12-2003

Multiperspective mosaics and layered representation for scene visualization

Jin-Choon Ng

To the Graduate Council:

I am submitting herewith a thesis written by Jin-Choon Ng entitled "Multiperspective mosaics and layered representation for scene visualization." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Mongi A. Abidi, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Jin Choon Ng entitled "Multiperspective Mosaics and Layered Representation for Scene Visualization." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

[illegible]

Dr Mongi A. Abidi, Major Professor

We have read this thesis
and recommend its acceptance:

[illegible]

Dr Andreas Koschan

[illegible]

Dr Hairong Qi

[illegible]

Dr Seong G. Kong

Accepted for the Council:

[illegible]

Vice Provost and Dean of Graduate Studies

Thesis
2003
.N4

Multiperspective Mosaics and Layered Representation for Scene Visualization

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Jin-Choon Ng
December 2003

Acknowledgements

First and foremost, I would like to thank my parents, Peter and Jocelyn Ng, who promptly stopped making any decisions for me when I turned 18, and have been a steady source of love and support ever since. Making my own choices has sometimes been difficult, and I pined once or twice for their veto to make my life a little easier, but I appreciate their trust a great deal. I can proudly claim all my successes and failures as my own, because, frankly, I now have no one else to blame them on.

Many thanks to Dr. Mongi Abidi, who has supported and counseled me during my two (plus) years with the IRIS Lab. I would not be where I am now without his patience, guidance, and support. Thanks, also, to Dr. Andreas Koschan for the discussions on research, conference papers, thesis revisions, as well as the conversations on various other non-technical topics. People need to talk about those things, too. Thanks also to Dr. Laura Morris Edwards, whose revision of my thesis did wonders for my italics. Many thanks to the other professors who were on my committee – Dr. Seong G. Kong and Dr. Hairong Qi - your time and consideration is greatly appreciated.

Thanks to all my friends at the IRIS Lab: Brad Grinstead, keep terrorizing those clueless free citizens. Faysal Boughorbel, you should be glad there are no more obese cats referring to you by name. Michael Roy, Wunderbar. Yohan Fougerolle, I will come get my Dreamcast back someday. Matt Schultz, please do not sit on me, I would not survive. Tak Motoyama, I have not seen you in months but I promise to come visit your dungeon someday. Justin, you know more about the Segway than any human being I have ever known, so rock on. David Lon Page, or should I say 'Dr. Page'. I forgot what it was I was going to say, Dr. Page, but thank you very much, Dr. Page.

Thanks to all my other friends from Knoxville: Brent Wood, propose to her, already. Kevin Zinn, propose to her, already. Matt Forsythe, bravo, you are a good man. David Kendall, we keep marching on. Marci Dandridge, you ask the darndest questions, and that is why everyone loves you. Lydia Fanning, you do not ask the darndest questions, and that is why everyone loves you. Gretchen Forsythe, I guess I ought to come sing for the kids, I have run out of excuses. Rachel Turner, Frisbee Goddess, Birthday Poet, what more do I need to say. Doug and Mary Terry, that must be ramen I smell. Cliff Davidson, high five, you big goof. Dee Lewis, stay in one lane, for crying out loud. Valerie Ling, I have not forgotten you, and I am sorry you had to say no to Mickey. Amy Herron, keep hauling in those bad guys. Paul Slay, Mike Sacks, and your respective families, you are my role models.

Thanks to my friends and family in Malaysia: Carol Ng, Adrian Lim, Jason Jaganathan, Sivakumar, Low Chun Kiat, Jason Koay. Your presence is felt. You can stop taunting me now. Chan Chun Liang, I do not know in which category you belong, you rascal, but you are being mentioned here too, just in case.

Last, but certainly not least, I want to thank my heavenly Father, who taught me and keeps reminding me to call Him 'Daddy'. May His grace and peace be with you all.

Abstract

This thesis documents the efforts made to implement multiperspective mosaicking for the purpose of mosaicking undervehicle and roadside sequences. For the undervehicle sequences, it is desired to create a large, high-resolution mosaic that may used to quickly inspect the entire scene shot by a camera making a single pass underneath the vehicle. Several constraints are placed on the video data, in order to facilitate the assumption that the entire scene in the sequence exists on a single plane. Therefore, a single mosaic is used to represent a single video sequence. Phase correlation is used to perform motion analysis in this case.

For roadside video sequences, it is assumed that the scene is composed of several planar layers, as opposed to a single plane. Layer extraction techniques are implemented in order to perform this decomposition. Instead of using phase correlation to perform motion analysis, the Lucas-Kanade motion tracking algorithm is used in order to create dense motion maps. Using these motion maps, spatial support for each layer is determined based on a pre-initialized layer model. By separating the pixels in the scene into motion-specific layers, it is possible to sample each element in the scene correctly while performing multiperspective mosaicking. It is also possible to fill in many gaps in the mosaics caused by occlusions, hence creating more complete representations of the objects of interest. The results are several mosaics with each mosaic representing a single planar layer of the scene.

Contents

List of Tables

List of Figures

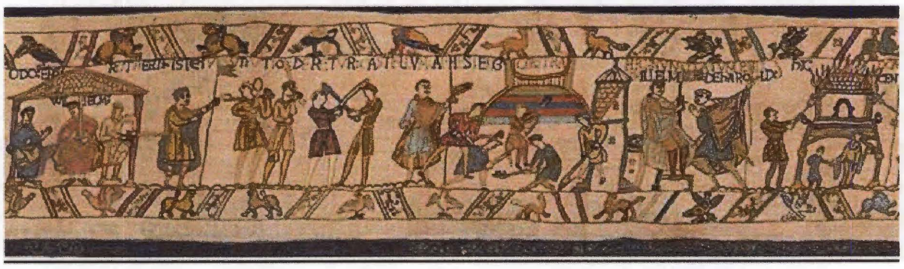
Chapter 1

Introduction

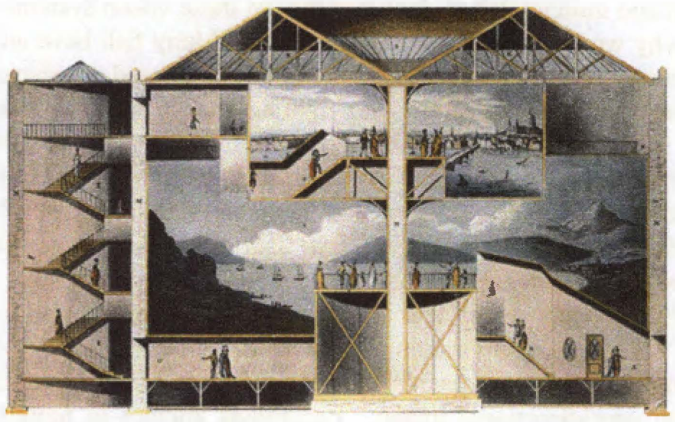
In order to perceive and experience the physical world, humans use what is classically known as the five senses: touch, taste, hearing, smell, and sight. Of these five, enhancing the sensory experiences of sight has been arguably one of the most engaging and popular challenges for all sectors of human effort, both in the arts and sciences. From monocles to telescopes, paintings to IMAX theaters, eye-drops to laser surgery; many innovations have been geared towards expanding the limits of what humans are capable of perceiving and experiencing with their eyes. Animal vision systems have provided inspiration in the past for many of these innovations. The vision systems of insects and fish, in particular, pose an interesting problem: how may human visual perception be enhanced by compensating for our limited field of view?

Before we explore answers to that question, let us briefly discuss the vision system of fish and insects. What characteristics of these vision systems make them unique, and why would we want to emulate them? Many fish have what is called *periscopic* vision: the eyes are on the sides of their heads and are shaped such that they bulge towards the pupil, refracting light in such a way that allows for an approximate 180° field of view for each eye [1]. Insect eyes are often made of patterns of photoreceptive cells that act as sensors, with each covering a specific direction, which also gives the eyes a wide-angle field of view [2]. These adaptations allow these animals to see in dim lighting conditions and detect threats from any direction, enhancing their ability to perceive the world around them.

It has long been recognized that a wide-angle visual representation of a scene enhances the observer's visualization of that scene. The wide-angle visualization concept is implemented most recognizably in the form of a *panorama*, or, more technically, an *omnidirectional* image. Panoramas attempt to represent a scene from a single perspective with a wide field of visibility, providing a unique representation of the scene at every possible viewing angle, and thus helping the observer 'feel' as though he or she is standing at the actual location of the scene. The earliest documented attempt to create an omnidirectional image go back to 1787, when an Irish painter named Robert Barker patented what he referred to as *La nature à coup d'œil*, or "A View of Nature" [3]. A panoramic picture was painted onto the inner wall of a circular rotunda, and visitors would view the picture from the center of the room. Thus, the panorama, as an art form, became a form of mass entertainment in 18th century Europe and America. The concept of the panoramic view itself was not new; the Bayeux Tapestry [4], dating back to roughly 1066, is a 70-meter long embroidery depicting the history of the Norman invasion of England, and may be considered an early realization of the panoramic view concept. Barker, however, extended this concept by attempting to deceive the eye into believing it was viewing reality, and not just a painting. Both Robert Barker's panorama and the Bayeux Tapestry are shown in Figure 1.1.



