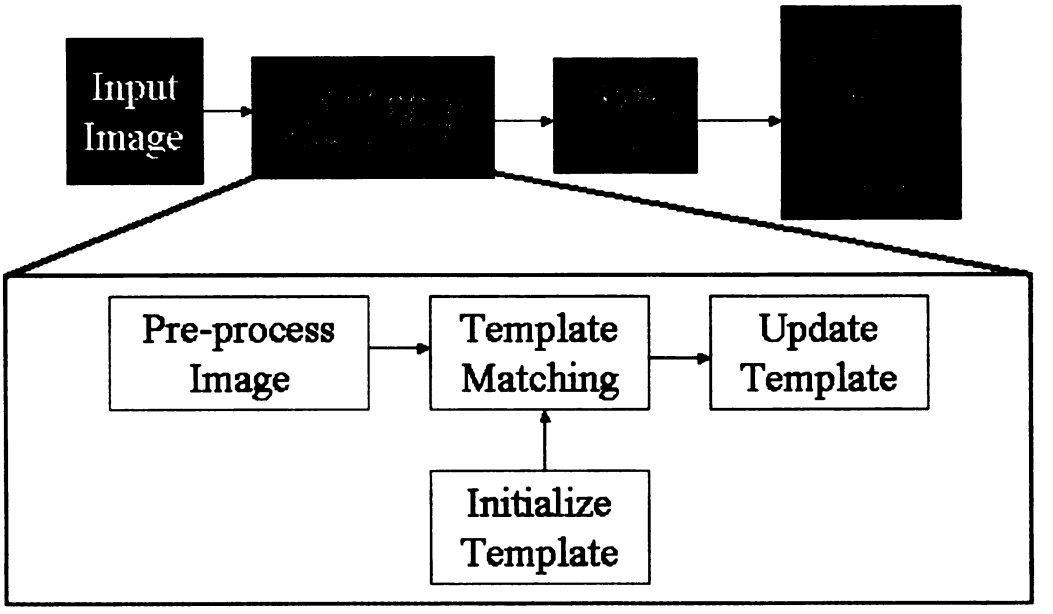


Figure 7.2: Template-based motion segmentation processing.

2-D templates attempt to perform a 2-D correlation between a pattern and the image to try to find the current location of the object of interest. Refer to Chapter 3 for the mathematical details of correlation processing. For most real-time applications, the use of 2-D templates for motion tracking is not feasible, due to the intense processing load associated with 2-D correlation. Additionally, template updating can be computationally intensive, since a transformation must be applied in a pixel-wise manner on the entire stored template. Hence we will not consider it for our application.

Line-based template matching represents the object as a collection of line segments, and attempts to track these segments over time. Since the processing attempts to only find lines, it is natural in this method to perform edge detection as a preprocessing stage. The resultant edge images are then analyzed to detect and track the line segments. As mentioned in Chapter 3, the most popular method for this

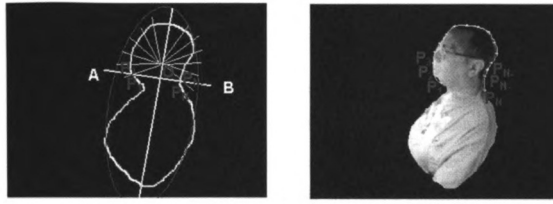
processing is the Hough transform. For the airbag suppression application, where we are tracking humans, a line segment-based representation is neither optimal, nor natural, since the human contour cannot be easily modeled with a closed-for expression. Consequently, we will not consider line templates, either.

Point-based matching reduces the image to a subset of points, and then tracks this set of points and their locations relative to each other over time. To reduce the image to points, it is common to perform edge detection or corner detection as pre-processing steps. Corners are more reliably tracked in two dimensions without any ambiguity, but for the airbag suppression problem the human occupant does not necessarily contain corners [156].

Due to the simplicity of the representation, and the ability to easily reduce the number of points in the template, point-based template matching will be used for the occupant motion estimation processing. The first stage of processing is the image pre-processing. For our application, the pre-processing will consist of the following steps:

1) compute the edge image and the difference image with the last frame,
2) combine them to eliminate edges that are not moving,
3) compute bounding ellipse of edge-difference image results, and
4) generate spokes every θ degrees, and record points of intersection between the template and the input image.

The outputs of steps (3) and (4) can be seen in Figure 7.3.



(a) (b)

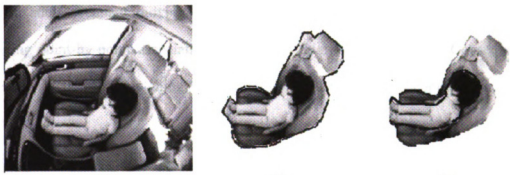
Figure 7.3: Mechanism for defining points to be tracked for template-based motion segmentation, (a) spoke arrangement, and (b) resultant points.

7.1.1 Template Initialization

The first step in template-based motion segmentation is template initialization. This is particularly difficult for the airbag suppression application, because we cannot use a pre-stored common object template, since many different occupants may enter the vehicle, ranging from children to very large adults. One possible solution is to use the segmentation from the classification processing as the initial template. Unfortunately, since the occupant and the seat are segmented as a single entity, the template would include more than just the occupant. Subsequent template matching would be corrupted by the inclusion of the seat and would lead to extremely poor contour matches that would not properly represent the occupant. This would also have a negative impact on the resultant track estimations and ASZ intrusion predictions.

This initialization problem can be overcome, however, through the use of the wrapper-based segmentation method we have defined in Chapter 6. In our wrapper-

based segmentation, the occupant is successfully removed from the seat, and this segmentation can then serve as the source of the initial template. Note that without the development of the wrapper-based segmentation algorithm, the use of templates for motion estimation would have only been feasible for adults, which implies the 4-class problem. Recall from Chapter 5, however, that the results for the 4-class classification did not appear to be generalizable, and that the 2-class system of infants versus adults was much more robust. This 2-class approach, however, requires tracking children.



(a) (b) (c)

Figure 7.4: Demonstration of difficulty of initializing templates from classifier-based segmentation, (a) original image, (b) segmented image with resultant initial template contour, and (c) true contour required for tracking occupant.

7.1.2 Template Matching

There are two common mechanisms for point-set matching: (i) deformable templates and (ii) Hausdorff distance. In each of these approaches a set of points from the template are matched to a set of points from the image. The position of the best match is then considered the new object location. The critical difference

between the two methods is that deformable templates perform point-by-point matching, while the Hausdorff distance performs set-level matching, which does not require point-by-point correspondence. The limitation of point-by-point matching is that if there are gaps in the image contour, then the algorithm will fail to find proper point matches and may not converge properly. Gaps in the contour are most often caused by object occlusions and illumination variations, both of which are very common in our application. Occlusions occur as the occupant turns her head or moves her hands towards her head, and illumination variations are constantly present in this environment. Figure 7.5 shows a typical sequence (roughly every 60 frames) of an adult occupant moving within the vehicle. Note the obvious difficulty in finding point-by-point associations throughout this image sequence. Consequently, we will use point-set matching (i.e. the Hausdorff distance) for our application.

Figure 7.5: Demonstration of difficulties in finding point-to-point matching in occupant template.

Recall, the Hausdorff distance metric does not require the same number of points in the template as in the input image. The Hausdorff distance is defined for a fixed set of points in set A relative to a fixed set of points in set B. The Hausdorff distance can also be used to find the best matching set of points in set A relative to a transformed set of points in set B by defining [43]:

$$H_G(A,B) = \min_{g \in G} H(A, g \circ B), \qquad (7.1)$$

where G is a group of transformations. The group G is the set of all translations if $\|\cdot\|$ in equation (7.1) is any norm, and the group G is the set of all rigid motions if $\|\cdot\|$ is the Euclidean norm [43].

For applications where the motion remains in the image plane, translation and rotation estimates are sufficient to match the template. For applications with weak perspective, affine transformations can adequately model the changes in the template [141]. A 2-D affine transformation is defined by [140]:

$$T(x,y) = \begin{bmatrix} x \cdot s_{1,1} & y \cdot s_{1,2} & d_x \\ x \cdot s_{2,1} & y \cdot s_{2,2} & d_y \end{bmatrix}, \qquad (7.2)$$

where d_x and d_y define the 2-dimensional translations, and the sub-matrix $\begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{bmatrix}$ defines a rotation if it is ortho-normal, otherwise it defines a combination of rotation, scale, and skew in the image.

In the airbag suppression application the deformations in the object template are due to the motions of occupant within the vehicle relative to the camera, as shown in Figure 7.6, and even more dramatically as shown in Figure 7.5. These occupant motions can sometimes result in dramatic changes in the shape of the occupant due to the close proximity of the occupant to the camera, which causes greater than para-perspective effects. For simplicity, however, and assuming a fast enough camera

update rate, the assumption of para-perspective is manageable. Note, however, that this still calls for us to find the best template match by attempting to match a template that is transformed by the matrix in Equation (7.2) for each camera frame. Typically the template matching is performed by: (i) defining the possible variation limitations in each of the parameters, and (ii) iterating through all combinations of these variations, which can be computationally prohibitive for a real-time system. Also, as the number of parameters, and the amount of allowed deformation in the images increases, the probability of false matches increases.

(a) (b) (c)

Figure 7.6: Typical adult motion in a vehicle, (a) seated normally, (b) sitting up and turning slightly, and (c) leaning far forward.

In light of our real-time requirements, and the increased chances of erroneous matching, we will limit the motion variations to be the set of rigid translations and rotations. This limitation greatly reduces our search space to only two degrees of freedom, rather than six. At each time instance, we allow the template to be translated by M pixels horizontally and rotated by some angle, θ. Since the occupant can only move a finite distance between frames, we set the maximum range for each of these values to be 5-10 pixels in translation and 2-5 degrees in rotation. Once the best match of the template is defined, the new template is updated, and then the point

set that comprises the current template is sent to the ellipse fitting stage of the processing.

7.1.3 Template Updating

Once the matching is performed, the template must be updated to account for any deformations in the object over time. For most applications, it is not sufficient to simply replace the most recent template with the transformed version of the last template, since we know our model of deformation is only an approximation. If we did not update the template with new pixel information, the contour of the occupant would eventually deform to a shape that can no longer be matched. Therefore, it is preferable to implement template updating as a process that integrates the new raw occupant contour information with transformed version of the last template used. The issue in integrating raw contour information is that the detected contours are often very noisy and may also be contaminated with extraneous motions as shown in Figure 7.7. For our initial implementation of template matching we will use the simple replacement strategy for the template updating. If the initial results for the template matching approach are not favorable, we will then investigate alternative template updating strategies.

Figure 7.7: Demonstration of difficulty in maintaining a template without controlling allowable deformations, (a) person seated normally, and (b) same person beginning to put on a sweater.

7.2 Optical Flow-based Motion Segmentation

The general processing flow for an optical flow-based motion segmentation algorithm is provided in Figure 7.8. The three methods we will investigate for computing the optical flows are: (i) phase-based, (ii) gradient-based, and (iii) correlation-based. Our results will confirm what other researchers have found that none of these methods are universally better [127][128][129]. Consequently, we will also investigate fusion methods for combining the results of each of these algorithms. The method for clustering the optical flow will then be discussed, as well. Lastly, we will describe an alternative method for motion estimation based on mutual information, and we will show how it can be applied to the airbag suppression problem.

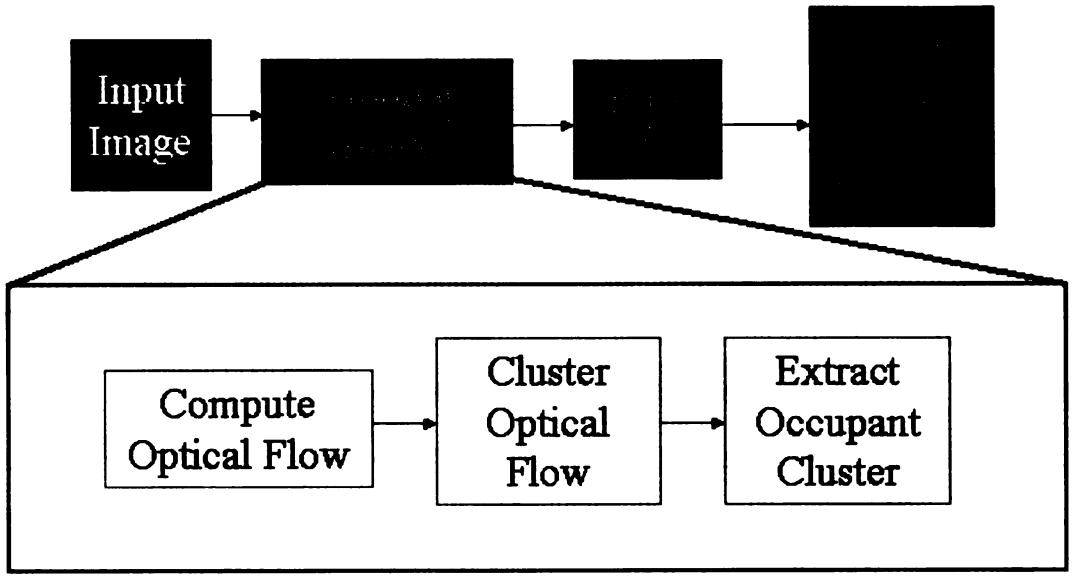


Figure 7.8: Processing for optical flow-based motion segmentation.

7.2.1 Compute Optical Flow

The first stage of the processing is to compute the optical flow fields. The flow fields are the U (horizontal) and V (vertical) component velocity fields. The velocity vector in terms of the U and V component is computed for every pixel in the input image, based on some number of previous input image frames.

A Phase-based Methods

While many optical flow estimation techniques have been proposed, the phase-based method has been shown to typically produce more accurate results [127][129] [130]. Fleet and Jepson defined the component velocity in terms of the gradient of the phase output of individual velocity-tuned linear filters (Gabor filters)

[130]. They used a 3-dimensional Gabor filter (two spatial dimensions and one temporal dimension), as defined by:

$$Gabor(\mathbf{x},t;\mathbf{k}_0,\omega_0,C) = \exp\left(i(\mathbf{x},t)\cdot(\mathbf{k}_0,\omega_0)\right)G(\mathbf{x},t;C), \qquad (7.3)$$

where $G(\mathbf{x},t;C)$ is a 3-dimensional Gaussian kernel with covariance C. These filters are used to detect motion in the input image sequence based on the scale, speed and orientation of the motion. The filtered result, R(x,t), is complex valued and defined as [130]:

$$R(x,t) = \rho(x,t)e^{i\phi(x,t)}, \qquad (7.4)$$

where $\rho(x,t)$ and $\phi(x,t)$ represent the amplitude and phase component of *R(x,t)*. The local phase gradient is then measured from the output of the individual filters to estimate the component velocity v_n [132]:

$$v_n = -\phi_t(x,t)\frac{\nabla\phi(x,t)}{\|\nabla\phi(x,t)\|^2}, \qquad (7.5)$$

where

$$\nabla\phi(x,t) = \frac{\mathrm{Im}[R^*(x,t)\nabla R(x,t)]}{|R(x,t)|^2} \qquad (7.6)$$

is the spatial derivative, and where $\phi_t(x,t)$ is the temporal derivative of the phase. Also, $R^*(x,t)$ denotes the complex conjugate of *R(x,t)*, $\nabla R(x,t)$ represents the gradient of *R(x,t)*, and Im[·] extracts the imaginary part of the complex result.

The use of phase information can generate more robust estimates of component velocity than amplitude information, as shown in Figure 7.9 [130]. Notice the near-perfect reproduction of the phase contours in this image relative to the original image in Figure 7.9 (f) and, then compare them to the poorly reproduced amplitude contours in Figure 7.9 (e). Instead of spatio-temporally filtering the images with 3-dimensional Gabor filters, Gautama and Hulle use a bank of 2-dimensional Gabor filters to only spatially filter the images [132]. It is a product of a Gaussian kernel and a complex plane wave. Its spatial form is expressed as [132].

$$Gabor(\mathbf{x};\mathbf{k}_0,C) = \exp(i\mathbf{x}\cdot\mathbf{k}_0)G(\mathbf{x};C). \qquad (7.7)$$

The Gabor filters are used to estimate the phase responses of each image in the image sequence individually. Expanding a signal using this basis set of directionally and spatially tuned filters provides a localized frequency description of each image in the sequence. Figure 7.10 shows Gabor filters with four different orientations and three different spatial scales [132].

The output of the Gabor filtering is a phase response at each point in the image for each Gabor filter at each time in the image sequence. From this set of phase histories for each point in the image, a least squares line fit through this sequence of phases over time is used to estimate the phase gradient for each Gabor filter, defined as: $\phi_{t,i}(\mathbf{x})$ [132]. Recall that the gradient of the phase history is the frequency of a signal, and, therefore, it defines the velocity at that point. This temporal gradient is now computed to estimate the velocity component, $\mathbf{v}_{c,i}(\mathbf{x})$, in

the direction orthogonal to the corresponding Gabor filter orientation, according to [132]:

$$\mathbf{v}_{c,i}(\mathbf{x}) = -\frac{\phi_{t,i}(\mathbf{x})}{2\pi\left(f_x^2 + f_y^2\right)}\left(f_x, f_y\right), \tag{7.8}$$

where f_x and f_y are the center frequencies of the Gabor filter.

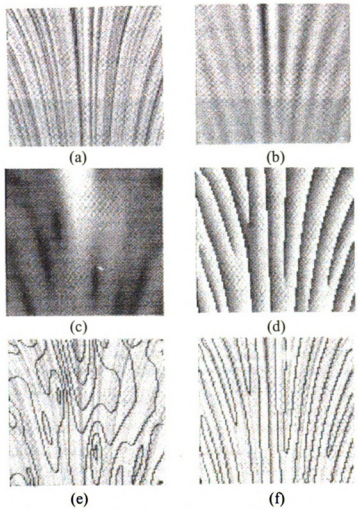
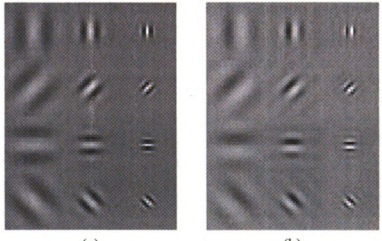


Figure 7.9: Demonstration of improved velocity detection, (a) time varying 1-D 'image' with additive Gaussian noise (time on the vertical axis), (b) tuned Gabor filter, (c) amplitude response of filter, (d) phase response of filter, (e) iso-amplitude lines, and (f) iso-phase lines [130].

The full velocity is estimated from these component velocities at each spatial location in the image. They also estimate the phase non-linearity, which can be used as a confidence measure for rejecting unreliable component velocity estimates [132]. Each component velocity, $\boldsymbol{v}_i$, constrains the full velocity vector, $\boldsymbol{v}$, to lie on a constraint line, L_i, in the multi-dimensional velocity space, with orientation orthogonal to $\boldsymbol{v}_i$ [138]. This line is defined as:

$$L_i \propto \frac{\mathbf{v} \cdot \mathbf{v}_i}{\|\mathbf{v}_i\|}. \qquad (7.9)$$



(a) (b)

Figure 7.10: Gabor filters with 3 scales and 4 orientations, (a) real part, and (b) imaginary part [132].

The complete velocity vector is generated by the intersection of the corresponding constraint lines. Gautama and Hulle formulated the constraint line clustering into a goal-programming problem, which is solved by the recurrent neural network [132]. Figure 7.11 shows the component velocity images generated from each of the directional Gabor filters. The resultant clustering of the component

velocities is reduced into traditional U and V (horizontal and vertical) velocity components as shown in Figure 7.12.

The major drawback of the phase-based method is the number of image frames that are needed to adequately estimate the phase gradient, $\phi_{t,i}(\mathbf{x})$, since it is computed by performing a least squares fit for each image point over time. We found that an image phase history of between 10 and 20 frames is required for reliable estimation, and this number of frames is similar to what has been used by Gautama and Hulle [132]. This deep history buffer creates two significant problems for our airbag suppression application: (i) the significant time delay resulting from such a deep image buffer, and (ii) the amount of system memory required to store all of these phase images.

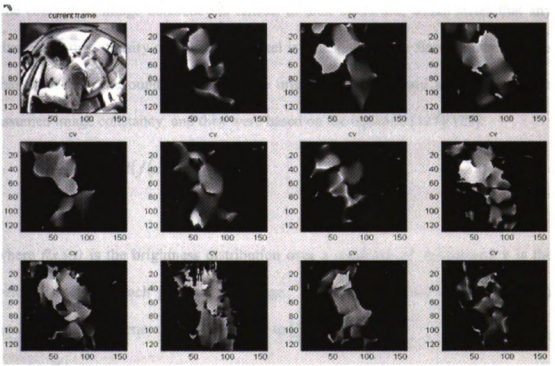
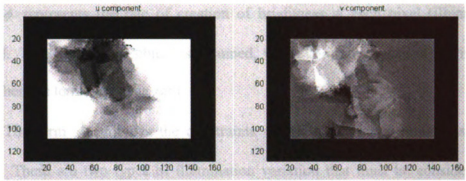


Figure 7.11: Resultant component velocities for various directions of Gabor filters.



(a) (b)

Figure 7.12: Combined motion from Gabor filters, (a) U direction, and (b) V direction.

B Gradient Methods

While the phase-based methods can provide superior accuracy in estimating the actual motion values, they tend to be sensitive to lighting variations [134][135]. The concept of image constancy is critical to their operation, which states that any changes in the intensity of a particular pixel is due solely to the flow of intensity into that pixel from surrounding pixels. The original gradient-based methods had also assumed image constancy, and they were based on the equation [127][129]:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot grad\left(f(x,y,t)\right) = 0\,, \tag{7.10}$$

where *f(x,y,t)* is the brightness distribution over a sequence of images and **v** is the velocity vector at each point in the image. This equation has been extended to handle lighting by generalizing this model to [134][135]:

$$\frac{\partial f}{\partial t} = -f(x,y,t) \cdot div(\mathbf{v}) - \mathbf{v} \cdot grad(f(x,y,t)) + \phi\,, \tag{7.11}$$

where ϕ represents the rate of creation of brightness at each pixel (illumination change). If a rigid body object is assumed, where the motion lies in the imaging plane, then the term $div(\mathbf{v})$ is zero.

The term ϕ provides the constraints on the illumination variations in the image. There are two types of illumination variation that must be considered: (i) changes in illumination due to changes in reflectance or diffuse shadowing (modeled as a multiplicative factor), and (ii) changes in illumination due to illumination highlighting (modeled as an additive factor) [134]. The term ϕ is then [134]:

$$\phi = f \cdot \frac{\partial m}{\partial t} + \frac{\partial c}{\partial t}, \tag{7.12}$$

where $\frac{\partial m}{\partial t}$ corresponds to the change in reflectance and $\frac{\partial c}{\partial t}$ corresponds to the illumination highlighting [134]. The actual motion and illumination field values are then computed by solving the following least squares problem [134]:

$$\sum_W \begin{bmatrix} I_x^2 & I_x I_y & -I_x I & -I_x \\ I_x I_y & I_y^2 & -I_y I & -I_y \\ -I_x I & -I_y I & I^2 & I \\ -I_x & -I_y & I & 1 \end{bmatrix} \cdot \begin{bmatrix} \delta_x \\ \delta_y \\ \delta m \\ \delta c \end{bmatrix} = \sum_W \begin{bmatrix} -I_x I_t \\ -I_y I_t \\ I_t I \\ I_t \end{bmatrix}, \tag{7.13}$$

where the summation is over a local window W about each pixel in the image. The explicit modeling of the effects of illumination can dramatically improve the motion estimation, as can be seen in Figure 7.13. Note, particularly in Figure 7.13 (e), there

is significantly more erroneous motion caused by illumination changes on the occupants legs, compared to Figure 7.13 (f) where these illumination effects were correctly modeled, and only the true motion is left.

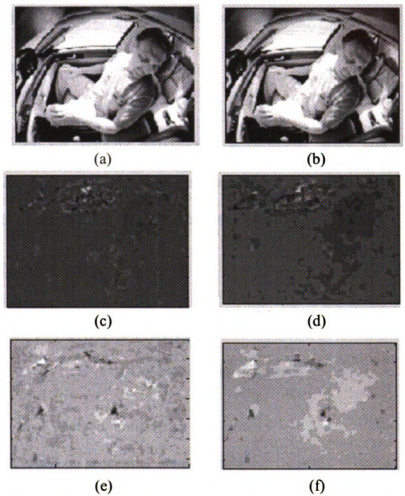


Figure 7.13: Standard versus extended gradient optical flow, (a) first image, (b) second image, (c) U-component for the basic gradient methods, (d) U-component for the illumination-enhanced gradient method, (e) V-component for the basic gradient method, and (f) V-component for the illumination-enhanced gradient method.

This improved gradient approach also has a mechanism for inferring the quality of the motion estimations based on the relative contributions of motion-based

optical flow and illumination based optical flow. The motion-based contribution is defined by [134]:

$$opt_{motion} = \delta x \cdot I_x + \delta y \cdot I_y . \quad (7.14)$$

The contributions due to illumination changes are defined by:

$$opt_{illum} = \delta m \cdot I + \delta c , \quad (7.15)$$

where $\delta x, \delta y$ are the velocity estimates at a particular pixel, and $\delta m, \delta c$ are the multiplicative and additive illumination contributions at that pixel. Regions of reliable optical flow occur in regions where the following ratio is large:

$$quality = \frac{opt_{motion}}{opt_{illum}} . \quad (7.16)$$

C Correlation-based Method

The final method considered for performing optical flow-based motion estimation is based on sequential image intensity correlation. The correlation-based method is less sensitive to the presence or absence of significant image features than gradient methods, where significant image features are defined as regions of high gradients. The correlation-based method is less sensitive to these discrete points since it operates on a patch in the image at a time [128]. Additionally, as with the gradient methods, it can be formulated with additional constraints that will support

the two types of illumination effects defined earlier. Let *f(x,y,t)* and *f(x,y,t-1)* denote two sequential images, then the relationship between them can be modeled as [128]:

$$m_3 \cdot f(x,y,t) + m_4 = f(x + m_1, y + m_2, t-1), \qquad (7.17)$$

where m_1 and m_2 denote the translations in x and y, and m_3 and m_4 represent the additive and multiplicative components of the illumination variation defined earlier. The correlation between the images, *f(x,y,t)* and *f(x,y,t-1)*, is defined as [140]:

$$\int_D f(x + m_1, y + m_2, t-1) \cdot (m_3 \cdot f(x,y,t) + m_4) dx, \qquad (7.18)$$

where D is a region of the image. Note that we can compute the minimum of the mean squared error (MSE) instead of the maximum of the correlation, since the MSE calculation includes the correlation as a component, and it is easier to compute. So, the target function to be minimized is defined as [128][140]:

$$E(\boldsymbol{m}) = \int_D (m_3 \cdot f(x,y,t) + m_4 - f(x + m_1, y + m_2, t-1))^2 dx, \qquad (7.19)$$

where $\boldsymbol{m} = (m_1, m_2, m_3, m_4)^T$. Using a first-order truncated Taylor series expansion, we get the approximated form of $E(\boldsymbol{m})$ [133]:

$$\begin{aligned} E(\boldsymbol{m}) \approx \int_D (m_3 \cdot f(x,y,t) + m_4 - \\ [f(x,y,t) + m_1 \cdot f_x(x,y,t) + m_2 \cdot f_y(x,y,t) - f_t(x,y,t)])^2 dx \end{aligned} \qquad (7.20)$$

$$= \int_D ([f_t(x,y,t) - f(x,y,t)] - [m_1 \cdot f_x(x,y,t) + m_2 \cdot f_y(x,y,t) - m_3 \cdot f(x,y,t) - m_4])^2 dx \quad (7.21)$$

$$= \int_D (k - \boldsymbol{c}^T \cdot \boldsymbol{m})^2 dx, \quad (7.22)$$

where

$$k = f_t(x,y,t) - f(x,y,t), \quad (7.23)$$

and

$$\boldsymbol{c} = [f_x(x,y,t),\ f_y(x,y,t),\ -f(x,y,t),\ -1]^T. \quad (7.24)$$

The discrete form of $E(\boldsymbol{m})$ is [133]:

$$E(\boldsymbol{m}) = \sum_{x,y \in D} (k - \boldsymbol{c}^T \cdot \boldsymbol{m})^2. \quad (7.25)$$

Equation (7.21) can now be minimized by differentiating with respect to $\boldsymbol{m}$, yielding [133]:

$$\frac{dE(\boldsymbol{m})}{d\boldsymbol{m}} = \sum_{x,y \in D} -2 \cdot \boldsymbol{c} \cdot (k - \boldsymbol{c}^T \cdot \boldsymbol{m}). \quad (7.26)$$

Setting this result equal to zero, and solving for $\boldsymbol{m}$, yields the MSE estimate for $\boldsymbol{m}$ [133]:

$$\boldsymbol{m} = \left[\sum_{x,y \in D} \boldsymbol{c} \cdot \boldsymbol{c}^T \right]^{-1} \cdot \left[\sum_{x,y \in D} \boldsymbol{c} \cdot k \right]. \quad (7.27)$$

The size of the neighborhood, D, must be chosen large enough to guarantee the first term is invertible, but not too large as to break the assumption of the coherence of $\boldsymbol{m}$ over the region [133]. Again, we must note that Equation (7.19) only models simple translations, not even rotations. The underlying assumption made in Equation (7.19) is that complex transformations of an object may appear as simple translations, when only small regions on the object are considered individually. Note, however, that regions may experience more complicated distortions in the sequential images as we had seen earlier in Figure 7.5 through Figure 7.7. Sutton [138] and Kalivas [140] proposed alternate correlation methods to account for deformations of the objects to allow for more complex motions to be modeled, but the impact is that the number of parameters, m, to estimate will increase accordingly, as we saw when modeling more complex transformations of the templates in the Hausdorff template matching.

7.2.2 Motion Clustering of the Optical Flow Fields

Motion clustering provides estimates of the extent of the regions in the motion fields that correspond to possible objects. Note that clustering is not required in the template-based motion segmentation approaches, since the template updating automatically provides an estimate of the current object boundary. In most real-world motion estimation problems there are often multiple moving objects in the image. This is clearly the case in the automotive airbag suppression application, where there is often motion outside the window, as well as the occupant motion. This causes

discontinuities in the motion field, as the multiple objects move past each other in space, and generate a distinct set of motions in the imaging plane [136]. Each of the motions can be considered a distinct cluster within the motion field image. The probability distribution for an optical flow motion estimate can be well modeled by a Gaussian distribution [142]. Therefore, it is reasonable to model the motion field as a mixture of Gaussians components. Recall, the EM algorithm is very effective in determining the parameters of a mixture of Gaussians, and will be used for this application [45] [136].

We use the U and V velocity components as a 2-dimensional space, within which we will perform clustering using EM. Jepson and Black [136] have also used EM for motion estimation, however, they apply EM on a set of linear constraints on the phase space, rather than directly on the U and V component motion fields. We model the actual velocity estimate for each pixel as a being associated with a Gaussian component, and then the multiple motions are modeled as a mixture of these Gaussian components.

7.2.3 Optical Flow Fusion

Each of the optical flow methods may have an advantage in a particular motion environment, but none have been shown to be consistently more robust. When the added complexity of varying illumination is added, it becomes even less clear if there is a single best approach. The lack of a universally best approach implies that benefits can be derived if multiple approaches are combined or fused

together. This may provide robust motion segmentation, since each algorithm is more reliable for different motion conditions.

There are three key design decisions that must be addressed in any data fusion problem: (i) at what level of abstraction will the data be combined, (ii) what mechanism will be used to combine the information, and (iii) how can the complexity of the problem be managed [102]. For motion segmentation, there is a range of data abstractions for performing fusion including [102]:

1) pixel-level fusion of the velocity information,
2) region-level fusion of regions of common velocity, and
3) object-level fusion, where the motion estimates are abstracted to objects and then combined.

We will concentrate on combining the information at the pixel level of the directional flow fields from each algorithm, where the data is at our lowest level of abstraction. Generally, operating at lower levels of abstraction implies that fewer constraints have been placed on the outputs and fewer assumptions have been made regarding the data, thereby minimizing the error propagation. Since we have successfully applied EM to the motion field from a single motion estimation algorithm, it is natural then to apply EM to combinations of the outputs from the various motion segmentation algorithms.

In the problem of motion field fusion, managing the complexity of the problem is important, since the collection of motion fields from the various segmentation techniques generates a high-dimensional data space. Since each motion segmentation algorithm generates at least a 2-dimensional space containing velocity vectors for each pixel in the image, the minimum data space is 6-dimensional. There

are three issues in using such a high dimensional space for fusion with the EM algorithm:

1) the number of parameters that must be estimated (when using EM, the number of free parameters increases as d^2, where d is the data dimensionality),
2) the real-time processing throughput and memory requirements for such a high dimensional space consisting of such a large number of data points, and
3) the quality of information from all the motion planes is not equal and some planes may provide no useful information.

Issue (3) can be addressed by applying an information theoretic approach to compressing the optical flow data, which may give insight into the true information content, while minimizing the effects of the noise. Figure 7.14 (a) and (b) show the U and V component images. Figure 7.14 (c) and (d) show the histograms for the U and V images, respectively. These histograms show that there is considerably more information (dispersion) in the U component. Therefore, the information in the U-component image should be considered more significant than the information in the V-component image.

We will use principal components analysis (PCA), since our goal is to determine the directions in the space of motion fields that provide the greatest information [45]. Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., x_N)$ represent the $n \times N$ data matrix, where each $\mathbf{x}_i$ is an image vector of dimension n, and N is the number of images in the

training set. The mean vector of the training samples, $\boldsymbol{\mu} = \sum_{i=1}^{N} \mathbf{x}_i$, is subtracted from each sample vector. PCA is a linear transformation from the original $n \times N$ data space to a $d \times N$ data space, with $d < n$, according to [45]:

$$\mathbf{Y} = \mathbf{W}_{PCA}^{T} \mathbf{X}, \tag{7.28}$$

where $\mathbf{W}_{PCA}$ is the transformation matrix. This matrix $\mathbf{W}_{PCA}$ holds the basis vectors of the PCA subspace. These basis vectors are generated from the eigenvectors of the scatter matrix, $\mathbf{S}_T$ [45]:

$$\mathbf{S}_T = \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T . \tag{7.29}$$

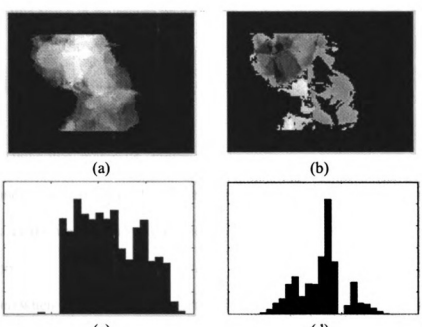


Figure 7.14: Flow fields and histograms, (a) U-component image, (b) V-component image, (c) histogram of U-component, and (d) histogram of V-component (note lack of structure beyond single Gaussian).

The transformation matrix $\mathbf{W}_{PCA}$ is composed of a subset of the eigenvectors of $\mathbf{S}_T$, corresponding to the d largest eigenvalues. After applying the projection, the n-dimensional input space is reduced to a d-dimensional subspace.

7.3 Information Theoretic Methods for Motion Segmentation

In many problems, the key to success is to find a proper space in which to interpret the incoming data stream. In pattern recognition, we define a feature space that will provide the best discrimination between the various classes [45]. In optical flow based motion estimation, the domain of analysis is the original grayscale image. The data in this space is generated by a complex interaction between the 3-D shape of the body of interest, its surface characteristics, and the illumination pattern [133][134][135]. Unfortunately, in our airbag suppression application, none of these underlying characteristics are known. In other words, we have unknown objects, covered in unknown materials, moving in an unknown manner, and in unknown lighting conditions. Consequently, there is an inherent uncertainty regarding whether the change in the image was due to motion of the object or due to changes in the illumination.

Even when the image is transformed into the edge domain, the results are still sensitive to lighting, since contrast is dependent on the lighting levels, which impacts the ability of the system to detect edges. What is required is to define an alternative

data space that captures the meaningful elements of the imagery, while ignoring (or at least greatly reducing) the effects of illumination.

Let us look at the problem of objects moving in a scene, not as the movement of actual objects, and not as the motion of grayscale intensities, but rather as the transfer of information across the image field of view over time. Elements of information theory may provide more meaningful data representations. Information theory has recently been applied to a number of image processing and pattern recognition applications ranging from image registration to feature selection in classification [143][145][146][147]. There are two methods we have investigated for estimating optical flow using information theoretic methods: (i) entropy- based flow and (ii) mutual information- based flow.

7.3.1 Entropy-based Motion Segmentation

One standard measure from information theory is the entropy. The entropy of a signal is defined as [143] [146]:

$$H = \sum_i p_i \cdot \log \frac{1}{p_i}. \qquad (7.30)$$

The entropy of a signal can have three different meanings [146]:

1) amount of information an event provides,
2) uncertainty in the outcome of an event, and
3) dispersion in the probability distribution function (visible in the histogram).

Since an additive and a multiplicative term can model the illumination effects in imagery, we must analyze the impact of these two effects on the image entropy [134][143].

Figure 7.15 (a) shows a region extracted from an image from our application, and Figure 7.15 (c) shows its histogram. Figure 7.15 (b) shows the same image region with additive illumination included, and Figure 7.15 (d) shows the resultant histogram. Note the structure of the histogram is preserved (it is merely shifted), assuming minimal image saturation. This implies that definition (3) will prove to be the most useful for motion segmentation, since the dispersions in the histograms are identical, thereby making the entropy in the two figures equal. This means that, by tracking objects in the entropy domain, the effects of additive illumination can be completely removed.

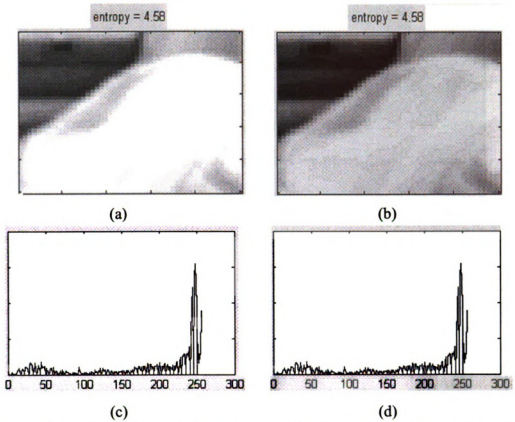


Figure 7.15: Effects of additive illumination, (a) original image, (b) modified image, (c) histogram of original image, and (d) histogram of modified image.

Figure 7.16 (b) shows the same region of the image, but now modified with multiplicative illumination effects. Notice that the general structure of the histogram is still preserved, but the total dispersion in the histogram has changed, since there is a net compression in its overall shape. This results in a change in the image entropy. While the resultant change in the histogram and its entropy is still minor, it encourages us to experiment with slightly more complicated illumination conditions to further test its robustness.

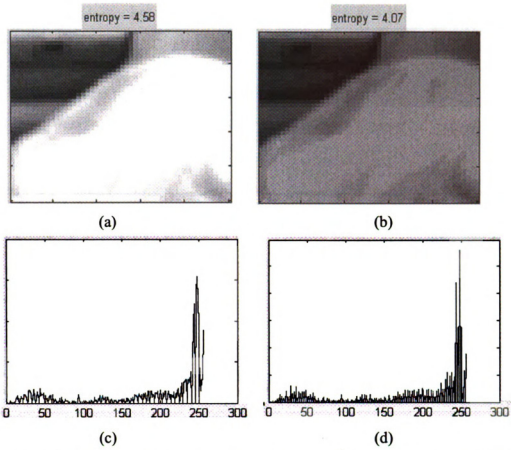


Figure 7.16: Effects of multiplicative illumination, (a) original image, (b) modified image, (c) histogram of original image, and (d) histogram of modified image.

Figure 7.17 (b) shows a more complex scenario, where multiplicative illumination occurs on only a portion of the image. This would be equivalent to a shadow cast across the subject from an object moving between the subject and the lighting source, or to different regions in the image having different coefficients of reflection due to two types of clothing materials. In this case, the effects on the structure and the dispersion of the histogram are dramatic, and there is a corresponding change in the entropy of the image, as shown in Figure 7.17 (c) and (d).

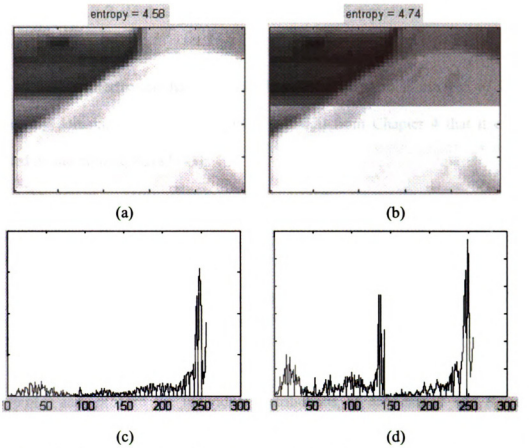


Figure 7.17: Effects of partial multiplicative illumination, (a) original image, (b) modified image, (c) histogram of original image, and (d) histogram of modified image.

The dramatic change in the histogram and the corresponding change in entropy imply that entropy alone is not a strong enough measure of information content in the image to adequately overcome the illumination effects. Consequently, a more powerful measure of the information content in an image is required. Since we are interested in the relative changes in an image from one frame to the next, the use of a comparative information metric may prove useful. The mutual information is such a measure, and will be investigated as a possible motion estimation metric.

7.3.2 Mutual Information for Motion Estimation

Mutual information has successfully been applied to a number of image processing domains [143][145][146][147]. Recall from Chapter 4 that it can be defined in one of three ways [143]:

1) $I(A;B) = H(B) - H(B|A) = H(A) - H(A|B)$,

2) $I(A;B) = H(A) + H(B) - H(A,B)$, and (7.31)

3) $I(A;B) = \sum_{a,b} p(a,b) \cdot \log\left(\frac{p(a,b)}{p(a)p(b)}\right)$.

where $p(a)$ and $p(b)$ are the individual distributions of data sets A and B, and $p(a,b)$ is the joint distribution of data sets A and B, and a is a value in dataset A and b is a value in dataset B.. Also, $H(A)$ is the entropy of data set A, and $H(A,B)$ and $H(A|B)$ are joint and conditional entropies, respectively. Notice that mutual information includes not only the relative entropy between the two signals, but also the entropy contained within each of the signals. Mutual information has the following useful properties [143][145]:

1) Symmetry: I(A,B) = I(B,A)
2) I(A,A) = H(A)
3) I(A,B) <= H(A); this implies the information that each image A contains about another image B cannot be greater than the information that image A contains about itself.
4) I(A,B) >= 0; Cannot increase uncertainty in A by knowing B

5) If A, B are independent then I(A,B) = 0

6) If A, B are Gaussian then:

$$I(A,B) = -\frac{1}{2} \cdot \log(1-\rho^2), \qquad (7.32)$$

where ρ is the standard cross-correlation between the two signals A and B.

A Phenomenology of Mutual Information in Motion-based Segmentation

For imaging applications, the mutual information is best calculated using either the second or third definition in Equation (7.29), where the joint probability density can be used to compute the joint entropy, using [143]:

$$H(A,B) = -\sum_{i,j} p(i,j) \cdot \log[p(i,j)]. \qquad (7.33)$$

As a baseline for comparison, the joint density function of an image with itself is shown in Figure 7.18. Note that when an image is compared to itself, the joint density function is a simple straight line of unit slope and intercept zero. Also note that the mutual information is equal to the entropy of the image in this case.

(a)

(b)

Figure 7.18: Phenomenology of the joint PDF, (a) incoming image, and (b) the joint density function of the image with itself.

The effects of illumination changes on the mutual information and the joint density function must now be determined. The effects on the joint density function of the three previously shown illumination conditions are provided in Figure 7.19. Figure 7.19 (b) (uniform additive illumination) shows that the joint PDF maintains the well-focused characteristics of a line, but the Y-intercept changed. In Figure 7.19 (c) (uniform multiplicative illumination) the joint PDF again maintains the well-focused characteristics of a line, but now its slope has changed. In both of these cases, the joint PDF maintains a well-focused line with no dispersion or distortion. Finally, Figure 7.19 (d) shows the effects of partial multiplicative illumination on the characteristics of the joint PDF. There are now two clearly distinguished lines in the joint PDF, one corresponding to the region in the image with ambient illumination and second line, with a different slope, corresponding to the region of the image effected by the multiplicative illumination.

Given that the effects of these illumination conditions on the mutual information are so well defined and detectable, we must now analyze the effect of the occupant motion on the mutual information. Figure 7.20 shows the effects of the

occupant's motion on the mutual information between successive image frames. Note the line is now blurred due to the relative motion, but the y-intercept and the slope of the line are preserved. This means the resultant effect on the mutual information due to occupant motion can very clearly be separated from the effects due to illumination.

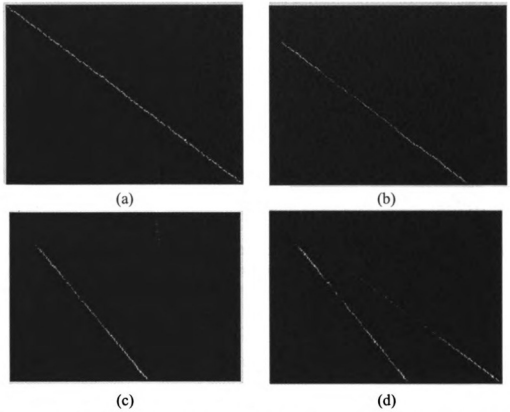
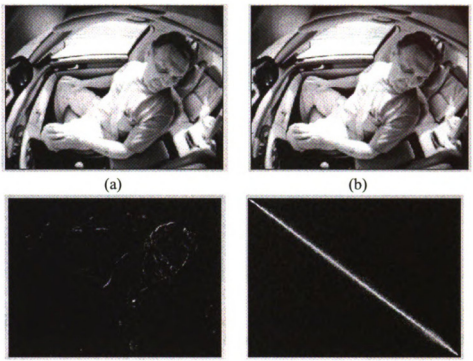


Figure 7.19: Joint density functions for various illuminations, (a) for an image with itself, (b) for an image with a version of itself with additive illumination, (c) for an image with a version of itself with multiplicative illumination, and (d) for an image with a version of itself with only a region with multiplicative illumination.



(a) (b)

(c) (d)

Figure 7.20: Effects of small occupant motions on joint density function, (a) first image, (b) image to be compared with first image, (c) difference image of image (a) and image (b), and (d) joint density function.

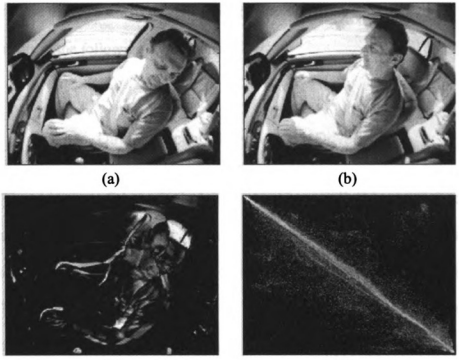


Figure 7.21: Effects of large occupant motions on joint density function, (a) first image, (b) image to be compared with first image, (c) difference image of image (a) and image (b), and (d) joint density function.

7.4 Data Collection for Occupant Tracking

There are two objectives for the data collection for the occupant motion segmentation and tracking analysis:

1) verify tracking of human occupants during normal behavior under various driving conditions, and
2) verify tracking of occupants during pre-crash braking events and the resultant performance of ASZ intrusion detection.

Objective (1) is easily performed using real human subjects, however, objective (2) cannot safely be met using human subjects. A mechanical crash test dummy must be

used for these tests. We will describe the collection process to support each of these requirements in the following subsections.

7.4.1 Data Collection Using Vehicle Drives

The objective of the vehicle drive testing is to collect images of occupants performing typical motions, under a variety of driving and illumination conditions. A segment of a typical input image sequence for this type of collection is shown in Figure 7.22. The images are collected at the full 30 Hz video rate of the camera. The images are stored as a sequence of bitmap files, where each file is a single image frame from the camera. The types of environmental and situational conditions that have been collected, include driving:

1) at night,
2) in modest sunlight,
3) in harsh unobstructed sunlight,
4) in harsh sunlight through tree lines (to generate moving shadow bands),
5) through freeway tunnels and underpasses (to generate global lighting variation over time),
6) in parking lots and along buildings (to generate structured background clutter from the parked vehicles and the buildings), and
7) on multi-lane roads (to have cars moving outside the passenger window at different speeds and directions).

Aside from these environmental and situational conditions, images are collected for multiple occupants in a variety of clothing to generate large variations in the images.

Figure 7.22: Part of the sequence of human motion during a vehicle drive test used for testing the motion estimation algorithms.

7.4.2 Data Collection Using the Robotic Test Fixture

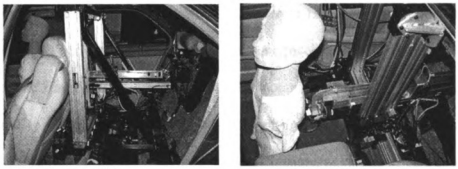
The robotic test fixture was designed to support continuous testing of the tracking subsystem for verifying the time delay of the detection of an intrusion into the Automatic Suppression Zone (ASZ). The known intrusion time is determined from an LED light beam, which is broken by the dummy, as it enters the ASZ. The light beam is oriented along the ASZ boundary, and as the dummy crosses the light beam, it triggers the intrusion alert. The relative time difference between the actual intrusion event and when the tracker predicts an intrusion is recorded. The

experiment can be run for thousands of iterations. The robotic test fixture is shown in Figure 7.23.

A mannequin is used in place of an actual Anthropomorphic Test Dummy (ATD) to reduce the weight load that is moved by the robotic arm. The system can move the dummy using both a constant velocity and a constant acceleration motion profile. We use constant acceleration, since it better matches the dynamics profile of actual vehicle braking. The start point of the dummy can also be varied, as well as the peaks of the acceleration profiles. The system can simulate any of the following motion profiles:

1) pure translational motion (simulates an unbelted occupant),
2) pure rotational motion (simulates an occupant with only the waist belt used), and
3) translational motion followed by rotational motion (simulates an occupant with only a loosely belted waist belt).

These profiles allow the dummy to simulate all of the interesting occupant dynamics conditions. An occupant that is properly belted is not simulated, since the occupant moves very little in a pre-crash event, and he is not likely to be placed at risk by the airbag deployment.



(a) (b)

Figure 7.23: Robotic test fixture for testing ASZ intrusion time, (a) view through driver's side rear door, and (b) view through sunroof.

7.5 Results of Segmentation Methods for Image Sequences

Using the data collected during the vehicle drives, we will test the motion segmentation algorithms on a variety of real-world driving conditions. These images provide excellent examples of combined foreground and background motions, complex motions of the occupants (limbs moving, head turning, etc.), and varied illumination conditions. We provide a series of output example images for each of the methods defined. This will allow a qualitative comparison of the algorithms. The same motion sequence is processed using all the methods proposed. The input image sequence for the motion processing was shown in Figure 7.22.

7.5.1 Results of Template Matching-based Motion Segmentation

The results of applying the Hausdorff-based motion segmentation algorithms on the sequence in Figure 7.22 are provided in Figure 7.25 and Figure 7.26. Figure

7.24 provides the exact frame for which the current Hausdorff distance-based point set matching is performed. Figure 7.25 shows typical results for the pre-processing stages of the Hausdorff processing. Note the incoming image and the last image are subtracted to obtain the difference image, Figure 7.25 (a), and then this image is filtered using the gradient image of the current frame, region limited based on the occupant dynamics from the tracker, and then thresholded. This set of thresholded points in Figure 7.25 (b) is then compared to the current occupant template, and the set of points with the lowest Hausdorff distance is then computed. These results are shown in Figure 7.26.