(a)



(b)

Figure 1.1 Artistic applications of the panoramic-view concept. (a) A section of the copy of the Bayeux Tapestry at the Museum of Reading [5]. In this early application of the panoramic-view concept, the story of William the Conqueror and Harold, Earl of Essex, is told in chronological fashion, from left to right (b). Cross section of Robert Barker's Panorama, Leicester Square, London, 1789 [3]. The panorama was painted inside the circular wall of a rotunda. Viewers stood on a platform in the center of the rotunda to view the panorama.

When the new technique known as photography was invented, interest turned towards creating panoramic representations using photographs. The first panoramic cameras were *swing lens cameras*, invented in 1843 by P. Puchberger, and *rotating cameras*, invented in 1857 by M. Garella, which had parts that could pivot about an axis of rotation to capture wide-angle images. In 1858, T. Sutton introduced a camera which used a spherical lens filled with water, eliminating the need for moving camera parts. In 1970, D. W. Rees patented an omnidirectional capture system that combined a hyberboloid mirror with a normal perspective camera. This was probably the first camera to utilize what is called a *catadioptric* configuration, a combination of mirrors and lenses, in order to capture wide-angle images. Such cameras have been researched and applied as visual sensors in various teleconferencing, monitoring, and robot navigation applications.

Today, there is a significantly more robust alternative to creating realistic representations of a scene than the mechanical, hardware-oriented solutions of the past. With the advent of fast, powerful, and easily accessible computers, it is now possible to use software-oriented approaches to facilitate scene visualization. The challenge of finding new ways to represent a scene from digital imagery has been the focus of much research over the past two decades, creating several new distinct, but related, fields of research. In the research topics of view interpolation, image morphing, stereo reconstruction, and plenoptic modeling, for instance, the goal is to synthesize novel views or even infer 3D information of a scene from a set of input images. It may be said that these techniques aim to create many images from a few. There is another field of research that takes the opposite approach to scene visualization: digital image mosaicking aims to create one large composite image out of many.

Using modern computing technology, it is possible to robustly create wide-angle representations scenes from sets of input images. Once we were restricted to using hardware-oriented approaches in order to create seamless wide-angle images (catadioptric systems, fish-eye lens). Today, images captured using ordinary, off-the-shelf cameras can simply be merged digitally into a single composite image. Such a composite image is referred to as a *digital mosaic*. What advantages are there to using a software-oriented approach as opposed to a hardware-oriented approach? Catadioptric cameras and fish-eye lenses typically distort images heavily and attempt to capture large amounts of information on a limited imaging surface. As a result, the overall resolution of these images suffer, and in the case of distorted images, is irregular throughout the image. So while these images provide a good overview of the scene, they are lacking in detail, and are thus unsuitable for many scene visualization purposes.

Digital image mosaicking techniques, on the other hand, allow us to quickly create a mosaic from multiple high-resolution images, producing a large image that not only gives a good overall view of the scene but preserves the level of detail seen in the original images. Mosaic seams that may appear due to perspective and lighting changes in the input images can be compensated for using image processing techniques, which is an obvious advantage over physically combining photographs to create a mosaic. Finally, a software-oriented approach allows us to create mosaics from video data fairly quickly, while such an undertaking using purely hardware-oriented or physical approaches would prove cumbersome and time-consuming.

The most well-known digital mosaicking method is the panoramic mosaicking method. This method stitches images together to form a wide-angle view, up to a full 360°, which represents the view of a scene from a single viewpoint. This method is commonly used for virtual tours on the internet, which visitors to a website may use to view panoramas of real-world locations, much like the visitors to Robert Barker's exhibits did in the 1800s, except this time, a computer screen replaces the large wooden rotundas of the past. Panorama creation, however, imposes an important restriction on the input images; because a panorama is meant only to represent the scene as seen from a single viewpoint, or perspective, all the input images are restricted to being taken from a single stationary position. In this work, we intend to deal with a different case: the camera moving past the scene of interest. As will be made clear, the case of a moving camera presents its own unique challenges to the image mosaicking process. The examples we saw in Figure 1.1 help to illustrate the difference between the case of a stationary camera and the case of a moving camera. Robert Barker's panorama, representing the 360° view from a single perspective, would be representative of the case of a stationary camera. The Bayeux Tapestry, on the other hand, depicts perspective changes as the locations change with the story, which follows the events surrounding William the Conqueror's invasion of England. This example is analogous of the concept of a wide-angle representation created from images taken by a moving camera.

This thesis documents the use of mosaics to represent video from moving platforms for visualization purposes. The remainder of this chapter describes the specific applications of mosaics in our work and the motivation for developing these mosaicking algorithms in Section 1.1. We conclude this chapter with a description of the document layout in Section 1.2.

1.1 Motivation

Scene visualization is an important problem in the field of digital image processing, and offers many technical challenges. Methods for creating large, high-quality images for the purpose of scene reconstruction and inspection are needed to enhance and support the functions of various automated tasks. There are two common methods of capturing images of a scene: one can either use a stop-and-shoot camera to take still images of the scene, or use a video camera to capture a video sequence. Individual still images typically offer sharper and more colorful images than individual video frames, while video is useful for capturing large amounts of information in a short amount of time. Unless a large quantity of still images are used, still imagery provides little relational information between images, while video sequences typically flow smoothly from one frame in the sequence to another, giving the viewer a better sense of how one image in the sequence relates to another. This large, smooth-flowing quantity of information is typically more useful when attempting to recreate a scene for the purposes of scene visualization and inspection.

Video is used for many purposes in day-to-day operations in the sectors of industry and security. Pan-tilt-zoom cameras are mounted on walls to keep a visual record of events in shopping malls, airports, and other busy public areas. Similar cameras, either mounted in a stationary position, or placed on moving platforms, are used in industrial areas for inspection purposes. Video is also used to keep visual records as supporting documentation for industrial or academic activities. For instance, if an urban area is being surveyed for the purpose of city planning, some video of the area may be captured to be included to complement the quantitative data. Another example would be using video in order to obtain a visual record of a particular building or area of historical importance, in conjunction with any other data recorded of the area or building of interest. Again, with video, it is easier to get a feel for the spatial proportions and positional orientations of the different parts of an area or building than if we were to look at various still shots. Alternatively, we may have a single image taken from a distance from the scene or area in order to get an all-encompassing view, which inevitably results in low-resolution images lacking in detail.

All these examples illustrate that video, with its ability to store visual information in an easily accessible format, is a very useful medium for recording visual information. However, there are times when video tends to be a cumbersome format to reference that information. Suppose, for instance, video is taken of an industrial area for the purpose of surveillance, and the video is updated and viewed periodically by inspection personnel in order to ensure there is nothing amiss. As the camera moves through the scene, the personnel involved must be watching the video sequence the entire time it is running, or risk missing important details that point to a situation that needs rectification, for instance, a malfunctioning machine part, or pipe leakage. If the personnel see that something is amiss, they may want to rewind the video or remotely move the camera back to center on the detail in question. If there are any other important details in the scene outside of the camera's limited field of view at this time, then those details will not be seen until the camera is moved again. To address this problem, the camera may be placed further from the scene to provide an all-encompassing view of the scene, but then smaller details would be difficult to see and inspect. Alternatively, several cameras may be mounted as different points in the scene so that the entire scene may be viewed without compromising detail. However, there are cases when it is not possible to use multiple cameras, for instance, in areas that are cramped, or even for simple budgetary concerns.

Suppose, however, that all the visual information in a single video sequence captured by the surveillance camera were somehow represented by a single, large, high-resolution image that encompasses the entire scene. This image would be a mosaic composed from all the individual video frames taken by that single camera. A mosaic representation eases the inspection process by removing the inter-frame redundancies seen in video sequences, since a mosaic represents each spatial point in the sequence only once. This representation of a video sequence shortens the inspection time by allowing inspection personnel to reference disparate spatial points quickly during inspection. This concept is illustrated in Figure 1.2.