Figure 7.24: Input image prior to Hausdorff pre-processing.

(a) (b)

Figure 7.25: Results of Hausdorff pre-processing, (a) difference image, and (b) thresholded and region limited difference image.

We also processed the template-matching algorithm on a more difficult sequence, where there is considerable motion out of the image plane (towards the camera), resulting in considerable perspective-based distortions. Figure 7.27 shows a segment of the motion sequence we are analyzing, and Figure 7.28 shows the exact image in this sequence when the Hausdorff distance fails. Figure 7.29 shows the results of the pre-processing required for Hausdorff point-set matching. Figure 7.30 shows the final results of the attempted match, with the template points from the best point-set match in the last image frame (squares in the image) and the best point-set match in the current image frame (circles in the image). Notice that the Hausdorff algorithm could only find background points (see circles) as the nearest match, and the point-set matching completely missed the actual occupant. The deformations in the shape of the occupant were too severe to allow the algorithm to find a point-set match on the actual occupant.

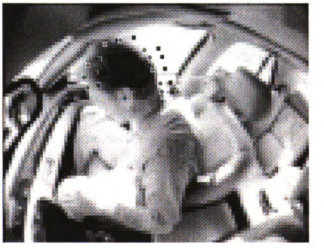


Figure 7.26: Final Hausdorff output from two consecutive frames; circles are from the current frame and squares are from the previous frame.

Recall, some of the underlying assumptions for template matching were: (i) the object should not move too much between frames, and (ii) the object's shape should not change too dramatically between frames. Clearly, in this example sequence from our airbag suppression application, both of these requirements were violated. The results are that the object is completely lost in the matching process.

Figure 7.27: Segment of motion sequence where the Hausdorff template matching fails.

Figure 7.28: Exact image from the sequence shown in Figure 7.28 with considerable out of the image plane motion where template matching fails.

(a)

(b)

Figure 7.29:Images for Hausdorff pre-processing, (a) difference image, and (b) thresholded and region limited difference image.

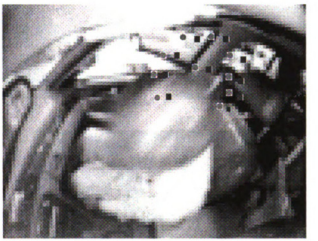


Figure 7.30:Final Hausdorff output for failed image sequence from two consecutive frames; circles are from the current frame and squares are from the previous frame.

7.5.2 Results of Optical Flow-based Motion Segmentation

The results for the optical flow-based motion segmentation are generated for the phase, gradient, and correlation methods. They are generated from the same sequence in Figure 7.22.

A Phase-based Results

The results of applying the phase-based motion segmentation algorithms on are provided in Figure 7.31 and Figure 7.32. Figure 7.31 shows the resultant U and V component fields in sub-figures (a) and (b), and the resultant vector representation of the motion field is provided in sub-figure (c).

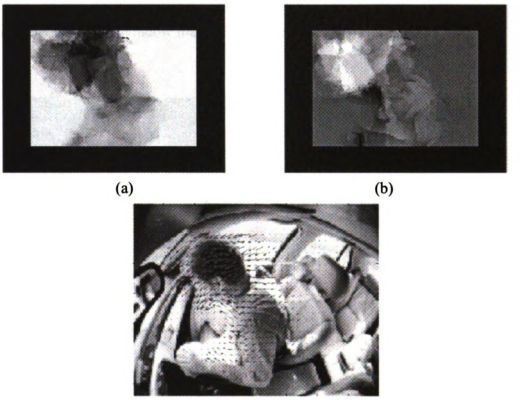


Figure 7.31: Phase-based motion estimation results. (a) U-component image, (b) V-component image, and (c) resultant flow field with velocity vectors shown.

Recall, we are using the motion fields to perform occupant segmentation, and not actual motion estimation (since this is performed by the tracker defined in Chapter 9). Therefore, we are interested in analyzing the clustering that results from

the motion estimation processing, which recall is performed using the EM algorithm. We performed the EM clustering using a mixture of three Gaussian components. We chose three components, where one component is for the stationary background, one is for the occupant, and the last is for the motion outside of the window. We retain the highest speed component of this mixture, which is shown in Figure 7.32 (c). Figure 7.32 (d) provides this EM cluster overlaid on the original image, showing the region that would be extracted by the algorithm. Notice the considerable amount of the region outside the window that was extracted with the occupant.

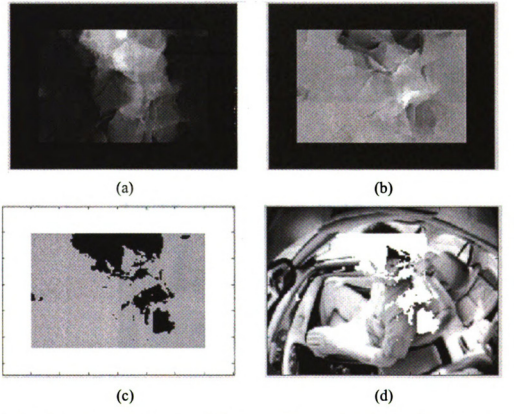
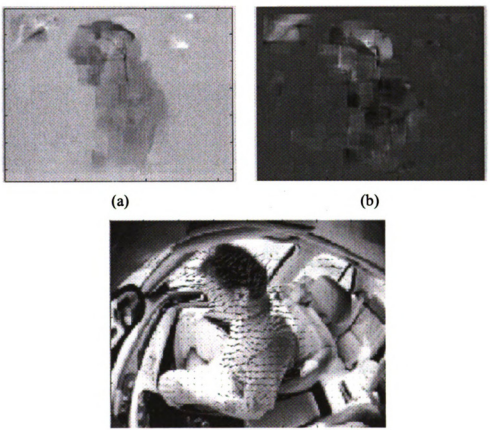


Figure 7.32: EM clustering for phased-based motion estimates. (a) U-component, (b) V-component, (c) EM cluster result, and (d) EM cluster result overlaid on the original image.

B Gradient-based Results

The results of applying the gradient-based motion estimation algorithms on the sequence in Figure 7.22 are provided in Figure 7.33 and Figure 7.34. Figure 7.33 shows the resultant U and V component fields in sub-figures (a) and (b), and the resultant vector representation of the motion field is provided in sub-figure (c).



(a) (b)

(c)

Figure 7.33: Gradient-based motion estimation results. (a) U-component image, (b) V-component image, and (c) resultant flow field with velocity vectors shown.

As with the phase-based processing, we performed the EM clustering using a mixture of three Gaussian components. We retain the highest speed component of

this mixture, which is shown in Figure 7.34 (c). Figure 7.34 (d) provides this EM mixture overlaid on the original image, showing the region that would be extracted by the algorithm. Notice the amount of the region outside the window that was extracted with the occupant is considerably less than for the phase-based methods (recall Figure 7.32 (d)). This is to be expected since the change in illumination is now explicitly modeled, and therefore can be removed.

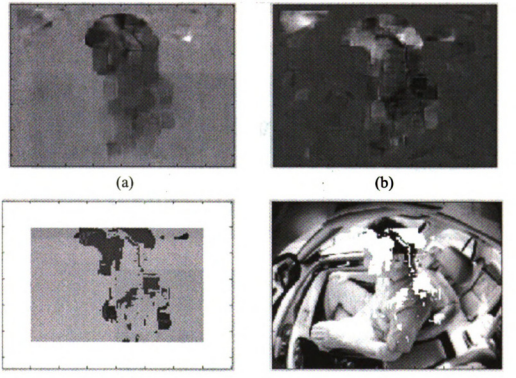
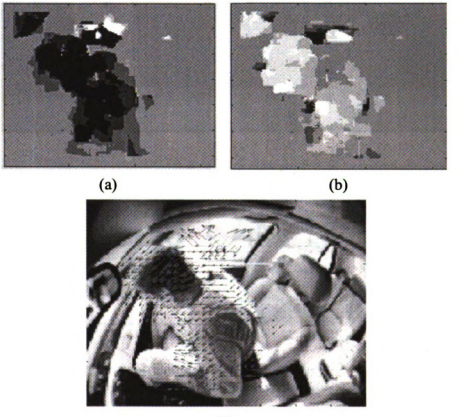


Figure 7.34: EM clustering for gradient-based motion estimates. (a) U-component, (b) V-component, (c) EM cluster result, and (d) EM cluster result overlaid on the original image.

C Correlation-based Results

The results of applying the gradient-based motion estimation algorithms on the sequence in Figure 7.22 are provided in Figure 7.35 and Figure 7.36. Figure 7.35 shows the resultant U and V component fields in sub-figures (a) and (b), and the resultant vector representation of the motion field is provided in sub-figure (c). Once again, note the results in Figure 7.36 (c), where the final cluster is overlaid on the original image, showing the region that would be extracted by the algorithm.



(a) (b)

(c)

Figure 7.35: Correlation-based motion estimation results. (a) U-component image, (b) V-component image, and (c) resultant flow field with velocity vectors shown.

Notice that when we compare Figure 7.36 (d) for the correlation processing with Figure 7.34 (d) for the gradient processing, there is very little overlap in the exact pixels defined in the segmentations (shown as the overlay). This clearly shows that our hypothesis that motion segmentation could benefit from fusion processing may be valid. It is also interesting that processing the same data through two algorithms, which are designed to extract the same information, can have such apparently orthogonal results.

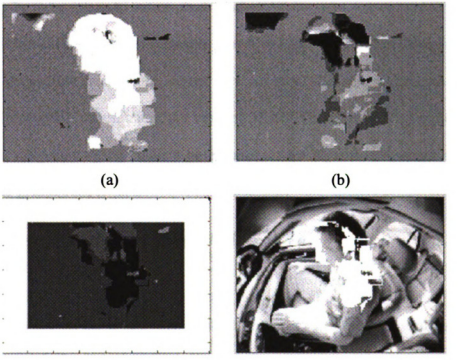


Figure 7.36: EM clustering for correlation-based motion estimates. (a) U-component, (b) V-component, (c) EM cluster result, and (d) EM cluster result overlaid on the original image.

D Demonstration of Optical Flow on Sequence that Causes Hausdorff Algorithm Failure

Recall that Figure 7.28 shows an image from a sequence for which the Hausdorff distance-based template-matching algorithm fails. Figure 7.30 shows the results of the Hausdorff template-match, and it is clear that the algorithm completely missed the occupant, and found only background clutter points as the closest matching. We will compare these results with the results using the gradient-based optical flow algorithm with the illumination effects modeled to determine if optical flow may be more robust to out-of-plane motions.

In this difficult motion sequence shown in Figure 7.28, all of the assumptions for the optical flow calculations are not met. In particular, this sequence has nearly all of the occupant motion out of the imaging plane. Therefore the assumptions used for the development of the gradient-based optical flow algorithm have not been met. Recall the gradient algorithm:

$$\frac{\partial f}{\partial t} = -f(x,y,t)\cdot div(\mathbf{v}) - \mathbf{v}\cdot grad(f(x,y,t)) + \phi\,, \qquad (7.34)$$

where we relied on the assumption that the motion of the object is restricted to the imaging plane, which led to the term $div(\mathbf{v})$ being zero, which is consequently not valid.

Notice in Figure 7.37 (c) that the vectors computed for the optical flow are nearly randomly oriented. This is due to the fact that the optical flow in the image was due to the first term in Equation (7.34) rather than the assumed second term. Therefore, if we were using this algorithm for accurate motion estimation, it would

have failed. In Figure 7.37 (a), (b), and (c) the motion segmentation of the occupant from the background is still quite complete, even though the actual estimates are wrong. This validates our earlier assumptions that, by using the optical flow for motion-based segmentation, rather than motion estimation, we may be able to relax some of the assumptions. Clearly, Figure 7.37 (c) provides a very good segmentation of the occupant, which is in stark contrast to the template-based approach, shown in Figure 7.30 where the output provided no useful information regarding the segmentation of the occupant.

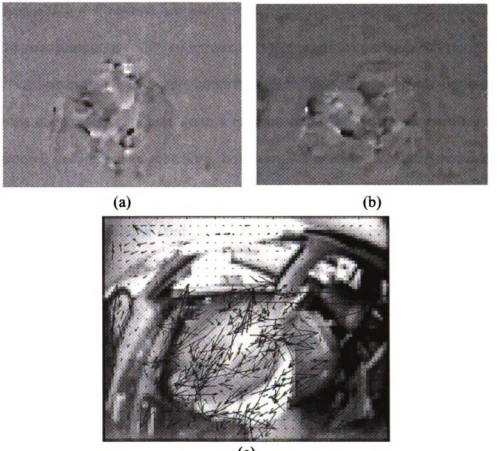


Figure 7.37: Optical flow-based motion segmentation for sequence where Hausdorff distance failed to perform, (a) U-component image, (b) V-component image, and (c) resultant flow field with velocity vectors shown.

It is important to revisit the underlying assumptions regarding the motion of the occupant, in order to determine which approach is potentially more appropriate for airbag suppression. Recall, from our earlier discussions, regarding the template-matching approach, that we cannot assume the occupant motion lies within the image plane. This issue highlights a significant difference between the template approach and the optical flow approach. The optical flow equations attempt to accurately estimate the motion vectors for every pixel in the image. In our application, however, we are using optical flow for a segmentation cue, rather than as an accurate motion estimator. If the equations are used as an accurate estimator, then violating the underlying assumptions results in erroneous estimates of the motion vectors.

If, however, we are only interested in segmentation, then the optical flow equations still result in motion vectors on the occupant being different than on the background. In template matching, however, we need to accurately estimate the transformation the occupant goes through, in order to properly update the template. If the underlying assumptions deviate too far from reality, then the matching of the template will eventually degrade to the point where a match is no longer possible, and the occupant is lost.

Additionally, since the optical flow algorithms are applied on windows within the image, rather than as global operators across the image, motions that generally violate the in-plane constraint, may still produce local changes in the image that can approximate the in-plane assumption. This means that violations of the assumption

of in-plane motion will be less severe for the optical flow than for template matching, since template matching is a global operation.

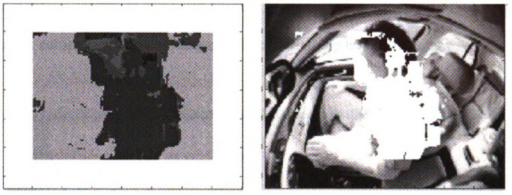
7.5.3 Results of Motion Field Fusion

We will initially show the results of combining all of the component velocities from the three motion estimation algorithms. Recall, in the results in Section 7.5.2, we applied the EM algorithm to the individual U and V component fields to derive the motion cluster estimates. Now we will apply the EM algorithm on the complete set of images (i.e., the U and V velocity component from all of the candidate algorithms), generating a total of 6 image planes. In order to reduce the computational burden of performing the fusion task, we will perform dimensionality reduction on this six dimensional space, and determine the effect on the final occupant segmentation.

A Results of EM on Combined Motion Flow Fields

The EM algorithm results on the combined motion estimation are provided in Figure 7.38. Figure 7.38 (a) is the EM clustering and Figure 7.38 (b) shows the current grayscale image frame overlaid on the segmentation results. Note the significant improvement over any of the individual methods in detecting the complete extent of the occupant. In particular, notice in Figure 7.38 (b) that there is no motion detected outside the rear window, and very little motion is detected outside the

passenger window as well. The drawback is that now we must perform the EM algorithm in a 6-dimensional space.



(a) (b)

Figure 7.38: EM clustering results based on the 6 U and V fields derived from the multiple methods (phase-based, correlation-based, and gradient-based), (a) EM clustering, and (b) EM clustering overlaid on the original image.

B Results of Dimensionality Reduction on Combined Motion Fields

We processed all of the input images through the principal components analysis algorithm. We experimented with the number of principal components that were required to provide a reasonable segmentation of the occupant. The relative values of the eigenvalues will show us the additional information content preserved with each additional principal component. The ordered eigenvalues are plotted in Figure 7.39, and Table 7.1 shows the percentages of information captured with each increasing eigenvalue. Note that 82% of the information in the dataset is contained within the first two eigenvalues, and the highest eigenvalue alone captures over 60% of the information. Figure 7.40 (b) shows the segmentation results maintaining only

the top eigenvalue for generating the fused motion segmentation output. Likewise, Figure 7.40 (d) shows the resultant segmentation from using the top two eigenvalues. The top two principal components provide a very good segmentation that is comparable in quality to the segmentation from the EM processing of the complete set of six component velocity images, with considerably less processing.

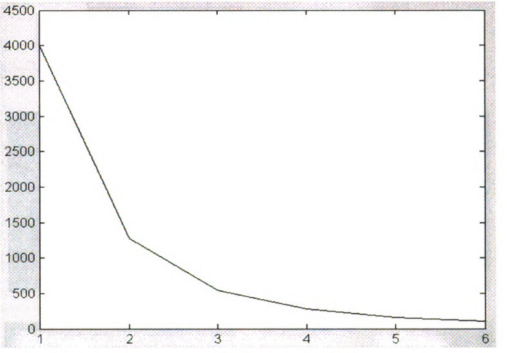
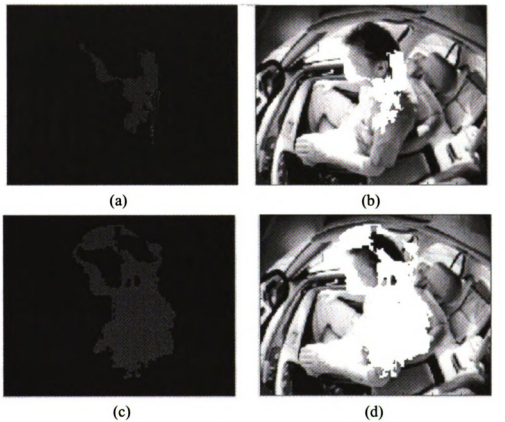


Figure 7.39: Ordered eigenvalues of PCA on 6 (U and V) component motion fields.

Table 7.1: Fraction of information captured by maintaining principal components up to the maximum number.

Principal Component	Fraction of Information Captured
1	0.620
2	0.822
3	0.910
4	0.957
5	0.988
6	1.000



(a) (b)

(c) (d)

Figure 7.40: Segmentation results of EM applied on the PCA projections of optical flows, (a) EM clustering of largest eigenvalue, (b) overlaid on the original image, (c) EM clustering of two largest eigenvalues, and (d) overlaid on the original image.

7.6 Summary

We have shown the results from a number of motion segmentation algorithms from two competing formalisms: (i) template matching, and (ii) optical flow. In template matching we found that the matching and template updating algorithms are limited in the amount of occupant motion allowed in the direction of the camera. In our application there are considerable perspective effects that can make it very difficult to match the template from one frame to the next, without allowing considerable deformation of the template. There are two drawbacks to allowing large deformations: (i) the computational time required for searching the high dimensional parameter space may be prohibitive, and (ii) when such large deformations are allowed, the algorithm can easily match to erroneous objects in the image and completely miss the occupant.

The optical flow algorithms have been shown to still be effective for motion segmentation, when the occupant motion contains a component normal to the imaging plane. This would not be the case if we were trying to actually estimate the motion vectors in the image, but recall we are merely using the occupant's motion to segment her from the vehicle background interior. Of the optical flow algorithms, the correlation method performed the best, in terms maximizing the amount of the occupant that was included in the segmentation while minimizing the amount of the background that was included due to illumination effects. By performing the correlation on blocks of pixels, rather than relying on the calculation of the gradient at each pixel, the algorithm is effectively increasing the available signal to noise ratio, and therefore deriving a better motion estimate. The drawback of correlation

methods is the reduction in the accuracy of the actual boundary of the object being segmented, but for our application the actual boundary is not overly critical.

We also demonstrated the benefits of combining the outputs of the three optical flow algorithms, and performing information fusion on the complete set of component vector fields. The fusion was accompanied by dimensionality reduction, using PCA to reduce the number of free variables estimated, in order to reduce the computational complexity and the effects of noise. While the added computational cost of running multiple motion segmentation algorithms is probably not feasible for our application, the research in motion fusion did reinforce the fact that there is no single best algorithm for motion segmentation, and when combined together the result was superior to any of the individual methods.

This is particularly interesting since, in theory, all three algorithms are providing redundant information (estimates of the U and V fields). In reality, however, they each are providing different information on the motion vector field due to the different methods used in the computation. An interesting area of future research would be to investigate means to reduce the computational burden of the multiple algorithms, possibly through reducing the image resolution, in order to offset the added computational burden of the fusion processing. This is similar in philosophy to the idea of combinations of multiple weak classifiers to build a strong classification system [47]. There may also be a benefit from using additional inputs, such as the input grayscale image, to provide additional constraints on the EM modeling. This concept of integrating additional, non-motion related data, into the EM processing is an interesting area for future research.

Lastly, we introduced the use of mutual information as a mechanism for reformulating the optical flow problem, from the tradition paradigm of the flow of grayscale values through the image sequence, to a paradigm of the flow of information through the image sequence. By reformulating the problem into this new representation, we were able to demonstrate a decoupling of the effects of illumination and the effects of motion in an image sequence. While the traditional algorithms tend to confuse these two sources of image change, the measurement of the mutual information between successive image frames shows a distinct phenomenological difference between the two sources of image change. An interesting direction of future work will be in the area of developing algorithms for motion segmentation that are based on mutual information.

Chapter 8.
Occupant Representation for Tracking

The objective of occupant representation is to define a model of the occupant that enables efficient tracking. There are two types of representation required for completely defining a tracking system for an object: (i) shape representation, and (ii) motion representation.

8.1 Occupant Shape Representation

There are three common methods used for modeling the shape of a human in an image scene: (i) contour-level modeling, (ii) blob-level modeling, and (iii) whole-body modeling (silhouettes) [108][117][117]. These approaches nearly span the entire set of modeling possibilities, from the finest to the coarsest resolutions. Pixel-level processing is not considered, since it would require tracking the occupant at too a low level to be practical.

8.1.1 Contour-Level Models

Contour-level processing models the human contour as a set of curve segments, most commonly accomplished using B-Splines [108] [117]. Figure 8.1

provides the graphical representation for this model. The system tracks high contrast points along the contour by searching along vectors normal to the contour, at equally spaced sample points. The system can then model two types of evolution of the contour: (i) translational and rotational motion of the entire contour, and (ii) deformations of the contour.

8.1.2 Blob Level Models

The blob-level processing defines the human by a set of connected, but separately behaving components, typically based on the joint locations in the human, as shown in Figure 8.2 [117]. The overall structure of the human is maintained through hard constraints, which force the blobs to remain connected, as shown in Figure 8.2 (g). This type of representation is very natural for systems that use motion based segmentation, since every segment of the human body would have a different motion [117]. For example, Figure 8.2 (c) through (f) show the blobs corresponding to the different motions detected for the various parts of the arm. Since different body components are naturally modeled in this approach, it is straightforward to perform body part tracking.

8.1.3 Whole-Body Models

The whole-body representation can be considered a limiting condition of the blob representation, where the human is modeled by a single blob or silhouette. This

representation is shown in Figure 8.3 [115]. The overall posture of the human is determined based on the orientation and shape of the silhouette. A standing human has a dramatically different silhouette than a sitting, or bending, human. From posture analysis, the probable locations of the various body parts may be detected and tracked. The complete silhouette can be used to perform other functions, such as symmetry and periodicity analysis [115]. The symmetry analysis can be used to determine if the human is carrying an object, such as a bag or a box. Periodic analysis can be used to identify body parts. For example, the motion of the legs will have a periodic nature to them, and hence can be isolated from the silhouette. Likewise, a region that does not exhibit the periodic nature may correspond to the torso.

Figure 8.1: Contour-level representation of human structure [108].

For the airbag suppression application, the whole body representation is the most logical. We are not interested in tracking limbs, since only the locations of the

head and torso are used for suppressing the airbag. We will further simplify the whole body representation by defining a bounding ellipse around the occupant.

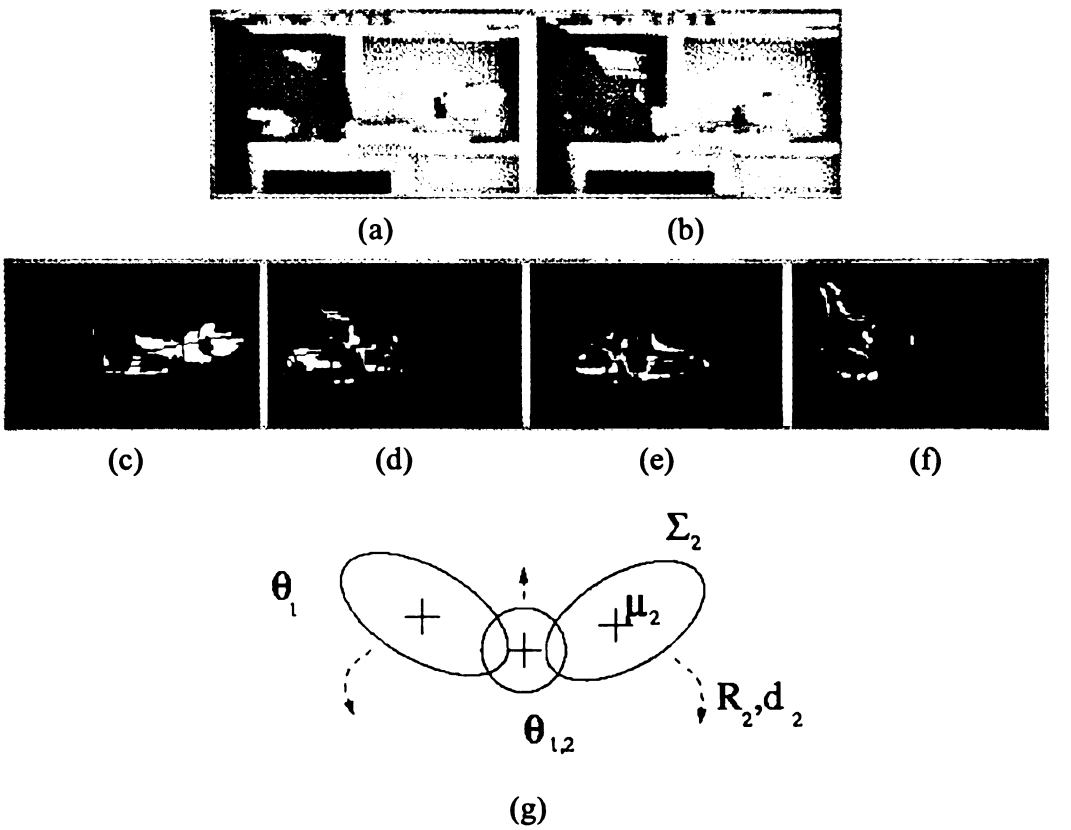
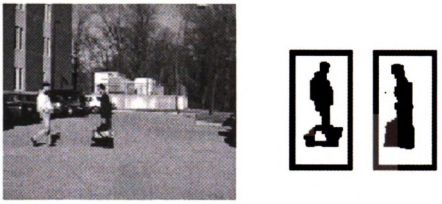


Figure 8.2: Blob representation of the human during motion showing the modeling of the various moving body parts as distinct blobs. (a) initial raw image, (b) raw image at next time instance, (c) blob for the upper arm, (d) blob for the hand, (e) blob for the elbow, (f) blob for the forearm, and (g) definition of two blobs and their interaction point [117].