There are several advantages to representing video sequences as mosaics. The first, which was just illustrated in the example above, is that mosaics facilitate visualization of the scenes captured on video. Another advantage, as Irani et al. [6] argue, is that mosaics of video are an *efficient* and *complete* representation of video sequences. By reducing the inter-frame redundancies inherent in video sequences, and presenting each spatial point in the sequence only once in a single image, a mosaic contains the same amount of visual information as a video sequence while requiring far less data storage space. Representing video as mosaics allows for

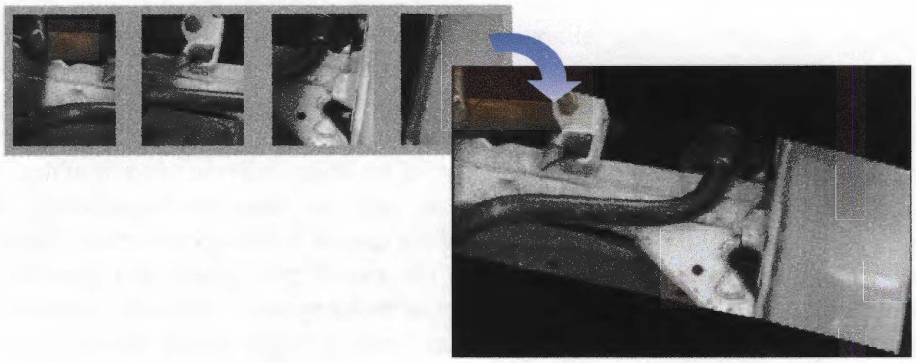


Figure 1.2 From a video sequence to a mosaic. Shown here are four sample frames from a 60-frame video sequence, depicting the underside of a vehicle, and a mosaic created from the video sequence. Simply observing each individual frame, it is difficult to ascertain how a scene in one frame is related to a scene in another frame. The mosaic gives us all this information in one concise representation.

information contained within that video to be referenced and transmitted more robustly for many applications.

The motivation for this work stems from the need to create mosaics from video data obtained from two main research efforts: undervehicle inspection and mobile laser-range scanning. For the undervehicle inspection effort, video was obtained from a mobile platform moving along the underside of vehicles for the purpose of threat detection, using both standard video as well as infrared modalities. A mosaic of the undervehicle video is desired in order to facilitate the process of inspection. For the laser-range scanning effort, video was shot of roadside environments from a moving vehicle. Mosaics are required in order to generate high-quality textures for the 3D data to improve the visual realism of the 3D models.

In both cases, the optical center of the camera moves. It is necessary that the camera moves past the scene if we intend to capture imagery of every part of a scene at high resolution; acquiring a single image of a large area would either result in our view of the area being limited to a small part of the scene, or the resolution of the image will suffer (if we used a wide-angle lens or captured the scene from a distance). A panoramic view of the scene would still leave us with the problem of lack of resolution, since parts of the scene that are of interest may be relatively far from the position from which the panorama is acquired. In order to acquire high-resolution imagery of the entire scene of interest, the camera is moved. However, when the optical center of the camera moves, this produces in the resulting video sequences a phenomenon called *motion parallax*. Motion parallax is a depth cue: as a camera's optical center moves past a scene, objects in the foreground move across the observer's field of view (FOV) faster than objects in the background. The magnitude of this perceived movement of elements in the scene is directly related to the distance of the elements from the camera. The closer an element is, the faster it moves past the FOV, and the further it is, the slower it moves.

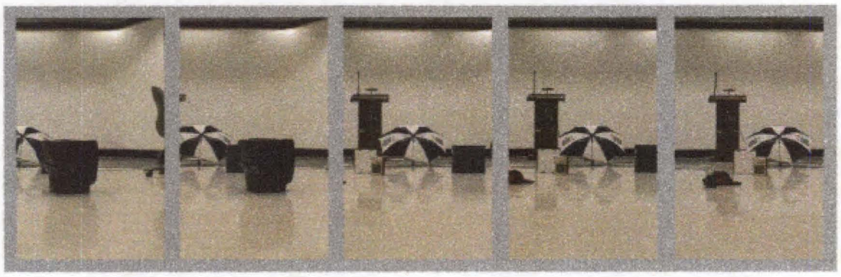
Because of motion parallax there is no one correct linear transformation that describes how consecutive frames may be aligned to one another. This phenomenon affects *registration*, the process of determining how consecutive frames are aligned with one another. For undervehicle video, motion parallax is relatively small. Still, due to that small motion parallax, whatever mosaicking technique we choose must be flexible in that it should settle for some sort of 'best-fit' registration between images, and not be too reliant on finding a uniformly correct registration.

For the roadside video sequences, large motion parallax in the sequence would produce more noticeable anomalies in any mosaic. Mosaics of such scenes might contain 'ghosts' or multiple occurrences of objects in the scene, since with objects recurring in video frames at different rates, it is impossible to sample them all in one mosaic correctly. Alternatively, a mosaic attempting to represent such a scene may warp objects with respect to their shape and size; objects in the distance appear truncated, and objects up close appear elongated, due to the same sampling problem. This problem is illustrated in Figure 1.3. There are also problems with *occlusion*; background elements may be occluded by foreground elements in one frame, but may be visible in another. In the presence of occlusions, is it possible to recreate an object of interest in its entirety in an intelligent manner?

We end this discussion of our motivation with a qualifier about the format of the data we wish to work with: although we speak of using video data, it isn't necessary that the data be in any one of the common digital video formats (avi, mpeg, etc). Instead, our intention is to take advantage of the dense image data commonly associated with video sequences: long sequences of images with significant overlap with one another. It is possible to acquire the same density of data using a still camera. Obviously, this would be extremely time-consuming and cumbersome in most cases, but it is possible nevertheless. Our focus is taking advantage of dense image data, and this kind of data happens to be most conveniently acquired in a video format.
In the work described in this thesis we address these challenges: using mosaics to creating concise and complete representations of video sequences, and to represent scenes captured on video displaying large motion parallax and occlusions. The solution needs to be implemented in software, and should not be hardware-specific (the solution should work with input from a large variety of imaging devices and modalities).

1.2 Document Layout

The remainder of this document will discuss some of previous work related to our efforts, as well as our algorithms and results. Chapter 2 presents a survey of major works produced in the areas of digital mosaicking, motion analysis, and layer extraction, as well as a general outline of our proposed solution. Chapter 3 outlines our algorithm used for a single-mosaic representation of a video sequence. Chapter 4 outlines our algorithm used for the layered-mosaics representation of a video sequence. Chapters 5 and 6 discuss the experimental setup and results of the single-mosaic and layered-mosaics representations, respectively. Finally, Chapter 7 summarizes our efforts and discusses possible future improvements to the algorithms presented here.


(a)

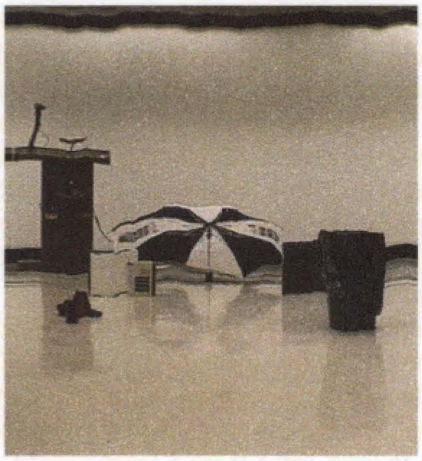

(b)

Figure 1.3 Motion parallax. (a) Sample frames from a test video sequence and (b) a crude mosaic created from the video sequence. The video was created by moving a camera sideways along a straight line parallel to the scene at a fairly constant speed. The mosaic was generated by sampling strips of pixels from the center of each frame in the sequence at a constant rate and pasting the strips together. Note that objects in the foreground (e.g. the baseball cap and dustbin) appear compressed, while objects in the background (e.g. the podium and umbrella) appear stretched. This is due to the fact that we are sampling each object at the same rate, even though the objects are moving past the camera's field of view at different rates.

Chapter 2

Related Work

The paradigms used to address the aforementioned problems were culled from various distinct, but inter-related, fields of digital image processing. Since we intend to use mosaics in order to create wide-angle representations, it is natural to begin with an overview of digital image mosaicking techniques in Section 2.1. We will discuss techniques developed to perform panoramic image mosaicking in Section 2.1.1, and multiperspective mosaicking in Section 2.1.2. Since the registration process of our chosen mosaicking algorithm relies heavily on motion analysis, we examine various motion analysis algorithms and their strengths and weaknesses in Section 2.1.3. Next, in order to deal with scenes displaying large motion parallax, we turn towards techniques that are geared towards represent video sequences as a series of planar layers. Layer extraction and representation, as these techniques are commonly called, are reviewed in Section 2.2. Finally, we conclude with a summary of key concepts and describe our proposed solution in Section 2.3.

2.1 Overview of Digital Image Mosaicking

Over the past two decades, much research has been conducted concerning the topic of digital image mosaicking. One convenient model for describing the digital image mosaicking process, described by Chen [7] is illustrated in Figure 2.1. According to this model, in general, image mosaicking methods follow a basic structure that consists of the following steps: image acquisition, image preprocessing, image registration, and image merging.

Image acquisition, as the term implies, is simply the process of acquiring images of a scene in such a way as to facilitate the creation of an image mosaic of that scene. This step takes into consideration such things as the type of camera used to take these images, the view angle and focal point of the camera, and the positioning and movements of the camera required to capture the images in the desired manner. Images taken using conventional cameras often exhibit properties that affect the mosaicking process, e.g., blurring, noise, lens warping, and irregular color constancy. Some preprocessing to reduce these effects is often necessary to ensure good mosaicking results. Image registration deals with finding the transformations that align adjacent images with one another. The transformation that relates two adjacent images is often called the *homography*. These transformations can take the form of simple translations and rotations as well as projective warps. Image merging is concerned with removing the inconsistencies that appear in the resulting mosaic after image registration is completed. Usually the image merging step includes blending and deghosting processes.