(a) (b)

Figure 8.3: Whole body representation of humans during motion, (a) raw image scene, and (b) segmented image showing two distinct whole-body blobs [115].

8.1.4 Ellipse Model of Occupant

The ellipse model of the occupant focuses on the head and torso motion. The motion of the limbs is not of concern, and can actually be considered foreground clutter that should be ignored, since a hand in the ASZ is not to be considered an intrusion. The two underlying assumptions for our approach to occupant modeling for dynamic suppression are: (i) the occupant can be modeled using a bounding ellipse, and (ii) the motion of this ellipse can be reliably tracked to predict when the occupant's head or torso may enter the automatic suppression zone (ASZ). Figure 8.4 shows the geometry of this fundamental assumption.

The elliptical model provides a considerable reduction in the complexity of the motion tracking problem, and allows us to only track the following parameters:

1) coordinates for x and y centroid,

2) major and minor axes, and

3) in-plane rotation, θ.

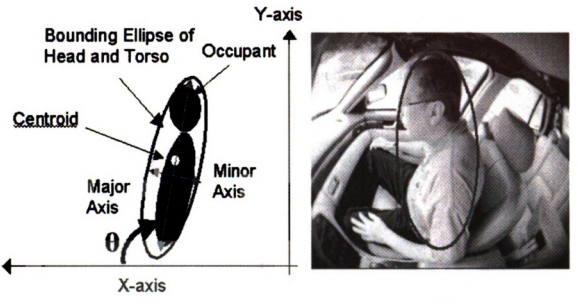


Figure 8.4: Human representation geometry for tracking, (a) definition of ellipse parameters, and (b) ellipse fit to the human occupant.

These parameters serve as the input measurement vector for the subsequent occupant tracking. Additionally, this simplifying assumption provides increased robustness against extraneous motion, such as that found outside the passenger window or from the motion of the occupant's arms.

There are two methods for computing the ellipse, depending on whether a template-based approach or an optical flow-based approach is used for motion segmentation.

1) for the template-based segmentation, the ellipse is computed from a set of points corresponding to the occupant's boundary, or

2) for the optical flow-based segmentation, the ellipse is computed from the binary region corresponding to the entire occupant.

Figure 8.5 shows the characteristics of these two data sets. For the contour-based segmentation, the ellipse-fitting approach is based on least squares fitting. For the motion-based segmentation, the ellipse-fitting approach is based on moment analysis.

(a) (b)

Figure 8.5: Ellipse fitting problem definition, (a) template matching-based segmentation as input, and (b) optical flow-based segmentation as input.

A Direct Least Squares Ellipse Fit

There are two methods for estimating the best-fit ellipse to a contour such as shown in Figure 8.5 (a): (i) Hough transform, and (ii) least squares fitting [41]. The Hough transform method has the advantages of being very robust to noise and occlusion, and also always generating an ellipse. The disadvantage of the Hough transform is that it is computational intensive, which is a critical issue in the airbag suppression application. The least squares method has the advantage of a lower processing burden, but a general least squares fitting of conics does not necessarily

generate ellipses, as is shown in Figure 8.6. This figure shows the deviation from a perfect ellipse for four data sets of increasing distortion [41]. The results demonstrated in this set of images are critical for the airbag suppression application, where the ellipse model is actually a simplifying assumption to the true underlying form of the human body. Recall from Figure 8.5 (a) that the actual contour data extracted from an image to represent an adult occupant, which is far from a perfect ellipse. Therefore our application will be very sensitive to the robustness of the ellipse-fitting algorithm.

Rather than performing the least squares fit to a simple conic, we will need to explicitly assume an elliptical form [40]. Recall a general conic is of the form [40]:

$$f(\mathbf{a},\mathbf{x}) = ax^2 + bxy + cy^2 + dx + ey + \mathrm{f}, \qquad (8.1)$$

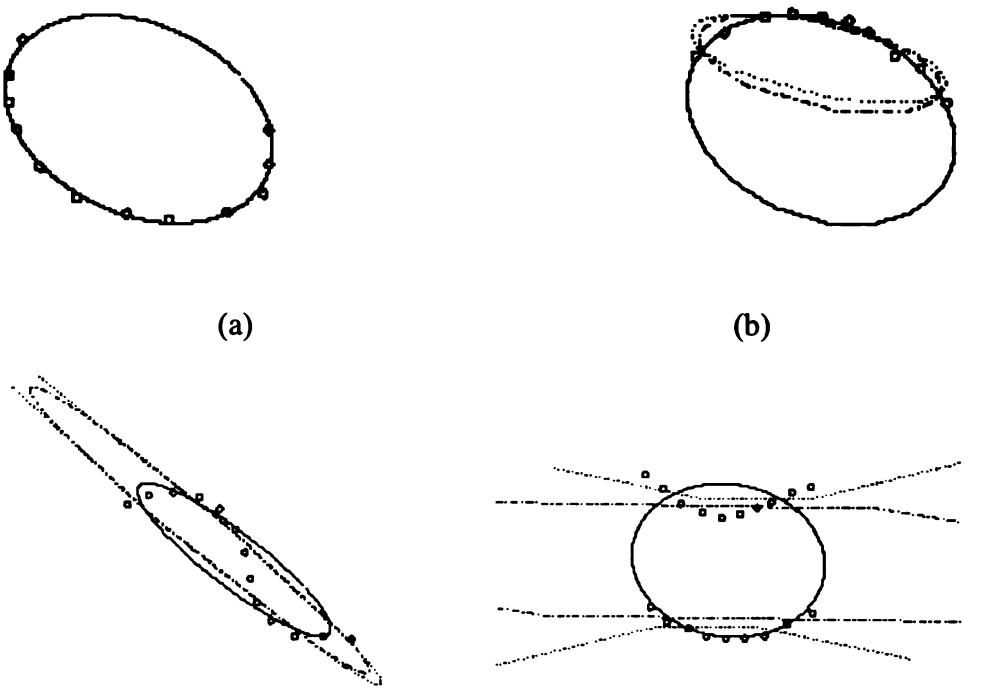
where we can write: $\mathbf{a} = [a\ b\ c\ d\ e\ f]$ and $\mathbf{x} = \left[x^2\ xy\ y^2\ x\ y\ 1\right]$. We define $f(\mathbf{a},\mathbf{x}_i) = D$, as the distance from a particular point, $\mathbf{x}_i$ to the conic defined by $f(\mathbf{a},\mathbf{x}_i) = 0$. The problem of fitting an input set of data points to the curve in Equation (8.1) can then be formulated as minimizing the total distance to all the points in the least squared error sense [40]:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}}\left\{\sum_{i=1}^{N} f(\mathbf{a},\mathbf{x}_i)^2\right\}. \qquad (8.2)$$

Specific constraints must be imposed on **a** to ensure a unique and elliptical solution. The constraints ensure that the ellipse is of the form: $b^2 - 4ac < 0$. This

quadratic constraint can be enforced on Equation (8.2) by minimizing the cost function [41]:

$$\frac{1}{b^2 - 4ac} \cdot \sum_{i=1}^{N} f(\mathbf{a}, \mathbf{x}_i). \qquad (8.3)$$



(a) (b)

(c) (d)

Figure 8.6: Examples of conic least squares fitting, (a) perfect set of ellipse points, (b) ellipse point-set with large amount of occlusion, (c) S-shaped point set, and (d) parallel C-shaped point set.

This can be formulated as an eigenvalue problem by defining the constraint equation in matrix form as [40]:

$$b^2 - 4ac = \mathbf{a}^T \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{a} = \mathbf{a}^T \mathbf{C} \mathbf{a} < 0 . \qquad (8.4)$$

If we also define a design matrix and a scatter matrix by [41]:

$$\mathbf{D} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \mathbf{x}_N \end{bmatrix}^T , \text{ and } \mathbf{S} = \mathbf{D}^T \mathbf{D} , \qquad (8.5)$$

then the eigenvalue problem can be written as [41]:

$$\mathbf{S}\mathbf{a} = \lambda \mathbf{C}\mathbf{a}$$

and (8.6)

$$\mathbf{a}^T \mathbf{C} \mathbf{a} = 1 .$$

For the constraint defined by Equation (8.4), there will be only one positive eigenvalue, and this eigenvalue defines the solution to the ellipse-specific least squares fit problem. The corresponding eigen-vector corresponds to the vector of the ellipse-fit parameters: $\mathbf{a} = [a\ b\ c\ d\ e\ f]$ [41].

B Moments-based Ellipse Fit

The bounding ellipse shape parameters can be calculated from the central moments of the segmented, *NxM* binary image in Figure 8.5 (b). The second order central moments are computed according to:

$$\Sigma_{xx} = \frac{1}{m_{00}} \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j) \cdot (x - \mu_x)^2, \tag{8.7}$$

$$\Sigma_{xy} = \frac{1}{m_{00}} \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j) \cdot (x - \mu_x)(y - \mu_y), \tag{8.8}$$

$$\Sigma_{yy} = \frac{1}{m_{00}} \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j) \cdot (y - \mu_y)^2, \tag{8.9}$$

where the lower order moments are:

$$m_{00} = \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j), \tag{8.10}$$

$$\mu_x = \frac{1}{m_{00}} \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j) \cdot x \quad \text{and} \quad \mu_y = \frac{1}{m_{00}} \sum_{i=1}^{N} \sum_{j=1}^{M} I(i,j) \cdot y. \tag{8.11}$$

Now the parameters of the ellipse are defined to be:

$$centroid_x = \mu_x \quad \text{and} \quad centroid_y = \mu_y, \tag{8.12}$$

with

$$L_{\text{major}} = \frac{1}{2} \cdot (\Sigma_{xx} + \Sigma_{yy}) + \frac{1}{2} \cdot \sqrt{\Sigma_{yy}^2 + \Sigma_{xx}^2 - 2 \cdot \Sigma_{xx} \cdot \Sigma_{yy} + 4 \cdot \Sigma_{xy}^2}, \tag{8.13}$$

$$L_{\text{minor}} = \frac{1}{2}\cdot(\Sigma_{xx} + \Sigma_{yy}) - \frac{1}{2}\cdot\sqrt{\Sigma_{yy}^2 + \Sigma_{xx}^2 - 2\cdot\Sigma_{xx}\cdot\Sigma_{yy} + 4\cdot\Sigma_{xy}^2}, \quad (8.14)$$

and

$$Slope = \tan^{-1}\left(\frac{L_{\text{major}} - \Sigma_{xx}}{\Sigma_{xy}}\right). \quad (8.15)$$

8.2 Occupant Motion Representation

The motion of the human in the vehicle can be modeled as a sequence of distinct motion events. For example, the occupant can be sitting still, and then move forward to open the glove box or the occupant can be moving normally, and then be suddenly propelled forward due to a pre-crash braking event. This structure of a sequence of motion types can be addressed in the framework of the human activity classification/recognition problem. A possible taxonomy of the methods available to solve this class of problem is provided in Figure 8.7 [112]. The two general approaches are: (i) template matching, and (ii) state space analysis. The state space method is more commonly used than template matching due to its flexibility [116][117][121][122]. Traditionally, state space representations of human motion are focused on posture states of the subject.

Within the state-space methods, however, we propos a further subdivision into two methods: (i) posture-based methods, and (ii) dynamics-based methods. The posture-based approach monitors the transitions from one posture to the next in an

image sequence [113][114]. The dynamics-based approach monitors the transitions between dynamics models during the image sequence [116][117][118].

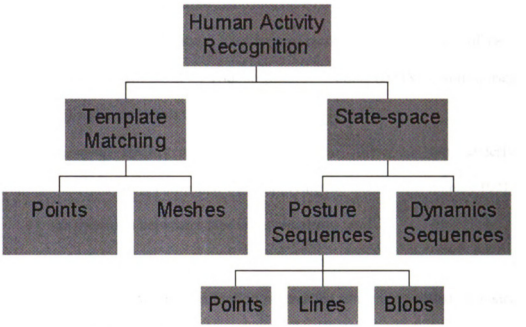


Figure 8.7: Structure of human motion analysis problem.

For the airbag suppression application, it is natural to use the dynamics model for state space analysis, since the posture of the occupant is fundamentally non-changing. The occupant's motion within the vehicle, and particularly his motion towards the airbag, is the critical behavior that must be modeled. Our proposed state space for the occupant consists of three possible states:

1) Stationary (no motion),
2) Human motion (roughly 0.1 g acceleration), and
3) Pre-crash braking motion (roughly 1.0 g acceleration).

Complex dynamics can be modeled in a state space paradigm by sequences of atomic dynamics events, sometimes referred to as "Movemes" in [122], or alternatively, elements in a motion alphabet in [117]. In the state space model, the dynamics of the occupant over time can be modeled by a set of atomic states, defined by specific dynamics and durations. This type of model, with these states of varying duration, is naturally addressed by Hidden Markov Models (HMMs), and is used in the majority of research on this topic [116][117][122] [125].

When using HMMs to model data streams, there are two underlying assumptions that are rarely confirmed when applying the technology, namely [92]:

1) the transition between states is instantaneous, and

2) there is only one state in operation at a time.

Consequently, with HMMs, human motion is modeled as a concatenated sequence of atomic motions, or postures, with an instantaneous transition between states [117] [125][126]. We believe that these assumptions are often not met in human motion analysis using real-time video, and, while they have had some success in the research community, another mechanism may be more effective at modeling the complexity of human motion dynamics. Recall that HMMs were originally used for modeling sequences of phonemes in human speech [123]. In the speech application, the requirement of instantaneous transitions and non-simultaneity of states is met. The human vocal track can only produce a single sound at a time, due to its physical limitations. Also the transitions between phonemes in a human speech sequence are quite rapid in relation to the duration of many of the phonemes [102].

However, when modeling the entire human body, it is possible to be in two states at the same time, when the motion is viewed within the context of a real-time sequence of video images. Note that most commercial cameras generate 30 image frames per second, while the fastest human reflex motion is roughly 10 Hertz, and most typical gross human motor events are typically in the 0.1 – 1.0 Hertz frequency range. For example, the simple motion sequence of a human getting out of a chair may actually be better modeled as a sequence of the combination of the sitting and standing states, where the probability of being in sitting versus standing states gracefully evolves over time.

Figure 8.8 provides a segment of a real-time video sequence, where an occupant is slowly moving in his seat, performing normal human motions. Likewise, Figure 8.9 provides a time history of the likelihood that the dynamics of the occupant corresponds to one of the three dynamics states defined earlier. Notice that while there are periods in the motion stream where the occupant is more clearly in a moving versus a stationary state, there are many transitory stages where there is no clear decision possible, and the dynamics would best be represented as a weighted average of the two most likely states. In general, it is extremely artificial to consider the occupant's motion to be represented by sequences of the occupant being stationary and being stopped, without first transitioning between intermediate states. In these cases it is difficult to define exactly which of the models should be used in an HMM architecture.

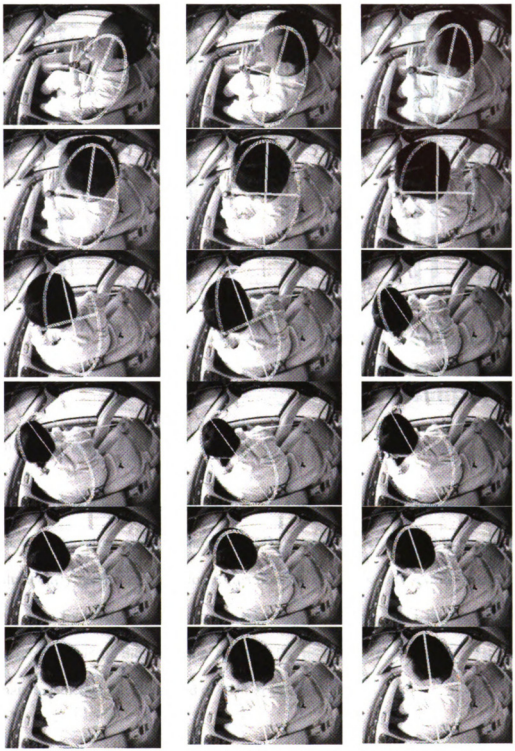


Figure 8.8: Sequence of an adult occupant moving normally in a vehicle.

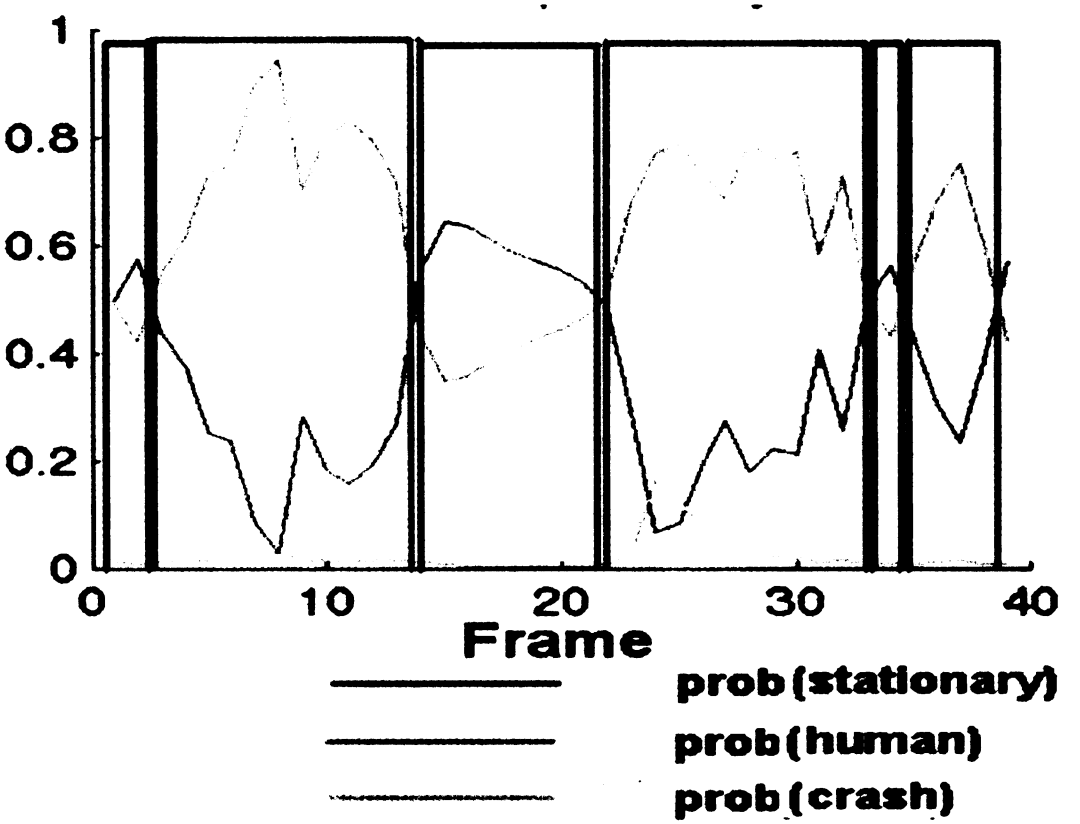


Figure 8.9: Variations in the model likelihoods for each of the motion models during normal human motion.

An alternative mechanism for modeling human motion in video sequences, called the interacting multiple models (IMM) tracking, is one where the system simultaneously generates tracks for all the possible motion states, and then generates a final state by mixing these atomic tracks together. The mixing is accomplished using the analysis of the deviation between the predicted track and the next track estimate to generate the likelihood of each of the track models.

8.2.1 Sequential Dynamics Modeling Using HMMs

The basic formulation of HMM, is that a time sequential system can be modeled as a sequence of individual atomic states, whose existence only depends on the immediate previous state. The transition between states is driven by a matrix of probabilities, defining the likelihood of the various state-to-state transitions [123][92]. The hidden component of the HMM corresponds to the fact that the actual state of the human is not observed, but rather it is a random variable defining the observed motion. The typical architecture of the HMM is shown in Figure 8.10 [92].

There are two tasks that must be performed for using the HMMs: (i) computing the probability of an observed sequence, given the model parameters (compute *P(O|model)*, and (ii) choosing the sequence of hidden states that is optimal, given an observation sequence (compute *P(S|O)*) [92][123].

Computing *P(O|model)* is performed using the forward-backward algorithm (actually only the forward part is needed) [123][92]. In this algorithm, we define the forward variable to be [123]:

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i \mid \lambda), \qquad (8.16)$$

where O is the observation sequence, $q = S$ is the definition of the state at time t, and λ is the model parameter.

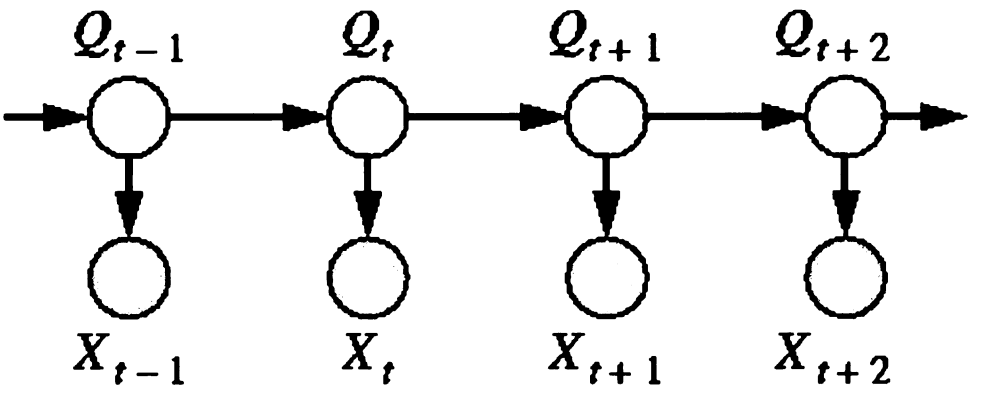


Figure 8.10: Hidden Markov Model (HMM) architecture [143].

The algorithm is initialized by [123]:

$$\alpha_t(i) = \pi_i b_i(O_1), \tag{8.17}$$

where $\pi_i = P(q_1 = S_i)$, and $b_i(O_1) = P[O_1 \mid q = S_j]$. Recall, q represents the hidden states, O represents the observations, and b is the conditional probability that we observe O given the system is in state q.

The second step of the algorithm for computing *P(O|model)* is to compute the value for α_{t+1} given α_t. This is accomplished through the iteration of the following equation [123]:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}] b_j(O_{t+1}) \text{ for } j = 1,...,N, \tag{8.18}$$

where a_{ij}, the state transition probability from state i to state j, is defined by [123]:

$$a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]. \tag{8.19}$$

The algorithm then terminates when the α_t is computed for all time instances, and the final probability of the observations, given the model, is defined by [123]:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i). \qquad (8.20)$$

Once the probability for the observation sequence, $P(O \mid \lambda)$, is computed, the optimal state sequence given the observations and the model, $P(S|O)$ can be derived via [123]:

$$P(S \mid O) = \frac{P_{s1} p_{s1}(O_1) \cdot \prod_{t=2}^{T} p_{st}(O_t) P(S_t \mid S_{t-1})}{P(O)}. \qquad (8.21)$$

8.2.2 Sequential Dynamics Modeling Using IMMs

The concept behind the IMM tracking is that the dynamics of a tracked object can be modeled as the weighted combination of a set of tracks, corresponding to multiple simultaneous models of the observed system. Each of these trackers is characterized by a different assumption of motion, i.e. a different dynamics model [18][103][104]. The true motion of the system is then estimated by combining these atomic tracks. The key to using the IMM framework is to define motion models for each possible set of dynamics the system may experience, and then define the mechanism for transitioning between these models. The dynamics model begins with

the state vector for a system $x(t)$ at time t. In the IMM framework there is now a different state vector for each possible model, $x(t|model\ i)$.

The interaction between the models depends on the *a priori* defined switching probabilities between each of the models, as shown in Figure 8.11, and is captured in a matrix defining the probabilities of transitions between all the models, $p(model\ i|model\ j)$. We must also define the likelihood that the system will be in a particular model, given the current sensor measurements and the current state vector, $L(model\ i|z,\ x(t|model\ i))$, where z is the current measurement. These likelihoods and transition probabilities, for each atomic model, are then used to compute two sets of mixing weights: (i) to determine the current state vector for each model based on the contributions from all the models, and (ii) to determine the final estimate of the system state by computing a weighted sum of all of the atomic model state estimates [18][103][104].

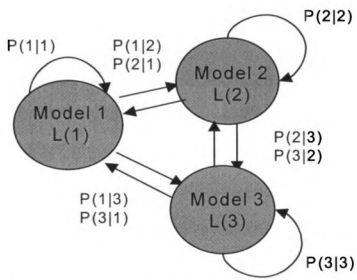


Figure 8.11: State transition diagram for a 3-model IMM system.

The mixing step is the critical difference between HMMs and IMMs. HMMs do not assume the final state of the system can be a mixture of the atomic states, but rather insist the system lies in only one of the states, that is selected based on which state is the most likely.

8.3 Summary

We have defined two models for representing the occupant for tracking, one model for the shape of the occupant over time, and one for the dynamics of the occupant. The shape representation for the occupant uses a whole body model, where the occupant is modeled by an ellipse. This is effective for our application, since the occupant is always seated, and therefore the head and torso are the components that we are modeling. Also we want to ignore the arms and hands, since, if they are tracked, we may accidentally disable the airbag.

For the dynamics modeling, we have reviewed the use of the two common methods, namely template matching and state space analysis. We have also extended the common definition of state space methods to include dynamics sequences, rather than just posture sequences. For state space modeling, we have reviewed the common use of HMMs to model human motion in video sequences. We have showen that for human motion sequences, the transition times between states is relatively slow, and this results in regions of ambiguity, where one is artificially choosing a single model in the HMM paradigm. To counter this, we introduced an alternative model called the Interacting Multiple Models (IMM) tracker that has been

developed for target tracking in surveillance applications. In this model we simultaneously consider all dynamics models to be active at the same time, and then we mix them together to devise a composite model. While we only proposed this model for dynamics, we believe an interesting area of future research would be to apply this model to posture-based state space analysis as well, where a person rising from a chair is not sitting and then standing, but rather is transitioning smoothly between these two states with a smoothly varying probability of being in each of the postures.

Chapter 9.
Occupant Tracking

The occupant tracker is responsible for determining when an occupant may enter the Automatic Suppression Zone (ASZ) of the airbag. The ASZ location is unique for each vehicle and for each airbag configuration. The tracker generates estimates and predictions for:

1) the motion of the occupant (where the human should appear in the next camera frame), and
2) the shape of the occupant (what is the shape of the bounding ellipse of occupant in the next camera frame).

This implies there is a natural split in the processing that separates the problem into position tracking (motion tracking) and shape tracking, as shown in Figure 9.1 [18]. The occupant tracker uses the position measurements of the bounding ellipse as inputs, and generates estimates of the current position, as well as predictions of future positions.

The measurements that the tracker subsystem uses are the five key parameters for defining an ellipse, in a plane $\{x_{\text{centroid}}, y_{\text{centroid}}, \theta, l_{\text{major}}, l_{\text{minor}}\}$. The outputs of the tracker are then the complete 3-dimensional representation of the bounding ellipsoid $\{x_{\text{centroid}}, y_{\text{centroid}}, \theta, \phi, l_{\text{major}}, l_{\text{minor}}\}$, where we assume an ellipsoid of rotation.

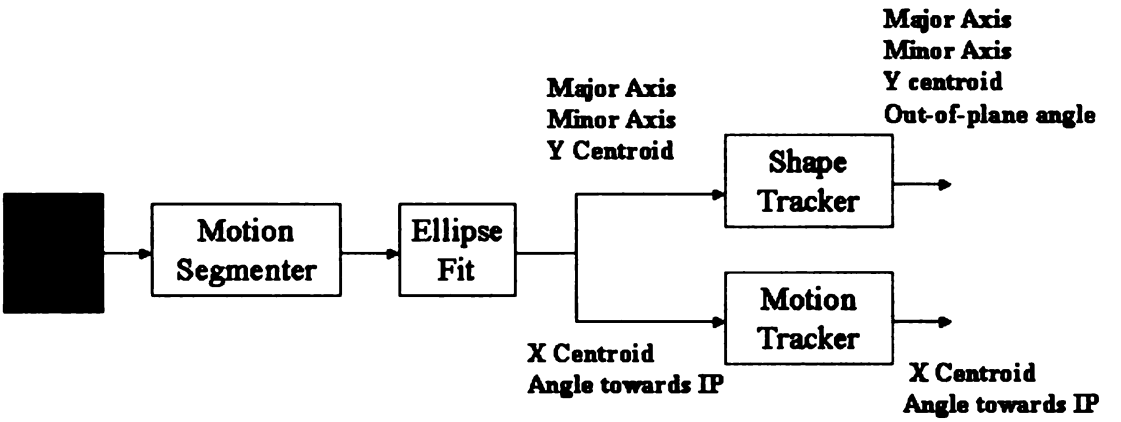


Figure 9.1: Occupant tracker showing the parallel motion and shape tracking functions.

The subset of parameters maintained by the motion tracker is those that is influenced by the vehicle dynamics, $\{x_{centroid}, \theta\}$. The accelerations these parameters experience, which include the pre-crash braking accelerations, can cause the occupant to be thrust forward in the x-direction. Likewise, the ellipse will tilt forward in the θ direction, due to these pre-crash braking dynamics, if the occupant is not using a shoulder harness. The remaining ellipse parameters, $\{y_{centroid}, \phi, l_{major}, l_{minor}\}$, are only influenced by the motions of the occupant and not the motions of the vehicle. Consequently, the shape tracker processes these parameters. It will be shown in subsequent sections that this division of 3-dimensional object tracking into distinct motion and structure components is common, and is exploited in other algorithms, such as the factorization method, for estimating shape from motion [151].

The most commonly used method of tracking any general object is the Kalman filter [98][100][101][102]. The Kalman filter is optimal if the expected underlying dynamics of the object being tracked are known, and if the noise

corrupting the measurements is Gaussian [98][108]. However, recall from Chapter 8 that the motion of the occupant can be modeled by three unique dynamics states. The transitions between these various models have been referred to as kinematic discontinuities that must be accounted for to minimize the prediction errors [111]. Deutscher et al. has also shown that even extended Kalman filter alone is unable to account for these transitions [111]. This is expected since the extended Kalman filter can linearize non-linear dynamics models, but still relies on a fixed underlying noise model that cannot support these discontinuities [98][99]. In Chapter 8 we proposed using interacting multiple models (IMM) filtering for tracking the occupant through these various model transitions [103][104]. The IMM filter is naturally able to handle these kinematic discontinuities.

Once the motion of the occupant is properly tracked, the shape of the occupant must also be estimated. Since the camera system generates only a 2-D representation of the occupant, the shape tracker must be capable of inferring the 3rd dimension of the occupant. A set of common shape-from-X processing methods will be described, and will serve as an introduction to our new method, called *shape from deformation*, which is based on a unique utilization of the IMM Kalman filter.

9.1 Motion Tracking

The motion tracker takes, as inputs, the measurements $\{x_{\text{centroid}}, \theta\}$ of the occupant, and generates smoothed estimates and predictions of these values, as shown in Figure 9.1. The system uses these inputs to estimate the current position of the

occupant, and to predict where the human should appear in the next camera frame. The motion tracker recursively integrates this time series of noisy measurements into a smooth estimate of the trajectory of the human occupant to accurately predict the position of the occupant at future times.

Recall that the underlying assumption of the Kalman filter is that the measurements of a system are corrupted by zero-mean, Gaussian noise. Since our measurements are generated from the integration either through moments analysis, or through least squares fitting, of a large number of pixel values, it is reasonable, through the weak law of large numbers, to assume that the errors associated with these measurements are Gaussian.

9.1.1 Underlying Technologies for Motion Tracking

There are four related technologies associated with applying the Kalman filter to the airbag suppression application, namely:

1) basic Kalman filtering,
2) extended Kalman filtering,
3) modeling the occupant dynamics to optimize the Kalman filter, and
4) interacting multiple models (IMM) Kalman filtering.

A Overview of the Basic Kalman Filter

The two basic equations for the Kalman filter are the *dynamics equation* and the *measurement equation*. The *dynamics equation* defines the mechanism by which

the system evolves through time, and the *measurement equation* defines the mechanism by which the tracked state and the measurements interact. Mathematically, they are defined, respectively, to be [98][100][101]:

$$\mathbf{x}(k) = \mathbf{\Phi}(k-1) * \mathbf{x}(k-1) + \mathbf{v}(k-1) \qquad (9.1)$$

and

$$\mathbf{z}(k) = \mathbf{M}(k) * \mathbf{x}(k) + \mathbf{w}(k), \qquad (9.2)$$

where $\mathbf{\Phi}$ is the state transition matrix, $\mathbf{x}$ is the state vector, $\mathbf{v}$ is the process noise, $\mathbf{z}$ is the measurement value, $\mathbf{M}$ is the measurement matrix, and $\mathbf{w}$ is the measurement noise [98][100]][101]. The state vector defines the extent of the dynamics required to be estimated for the particular application. For our application, the state vector will be: $\mathbf{x}(k) = [x,\ \dot{x},\ \ddot{x}]$, consisting of the position, velocity, and acceleration of the particular variable.

The dynamics equation, (9.1), is composed of two terms: (i) the state propagation term, $\mathbf{\Phi}(k-1) * \mathbf{x}(k-1)$, and (ii) the process noise term, $\mathbf{v}(k-1)$. The state propagation term defines how the state vector, at the time *k-1* of the last measurement, is propagated forward to a future time. It accomplishes this through the state transition matrix, which defines the Newtonian mechanics that are involved during the transitions from time *k-1* to time *k*. The state transition matrix for a system that models position, velocity, and acceleration is provided below [98][100][101]:

$$\mathbf{\Phi}(k) = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \qquad (9.3)$$

where Δt is the time between the two updates, *k-1* and *k*. Note that this transition matrix represents the classical Newtonian equations of motion for a system undergoing velocity and acceleration,

$$x(t) = x(t-1) + v(t-1) * \Delta t + a(t-1) * \Delta t^2 / 2 . \qquad (9.4)$$

The second term in the dynamics equation, Equation (9.1), is the process noise term. It models the white-noise process that drives the underlying dynamics model. In these basic Kalman equations, the process noise is assumed to be zero-mean Gaussian and white (i.e., uncorrelated in time). The process noise covariance is then defined by [105][106]:

$$Cov_{v}(k,j) = E\{v(k) \cdot v(j)\} = Q(k) \cdot \delta(k,j), \qquad (9.5)$$

where $\delta(k,j)$ is the Kronneker delta function, which defines a diagonal matrix with parameters *Q(k)* defining the correlation for the position, velocity, and accelerations independently.

The measurement equation, (9.2), defines the transformation through which the tracked state and the actual sensor measurements interact. Recall, in our airbag suppression application, the measurements are $\{x_{centroid}, \theta\}$ while our state vector is $\mathbf{x}(k) = [x,\ \dot{x},\ \ddot{x}]$. The measurement matrix, ***M***, is then defined to extract the position value from the state vector, prior to when we combine the state with the incoming measurement, according to [98][100][101]:

$$\mathbf{M}(k) = [1 \quad 0 \quad 0]. \tag{9.6}$$

The dynamics equation and the measurement equation are realized in the Kalman filter implementation via the state prediction equation and the state estimation equation, which are respectively defined to be [98][100][101]:

$$\mathbf{x}(k \mid k-1) = \mathbf{\Phi}(k-1) * \mathbf{x}(k-1 \mid k-1), \tag{9.7}$$

and

$$\mathbf{x}(k \mid k) = \mathbf{x}(k \mid k-1) + \mathbf{G}(k) * \mathbf{Z}(k), \tag{9.8}$$

where

$$\mathbf{Z}(k) = \mathbf{z}(k) - \mathbf{M}(k) * \mathbf{x}(k \mid k-1), \tag{9.9}$$

where **G** is called the gain of the Kalman filter, and **Z**(*k*) is called the innovation or residual of the filter. Note that **Z**(*k*) measures the difference between the expected measurement and the actual measurement. Also the term **x**(*k-1*|*k-1*) is the value of the state vector at time instance *k-1* given all the measurements up to time *k-1*, and **x**(*k*|*k-1*) and **x**(*k*|*k*) follow the same convention regarding time. The equations for the gain, **G**, and the covariance matrix for the predictions, **P**, are then, respectively [98][100][101]:

$$\mathbf{G}(k) = \mathbf{P}(k \mid k-1) * \mathbf{M}(k)^T * (\mathbf{M}(k) * \mathbf{P}(k|k-1) * \mathbf{M}(k)^T + \mathbf{R}(k))^{-1}, \tag{9.10}$$

and

$$\mathbf{P}(k \mid k-1) = \mathbf{\Phi}(k-1) * \mathbf{P}(k-1 \mid k-1) * \mathbf{\Phi}(k-1) + \mathbf{Q}(k-1), \tag{9.11}$$

where **Q**(k) is the process noise, **R**(k) is the measurement noise, **M**(k) is the measurement matrix, and $\mathbf{\Phi}(k)$ is the state transition matrix.

Recall, the innovation term is the difference between the input measurement and the next predicted measurement. If the dynamics of the system has been properly modeled, this term should be a zero-mean, uncorrelated Gaussian [98][100][103][104]. If the sequence of innovations over time is either correlated, or not zero-mean, or both, then the Kalman filter equations are not properly modeling the underlying dynamics [103][104]. The behavior of this sequence of innovation terms will be critical to the operation of the IMM Kalman filtering.

B The Extended Kalman Filter

The basic Kalman filter equations assume a linear dynamics model for the system. In other words, the state transition matrix and the measurement matrix can be implemented as simple matrix multiplications. If each of these matrices is replaced by a general function defining the dynamics, then we must define the extended Kalman filter. The extended Kalman filter allows us to model much more complicated system behaviors. The dynamics equation and the measurement equations of the Kalman filter, previously defined in Equations (9.1) and (9.2), are now respectively defined to be [98][100][101]:

$$\mathbf{x}(k) = f[\mathbf{x}(k-1)] + \mathbf{v}(k-1), \tag{9.12}$$

and

$$\mathbf{z}(k) = h[\mathbf{x}(k)] + \mathbf{w}(k), \tag{9.13}$$

where the functions $f[\cdot]$ and $h[\cdot]$ can model any desired dynamics. The process noise and measurement noise terms maintain the same form as for the basic Kalman

implementation, while the prediction and estimate of the state vector become [98][100][101]:

$$\mathbf{x}(k \mid k-1) = f[\mathbf{x}(k-1 \mid k-1)], \tag{9.14}$$

and

$$\mathbf{x}(k \mid k) = \mathbf{x}(k \mid k-1) + \mathbf{G}(k) * \mathbf{Z}(k), \tag{9.15}$$

where now the innovation term is:

$$\mathbf{Z}(k) = \mathbf{z}(k) - h[\mathbf{x}(k \mid k-1)]. \tag{9.16}$$

Recall that in Equations (9.10) and (9.11) that the gain and covariance matrices also depend on $\mathbf{\Phi}(k)$ and $\mathbf{M}$, which have now been replaced by $f[\cdot]$ and $h[\cdot]$. In order to simplify the calculations of the Kalman filter, we linearize the state transition and measurement functions, using the Jacobian of these functions, $f[\cdot]$ and $h[\cdot]$, according to [98][100][101]:

$$\hat{\mathbf{M}}(k) \cong \frac{d}{d\mathbf{x}} h(\mathbf{x})\Big|_{\mathbf{x} = \mathbf{x}(k \mid k-1)}, \tag{9.17}$$

and

$$\hat{\mathbf{\Phi}}(k) \cong \frac{d}{d\mathbf{x}} f(\mathbf{x})\Big|_{\mathbf{x} = \mathbf{x}(k-1 \mid k-1)}. \tag{9.18}$$

The gain and covariance matrices can now be computed using $\hat{\mathbf{\Phi}}(k)$ and $\hat{\mathbf{M}}(k)$ instead of $\mathbf{\Phi}(k)$ and $\mathbf{M}$.

C Modeling the Dynamics in the Kalman Filter

The dynamics model of the system is characterized by the process noise. The process noise appears in the following three terms: (i) covariance matrix for the predictions, (ii) the gain matrix, and (iii) the state transition matrix. The form of the process noise can accommodate a variety of possible dynamics models, including [105][106]:

1) constant position,
2) constant velocity,
3) constant acceleration, and
4) time-correlated acceleration.

The choice of the model used depends on the complexity of the dynamics expected for the object being tracked. Clearly, for our occupant tracking application, we are not dealing with either constant position or constant velocity objects. Occupants in the vehicle can start and stop moving at anytime. Likewise, a constant acceleration model is not optimal, since we know the dynamics of humans can be modeled as a concatenated series of different motions, as shown in Chapter 8. Therefore, the model that is most attractive is the time-correlated acceleration model.

In this model, the acceleration that an object experiences is assumed correlated over time. For example, the sequence of an occupant first sitting still, then moving forward, and finally stopping, is not a series of instantaneous transitions, but rather is a set of correlated transitions based on the muscle reaction time of the occupant and the occupant's inertia. This sequence of motions can be modeled by a

time-correlated history of accelerations and decelerations. A very common model, referred to as the Singer model, uses exponentially correlated accelerations, where the accelerations, are defined by [105][106]:

$$r_a(\tau) = E\{\nu(t)\cdot\nu(t+\tau)\} = \sigma_a^2 \cdot e^{(-\alpha\,|\,\tau\,|)}, \tag{9.19}$$

where σ_a^2 is the variance, and α is the exponential decay coefficient of the expected accelerations. Recall, the correlations of the acceleration are explicitly represented in the process noise and state transition matrices, $\mathbf{Q}$ and $\mathbf{\Phi}(k)$. The state transition matrix can now be defined by [105][106]:

$$\mathbf{\Phi}(\Delta t) = \begin{bmatrix} 1 & \Delta T & \frac{1}{\alpha^2}[\exp(-\alpha\Delta T) + \alpha\Delta T - 1] \\ 0 & 1 & \frac{1}{\alpha}[1 - \exp(-\alpha\Delta T)] \\ 0 & 0 & \exp(-\alpha\Delta T) \end{bmatrix}. \tag{9.20}$$

Note that in the limit of $\alpha \to 0$ (i.e., infinitely correlated or constant accelerations), the state transition matrix reduces to the conventional Newtonian mechanics model. Recall the covariance of the process noise in the basic Kalman equations was a diagonal matrix of constant variances, while it now becomes [105][106]:

$$\mathbf{Q} = 2\alpha\sigma_m^2 \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{1,2} & q_{2,2} & q_{2,3} \\ q_{1,3} & q_{2,3} & q_{3,3} \end{bmatrix}, \tag{9.21}$$

where the q terms are given by [105][106]:

$$q_{11} = \frac{1}{2\alpha^5} \cdot [1 - \exp(-2\alpha\Delta T) + 2\alpha\Delta T + \frac{2}{3}\alpha^3\Delta T^3 - 2\alpha^2\Delta T^2 - 4\alpha\Delta T \exp(-\alpha\Delta T)],$$

$$q_{12} = \frac{1}{2\alpha^4} \cdot \left[\exp(-2\alpha\Delta T) + 1 + 2(\alpha\Delta T - 1)\exp(-\alpha\Delta T) - 2\alpha\Delta T + \alpha^2\Delta T^2\right],$$

$$q_{13} = \frac{1}{2\alpha^3} \cdot [1 - \exp(-2\alpha\Delta T) - 2\alpha\Delta T \exp(-\alpha\Delta T)],$$

$$q_{22} = \frac{1}{2\alpha^3} \cdot [4\exp(-\alpha\Delta T) - 3 - \exp(-2\alpha\Delta T) + 2\alpha\Delta T], \quad (9.22)$$

$$q_{23} = \frac{1}{2\alpha^2} \cdot [\exp(-2\alpha\Delta T) + 1 - 2\exp(-\alpha\Delta T)], \text{ and}$$

$$q_{33} = \frac{1}{2\alpha} \cdot [1 - \exp(-2\alpha\Delta T)]$$

In order to implement these motion models, we define a method for computing the decay constant and variance terms, α and σ_m^2, according to [105][106]:

$$\alpha = \frac{1}{\text{average time duration of a maneuver}}, \quad (9.23)$$

and

$$\sigma_m^2 = \frac{A_{\max}}{3} \cdot (1 + 4 \cdot P_{\max} - P_0), \quad (9.24)$$

where A_{max} is the maximum acceleration experienced by the system. This maximum acceleration occurs with probability P_{max}, while P_0 is the probability the system experiences zero acceleration. The probability density function for the system acceleration is shown graphically in Figure 9.2 [105][106]. It consists of the probability peaks at P_{max} and P_0, and a uniform distribution for all the remaining accelerations.

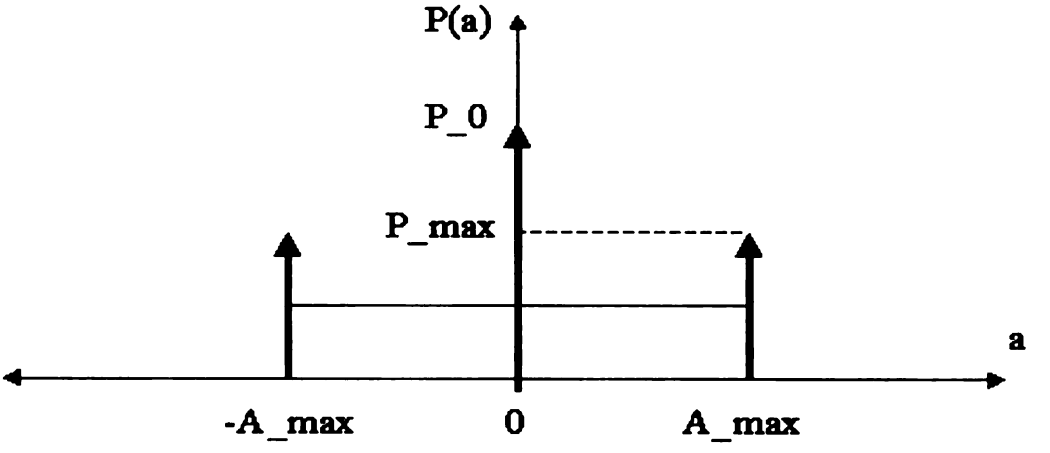


Figure 9.2: Probability density function for the accelerations in the exponentially correlated acceleration dynamics model.

D Interacting Multiple Models (IMM) Kalman Filtering

Recall the IMM Kalman filter is based on processing tracks associated multiple possible motion models in parallel. For the IMM implementation, there is a complete set of Kalman equations, namely Equations (9.7) through (9.11), for each model. These models are defined by the possible dynamics that they allow, and they

are implemented by tuning the process noises that model these dynamics. In Kalman filtering, the innovations (the difference between the predicted state and the measured state) are a zero-mean Gaussian process, if the possible dynamics of the system have been properly modeled [98][100][101]. In the IMM implementation of the Kalman filter, the system uses the deviation of the innovations from a zero-mean Gaussian process to generate weights for the contributions of each of the underlying models [103][104]. The architecture of the IMM Kalman filter is provided in Figure 9.3.

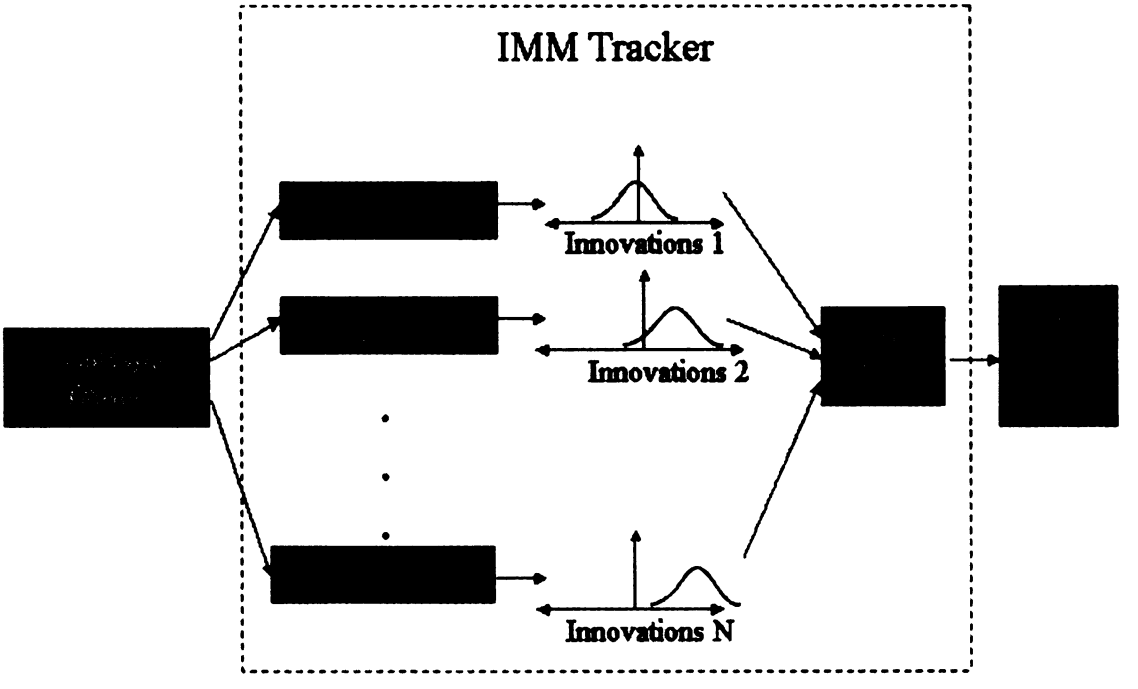


Figure 9.3: Architecture of the Interacting Multiple Model (IMM) Tracker.

The innovations of the track for each model m, for models $m=1,...,N$, are defined to be [98][99][100][101]:

$$\mathbf{Z}_m(k) = \mathbf{z}(k) - \mathbf{M}(k) * \mathbf{\Phi}_m(k-1) * \mathbf{x}_m(k-1 \mid k-1), \qquad (9.25)$$

where $\mathbf{M}$ is the measurement matrix, $\mathbf{\Phi}_m(k-1)$ is the state transition matrix for model m, and $\mathbf{x}_m(k-1\,|\,k-1)$ is the state vector for model m. The innovations for each model are then used to derive the likelihood of that model, according to [99][103][104]:

$$L_m(k) = N[\mathbf{Z}_m(k);\ 0,\ \mathrm{cov}_m(k)], \tag{9.26}$$

where this equation is the evaluation of a zero-mean Gaussian at the point $\mathbf{Z}_m(k)$.