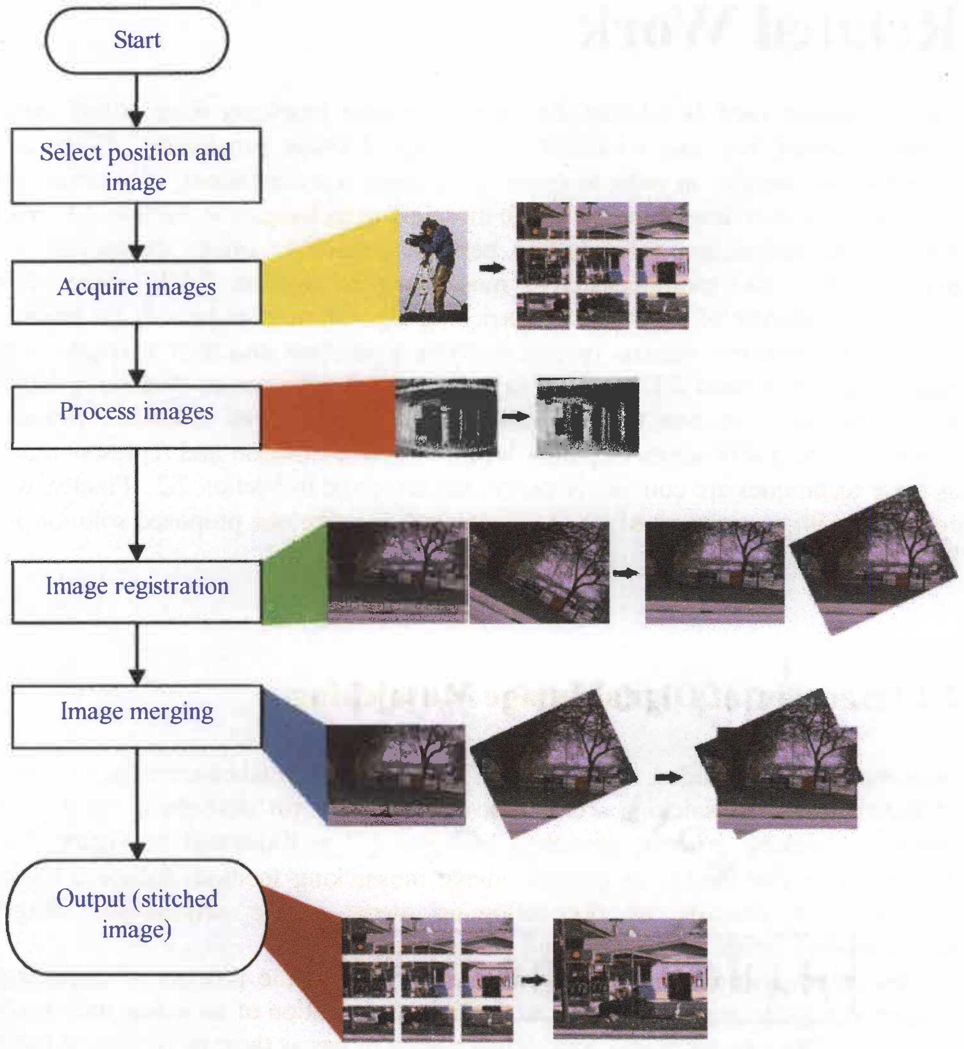


Figure 2.1 General Approach to Image Mosaicking. Adopted from [7].

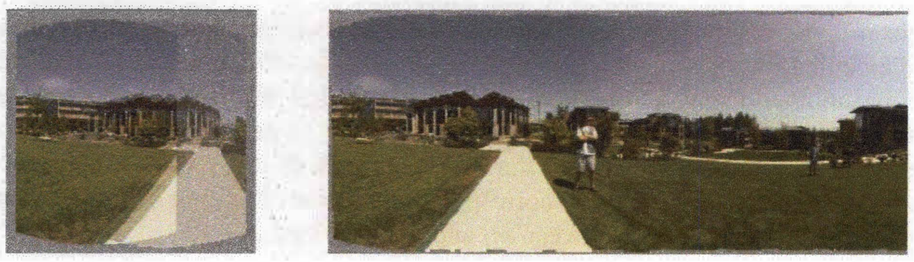

(a) (b)

Figure 2.2 Construction of a cylindrical panorama. Adopted from [8]. (a) Two images warped into cylindrical coordinates, and (b) a panoramic image created by aligning a series of warped images.

2.1.1 Panoramic Mosaicking

A popular example of digital mosaicking is *panorama* generation. The panorama, as explained earlier, is a popular form of scene visualization, with various artistic and scientific applications. We now describe the construction and restrictions of panorama generation in more detail.

A panorama is an image representing the wide-angle view of a scene from a single point. A digital panoramic mosaic, hence, is created by combining pictures taken by a camera whose optical center remains stationary. The camera's motion is restricted to rotating and panning motion about its optical center. Input images for panoramic mosaic generation are typically taken using a tripod or other rigid mounting device in order to limit movement of the camera's optical center, though it is possible to acquire such images by hand if the person taking the pictures stands still.

A simple method of generating panoramic mosaics, described by Szeliski and Shum [8], is to project input images onto cylindrical or spherical manifolds. Each input image is reprojected into cylindrical or spherical coordinates, according to the focal length of the camera used to capture the images. The focal length may already be known, if the camera was precalibrated, or estimated using sets of two or more input images [9]. Example results of warping images into cylindrical coordinates are shown in Figure 2.2. Once each input image has been warped, registration of each image becomes a pure translation problem, since perspective differences between images in this coordinate system are eliminated. Images are aligned by minimizing the sum of intensity errors between images:

$$E = \sum_i [I_2(x_2, y_2) - I_1(x_1, y_1)]^2 . \qquad (2.1)$$

where I_1 and I_2 are intensity values at two corresponding points between adjacent images, x and y are the coordinate vectors in the images, and i encompasses the overlapping pixels of the two adjacent images. A panorama created in this manner is also shown in Figure 2.2.

Another panoramic mosaic generation technique that does not rely as heavily on knowing the focal length is to use full planar perspective motion models to register images, a method which Szeliski [11] discusses, and Irani et al. [12] have implemented for video mosaics as well. Using this technique, images are related to one another using a transformation matrix called the *8-parameter motion model*. This motion model uses a linear projection to describe the motion of a rigid planar surface as either it or the camera moves. In this model, a point $m_1 = (x_1\ y_1\ z_1)^t$ of the first image has a corresponding point $m_2 = (x_2\ y_2\ z_2)^t$ in the second image, and the relationship between m_1 and m_2 is defined as

$$m_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = Hm_1 = \begin{pmatrix} h_0 & h_1 & h_2 \\ h_3 & h_4 & h_5 \\ h_6 & h_7 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \tag{2.2}$$

where H is the homography that relates the two images. Again, in order to determine this homography, the goal is to minimize the sum of the squared intensity errors given in Equation 2.1.

An alternative to minimizing Equation 2.1 is to detect features and track those features between adjacent images, using the feature points to compute the parameters of the homography. The problem of finding the homography is treated as finding four corresponding feature points between two adjacent images. This is because a pair of correspondence points produces two equations:

$$\begin{cases} x_2 / z_2 = \dfrac{h_0 x_1 + h_1 y_1 + h_2}{h_6 x_1 + h_7 y_1 + 1} \\[2ex] y_2 / z_2 = \dfrac{h_3 x_1 + h_4 y_1 + h_5}{h_6 x_1 + h_7 y_1 + 1} \end{cases}, \tag{2.3}$$

and eight equations are needed to solve for the 8 parameters of the 8-parameter motion model. Therefore, four sets of corresponding points would provide the eight equations required. Acquiring an automated search method for corresponding points between two images is not a trivial problem. A brute force comparison between all possible homographies between all feature points (extracted using some feature point extractor, e.g. the Harris detector [14]) of two adjacent images can be a very time-consuming process.

Zoghlami et al. [15] reduced the problem to finding two feature points by using a corner model to represent feature points. Corners are computed using a blurring filter for modelization, and models are fitted by non-linear minimization. The quality of a corner is determined by computing the least-squares difference between the grey-levels of the image and those of the model, with small measures indicating 'good' corners. Between a pair of corresponding 'good' corners between adjacent images, four lines based on those corners are computed, and the intersections of those lines can be used to compute a homography. By using more information from feature points than just their coordinates, the number of matched features required to obtain a good homography between two images is greatly reduced, which in turn reduces the complexity of the homography search.

Tian et al.[16] use local binary patterns (LBP) to find correspondence points between images. An LBP for a feature point is created by computing the Harris detector values of that point's eight neighboring interest points, labeling those values as ones or zeros based on a threshold value, and rotating those values to form the smallest possible binary number. This number is the LBP of that feature point, and is rotation and illumination-invariant across overlapping images. The advantage of this method is that images with large illumination differences can still be registered, since the LBPs for feature points are mostly illumination-invariant. The use of the LBP as a match criteria for identifying correspondence points is more reliable than the use of, say, single Harris values alone.

Peleg et al. [17] proposed an alternative to computing homographies from intensity differences and feature points. They use intelligently sampled dense video sequences in order to form their panoramas. This is done by combining strips sampled from each frame in the image and then aligning and adding the strips to the final mosaic. Peleg's implementation could perform in real-time: a mosaic could be created as a video camera pans to view the scene. This technique is a special case of Peleg's manifold projection technique. Manifold projection will be discussed in more detail in Section 2.1.2.

The panoramic mosaicking problem is not always restricted to that of registering images of static environments, as there can be objects moving in the scenes of real-world environments. These objects may appear as 'ghosts' or 'doubles' in the resulting panoramas, since they appear multiple times across the scene depicted in the panorama. It is usually desired to deal with these elements either by segmenting them out of the panorama or retaining single representatives of these elements within the panorama. Odone and Fusiello [18] take the former approach, and tackle the problem by back-registering their mosaic (with moving objects) onto the images from which it was constructed, and using a median-based temporal filter to remove the moving objects from the mosaic. As an aside, they also retrieve the foreground (the moving objects) using a local misalignment measure developed by Irani et al. [6],

$$S_n(x,y) = \frac{\sum_i \left| I_n(x_i,y_i) - I_n^{pred}(x_i,y_i) \right|}{\sum_i \left| \nabla I_n(x_i,y_i) \right| + C}, \tag{2.4}$$

where I_n is the nth input frame, I_n^{pred} is the input frame from the mosaic, $\nabla I_n(x,y)$ is the spatial intensity gradient at the pixel (x, y), i denotes a small neighborhood of pixels around (x, y), and C is a constant used to avoid numerical instabilities and to suppress noise. This measure, when used to detect local misalignments between individual frames and the final still mosaic, can easily detect the moving objects and segment them out of the input images.

Davis [19] suggests a slightly different approach to the problem. Firstly, a registration method that is unbiased by movement is employed to register the images. Images are reprojected into spherical coordinates and registration is performed using the Mellin transform, which is an extension of the phase correlation algorithm that recovers both translation and rotation (the phase correlation algorithm is discussed further in Section 2.1.3). To compensate for the moving objects, the resulting mosaic is compared with each input image, and difference images between the mosaic and the input images are computed. Dijkstra's algorithm (an algorithm used commonly to compute the shortest path between two vertices of a graph) is then used to compute the best path dividing each of the difference images, which in turn produces a segmentation of the mosaic into disjointed regions. The boundaries of each of these regions mark places where there are representations of moving objects in the input sequence which do not overlap one another spatially in the final mosaic. 'Correct' representations of these moving objects are placed within the final mosaic to create a final, focused image, without the ghosts or blurring that would result from a simple averaging or median sampling of the input images.

Because of the immersive nature of panoramic mosaics, they are used frequently to create image-based environment maps that may be viewed and manipulated in real-time. Chen [20] describes such a system that uses Apple's QuickTime VR. In this system, panoramic images are mapped onto QuickTime's cylindrical environment maps, which may then be warped in real-time to simulate panning or zooming to correspond with the viewer's input. Hotspots may be added to the environment maps, allowing viewers to 'hop' to the location within the environment map, thus loading up a new environment map to represent that location. Chen's method is implemented frequently on internet websites to provide virtual tours of real-world locations.

We have briefly reviewed digital panoramic mosaicking and some important works that attempt to address the various challenges associated with this field of study. It may be apparent at this point that panoramic mosaicking techniques do not readily address several phenomena present in many video sequences of real-word scenes. One phenomenon in particular, motion parallax, is heavily present in the video sequences used in this work. The shortcomings of panoramic mosaicking with regards to dealing with this phenomenon, and an alternative solution, are discussed next.

2.1.2 Multiperspective Mosaicking

Most panoramic mosaic generation techniques, with the exception of Peleg's technique, impose a major restriction on the input images: they must be taken from a stationary position. The camera can only exhibit panning and tilting motion. Any translational motion results in motion parallax. Recall from Section 1.1 that motion parallax arises from depth disparities of elements in the scene: when the camera moves, the speed of objects in the distance moving across the camera's field of vision appears slower relative to the speed of objects up close. The result is that the homography between two adjacent frames can no longer be adequately represented by planar projection equations such as the 8-parameter motion model.

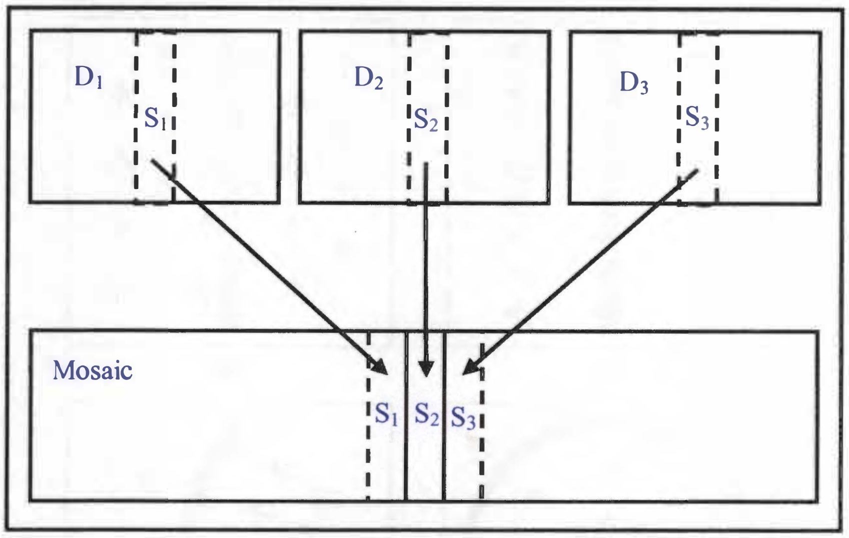


Figure 2.3 Illustration of the multiperspective mosaicking process[21]. Strips (S_1, S_2, S_3)are sampled from images (D_1, D_2, D_3) in a sequence and combined to form the mosaic.

Another method, called *multiperspective mosaicking*[†], provides a way to mosaic image sequences exhibiting motion parallax. This method simulates the function of a pushbroom camera, used in aerial photography. Pushbroom cameras are used to capture 1D line scans of an area and combine them to form a mosaic as the camera moves across the area of interest. Multiperspective mosaicking performs the same basic function by combining thin strips sampled from larger video frames and combining them into a mosaic. An illustration of this concept, given by Peleg and Ben-Ezra [21], is shown in Figure 2.3.

Zhu et al. [22] use multiperspective mosaicking to create stereoscopic mosaics of aerial video sequences. First, they perform image rectification on each frame of the video sequence, to make it appear as though the camera is undergoing pure 2D translational motion. Then, using a pyramid-based matching algorithm, they compute the displacements, and hence the registrations, between images. Instrumentation data from a GPS (global positioning system) and an INS (inertial navigation system) are used to correct for accumulated errors from pair-wise registrations of images. To compensate for local motion parallax between two adjacent strips, point correspondences between the two strips are identified close to the 'stitching line', the line marking the boundary between strips, and these corresponding points are mapped to one another to form a triangulation relating the two strips (a triangulation here being the segmentation of both strips into triangular regions with each region in one strip having a corresponding region in the other strip). Using this triangulation, the two strips are warped together to form the final

[†] In the literature, the terms free mosaic and panoramic-view image, among others, are also used in place of the term multiperspective mosaic. For clarity, in this work, the term panorama is used exclusively to mean a mosaic created from images taken from a camera with a (relatively) stationary optical center, while the term multiperspective mosaic is used exclusively to mean a mosaic created from images taken using a camera whose optical center moved during acquisition.