We must also compute two model probabilities: (i) $\mu_{s\,|\,m}(k-1)$, the probability of model s is correct at time k, given that model m was correct at time k-1, and (ii) the overall model probability, $\mu_m(k-1)$. These are computed by [99][103][104]:

$$\mu_{s\,|\,m}(k-1) = \frac{1}{\sum_{s=1}^{N} p(s\,|\,m) * \mu_m(k-1)} * p(s\,|\,m) * \mu_m(k-1), \tag{9.27}$$

and

$$\mu_m(k) = K * L_m(k) * \sum_{s=1}^{N} p(s\,|\,t) * \mu_s(k-1), \tag{9.28}$$

where

$$K = \frac{1}{\sum_{s=1}^{N} L_s(k) * \sum_{t=1}^{N} p(s\,|\,t) * \mu_t(k-1)}, \tag{9.29}$$

K normalizes the probability to one, $p(s \mid m)$ is the state transition probability (i.e., the probability of transitioning from state m into state s at any given time), and $L_m(k)$ is the likelihood of the model defined by Equation (9.28).

Once the probabilities of each model are computed, it is possible to compute the estimate of the state vector for each model. The state vector for each model is a weighted combination of the state vectors for all of the models, weighted by the probability of the current model being valid, given that another model was valid at the last time instance. The state vectors for each model are computed according to [99][103][104]:

$$\mathbf{x}_m(k-1 \mid k-1) = \sum_{s=1}^{N} \mathbf{x}_s(k-1 \mid k-1) * \mu_{m \mid s}(k-1). \qquad (9.30)$$

The state vector for each model must now be propagated forward in time to the next sensor input time, using the state transition matrix defined for each dynamics model:

$$\mathbf{x}_m(k \mid k-1) = \mathbf{\Phi}_m(k-1) * \mathbf{x}_m(k-1 \mid k-1). \qquad (9.31)$$

This prediction can now be updated to the current time instance, using the most recent measurement:

$$\mathbf{x}_m(k \mid k) = \mathbf{x}_m(k \mid k-1) + \mathbf{G}_m(k) * \mathbf{Z}_m(k), \qquad (9.32)$$

where $\mathbf{G}_m(k)$ is the gain of the filter for model m, and $\mathbf{Z}_m(k)$ is the innovations for model m, that was defined in Equation (9.26). The final output state vector of the system is a combination of the state vectors for each of the models. It is a weighted

sum of the state vectors for each of the models, weighted by their current model probabilities, according to [99][103][104]:

$$\mathbf{x}(k \mid k) = \sum_{s=1}^{N} \mathbf{x}_s(k \mid k) * \mu_s(k). \qquad (9.33)$$

The corresponding covariance matrix for this state vector is then the weighted sum of the covariance matrices for each model, plus the covariance between the mixed state vector and the individual model state vectors, according to [99][103][104]:

$$\mathbf{P}(k \mid k) = \sum_{s=1}^{N} \mu_s(k) * \{\mathbf{P}_s(k \mid k) + [\mathbf{x}_s(k \mid k) - \mathbf{x}(k \mid k)][\mathbf{x}_s(k \mid k) - \mathbf{x}(k \mid k)]^T\}. \qquad (9.34)$$

Equation (9.35) identifies a critical difference between covariance of the final state predictions for an IMM filter and the traditional Kalman filter. In the traditional Kalman filter, the covariance of the predictions is independent of the state vector, (recall equation (9.11)). Now, for the IMM tracker, there is an explicit dependence on the state vector, which implies a dependence on the incoming measurement stream.

9.1.2 Implementation of Motion Tracker

We must now define the key Kalman filter and IMM parameters for the motion tracker variables $x_{centroid}$ and θ, for the three dynamics models: (i) stationary, (ii) normal human dynamics, and (iii) pre-crash braking dynamics.

First, we need to decide whether we can track $x_{centroid}$ and θ separately, or whether there may be any interactions between the variables. The advantage of treating them independently is that the state transition matrix reduces from a 6x6 matrix into two 3x3 matrices, when processing the position, velocity, and acceleration terms.

Two different and mutually exclusive, processes influence the dynamics for $x_{centroid}$ and θ. The motion of the **unbelted** occupant influences, the dynamics of $x_{centroid}$, as the occupant is propelled into the ASZ during a pre-crash braking event. The dynamics of the tilt angle, θ, is influenced by the **belted** occupant who is tilted forward in a pre-crash braking event, when their shoulder restraint is not latched, and their lap-belt is holding their hips in place. Therefore, since two separate external processes drive these two variables, we can track them separately.

For each of these trackers, we will implement the IMM Kalman filter, which requires us to define the following parameters for each of the m models, and for both $x_{centroid}$ and θ:

1) $\mu_m(0)$, the probability of the system being in model m at the initial time.

2) $p_x(s \mid m)$ and $p_\theta(s \mid m)$, the model transition probabilities.

3) α_m and σ_m^2, the exponential decay and variance

4) Update rate of the inputs and the outputs of the tracker.

The initial model probabilities for three dynamics model for $x_{centroid}$ and θ trackers are assumed to be the same. The eventual behavior of the system over time is not overly dependent on these parameters, since the tracker will be operating for very long periods of time when driving. We will define them to be:

$$\begin{aligned} \mu_{stat}(0) &= 0.495, \\ \mu_{human}(0) &= 0.495, \text{ and} \\ \mu_{crash}(0) &= 0.01. \end{aligned} \qquad (9.35)$$

We must next define the matrix for the model transition probabilities, $p_x(s \mid m)$ and $p_\theta(s \mid m)$:

$$p(s \mid m) = \begin{bmatrix} p_{stat \mid stat} & p_{stat \mid human} & p_{stat \mid crash} \\ p_{human \mid stat} & p_{human \mid human} & p_{human \mid crash} \\ p_{crash \mid stat} & p_{crash \mid human} & p_{crash \mid crash} \end{bmatrix}. \qquad (9.36)$$

No simple equations exist for computing each of these model transition probabilities. For these we empirically tested a range of values, and used the values that minimize the tracking error for a set of training crash video sequences captured with our ASZ tester defined in Chapter 7. Additionally, we imposed non-symmetry in the

probabilities to and from the pre-crash braking model. Our philosophy behind this non-symmetry is that once the system is in a pre-crash braking dynamics, there is a high probability that there will be additional high acceleration maneuvers (e.g., a mild crash, a set of concatenated minor crashes, or the occupant rebounding into the seat back). Therefore, we leave the system in the pre-crash braking state, until there is an overwhelming amount of evidence that the system is no longer in that model, and then transition the system back to the stationary or the human motion states. Figure 9.4 demonstrates the 'hysteresis' that is built into the pre-crash braking transition probabilities.

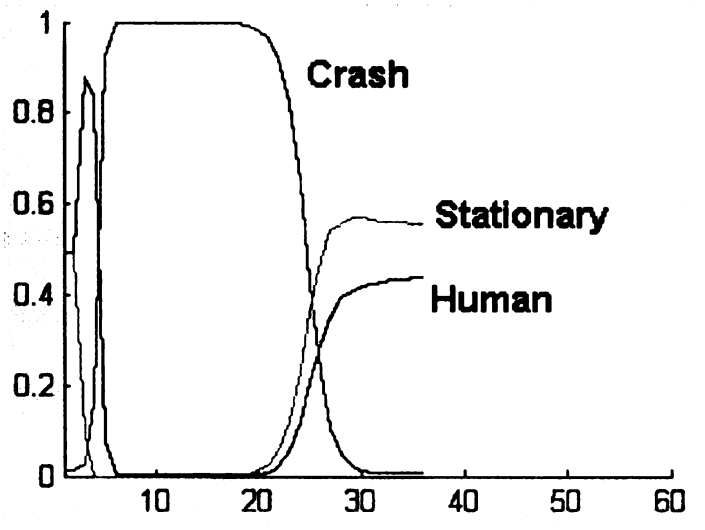


Figure 9.4: Model probabilities during a pre-crash braking motion sequence.

We use the same model transition probabilities for both $x_{centroid}$ and θ trackers, and the elements of $p(s \mid m)$ are:

$$
\begin{aligned}
p_{stat \mid stat} &= 0.75, \\
p_{stat \mid human} &= 0.245, \\
p_{stat \mid crash} &= 0.005, \\
\\
p_{human \mid stat} &= 0.245, \\
p_{human \mid human} &= 0.75, \\
p_{human \mid crash} &= 0.005, \\
\\
p_{crash \mid stat} &= 0.0005, \\
p_{crash \mid human} &= 0.0005, \text{ and} \\
p_{crash \mid crash} &= 0.999.
\end{aligned}
\qquad (9.37)
$$

Note that for each model, transition probabilities, the sum of the probabilities into that model state must sum to one, to ensure that all possible system transitions from one model to another are considered, i.e.:

$$p_{stat \mid stat} + p_{stat \mid human} + p_{stat \mid crash} = 1. \qquad (9.38)$$

The third set of parameters that we must define is α_m and σ_m^2. These are easily computed for each of the models of dynamics using the equations in (9.24) and (9.25). For these time constants we use the same values for the centroid and the angle, since the correlation characteristics of the motions should be similar, since they are driven by the same underlying dynamics, and they are:

$$\begin{aligned} \alpha_{stat} &= 0.01, \\ \alpha_{human} &= 0.1, \text{ and} \\ \alpha_{crash} &= 0.5. \end{aligned} \tag{9.39}$$

Likewise, the values for variances for the system process noise for each model for $x_{centroid}$ are then:

$$\begin{aligned} \sigma^2_{stat}(x) &= 0.001, \\ \sigma^2_{human}(x) &= 0.03, \text{ and} \\ \sigma^2_{crash}(x) &= 2.5. \end{aligned} \tag{9.40}$$

The values for variances for the system process noise for θ are slightly larger, due to lower sensitivity in the angular direction compared with the lateral direction. They are measured in degrees, and are:

$$\begin{aligned} \sigma^2_{stat}(\theta) &= 0.1, \\ \sigma^2_{human}(\theta) &= 1.0, \text{ and} \\ \sigma^2_{crash}(\theta) &= 5.0. \end{aligned} \tag{9.41}$$

These values were initially estimated based on Equation (9.25), and then optimized based on empirical data regarding predicted versus actual ASZ intrusion times using the test fixture defined in Chapter 7.

The fourth parameter to define is the input rate for the tracker. It is simply the update rate of the camera, which is 30 Hz (33 milliseconds per frame). The output rate of the tracker can be different than the input rate, and actually must be different for the airbag suppression application to allow the system to detect an ASZ intrusion within 10 milliseconds, which is roughly a three times faster rate than the input measurements. This requires us to define two different state transition matrices: (i) $\Phi_m^{input}(k)$ for transitioning the state vector to the time of the next sensor input, and (ii) $\Phi_m^{output}(k)$ for transitioning the state vector at a data rate that will support predicting when the occupant may enter the ASZ.

We have designed the output rate of the motion tracker to be 5 milliseconds, which gives us a margin of error. This update rate equates to roughly six updates per input image frame. We also need to provide some prediction capability ahead of the next frame. Therefore, we extrapolate the position of the occupant 8 high-speed outputs ahead, which allows us to extrapolate into the middle of the next image frame. The position predictions provided for each input frame are graphically shown in Figure 9.5.

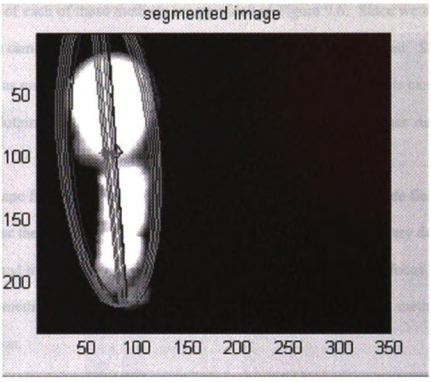


Figure 9.5: Sequence of predicted positions for one sensor input for the motion tracker. Ellipses show the set of high rate updates for a single image input.

9.2 Shape Tracking

If a system does not use an active ranging mechanism, then the 3-dimensional definition of the shape and orientation of the object must be inferred from 2-dimensional imagery. There are four mechanisms for inferring 3-dimensional shape and orientation from 2-dimensional images, namely [148][166]:

1) shape from stereo,
2) shape from shading,
3) shape from defocus, and
4) shape from motion.

Examples of each of these methods are provided in Figure 9.6. Since we are limited to a single camera for this system, shape from stereo was not considered. Shape from shading was not considered for the following two reasons: (i) occupants can have any type of clothing, and illumination conditions are uncontrolled (either direction or intensity).

Shape from focus was also considered infeasible due to the wide field of view required for the camera. Wide field-of-view cameras tend to have a very deep depth-of-focus, and this greatly limits their ability to perform shape from defocus. Since all the other methods were discounted, we then must rely on the final method, shape from motion.

9.2.1 Overview of Methods for 3-D Shape/Pose from Motion Estimation

The techniques of shape from motion (SFM) are well established in the fields of photogrammetry and 3-D image reconstruction [148]. These techniques all rely on point correspondences of features that are deemed to be reliable. Reliable features are those that have the following characteristics [156]:

1) they appear continuously in the image sequence (no occlusion),
2) they are uniquely defined in two directions (straight edges, for example, are only uniquely defined in the direction perpendicular to their orientation, which means corner features are the best to track), and
3) they do not change shape dramatically in the image sequence (minimal deformation).

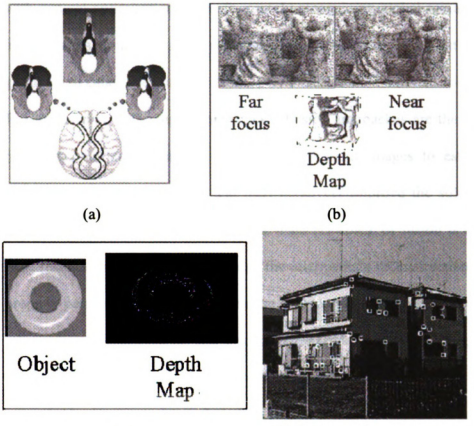


Figure 9.6: The four basic methods of deriving 3-D structure from 2-D imagery, (a) stereo, (b) shape from de-focus, (c) shape from shading, and (d) shape from motion [28].

Another common requirement of SFM algorithms is that the object being imaged must be a rigid body. Deformations in the body of interest will result in errors in the point correspondence problem. There are a series of established algorithmic approaches to solving the shape from motion problem, including [148]:

1) **Optimization Approaches** – Optimization approaches create 'optimal' reconstructions by minimizing some criterion function, typically the least squares image error. They tend to solve the SFM problem with no a priori knowledge of the imaging phenomenon, but rather apply brute force

optimization algorithms. They suffer from the classic problems associated with general optimization methods, namely recognizing global minima from local minima, since they have no underlying phenomenology to test the true goodness of candidate minima.

2) **Fusing/Kalman Filtering Approaches** – Fusion approaches are the class of reconstruction algorithms that use subsets of input images to estimate a solution to the reconstruction, and then iteratively improve the solution by integrating additional image data. The main drawback of fusion is that the final result is limited by the quality of the intermediate reconstructions. To properly perform the fusion, one must be able to calculate the quality of each intermediate reconstruction, and then determine the proper weighting that it should be given. If each of the sub-estimates are assumed to have been corrupted by Gaussian errors, then the Kalman filter can be used as an optimal means of performing the fusion task.

3) **Projective/Euclidean Reconstruction Approaches** – Projective reconstruction uses un-calibrated cameras, while the Euclidean reconstruction uses calibrated cameras. In projective reconstruction, not only is the camera calibration assumed unknown, but it is also allowed to be arbitrarily different for each image used in the reconstruction sequence. Clearly, this is not true for the airbag suppression application, where we are using the image stream from a single camera, so we will define the Euclidean approach. The basic problem of reconstruction is to solve the equation [148]:

$$\mathbf{I} = \mathbf{M} \cdot \mathbf{S}, \tag{9.42}$$

where **I** is a matrix of the actual 2-dimensional image points in homogeneous coordinates, **M** is the camera matrix, and **S** is the structure matrix. The structure matrix is the desired matrix of the actual object points defined in a 3-dimensional space. The camera matrix, **M**, contains the calibration parameters of the camera. It has the same form as the general 3-dimensional affine transformation [159]:

$$\mathbf{M}(x,y) = \begin{bmatrix} x \cdot s_{1,1} & y \cdot s_{1,2} & z \cdot s_{1,3} & d_x \\ x \cdot s_{2,1} & y \cdot s_{2,2} & z \cdot s_{2,3} & d_y \\ x \cdot s_{3,1} & y \cdot s_{3,2} & z \cdot s_{3,3} & d_z \end{bmatrix} . \qquad (9.43)$$

For Euclidean reconstruction, the following sub-matrix is extracted from equation (9.36):

$$\hat{\mathbf{S}} = \begin{bmatrix} s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}, \qquad (9.44)$$

where $\hat{\mathbf{S}}$ is a simple orthogonal rotation matrix. The use of a variation of Equation, (9.43) is also the core idea behind the factorization methods defined below.

4) **Invariants-based Approaches** – In these methods polynomial constraints on the image behavior from one frame to the next are defined. Then one algebraically eliminates some set of unknowns, and solves for the remaining

values [148]. Invariants-based approaches typically follow a three-stage process consisting of: (i) estimate the essential matrix, (ii) use the essential matrix to recover the motion parameters, and (iii) compute the actual structure parameters. These algorithms typically do not use the entire image stream, but rather use only two or three frames. Additional data from an image stream must be incorporated via another process, such as a fusion algorithm defined above [148].

While there are many structure from motion algorithms, the factorization method will be reviewed here since it solves the problem using a similar philosophy, namely by decomposing the image effects into camera (or subject) motion and subject structure. It is also very flexible in terms of the number of image frames and the number of object points per image frame that are utilized. The basic problem addressed by the factorization method is to solve the imaging equation [148]:

$$\mathbf{I} = \mathbf{M} \cdot \mathbf{S}, \tag{9.45}$$

where **I** is a matrix of the actual 2-dimensional image points in homogeneous coordinates, **M** is the camera matrix, and **S** is the structure matrix. The factorization method is capable of simultaneously extracting the structure and the motion in an image [149][150][151]. Variants of the algorithm have been developed for images collected under a variety of imaging geometries, including [149][150][151]:

1) perspective,

2) para-perspective,

3) weak perspective,

4) orthographic projection, and

5) scaled- orthographic projection.

For all of these geometrical models there is an upper and lower bound on the amount of image motion that is allowed. The camera (or object) motion must be large enough to overcome noise issues in measuring the correspondences, while the motion must not be so large as to cause either: (i) dramatic changes in points that make the point correspondence difficult, or (ii) occlusion of some of the points being tracked.

The approach of the factorization method is to track P feature points over a sequence of F image frames, thereby generating an array of the form [151]:

$$\{(u'_{fp}, v'_{fp}) \mid f = 1,...,F; p = 1,...,P\}, \tag{9.46}$$

where u is the horizontal location and v is the vertical location of the feature points in the image. Note that this list only contains points that have been tracked through the entire image stream. Any points that are lost due to occlusion or poor correspondence quality are dropped from the entire stream of data points. The images are aligned (only up to a simple in-plane translation), by subtracting the means of the u and v coordinates, through [151]:

$$u_{fp} = u'_{fp} - \bar{u}_f \text{ and } v_{fp} = v'_{fp} - \bar{v}_f, \tag{9.47}$$

where

$$\bar{u}_f = \frac{1}{P} \cdot \sum_{p=1}^{P} u'_{fp} \text{ and } \bar{v}_f = \frac{1}{P} \cdot \sum_{p=1}^{P} v'_{fp} \quad . \tag{9.48}$$

From this data array, a pair of matrices is produced. One matrix contains all the u-values, and the other matrix contains all of the v-values of the image points. These two matrices are then stacked on top of each other to generate the measurement matrix (not to be confused with the measurement matrix in Kalman filtering), which takes on the form [151]:

$$\mathbf{W} = \begin{bmatrix} \mathbf{U}_{matrix} \\ \mathbf{V}_{matrix} \end{bmatrix}. \tag{9.49}$$

At any image frame, f, the camera pointing-angle can be modeled as a Cartesian unit vector, $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$, in some fixed global coordinate system. Then the relative motion between the camera and the object can be represented by a series of unit vectors, which will capture only the relative motion in the $\mathbf{i}_f$ and $\mathbf{j}_f$ coordinates. These values are then captured in a motion matrix, **M**, which is of the form [150]:

$$\mathbf{M} = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_f^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_f^T \end{bmatrix} \tag{9.50}$$

Note that this model assumes an orthographic, or very weak-perspective, projection

system, which assumes very little change in the $\mathbf{k}_f$ direction (depth) of the real object points. Now the locations of each feature point can be represented by a unit vector, $\mathbf{s}_i$, in this same coordinate system, by a structure matrix, **S**, of the form [150]:

$$\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_P]. \tag{9.51}$$

Since we had previously removed the means from all of the u and v coordinates, the following constraint will also hold [150]:

$$\sum_{p=1}^{P} \mathbf{s}_p = \mathbf{0}. \tag{9.52}$$

The measurement matrix can now be written as the product of the motion and structure matrices [150]:

$$\mathbf{W} = \mathbf{M} \cdot \mathbf{S}. \tag{9.53}$$

Note that the motion matrix, **M**, is of size (2xF)x3, and the structure matrix, **S**, is of size 3xP, which implies that the matrix **W** is at most a rank 3 matrix (i.e., definable by three unique eigen-vectors) [150][151]. The factorization algorithm then represents the matrix **W** as a product of three matrices of special form, rather than the two general matrices, according to [150][151]:

$$\mathbf{W} = \mathbf{L} \cdot \mathbf{\Sigma} \cdot \mathbf{R}, \tag{9.54}$$

where $\mathbf{\Sigma}$ is the matrix whose diagonal elements are the singular values, and **L** and **R** are the matrices containing the left and right eigen-vectors, respectively. Since the

rank of **W** is three, all of the relevant information regarding the structure and motion of the image sequence is contained in the top three singular values and their corresponding left and right eigenvectors [150][151]. A new approximate measurement matrix is then computed from these top three singular values, giving [150][151]:

$$\hat{\mathbf{W}} = \hat{\mathbf{M}} \cdot \hat{\mathbf{S}}. \tag{9.55}$$

This representation, however, is only unique up to an affine transformation. The second component of the factorization method algorithm derives the transformation matrix, which transforms the $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ into the true motion and structure matrices, **M** and **S**, via [150][151]:

$$\mathbf{M} = \hat{\mathbf{M}} \cdot \mathbf{A},$$

and (9.56)

$$\mathbf{S} = \mathbf{A}^{-1} \cdot \hat{\mathbf{S}}.$$

The matrix **A** can be calculated from a set of constraints, called the metric constraints, which are of the form [150][151]:

$$\hat{\mathbf{i}}_f \cdot \mathbf{A} \cdot \mathbf{A}^{\mathbf{T}} \cdot \hat{\mathbf{i}}_f = 1,$$

$$\hat{\mathbf{j}}_f \cdot \mathbf{A} \cdot \mathbf{A}^{\mathbf{T}} \cdot \hat{\mathbf{j}}_f = 1, \tag{9.57}$$

$$\hat{\mathbf{i}}_f \cdot \mathbf{A} \cdot \mathbf{A}^{\mathbf{T}} \cdot \hat{\mathbf{j}}_f = 0.$$

Once **A** is computed, the true structure matrix can be derived, and the 3-dimensional global locations of all the tracked feature points have been computed.

9.2.2 Shape from Deformation

The algorithms defined in Section 9.2.1 to derive shape from motion relied on point correspondences between image frames. The 3-dimensional structure is computed from the relative motions of these points, as the viewing angle is changed. Consequently, it is imperative to define image points that are suitable for tracking [156]. Recall the results of the Hausdorff point-set matching for motion estimation that were presented in Chapter 7. We saw that during severe 3-dimensional motion, the point-set matching approach failed to find point correspondences, even on a point-set level. The structure from motion algorithm requires point-to-point matching which is even more difficult, due to the severe deformations in the object being matched. For our application, we need to develop an algorithm that will be able to operate through these extreme deformations.

There has recently been research work on using the deformations in an image sequence to infer the direction of motion of an object [151]. Unfortunately, as in the traditional SFM algorithms, this deformation-based approach still relies on point tracking to define the deformations. Due to the unique position of the camera relative to the object of interest (the occupant) in the airbag suppression application, dramatic changes are witnessed in the image, as the occupant moves in the vehicle.

Of particular interest is the behavior of the bounding ellipse of the occupant, as she leans in the direction towards or away from the camera (into and out of the image plane). As shown in Figure 9.7, motion in the direction perpendicular to the image plane (toward the camera) causes significant deformations in the bounding ellipse. Specifically, the ellipse attains maximum eccentricity when the occupant is leaning toward the window, which, due to our camera angle, is when the major axis of the occupant is nearly aligned within the imaging plane of the camera. The ellipse attains minimum eccentricity (becomes a circle) when the occupant is leaning towards the camera, and hence the major axis of the occupant is oriented nearly perpendicular to the imaging plane.

Note also that the turning of the occupant's head has no net effect on the ellipse shape. This can be seen when comparing the first and the last image in the sequence in Figure 9.7. The occupant's head is slightly turned towards the driver, but there is no net effect in the bounding ellipse. However, this type of motion would have a dramatic effect if point feature correspondences had been used.

It has been recommended that the approach taken when designing reconstruction algorithms should be based on phenomenological analysis [148]. It is also recommended that one should "design [SFM] algorithms specifically for their problem domains" [148]. In light of this recommendation, we specifically utilize the observed phenomenology of the occupant airbag suppression problem that was shown in Figure 9.7. Based on the deformations of the bounding ellipse of the occupant, we propose a new method, called *structure from deformation*. The power of our structure from deformation algorithm lies in the fact that it does not require feature-

level tracking and point correspondences, but rather works on the global deformations of the occupant as she moves within the vehicle.

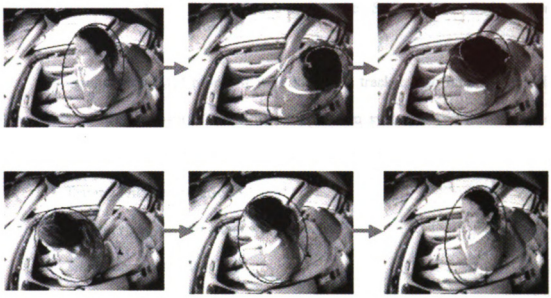


Figure 9.7: Demonstration of the effects of ellipse deformation resultant from out-of-plane rotations of the occupant.

The shape from deformation algorithm that we have defined uses the fact that, as the occupant moves in and out of the image plane of the camera, the projected ellipse of the occupant changes shape. There are three key technical elements that must be developed to support the shape from deformation algorithm:

1) mathematical representation of the projective effects in the imagery,
2) developing an IMM Kalman filter implementation to describe the out-of-plane motions, and
3) defining the extended Kalman filter mathematics to support the IMM tracking.

At this point in our research we will define the IMM architecture, and propose the other two algorithms for a future research effort.

A IMM Modeling of the Shape from Deformation Problem

The goal of the IMM processing for the shape tracker is to infer the out-of-plane angle of the occupant. The airbag suppression system uses three states to explain the possible 3-dimensional orientation of the occupant, namely:

1) leaning left (toward driver),

2) leaning right (toward window), and

3) sitting upright (no leaning).

This implies that the matrix for the state transition probabilities for the out-of-plane angle will be a 3x3 matrix:

$$p(s \mid m) = \begin{bmatrix} p_{left \mid left} & p_{left \mid center} & p_{left \mid right} \\ p_{center \mid left} & p_{center \mid center} & p_{center \mid right} \\ p_{right \mid left} & p_{right \mid center} & p_{right \mid right} \end{bmatrix}. \qquad (9.58)$$

Note that unlike in the case of the motion tracker, there are some transitions that are not possible for the shape tracker. The occupant cannot transition from a leaning-left to a leaning-right orientation, without first transitioning through the center orientation. Therefore, the state transition matrix for the shape tracker reduces to:

$$p(s \mid m) = \begin{bmatrix} p_{left \mid left} & p_{left \mid center} & 0 \\ p_{center \mid left} & p_{center \mid center} & p_{center \mid right} \\ 0 & p_{right \mid center} & p_{right \mid right} \end{bmatrix}. \qquad (9.59)$$

As with the motion tracker, there are no simple equations for computing each of these state transition probabilities. For these we empirically test a range of values, and use the values to minimize the orientation error for a set of training crash video sequences.

9.2.3 Prediction of Intrusion into the ASZ

The final task of the tracking subsystem is to predict potential intrusions into the Automatic Suppression Zone (ASZ) by the occupant's head or torso. The motion tracker provides the predictions of the occupant's center of mass through the state variable $x_{centroid}$, and his angular tilt forward towards the airbag through the state variable θ. In order to predict the time of the intrusion into the ASZ, two tasks must be performed: (i) provide predictions of the occupant's position and orientation of his bounding ellipse at an output rate of 5 milliseconds, and (ii) detect if the forward-most edge of his bounding ellipse has entered the ASZ.

Since the shape of the ellipse is defined by the occupant's motion, the dynamics occur at a much slower rate. Therefore, for the purpose of ASZ intrusion detection, we maintain the shape of the ellipse to be constant between camera frames. As the test results will show, this was a reasonable simplifying assumption.

A High Speed Ellipse Prediction

The prediction of the position and orientation of the bounding ellipse at the high output data rate is relatively straightforward. Recall, the state vector prediction equation:

$$\mathbf{x}(k \mid k) = \sum_{s=1}^{N} \mathbf{x}_s(k \mid k) * \mu_s(k), \qquad (9.60)$$

where the term $\mathbf{x}_s(k \mid k)$ is the state vector for dynamics model *s*, and the final state vector is the output of the weighted contribution of all of the states. In order to determine the position of the ellipse at each intermediate point, we must use the IMM prediction equation:

$$\mathbf{x}_m(k \mid k-1) = \Phi_m^{output}(k-1) * \mathbf{x}_m(k-1 \mid k-1), \qquad (9.61)$$

where now the time between *k-1* and *k* is 5 milliseconds, which is factored into the output state transition matrix, $\Phi_m^{output}(k-1)$.

At each of these updates, we must compute if the forward-most point of the ellipse has crossed the ASZ boundary.

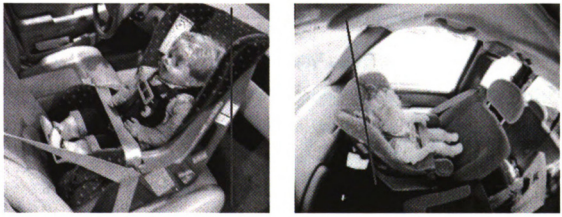
B Calculation of Ellipse Intersection with the ASZ

Recall that a vertical plane in front of the airbag defines the ASZ boundary. Unfortunately, a vertical line in the vehicle is warped into a line with a slope in the image of the interior of the vehicle due to the perspective effects of the camera, as shown in Figure 9.8. Therefore, we must compute the intersection of an ellipse at

some orientation angle, θ, with a line of a pre-defined slope. Recall the equation for an ellipse at some orientation θ is:

$$f(x,y) = 0 = \frac{(x\cos(\theta) + y\sin(\theta))^2}{a^2} + \frac{(y\cos(\theta) - x\sin(\theta))^2}{b^2} - 1, \quad (9.62)$$

where a and b are the major and minor axes, and θ is the orientation angle of the ellipse. Now we must solve this equation for y, compute the slope dy/dx, and determine the point along the ellipse that has the same slope. We want to find the point of equal slopes, because this would be the very first point to cross the ASZ boundary line on the ellipse, as shown in Figure 9.9.



(a) (b)

Figure 9.8: Deformation of the vertical ASZ boundary due to camera perspective effect, (a) ASZ as vertical line in the 'real-world' view, and (b) the ASZ line sloped due to perspective.

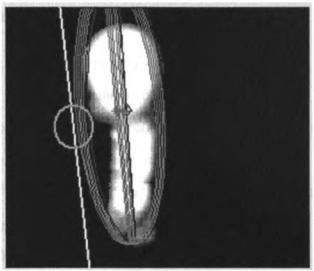


Figure 9.9: Geometry of occupant ellipse and ASZ boundary intersection, with intersection point highlighted by a circle.

First, define the ellipse as a function *y(x)* as follows:

$$y(x) = \frac{1}{2(a^2 \cdot \sin(\theta)^2 - b^2 \cdot \sin(\theta)^2 - a^2)} \cdot \{2b^2 \cdot x \cdot \cos(\theta)\sin(\theta) - 2a^2 \cdot x \cdot \cos(\theta)\sin(\theta) + 2 \cdot [b^4 \cdot x^2 \cdot \cos(\theta)^2 \sin(\theta)^2 - 2 \cdot b^2 \cdot a^2 \cdot x^2 \cdot \cos(\theta)^2 \sin(\theta)^2 + a^4 \cdot x^2 \cdot \cos(\theta)^2 \sin(\theta)^2 - 4 \cdot a^4 \cdot b^2 \cdot \sin(\theta)^2 + a^2 \cdot b^2 \cdot x^2 \cdot \sin(\theta)^2 - 2 \cdot a^2 \cdot b^2 \cdot x^2 \cdot \sin(\theta)^4 + a^4 \cdot x^2 \cdot \sin(\theta)^4 + a^2 \cdot b^4 \cdot \sin(\theta)^2 - b^4 \cdot x^2 \cdot \sin(\theta)^2 + b^4 \cdot x^2 \cdot \sin(\theta)^4 + a^4 \cdot b^2 - a^2 \cdot b^2 \cdot x^2 - a^4 \cdot x^2 \cdot \sin(\theta)^4]^{1/2}\}. \quad (9.63)$$

Now the value of x on the ellipse where the slope of the ellipse is equal to the slope of the line is defined by the equation:

$$
\begin{aligned}
x(\frac{dy}{dx} = slope) = -\Big[& a^2 \cdot \sin(\theta)^2 - slope^2 \cdot a^2 \cdot \sin(\theta)^2 - \\
& 2 \cdot a^2 \cdot slope \cdot \cos(\theta)\sin(\theta) + slope^2 \cdot a^2 + slope^2 \cdot b^2 \cdot \sin(\theta)^2 + \\
& 2 \cdot b^2 \cdot slope \cos(\theta)\sin(\theta) - b^2 \cdot \sin(\theta)^2 \Big]^{1/2} \cdot \\
& \Big[b^2 \cdot \cos(\theta)\sin(\theta) + a^2 \cdot \cos(\theta)\sin(\theta) + slope \cdot a^2 \cdot \sin(\theta)^2 - \\
& slope \cdot b^2 \cdot \sin(\theta)^2 - slope \cdot a^2 \Big] / \\
& \Big\{ -a^2 \cdot \sin(\theta)^2 + slope^2 \cdot a^2 \cdot \sin(\theta)^2 + 2 \cdot a^2 \cdot slope \cdot \cos(\theta)\sin(\theta) \\
& - slope^2 \cdot a^2 + b^2 \cdot \sin(\theta)^2 - slope^2 \cdot b^2 \cdot \sin(\theta)^2 \\
& - 2 \cdot b^2 \cdot slope \cdot \cos(\theta)\sin(\theta) - b^2 \Big\},
\end{aligned}
\tag{9.64}
$$

where *slope* is the slope of the ASZ boundary line in the image. We will define $x_{forward\_most} = x(\frac{dy}{dx} = slope)$, and we must now offset this position by the centroid of the ellipse by:

$$
x_{forward\_most} = x_{centroid} - x_{forward\_most} \cdot \tag{9.65}
$$

Lastly, we must now compute the y-value on the ellipse using Equation (9.75) above, $y_{forward\_most} = y(x_{forward\_most})$. Then we need to offset this y-value by its centroid according to:

$$
y_{forward\_most} = y_{forward\_most} + y_{centroid} \cdot \tag{9.66}
$$

Once we have computed this location, $(x_{forward\_most}, y_{forward\_most})$, on the bounding ellipse of the occupant, we can determine if this point is to the right or to the left of the line defining the ASZ, by:

$$
x_{ASZ} = C_{ASZ}(1) * y_{forward\_most} + C_{ASZ}(2), \tag{9.67}
$$

where $C_{ASZ}(1)$ and $C_{ASZ}(2)$ are the two coefficients to define the ASZ line. Now if $x_{forward\_most} \leq x_{ASZ}$ then we declare an ASZ intrusion.

9.3 Tracking Results

The tracking results focus on the performance of the motion-tracking component for the airbag suppression application. The results for the motion tracking are divided into two categories: (i) results for the model switching behavior of the system, and (ii) results for the time delay in detecting intrusions into the ASZ. The results for each of these key performance metrics are presented in the next two subsections. The tracker subsystem is tested in two different methods: (i) a highly repeatable and quantitative method based on using our ASZ tester, and (ii) a more variable and qualitative method involving actual drive testing. The quantitative testing allows the intrusion detection capability to be rigorously tested against a known target motion path. The qualitative method involves using real people in drive tests and monitoring the overall quality of the ellipse fit about the occupant, and also monitoring the behavior of the ellipse as the occupant moves within the vehicle.

9.3.1 Results of Model Transitioning in IMM

Figure 9.10 shows the model probabilities for the crash from a video sequence from the robotic test fixture that simulates a constant acceleration pre-crash braking event at 0.86g. Recall from Chapter 1 that 0.86g is a typical deceleration rate for pre-crash braking. In this sequence, the dummy is stationary for the first two frames, and then begins its motion profile. After these initial frames, the tracker quickly transitions to human motion, as the dummy is initially moved in the crash sequence. After only two additional frames, the tracker determines that the dynamics exceed what is expected for natural human motion, and the system transitions to pre-crash model. The system remains in the pre-crash model through the crash event, and for a few moments afterwards, as is expected from the intentional crash model hysteresis discussed earlier.

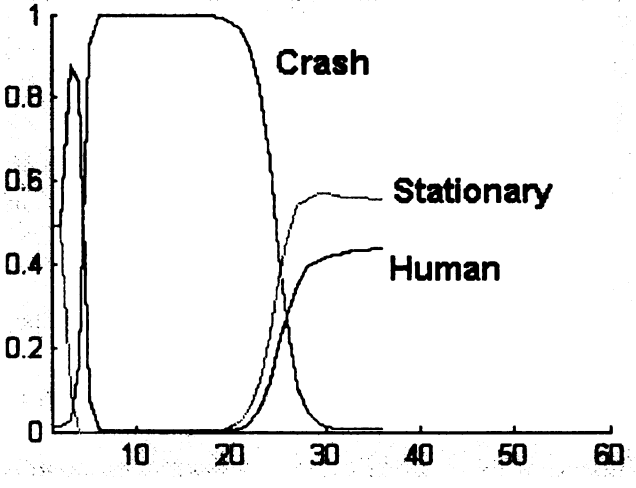


Figure 9.10: Variations in the model likelihoods for each of the motion types through a pre-crash braking sequence.

Figure 9.11 shows the same model probabilities plot for a human occupant moving in the seat at normal human motion rates, and also sitting still for periods of time. Note the transitions (e.g., frame *15*) from the human model to the stationary model during these intervals. Also, note that there is no hysteresis during the transitions between these states. The use of the IMM architecture, rather than the HMM model, is particularly useful in this sequence.

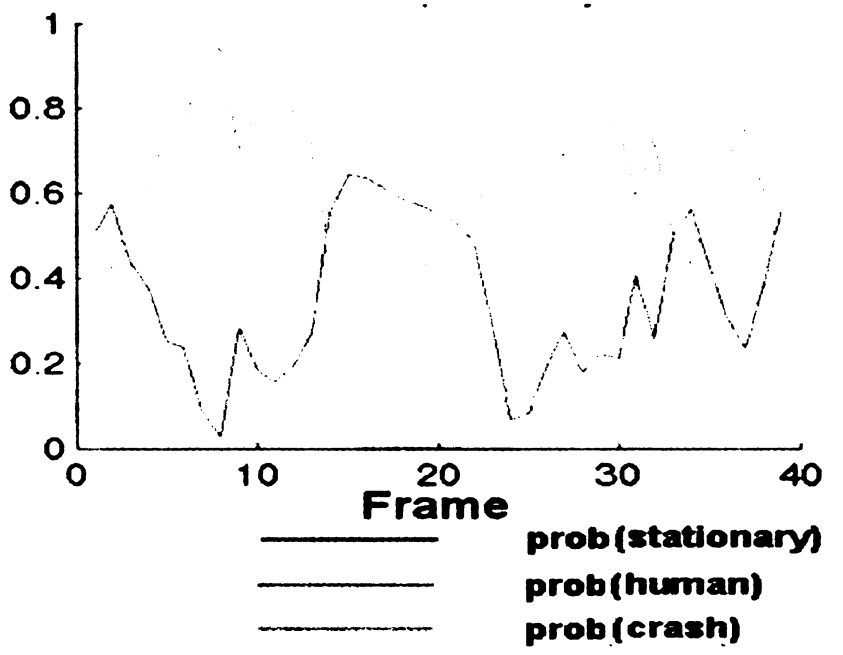


Figure 9.11: Variations in the model likelihoods for each of the motion models during normal human motion.

In the interval from roughly frame 15 to frame 22, there is only a slightly higher probability that the occupant is stationary. This is because the occupant is moving very slowly through a period of transitioning from moving in one direction, stopping, and then reversing direction. The lack of a clear transition to stationary state shows why the IMM architecture is superior to the HMM architecture. The mixing of the stationary and human motion states into a final state allows us to model

intermediate dynamic states smoothly, and without any artifacts, from the discretization of the motions.

9.3.2 Results of Intrusion Time and Positional Accuracies

The results for the intrusion prediction time errors are provided in Figure 9.12. Note the relative Gaussian distribution of the intrusion detections, and the fact that there are negative time errors. This is due to the fact that the tracker system is actually predicting ahead in time, when the intrusion is to occur. When the tracker predicts that the intrusion will be sooner than it actually occurs, a negative intrusion time error is recorded. Figure 9.13 shows the resultant track accuracies during the entire crash event. The worst-case errors were on the order of less than ± 5 pixels during a sequence that includes a transition from fully stationary state to the high-speed pre-crash braking state.

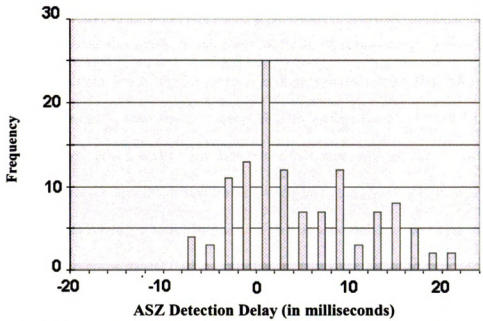


Figure 9.12: Number of the times that the tracker computed an ASZ intrusion compared to the true time of intrusion.

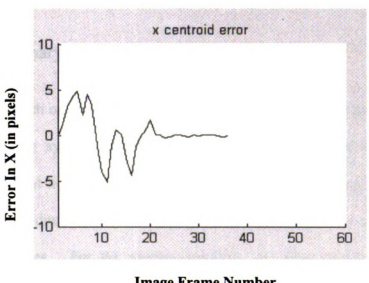


Figure 9.13: Error of the tracker in the position of the dummy during a pre-crash braking sequence in the direction towards the airbag (units are in pixels).

Also note from Figure 9.12 that the system has a mean intrusion delay of 0 milliseconds, and the standard deviation of the intrusions is roughly 7 milliseconds. Additionally, very few of the intrusion delays ever exceeded the NHTSA mandate of 20 milliseconds. Thus, we have been able to successfully demonstrate a system capable of typically detecting the intrusion of the occupant within 10 milliseconds, and rarely worse than 20 milliseconds, while operating a sensor system that only provides updates every 33 milliseconds. This was accomplished by using an effective predictive dynamics model. Also note that by using the IMM architecture, we have been able to simultaneously minimize the track lag (the positive delay in the intrusion detection), as well as the track jitter (the negative delay in intrusion detection due to the tracker being too sensitive).

9.4 Summary

Our research on occupant tracking has been focused in two areas: (i) defining a motion tracking system capable of accurately predicting the intrusion of the occupant into the at risk zone of the airbag, and (ii) a shape tracking system that can predict the three-dimensional motions of an occupant from a stream of two-dimensional images. For the motion tracking, we have shown that using the interacting multiple model (IMM) architecture for tracking the occupants provides excellent results in terms of prediction of intrusion into the at risk zone. The tracker simultaneously minimizes the effects due to track lag (delay in estimated intrusion)

and track jitter (false alarms of intrusion). The tracker was also shown to be capable of meeting the 20 milliseconds intrusion detection time mandated by NHTSA.

In the shape tracking, we have proposed a new mechanism for inferring three-dimensional shape, called shape from deformation. We demonstrated the effects on the bounding ellipse due to the occupant performing three-dimensional motions of the occupant. The placement of the camera system and the modeling of the occupant by the bounding ellipse model allow us to use this unique pose estimation algorithm. The phenomenology of the shape from deformation was demonstrated, and the architecture for using it for deriving estimates of the 3-D pose of the occupant was introduced. As an area of future research, the actual mathematics of the shape from deformation will be studied and formalized.

Chapter 10.
Conclusions and Future Work

We have provided the system architecture for a real-time occupant classification and tracking system for smart airbag suppression. We have developed the theory and provided results for each of the critical subsystems that comprise this smart airbag system. We will review the contributions to pattern recognition and human motion tracking that arose through this research. We will also outline areas of interest for future research.

10.1 Research Contributions

The field of automatic suppression of airbags is relatively new. To date these systems have performed static suppression where the classification of the occupant is determined by a contact-based sensor such as a seat weight sensor. The application of computer vision to this application is very new, and has also been focused primarily on performing occupant classification. Machine vision systems have been successfully applied to relatively controlled environments, such as for manufacturing. In uncontrolled environments, such as for surveillance, vision systems generally have a human in the loop to monitor the performance of the system. The airbag suppression application is unique in that it attempts to use computer vision in an

application that is neither in a controlled environment, nor has a human operator to assist in decision-making.

In this thesis we have developed a computer vision-based approach to airbag suppression that provides the robustness required for an autonomous system fielded in a relatively uncontrolled environment. The system utilizes the available constraining information in multiple ways, such as the known vehicle interior, the knowledge of the types of occupants we must detect, the model of the types of motion an occupant can experience, and lastly the contextual information derived from the real-time image sequence of the occupant in the vehicle. In this last level of contextual processing, the real-time video information provides an estimate of the occupant's dynamics, as well as a contextual framework for improving the classification accuracy over time. This is accomplished through the temporal fusion of the sequence of classification results from multiple frames.

The knowledge of the vehicle is used in the image segmentation processing to extract the occupant from the vehicle interior. This is implemented as a background correlation test that uses local correlation information to identify the occupant. It was shown to provide superior performance over other background subtraction methods, such as eigen-image representation. To further enhance the segmentation, we developed a contextual segmentation algorithm based on the wrapper methods for feature selection.

In the proposed wrapper-based segmentation, we label the output of the background correlation into blobs of common grayscale using the EM algorithm. We then use the forward sequential search feature selection method to choose the

grouping of these blobs that provides the highest classification accuracy for each of the known classes we are detecting. This algorithm was shown to provide segmentations as accurate as human hand segmentation. Since it uses the full contextual information of the object being segmented, the classifier is used to provide the metric for the quality of the segmentation.

The interacting multiple models (IMM) tracker captures the type of motion experienced by the occupant. It is similar to the HMM modeling that is common in human motion analysis, but differs critically in that the final state of the occupant is not limited to one of the known states of the HMM, but can be a probabilistic combination of states. We model the occupant as being capable of stationary, human, or pre-crash braking motion dynamics. The final motion predicted by the tracker is then a linear combination of the predictions of each of these states of motion. The IMM mechanism was shown to be capable of meeting the very strict NHTSA requirement of 20 milliseconds reaction time, while minimizing false suppressions.

We also derived considerable contextual information to be from the real-time image sequence of the occupant. By recognizing the fact that the vehicle has been in motion, and that the door has not been opened, then it is highly unlikely that the true occupant class has changed. While we cannot know for sure at any time what the true class is, we have developed a mechanism, based on belief revision from the theory of evidential reasoning, to optimally integrate the temporal stream of classification results. This stream of classification results is integrated using the Dempster-Shafer rules of belief revision, which allows us to classify the occupant to a level of abstraction supported by the image information, rather than to the a priori atomic

classes. This method of contextual processing was shown to provide excellent results for our challenging application, where the true class of the occupant is not changing, but their motions and sequence of postures can make them appear as a different class.

To enhance the performance of the classification system against the various illumination effects, we investigated numerous feature spaces and chose the edge-based representations with an additional constant-false alarm rate edge detector. We also developed a two-stage filter-based feature selection algorithm that ranks features based on their discrimination ability using the Mann-Whitney test, and then selects features based on minimizing the correlation between the final feature set using the Spearman-R test. This algorithm was tested in comparison to a number of existing filter and wrapper methods for feature selection, and was shown to provide equivalent or superior performance, with a dramatically reduced computational burden.

To enhance the information derived from the classification process, we defined an adaptation of the k-nearest neighbor rule that uses the k-nearest distances to each of the classes to provide a continuous metric of classification quality, rather than the simple voting mechanism provided by the traditional k-nearest neighbor decision rule. This provides classification information at a much finer resolution than is possible with simple voting, which is useful for both the temporal contextual processing and the integrated classifier\segmenter.

We also defined a new mechanism for motion segmentation based on mutual information. In this method we propose analyzing image streams based on the flow of information in the image, rather than based on the flow of grayscale values, as in

traditional optical flow. The initial analysis of the approach shows that it may be inherently immune to illumination changes.

Lastly, we developed a model of the occupant for tracking based on fitting the head and torso of the occupant to a bounding ellipsoid. This model provided a simplification of the occupant that allows the motion tracker to be relatively immune to hand and arm motions that may confuse a system into thinking the occupant is too close to the airbag. Additionally, we have defined a new mechanism for inferring the 3-dimensional pose of the occupant based on the relative deformations of the ellipse over time, and have referred to this new approach as shape from deformation.

10.2 Future Research

While the system we have developed to date provides excellent performance, there is always room for additional improvement, since the system is safety related, and 100 percent protection of all occupants is always a goal to be reached. Since this is a system where there is no possibility for a reject option, we must continue to find ways of ensuring the proper deployment at all times, either through static or dynamic suppression algorithms, or through a combination of both. We conclude this thesis by identifying some of the areas where continued research is possible, and we also identify other areas of research, where some of the technologies we have developed may be applied.

The areas of research that may improve the overall system performance include:

i) In occupant segmentation and classification we showed excellent results for the stand-alone classification system. However, in order to ensure generalizability to all illumination conditions, the integrated wrapper-based segmenter/classifier may be the preferred system. To improve the performance of the wrapper approach, we propose to investigate means for including additional information, such as the interior edge features that were used for the stand-alone classifier to supplement the silhouette features currently used. In addition, we propose investigating other random blob combination mechanisms, such as genetic algorithms that may perform as well as the FSS method, but with reduced computational complexity.

ii) For occupant tracking the first area of continued research is in the utilization of mutual information for motion segmentation. The phenomenology we demonstrated in this thesis shows this method holds considerable potential in developing a method of motion segmentation that is inherently immune to illumination effects. Specific algorithms for utilizing this measure must be investigated, and then the method must be compared against existing methods.

iii) A second area of research in occupant tracking is the continuation of the research in the shape from deformation method. Again, the phenomenology of the approach was demonstrated in this thesis, and now the specific implementations of the extended Kalman filter must be developed to handle the variety of perspective models, including Euclidean, para-perspective, and full perspective.