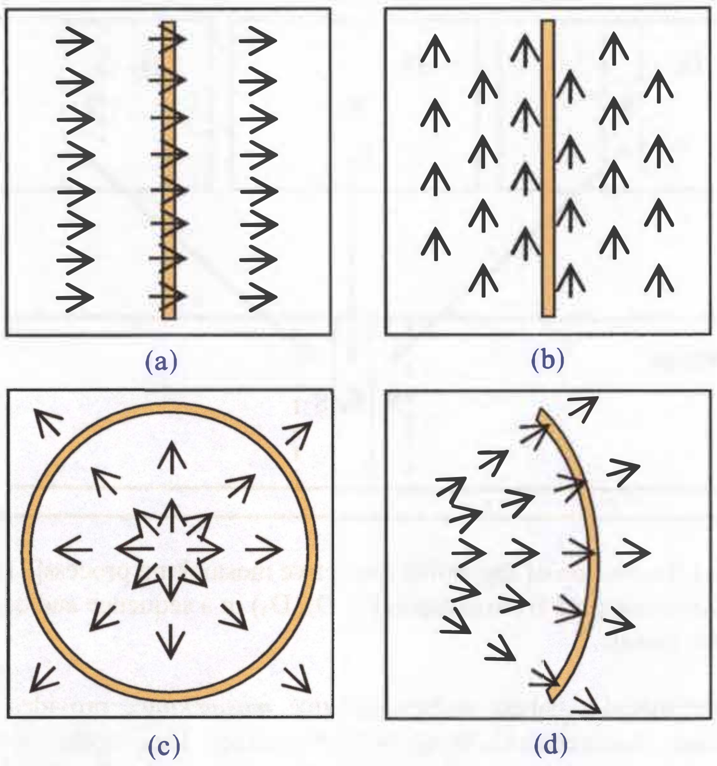


Figure 2.4 Warping strips for mosaicking onto adaptive manifolds. Adopted from [23]. Strips are warped to match the motion flow of the sequence (represented by the arrows): a) A straight, vertical strip is the logical choice to use in order to mosaic a scene displaying a horizontally-oriented flow field, whereas (b) the same strip is a poor choice to use for a vertically-oriented flow field. (c) A sequence created by a camera moving forward can be mosaicked using a circular strip, while (d) a sequence created by a camera oriented at an angle from the direction of movement can be mosaicked using a curved strip.

stitched strip pair. When composing the mosaics, two strips, instead of just one, are sampled from each frame, to create two stereo mosaics. A 3D effect is generated in the brain of the viewer when the two mosaics are viewed through special 3D lens. A depth map is also obtained from the stereo mosaics by finding correspondences along the epipolar curves constraining each pair of video frames, and estimating 3D range values from those correspondences.

Peleg et al. [23] proposed mapping mosaics onto manifolds adapted to the camera motion. These mosaics are projected onto an adaptive manifold by warping each strip so that each point is perpendicular to the optical flow of the image sequence. This process is illustrated in Figure 2.4. This manifold may take the shape of a cylinder for panning motion, a plane for translational motion, a tunnel for forward motion, and so on. These manifolds may be computed explicitly, and the images from the video sequence projected onto the manifold, or the manifolds may be computed implicitly by cutting and warping strips appropriately. This technique is more robust than Zhu's technique with regards to camera motion, but less geographically accurate. One important thing to note is that Zhu's work was

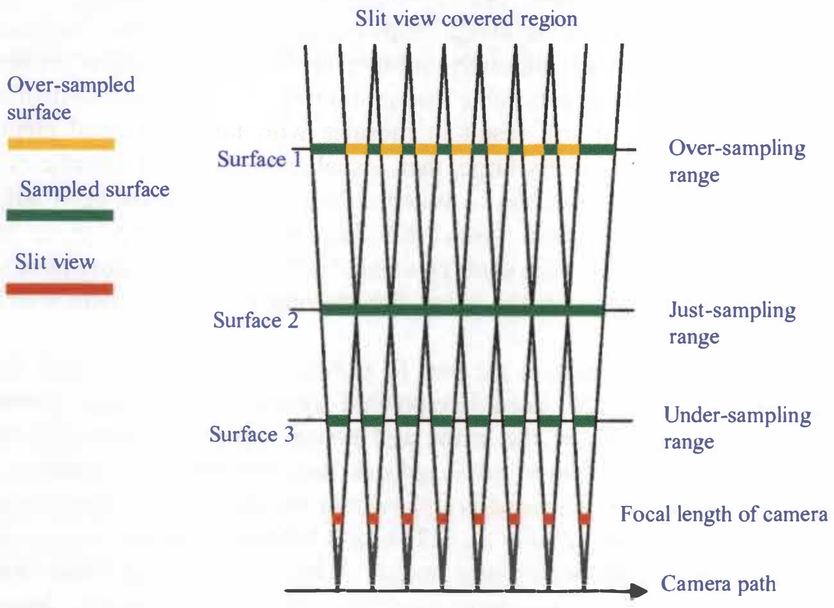


Figure 2.5 Just-sampling, over-sampling, and under-sampling of elements for creating a route map. Adopted from [25]. Each of the points along the camera path represents a point at which the camera slit samples the scene. These slit samples are pasted together to form the mosaic.

focused on creating geographically accurate mosaics and depth maps for land surveillance purposes, while Peleg's technique is geared towards general viewing applications, which only require that the mosaics look pleasing to the eye.

Zheng and Tsuji [24] used a technique similar to Peleg's in order to generate mosaics of outdoor environments to create a route map for robotic navigation. No motion analysis is performed here: the mosaic is created by simply pasting thin slits captured as the mobile platform moves pass the scene of interest. In most cases, the camera's principal axis (or plane of sight as they refer to it) is roughly orthogonal to the camera's motion. In a more recent paper, Shi and Zheng [25] attempted to investigate the sampling effects that arise due to motion parallax. They identified three distinct sampling cases, according to the depth of the elements in the scene from the camera: the depth at which objects are over-sampled, the depth at which they are just-sampled, and the depth at which they are under-sampled (Figure 2.5). As an aside, this is the same effect observed in the test case discussed earlier in Figure 1.3: elements in the background are over-sampled, hence they appear elongated in the final mosaic, while elements in the foreground are under-sampled, and hence appear truncated. Zheng has not yet attempted to compensate for the warping of elements caused by the sampling effect using post-capture image processing. He does intend to use his quantitative investigation of the sampling effects to better plan the capture of specific scenes, by adjusting the camera's focal length, speed, etc. For the purposes of building a route panorama for robotic navigation, this approach would suffice.

Multiperspective mosaicking alone does not 'solve' the problem of motion parallax, rather, it reduces the visual discontinuities caused by motion parallax by distributing misalignments more evenly throughout the entire mosaic, or, as in Zhu's stereo mosaics, by warping image strips locally based on joint triangulations between the strips. However, although multiperspective mosaicking is capable of producing mosaics with less noticeable discontinuities, image sequences displaying large motion parallax will still result in mosaics with fairly distorted elements. Also, if the strips are sufficiently large, then visual anomalies will become evident in the form of '*ghosts*' or '*doubles*' - visual elements recurring between adjacent strips. These problems stem from the fact that each strip in a single multiperspective mosaic samples each element in the strip at the same rate, while elements at different depths in the scene are moving past the camera's field of vision at different rates.

Attempts have been made in the past to address the problem of large motion parallax. Rousso et al. [26] used interpolated views between video frames to increase the sampling rate of the strips and hence improve continuity between strips. This approach, however, only reduces the 'doubles' in the mosaics; the warping of the elements in the scene is still evident, because those elements are still being incorrectly sampled. Zhu et al. [27] take a broader approach to solving the problem by generating epipolar-plane images (EPI) to complement the mosaics, and combining them into a representation they call the 3D Layered Adaptive-resolution and Multiperspective Panorama (3D LAMP). An EPI is a mosaic created by combining slices sampled from each frame that are parallel to the dominant motion of the camera. The resulting image has several straight loci, the orientations of which are used to determine the speed at which elements in the scene moved pass the camera, which also may be used to infer depth information. Thus, using many of these epipolar-plane images (the number of images depends on the desired resolution of the depth map; probably equal to the pixel-wise width/height of the original input images), 3D depth maps are generated which can then be used to segment the scene into layers of depth. On each of these layers, the various elements of the panorama can then be represented according to their correct sampling rates. The EPI for an example scene with varying depths is shown in Figure 2.6. The key idea here is that the speed at which an element in the scene moves past the camera's field of view determines the rate at which the element is sampled.

From this review of multiperspective mosaicking techniques, it is clear that the problem of motion parallax and occlusion is still left fairly open, though Zhu's approach to composing the scene as layers points to an obvious solution. However, before we examine techniques for representing video sequences as layers, we need to address the problem of finding correspondences between video frames so that registration between frames may be performed. If GPS data or some other external velocity measurements were available, those could be used in order to determine the inter-frame motions. However, in most of the video sequences used in this work, no such measurements were available. Also, many of these video sequences displayed time-varying velocities, and therefore constant-motion assumptions could not be used to simplify registration. Therefore, motion analysis techniques were used to determine the sampling rates used in the mosaicking process. We therefore proceed with a short overview of relevant motion analysis techniques.