In addition to these research areas within airbag suppression, there are a number of interesting areas where some of the technologies developed in this thesis may be applied. These research areas of interest include:

i) Application of the Mann-Whitney and feature correlation processing to massive-scale feature selection problems. We demonstrated a significant speed improvement for this method over wrapper methods. It would now be interesting to apply the proposed method to some existing large and massive scale feature selection problems (> 1000 features) to determine the performance. These large feature spaces are common in data mining applications, which is currently a very popular area of research.

ii) The application of the wrapper-based segmenter/classifier to image database query and retrieval. Image database retrieval is also currently a very important research area, and the wrapper method provides a significant advantage over existing methods of retrieval, since it allows the simultaneous segmentation and recognition of the objects of interest.

iii) The application of the concepts of evidential reasoning, and its set theoretic concepts to classifier combination, is another area of potential interest. The approach worked well for the temporal integration of information, and its underlying set theory approach may prove powerful for integrating classifiers that provide object classification at different levels of abstraction.

iv) The concepts of the IMM framework can be used to perform tasks such as human gait analysis, since it is capable of providing a detailed temporal model of the transitions between states, rather than only an identification of

the sequence of states. This feature may provide an additional cue to make gait recognition more robust.

v) Lastly, the concepts behind the shape from deformation algorithm may be applicable to the more general problem of human motion analysis, since traditional methods for pose estimation require rigid bodies, while our method actually uses the deformations of the body to infer the posture of the subject.

BIBLIOGRAPHY

[1] P. Mengel, G. Doemens, and L. Listl, "Fast range imaging by CMOS sensor array through multiple double short time integration (MDSI)", *Proc. IEEE International Conference on Image Processing*, pp. 169-172, 2001.

[2] A.P. Corrado, S. Decker, and P. Benbow, "Automotive occupant sensor system and method of operation by sensor fusion", *US Patent 5482314.*

[3] J.H. Semchena, E. Faigle, R. Thompson, J. Mazur, and C. Steffens Jr., "Apparatus and method for controlling an occupant restraint system", *US Patent 5531472.*

[4] L. Eisenmann, Y. Lu, S. Sauer, and C. Marschner, "Process for the capacitive object detection in the case of vehicles", *US Patent 6442464.*

[5] V.C. Patel, T. Thuen, J.K. Hanninen, and H.T. Kuisma, "Force sensor assembly'', *US Patent 6089106.*

[6] P.B. Blakesley, "Vehicle seat weight sensor", *US Patent 6407347.*

[7] Discussions with fellow engineers while author was employed at Takata, Inc., 1995-1999.

[8] J. Krumm and G. Kirk, "Video occupant detection for airbag deployment," *Proc. IEEE Workshop on Applications of Computer Vision*, pp. 30-35, 1998.

[9] Y. Owechko, N. Srinivasa, S. Medasani, and R. Boscolo, "Vision-based fusion system for smart airbag applications", *IEEE Intelligent Vehicle Symposium*, 2002.

[10] http://www.uspto.gov/patft/index.html (USPTO patent search website)

[11] http://www.highwaysafety.org/safety_facts/qanda/airbags.htm

[12] National Highway Transportation & Safety Administration, Federal Motor Vehicle Safety Standard # 208, Dec 2001.

[13] General Accounting Office, "*Vehicle Safety - Technologies, Challenges, and Research and Development Expenditures for Advanced Air Bags*", June 2001.

[14] National Highway Transportation & Safety Administration, *1998 Motor Vehicle Occupant Safety Survey, Volume 3, Child Safety Seat Report*, July, 2000.

[15] http://www.nhtsa.dot.gov.

[16] http://www.exponent.com/practices/vehicles/TEC.

[17] Eaton Corporation, *The NHTSA Positioning and Orientation of the Front Outboard Passenger Seat Occupant*, 2002.

[18] M. Farmer, R.L. Hsu, and A. Jain, "Interacting multiple models (IMM) Kalman filter for robust high speed human motion tracking", *Proc. IEEE International Conference on Pattern Recognition*, vol. 2, pp. 20-23, 2002.

[19] M. Farmer and A. Jain, "Occupant classification system for automotive airbag suppression", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 756-761, 2003.

[20] M. Farmer and A. Jain, "Integrated segmentation and classification for automotive airbag suppression", *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 1053-1056, 2003.

[21] A.K. Jain and C. Dorai, "Practicing vision: Integration, evaluation and applications", *Pattern Recognition*, vol. 30, no. 2, pp. 183-196, 1997.

[22] O. D. Trier, A.K. Jain and T. Taxt, "Feature extraction methods for character recognition – A survey", *Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.

[23] R.C. Veltkamp and M. Hagedorn, "State-of-the-art in shape matching", In *Principles of Visual Information Retrieval*. pp. 87-119, Springer, 2001.

[24] S. Loncaric, "A survey of shape analysis techniques", *Pattern Recognition*, vol. 31, no. 8, pp. 983-1001, 1998.

[25] M. Safar, C. Shababi, and X. Sun, "Image retrieval by shape: A comparative study", *Proc. IEEE International Conference on Multimedia and Exposition* pp. 141-144, 2000.

[26] M. Safar, C. Shababi, and C. Tan, "Resiliency and robustness of alternative shape-based image retrieval techniques", *Proc. IEEE International Database Engineering and Applications Symposium*, pp. 337-348, 2000.

[27] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical pattern recognition: A review", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, 2000.

[28] A.K. Jain. *Image Processing and Pattern Recognition,* 2-day class presented to Eaton Corporation, April 2003.

[29] M.R. Teague, "Image analysis via the general theory of moments", *J. Opt. Soc. Amer.* vol. 70, no. 8, pp. 920-930, 1980.

[30] R. Mukundan, S.H. Ong, and P.A. Lee, "Image analysis by Tchebichef moments", *IEEE Trans. on Image Processing*, vol. 10, no. 9, pp. 1357-1364, 2001.

[31] C-H Teh and R.T. Chen, "On image analysis by methods of moments", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 496-512, 1988.

[32] G. Arfken, *Mathematical Methods for Physicists*, Academic Press, 1970.

[33] R. Mukundan and K.R. Ramakrishnan, *Moment Functions in Image Analysis*, World Scientific, 1998.

[34] R.R. Bailey and M. Srinath, "Orthogonal moment features for use with parametric and non-parametric classifiers", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 389-398, 1996.

[35] M. Tuceryan, "Moment based texture segmentation", *Pattern Recognition Letters*, vol. 15, no. 7, pp. 659-668, July 1994.

[36] Q. Lu and K.H. Lee, "Recognition of Chinese characters by moment feature extraction", *IEEE International Conference on Computer Processing of Oriental Languages*, pp. 566-571, 1997.

[37] J. Flusser and T. Suk, "On the calculation of image moments", Research Report #1946, Institute for Information Theory and Automation, Academy of Sciences of the Czech Republic, 1999.

[38] I.M. Spiliotis and B.G. Mertzios, "Real-time computation of two-dimensional moments on binary images using image block representation", *IEEE Trans. Image Processing*, vol. 7, no. 11, pp1609-1615, 1998.

[39] I. Pitas, and A. Venetsanopulios, "Morphological shape decomposition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 38-45, 1990.

[40] M. Pilu, A.W. Fitzgibbon, and R.B. Fisher, "Ellipse-specific direct least-square fitting", *Proc. IEEE International Conf. On Image Processing*, 1996.

[41] A.W. Fitzgibbon, M. Pilu, and R.B. Fisher, "Direct least-square fitting of ellipses", *Proc. IEEE International Conf. On Image Processing*, 1996.

[42] C. Lee, M. Eden and M. Unser, "High-quality image resizing using oblique projection operators", *IEEE Trans. Image Processing*, vol. 7, no. 5, pp. 679-692, 1998.

[43] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge, "Comparing images using the Hausdorff distance", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850-863, 1993.

[44] H. Vincent Poor, *An Introduction to Signal Detection and Estimation*, Springer, 1994.

[45] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2nd edition, Wiley, New York, 2000.

[46] R.P.W. Duin, "A note on comparing classifiers," *Pattern Recognition Letters*, vol. 17, no. 5, pp. 529-536, 1996.

[47] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[48] E. Pekalska and R.P.W. Duin, "Prototype selection for finding efficient representations of dissimilarity data," *Proc. IEEE International Conference on Pattern Recognition*, pp. 37-40. 2002.

[49] J.C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery 2*, pp. 121-167, 1998.

[50] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods,* Cambridge University Press, 2000.

[51] D. Cook, D. Caragea, and V. Honavar, "Visualization for classification problems, with examples using support vector machines", http://www.public.iastate.edu/~dicook/Limn/svm/paper.pdf

[52] Http://svmlight.joachims.org/

[53] R.P.W. Duin and D. Tax, "Experiments with classifier combining rules", *Proc. Multiple Classifier Systems 2000*, LNCS 1857, Springer, pp. 16-29, 2000.

[54] S. Bay, "Combining nearest neighbor classifiers through multiple feature subsets", *Proc. Fifteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 37-45, 1998.

[55] L.I. Kuncheva and L.C. Jain, "Nearest neighbor classifier: Simultaneous editing and feature selection", *Pattern Recognition Letters*, vol. 20, no. 11, pp.1149-1156, 1999.

[56] A.K. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153-158, Feb. 1997.

[57] M. Dash and H. Liu, "Feature selection for classification", *Intelligent Data Analysis*, vol. 1, pp. 131-156, 1997.

[58] E.M. Bicici, "A tutorial on feature selection", http://www4.ncsu.edu/~embicici/Academic/A_turorial_on_feature_selection.htm, 2001.

[59] D. B. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms", *Proc. Eleventh International Conference on Machine Learning*, pp. 293-301, 1994.

[60] K. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, May 1983.

[61] A.L. Blum and P. Langley, "Selection of relevant features and examples in machine learning", *Artificial Intelligence*, vol. 20, pp. 245-272, 1997.

[62] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection", *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119-1125, Nov 1994.

[63] P. Somol, P. Pudil, J. Novovicova, and P. Paclik, "Adaptive floating search methods in feature selection", *Pattern Recognition Letters*, vol. 20, no. 11-13, pp.1157-1163, 1999.

[64] D. Koller and M. Sahami, "Toward optimal feature selection", *Proc. 13th International Conference on Machine Learning*, Morgan Kaufman, pp. 197-243, 1996.

[65] D.W. Aha and R.L. Bankert, "A comparative evaluation of sequential feature selection algorithms", in *Learning from Data: AI and Statistics*, Springer-Verlag, 1996.

[66] R. Kohavi and G.H. John, "The wrapper approach", In *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer Academic, pp.33-50, 1998.

[67] M. Kudo and J. Sklansky, "Classifier-independent feature selection for two-stage feature selection", *Lecture Notes in Computer Science*, vol. 1451, pp. 548-554, 2000.

[68] J. Bins and B.A. Draper, "Feature selection from huge feature sets", *Proc. IEEE International. Conference on Computer Vision,* vol. 2, pp. 159-165, 2001.

[69] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection", In *Pattern Recognition in Practice*, vol. 4, pp. 403-413, Elsevier, 1994.

[70] K. Torkkola, "On feature extraction by mutual information maximization", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 821-824, 2002.

[71] D.P.W. Ellis and J. A. Bilmes, "Using mutual information to design feature combinations", *Proc. Int. Conf. on Spoken Language Processing*, vol. 3 pp.79-82, 2000.

[72] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers", *Pattern Recognition*, vol. 33, no. 1, pp. 25-41, 2000.

[73] R. Lowry, *VassarStats: Web Site for Statistical Computation*, http://faculty.vassar.edu/lowry/VassarStats.html.

[74] D.Q. A and B.T. Hong, "Statistical data analysis", http://www.netnam.vn/unescocourse/statistics/stat_frm.htm.

[75] R. J. Larsen and M.L. Marx, *An Introduction to Mathematical Statistics and its Applications*, Prentice-Hill, 1986.

[76] K.J. Han and A.H. Tewfik, "Eigen-image based video segmentation and indexing", *Proc. IEEE International Conference on Image Processing*, pp. 147-151, 1999.

[77] K. Ohba and K. Ikeuchi, "Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.19, no.9, pp.1043-1048, 1997.

[78] Y. Weiss, "Segmentation using eigenvectors: a unifying view", *Proc. IEEE International Conference on Computer Vision,* pp. 975-982, 1999.

[79] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance", *Proc. IEEE International Conference on Computer Vision*, pp. 255-261, 1999.

[80] K.S. Fu and J.K. Mui, "A survey of image segmentation", *Pattern Recognition*, vol. 13, pp. 3-16, 1981.

[81] R.M. Haralick and L.G. Shapiro, "Survey: Image segmentation techniques", *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 100-132, 1985.

[82] N. R. Pal and S. K. Pal, "A review of image segmentation techniques", *Pattern Recognition*, vol. 26, no. 9, pp. 1277-1294, 1993.

[83] L. Shapiro and G. Stockman, *Computer Vision*, Prentice Hall, 2001.

[84] J.B.T.M. Roerdink and A. Meijster, "A Watershed transform: definitions, algorithms, and parallelization strategies", *Fundamenta Informaticae*, vol. 41, pp. 187-228, 2000.

[85] S. Beucher, "The watershed transformation applied to image segmentation", *Scanning Microscopy International*, suppl. 6, pp. 299-314, 1992.

[86] J. Serra, "Introduction to mathematical morphology", Computer Vision, Graphics, and Image Processing, vol. 35, pp. 283-305, 1986.

[87] J. Shi and J. Malik, "Normalized cuts and image segmentation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol 22, no. 8, pp. 888-905, Aug. 2000.

[88] S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color and texture-based image segmentation using EM and its application to content-based image retrieval", *Proc. IEEE International Conference on Computer Vision*, pp. 675-683, 1997.

[89] C. Carson, M. Thomas, S. Belongie, J. H. Hellerstein, and J. Malik, "Blobworld: A system for region-based image indexing and retrieval", *Proc. International Conference on Visual Information Systems*, 1999.

[90] J.A. Blimes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models", Tech Report TR-97-021, Dept. of Electrical Engineering and Computer Science, Univ. of Washington, April 1998.

[91] M.A.T. Figueiredo and A.K. Jain, "Unsupervised learning of finite mixture models", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol 24, no. 3, pp. 381-396, Mar. 2002.

[92] J. Bilmes, "What HMMs can do", Technical Report UWEETR-2002-0003, Dept. of Electrical Engineering, Univ. of Washington, 2002.

[93] C.A. Bouman and M. Shapiro, "A multi-scale random field model for bayesian image segmentation, " *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp. 162-177, Mar. 1994.

[94] O.D. Trier and A.K. Jain, "Goal-directed evaluation of binarization methods", *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 12, pp. 1191-1201, 1995.

[95] S.X. Yu, R. Gross, and J. Shi, "Concurrent object recognition and segmentation by graph partitioning", *Advances in Neural Information Processing Systems 15*, 2002.

[96] J. Luo and C. Guo, "Perceptual grouping of segmented regions in color images", *Pattern Recognition*, vol. 36, pp. 2781-2792, 2003.

[97] N. Sprague and J. Luo, "Clothed people detection in still images", *Proc. IEEE International Conference on Computer Vision,* vol. 3, pp. 585-589, 2002.

[98] A. Gelb, *Applied Optimal Estimation*, MIT Press, 1974.

[99] M. Pekkarinen, "Multiple model approaches to multisensor tracking", Masters Thesis, Tampere University of Technology, 1999.

[100] G. Welch and G. Bishop, "An introduction to the Kalman filter", TR-95-041, Dept. of Computer Science, University of North Carolina, 2002.

[101] K. Miller and D. Leskiw, *An Introduction to Kalman Filtering with Applications*, Robert E. Kreiger Publishing, 1987.

[102] Y. Bar-Shalom, *Multitarget-Multisensor Tracking*, Artech House, 1990.

[103] H.A.P Blom, and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients", *IEEE Trans. Automatic Control*, vol. 33, no. 8, pp. 780-783, Aug. 1988.

[104] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A survey", *IEEE Trans. on Aerospace and Electronics*, vol. 34, no. 1, pp. 103-122, January 1998.

[105] R.A. Singer, "Estimating optimal tracking filter performance for manned maneuvering targets", *IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-6, no. 4, pp. 473-483, 1970.

[106] M. Moore and J. Wang, "An extended dynamic model for kinematic position", *Journal of Navigation*, vol. 56, pp. 79-88, 2003.

[107] M. Vincze, "Robust tracking of ellipses at frame rate", *Pattern Recognition*, vol. 34, no. 2, pp. 487-498, 2001.

[108] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking", *Int. Journal Computer Vision*, vol. 29, no. 1, pp. 5-28, 1998.

[109] M. Isard and A. Blake, "A mixed-state condensation tracker with automatic model-switching", *Proc. IEEE International Conference on Computer Vision*, pp. 107-112, Jan. 1998.

[110] A. Blake and M. Isard, *Active Contours: The Application of Techniques from Graphics, Control Theory & Statistics to Visual Tracking of Shapes in Motion*, Springer-Verlag, 1998.

[111] J.B.N. Deutscher, B. Bascle, and A. Blake, "Tracking through singularities and discontinuities by random sampling", *Proc. IEEE International Conference on Computer Vision*, vol. 2, pp. 1144-1149, 1999.

[112] J.K. Aggarwal and Q. Cai, "Human motion analysis: A review", *Computer Vision and Image Understanding*, vol. 73, no 3, pp. 428-440, March 1999.

[113] C. Bregler, "Learning and recognizing human dynamics in video sequences", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 568-574, 1997.

[114] S. Wachter and H. Nagel, "Tracking of persons in monocular image sequences", *Proc. IEEE Non-Rigid and Articulated Motion Workshop*, pp. 1-9, 1997.

[115] I. Haritaoglu, D. Harwood, and Larry S. Davis, "W4: Real-time surveillance of people and their activities", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no 8, pp. 809-830, Aug 2000.

[116] N. Oliver, B. Rosario, and A. Pentland, "A Bayesian computer vision system for modeling human interactions", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831-843, Aug 2000.

[117] C.R. Wren and A.P. Pentland, "Understanding purposeful human motion", *Proc. IEEE International Workshop on Modeling People (MPEOPLE)*, pp. 19-25, 1999.

[118] J. Rittscher and A. Blake, "Classification of human body motion", *Proc. IEEE International Conference on Computer Vision*, pp. 634-639, 1999.

[119] K. Rohr, "Human movement analysis based on explicit motion models", In *Motion-based Recognition*, Kluwer Academic, pp. 171-198, 1997.

[120] D.M. Gavrila, "The visual analysis of human movement: A survey", *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82-98, 1999.

[121] A. Pentland and A. Liu, "Modeling and prediction of human behavior", Perceptual Computing Technical Report No. 433, Media Lab, MIT, 1995.

[122] C. Bregler, "Probabilistic recognition of human actions", Technical Report, Dept. of Computer Science, Univ. of California-Berkeley, 1996.

[123] C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 780-785, 1997.

[124] L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", In *Readings in Speech Recognition*, Science & Technology Books, 1989.

[125] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human actions in time-sequential images using hidden markov models", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 379-385,1992.

[126] A. Witkin, M. Gleicher, and W. Welch, "Interactive dynamics", In *ACM SIGGraph, Computer Graphics*, vol. 24, no. 2, pp. 2-11, March 1990.

[127] S. Beauchemin and J. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, no. 3, pp. 433-467, 1996.

[128] B. Horn and B. Schunck, "Determining optic flow," *Artificial Intelligence*, vol. 17, pp. 185-204, 1981.

[129] J. Barron, D. Fleet and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43-77, 1994.

[130] D. Fleet and A. Jepson, "Computation of component image velocity from local phase information," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 77-104, 1990.

[131] M.J. Black, D.J. Fleet, and Y. Yacoob, "Robustly estimating changes in image appearance", *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 8-31, 2000.

[132] T. Gautama and M. Hulle, "A Phase-based approach to the estimation of the optical flow field using spatial filtering," *IEEE Trans. on Neural Networks*, vol. 13, no. 5, pp.1127-1136, 2002.

[133] S. Periaswamy and H. Farid, "Elastic registration in the presence of intensity variations, " *IEEE Trans. on Medical Imaging*, in press, 2003.

[134] S. Negahdaripour, "Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20 no. 9, pp. 961-979, Sept 1998.

[135] L. Zhang, T. Sakurai, and H. Miike, "Detection of motion fields under spatio-temporal non-uniform illumination", *Image and Vision Computing*, vol. 17, no. 3-4, pp. 309-320, 1999.

[136] A. Jepson and M. Black, "Mixture models for optical flow computation", Tech Rep RBCV-TR-93-44, University of Toronto, Dept of Computer Science, 1993.

[137] Y. Weiss and E.H. Adelson, "A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models" *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 321-326, 1996.

[138] C.L. Fennema and W.B. Thompson, "Velocity determination in scenes containing several moving images," *Computer Graphics and Image Processing*, vol. 9, pp. 301-315, 1979.

[139] M.A. Sutton, W.J. Walters, W.H. Peters, W.F. Ranson, and S.R. McNeil, "Determination of displacement using an improved digital correlation method", *Image and Vision Computing*, vol. 1, no. 3, pp.133-139, 1983.

[140] D.S. Kalivas and A.A. Sawchuk, "A region matching motion estimation algorithm", *Computer Vision Graphics and Image Processing*, vol. 54, no. 2, pp. 275-288, 1991.

[141] B. Bascle, P. Bouthemy, R. Deriche, and F. Meyer, "Tracking complex primitives in an image sequence", Technical Report # 2428, INRIA, 1994.

[142] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger, "Probability distributions of optical flow", *Proc. IEEE Conference on Computer Vision and Pattern Recognition,* pp. 310-315, 1991.

[143] J.P.W. Pluim, J.B.A. Maintz, and M.A. Viergever, "Mutual information based registration of medical images: A survey", *IEEE Trans. Medical Imaging*, vol. 22, pp. 986-1004, 2003.

[144] M.D. Esteban and D. Morales, "A summary of entropy statistics", *Kybernetika*, vol. 31, no.4, pp. 337-346, 1995.

[145] G.A. Tourassi, E.D. Frederick, M.K. Markey, and C.E. Floyd, "Application of the mutual information criterion for feature selection in computer-aided diagnosis", *Medical Physics*, vol. 28, no. 12, pp. 2394-2402, Dec. 2001.

[146] M. Ferraro, G. Boccignone, and T. Caelli, "Entropy based representation of image information", *Pattern Recognition Letters*, vol. 23, no. 12, pp. 1391-1398, 2002.

[147] P. Viola, "Alignment by maximization of mutual information", A.I. Tech Report No. 1548, Artificial Intelligence Laboratory, MIT, June 1995.

[148] J. Oliensis, "A critique of structure-from-motion algorithms", *Computer Vision and Image Understanding,* vol. 80, pp. 172-214, 2000.

[149] S. Christy and R. Horaud, "Euclidean shape and motion from multiple perspective views by affine iterations", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18 no.10, pp.1098-1104, 1996.

[150] T. Morita and T. Kanade, "A sequential factorization method for recovering shape and motion from image streams", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 8, pp. 858-867, 1997.

[151] C. Tomasi and T. Kanade, "Shape and motion from image streams: a factorization method", CMU-CS-91-105, School of Computer Science, Carnegie Mellon University, Jan 1991.

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

[153] R. Hartley, "Projective reconstruction and invariants from multiple images", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 10, pp. 1036-1041, 1994.

[154] H.C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", *Nature*, vol. 293, no. 10, pp. 133-135, Sept 1981.

[155] R. Mohr and B. Triggs, "Projective geometry for image analysis", Tutorial given at *International Society of Photogrammetry and Remote Sensing*, June 1996.

[156] J. Shi and C. Tomasi, "Good features to track", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600, June 1994.

[157] C. Tomasi and J. Shi, "Direction of heading from image deformations", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4326-4333, June 1993.

[158] O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, 1993.

[159] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[160] J. Kohlas, "Mathematical foundations of evidence theory – A theory of reasoning with Uncertain Arguments", *Lectures Presented at the International School of Mathematics*, Erice, Sicily, June 19-25, 1994.

[161] C. Largouet and M.Cordier, "Combining observations and expectations: application to the refinement of an image sequence classification", *Workshop on Fusion of Domain Knowledge with Data for Decision Support*, UAI-2000, June 2000.

[162] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[163] P. Smets, "What is Dempster-Shafer's model", *Advances in the Dempster-Shafer Theory of Evidence*, Wiley, pp. 5-34, 1994.

[164] C. Castel, L. Chaudron, and C. Tessier, "What is going on here? A high level interpretation of sequences of images", *4th European Conference on Computer vision, Workshop on Conceptual Descriptions from Images*, pp. 13-27, 1996.

[165] J. Giarratano and G. Riley, *Expert Systems Principles and Programming*, PWS Publishing Co., 1993.

[166] www.vision3d.com/stereo.html

[167] S. Bapna, Discussions while working together at Eaton Corp.

[168] M. Trajkovic, Discussions while working together at Eaton Corp.

[169] M. Dell'Eva, Discussions while working together at Eaton Corp.

[170] Y. Nievergelt, *Wavelets Made Easy*, Birkhauser, 1999.

[171] B. B. Hubbard, *The World According to Wavelets*, A.K. Peters, 1998.

[172] S. Mallat, "Wavelets for a vision", *Proc. of IEEE*, vol. 84, no. 4, April 1996.

[173] P. Maass and H-G. Stark, "Wavelets and digital image processing", *Surv. Math. Ind.*, vol. 4, pp.195-235, 1994.

[174] D. Shen and H.H.S. Ip, "Discriminative wavelet shape descriptors for recognition of 2-D patterns", *Pattern Recognition*, vol. 32, pp. 151-165, 1999.