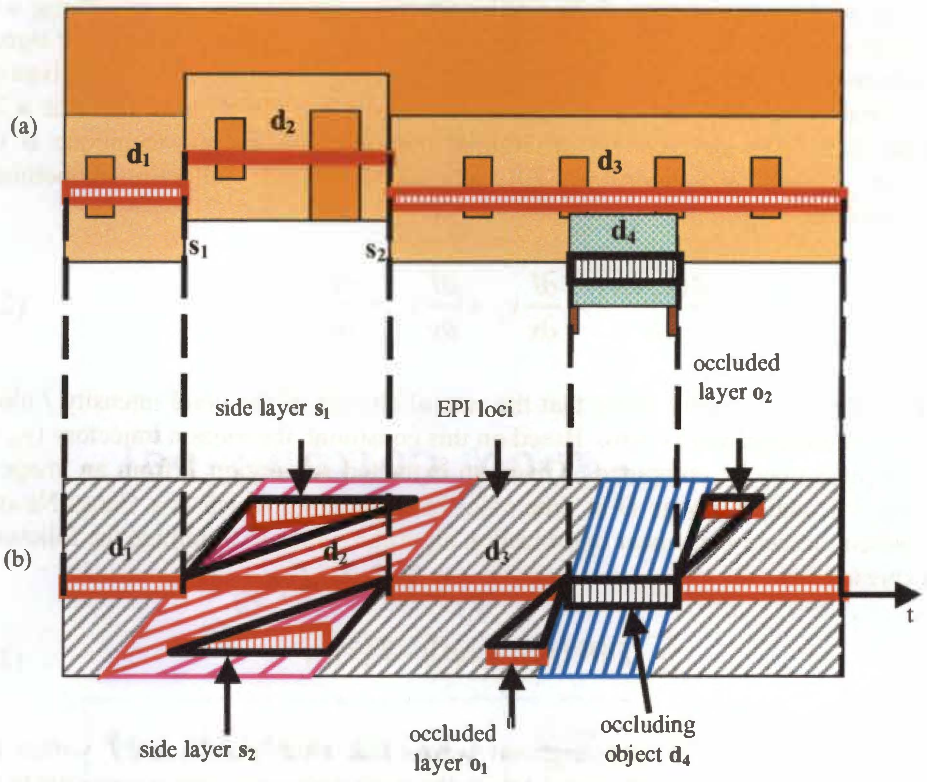


Figure 2.6 3D LAMP Representation. (a)A sample scene with four depth regions, d1, d2, d3 and d4, and (b) its corresponding epipolar-plane image (EPI) from the 3D LAMP, adopted from [27]. Using the depth and occlusion information inferred from the orientations of the loci in an EPI, the resolution of the mosaic may be adapted accordingly. The resulting mosaic is split into 'layers' of depth/resolution.

2.1.3 Motion Flow Analysis

Motion flow analysis techniques aim to determine the movement of pixels in an image sequence. This problem is well-known in the digital imaging community and has been addressed extensively. A good deal of research has been focused on developing differential techniques to create dense 2D flow fields, describing the motion for every pixel in a given image from a sequence of images. By 'differential' we mean that the techniques rely heavily on computation of the intensity gradients of the images, usually to find parameter updates in an iterative minimization process. According to a survey conducted by Barron et al. [28], many of these techniques share similar processes: prefiltering or smoothing with lowpass/bandpass filters to extract signal structure of interest and enhance signal-to-noise ratio, extraction of basic measurements such as spatiotemporal derivatives or local correlation surfaces, and integration of these elements to produce a 2D flow field. A widely-known differential motion flow analysis technique is the Lucas and Kanade algorithm [29], which assumes the following smoothness constraint:

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0, \tag{2.5}$$

This constraint simply states that the rate of change of the pixel intensity I along the motion trajectory is zero. Based on this constraint, the motion trajectory (v_x, v_y) of a pixel may be computed. Given an extracted sub-region T from an image at time t=1, and an image D at time t=2, this algorithm, which is a Gauss-Newton gradient descent non-linear optimization algorithm, aims to minimize the following expression

$$\sum_x [D(W(x;p+\Delta p)) - T(x)]^2, \tag{2.6}$$

where W is a set of parameterized warps that align pixel x in T within the coordinate frame of D, and p and Δp are the warp parameters and increments to the parameters, respectively. The minimization is performed with respect to Δp, with iterations being performed until the vector Δp is below a set threshold. This method was first developed in 1981, and many extensions have been made since by various authors to improve performance in the presence of noise, changing image capture conditions, and lower frame rates, as well as to improve computation efficiency.

Any differential motion analysis technique will produce good results in image sequences with sufficient detail and high frame rates. If detail is lacking (there are large homogenous areas in the image sequence) or the frame rate is low relative to the speed of the elements in the sequence (pixel movement between frames is large), then the performance of differential techniques tends to suffer. Also, differential techniques tend to impose other constraints, such as assumed color constancy and linear motion. One solution to these problems would be to use spatiotemporal segmentation that relies on the homogeneity over consecutive frames in a video sequence. Valencia et al. [31] describe such a technique. They use a pyramidal hierarchical structure to link homogenous regions in one frame to homogenous regions in the next frame. Based on the assumption that a pair of linked regions correspond to the same object, the motion of pixels representing that object then becomes the displacement of the centroids of those regions. Using this approach, it is possible to compute the motions for video sequences with large homogenous regions, changing illumination conditions, and low frame rates. If, however, the image sequence does meet the constraints imposed by differential techniques, the authors concede that the results of differential techniques are typically more accurate.

The phase correlation technique, first described by Kuglin and Heines [32], is another motion analysis technique that does not rely on the constraints of differential techniques. Phase correlation relies on the translation property of the Fourier transform, also known as the Fourier shift theorem. Suppose we have two images, one being a translated version of the other, with a displacement vector (x_0, y_0). Given the Fourier transforms of the two images, F_1 and F_2, then the cross-power spectrum of these two images is defined as

$$\frac{F_1(\xi,\eta)\cdot F_2^*(\xi,\eta)}{\left|F_1(\xi,\eta)\cdot F_2^*(\xi,\eta)\right|} = e\{j2\pi(\xi x_0 + \eta y_0)\}, \tag{2.7}$$

where F_2* is the conjugate of F_2. The inverse Fourier transform of cross-power spectrum would, ideally, produce an impulse function, with the position of the impulse indicating the displacement (x_0, y_0). A plot of this function, created by L. Hill, is shown in Figure 2.7. This function is sometimes referred to as the phase correlation surface. If there are several elements moving at different velocities in the picture, then the phase correlation surface will produce more than one peak, with each peak corresponding to a motion vector. By isolating the peaks, a group of dominant motion vectors can be identified. This information does not specify individual pixel-vector relationships, but does provide information concerning motion in the frame as a whole.

A simple extension to the phase correlation technique, proposed by Reddy and Chatterji [34], allows for the rotation and scale changes between two images to be recovered as well. By remapping the Fourier transforms of two adjacent images to log-polar coordinates, and then performing phase correlation on the images of the remapped Fourier transforms, it is possible to recover the scale and rotation factors between those two images. Once the scale and rotation changes have been compensated for, then phase correlation can be performed again to recover the translation between those images. This extension may be useful if there is a large amount of zoom or directional change exhibited by a video sequence.

The motion analysis techniques reviewed here provide a basis for addressing the registration problems associated with our mosaicking algorithm. We now turn our attention to the problem of representing scenes displaying large motion parallax. The solution lies in a class of techniques that have traditionally been proposed as video compression techniques: layer extraction and layered representation of video.

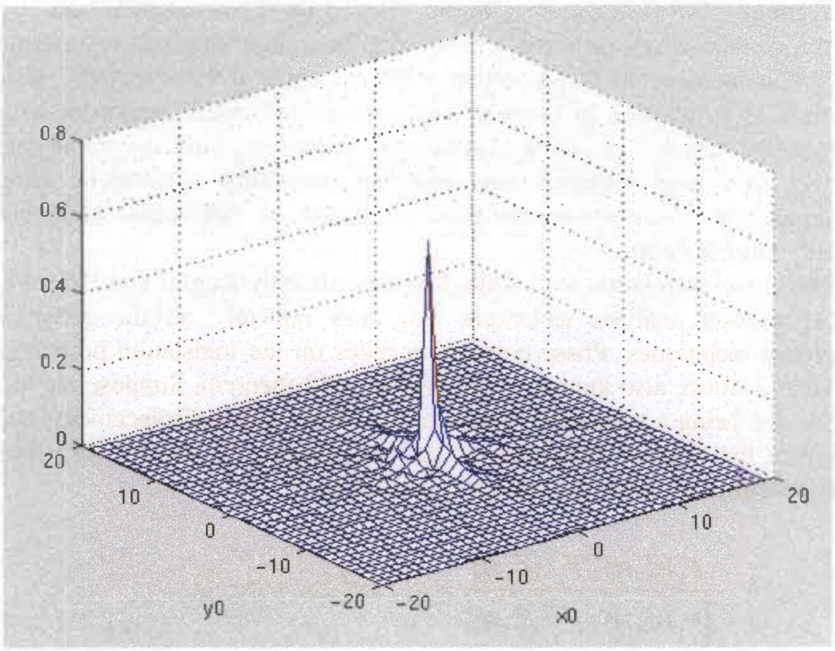


Figure 2.7 A typical phase correlation surface. Adopted from [33]. The x and y axes correspond to the height and width of the images, and the peak of the function indicates the displacement vector between the two images.

2.2 Layer Extraction

Representing video as layers is a relatively new field of study that was first introduced in the early 1990s. Wang and Adelson [35] were the first to propose the layered representation of video as an efficient video compression method. The paradigms associated with this technique are not without precedent, however, as many of the problems addressed in this field of study are the same as the more classical problems of segmentation by motion and multiple motion analysis. Layer extraction aims to represent a video scene as a series of planar layers corresponding to different depths or planar elements in a scene. Decomposing an image sequence into layers has been proposed as an efficient video and 3D representation method. A basic assumption is that the scene in a given video sequence may be represented as being piecewise planar. Each planar layer typically consists of a color map that specifies the pixel values of elements in that layer and a transparency map to define how each layer is occluded by all the others. There may also be a velocity map that defines how the layer is warped over time if one wishes to recreate the original sequence from which the layers were generated. These elements of the layered representation method are illustrated in Figure 2.8.

Wang and Adelson liken the layered representation of video to the process of traditional cell animation. In traditional cell animation, sequences of images are painted on clear celluloid and then placed over a painted background. The effect on film is of an animated foreground moving against a static background. The layer extraction process would aim to take resulting animation and reverse the

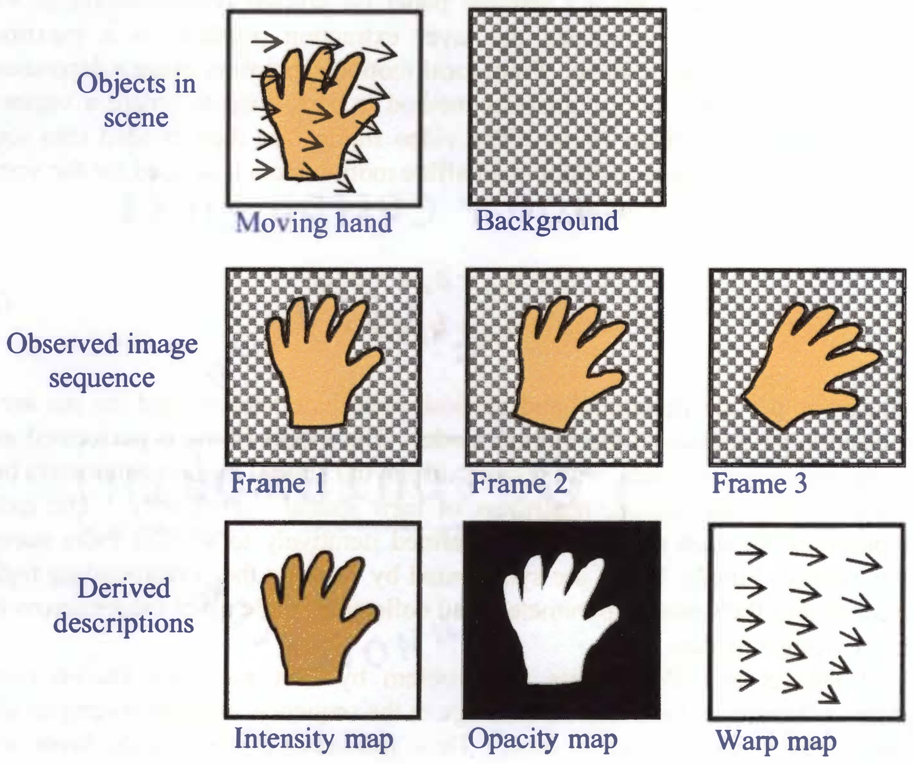


Figure 2.8 Layered Representation of Video. Adopted from [35]. The scene is decomposed into two layers: one to represent the hand and one to represent the checkerboard background Each layer consists of an intensity map that defines the pixel information of elements in that layer, an opacity map that determine occlusion relationships with other layers, and a velocity map that defines the motion of the layer with time.

process, decomposing the scene into their composite layers: the background and the celluloid. Then, using layered representation, the animation can be recreated using the extracted layers.

Layer extraction methods typically deal with the issues of *layer model determination* and *spatial support determination* for each layer. The layer model determines how many layers are required to represent the scene as well as the model-based motion of each layer. Determining spatial support simply means determining which layer each pixel in a given frame is associated with. The processes may be performed one after the other, or may be part of one organic process. Wang and Adelson's seminal paper on layered representation of video sequences basically formulates the layer extraction problem as a maximum-likelihood estimation problem. First, local motion estimation using a derivation of Lucas and Kanade's motion analysis method is performed to obtain a vector for each pixel in each video frame. The video frames are then divided into square regions, and each region is fitted to an affine motion model, defined for the vertical and horizontal components as

$$\begin{aligned} V_x(x,y) &= a_{x0} + a_{x1}x + a_{x2}y \\ V_y(x,y) &= a_{y0} + a_{y1}x + a_{y2}y \end{aligned} \tag{2.8}$$

where x and y are horizontal and vertical coordinate vectors, and the a_ks are the respective parameters of the motion model. The model fitting is performed using linear regression methods, with regions displaying similar motion parameters being clustered into one region, regardless of their spatial connectivity. The motion hypotheses for each region are then refined iteratively to acquire more accurate parameters. Finally, layers are synthesized by warping the corresponding regions according to their motion parameters and collecting stable pixel values across each frame using a median filter.

Baker et al. [36] address the problem by first assuming known camera projection matrices for every input image in the sequence, and performing an initial segmentation of the scene by hand. Thus, instead of computing the layer model and the spatial support concurrently as Wang and Adelson do, model initialization is performed manually first. From there, the inter-frame registration of layers and computation of their warp parameters are performed using gradient descent methods, such as Gauss-Newton minimization, using knowledge of the collineation between the camera matrices. In addition to the computation of pixel assignments and warp parameters, they also compute the residual 3D depth for each layer, providing a '2.5D' layered representation of the scene. Baker's method produces a more geometrically accurate representation of the scene, complete with depth information, as opposed to Adelson's method, which only aims to reproduce the video sequence in a visual sense.

Torr et al. [37] later address the question of model initialization posed by Baker. They formulate prior assumptions about the number of layers and their associated parameters within a Bayesian decision making framework in order to automatically perform model initialization. Several different models (1 layer, 2 layers, 3 layers, etc...) are evaluated individually, with parameters for each model estimated robustly from the feature points of the first image sequence. The posteriori-likelihoods of each given model is then calculated from these parameters, which then indicate the number of layers that should be used to represent the scene.

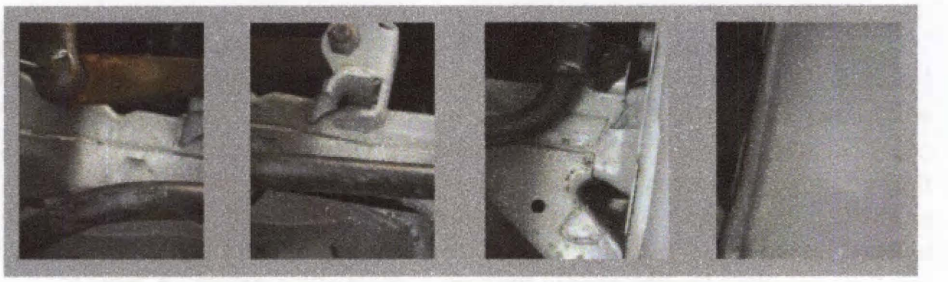
Ayer and Sawhney [38] also formulate the problem using Bayesian methods. However, instead of evaluating various layer models at the same time, they start with a user-defined number of layers which begin as non-overlapping regions that evenly divide an input image. The ownership probability for each of these layers is then computed for every pixel in the image, with pixels being assigned to layers producing the largest ownership probabilities. The initial layer model is then updated by removing each of the layers from the model in turn, and calculating the resultant encoding lengths needed for each version of the model. A large reduction in the encoding length results in that layer's removal from the representation, and the process is repeated until no more large reductions in encoding length occur. Ayer and Sawhney's method selects its layer model based on the minimum encoding length (MEL) criteria: the least complicated model that adequately represents the scene is selected to be the correct layer model.

So far our overview of layer extraction techniques has described some of the most influential works produced in this relatively new field. In particular, we are interested in the general paradigms the authors used to formulate their solutions to the problem of representing scenes as layers. The concept of combining mosaics with layered representations is a natural approach to solving the problem of motion parallax in long video sequences. Many of these works do describe their final layers as mosaics created from the accumulation of layer elements from all the frames in the input sequences. However, the mosaicking aspect is treated as a natural byproduct of the algorithms presented here. This brings us back to the recent efforts of Zhu et al. [27, 41], which were discussed in Section 2.1.2, to explicitly combine mosaicking and layer extraction concepts into one unified framework. As was explained before, they generate epipolar-plane images in order to compute depths and occlusions for each point of a multiperspective mosaic. These depth and occlusion relationships then directly determine the layers and their spatial support. They apply this technique, in particular, to roadside sequences displaying a mix of translational movement of the vehicle and vibrational movement of the camera, and place practical movement constraints on their system to ease EPI analysis of the video sequence. By placing these constraints, they achieve better results than more general techniques for EPI analysis.

We have now concluded our review of various efforts that have been made in the various fields of research that have contributed significantly to our work. We now describe the problems addressed in this research in more specific detail and review the most important ideas that were used to formulate our approach to solving these problems.

2.3 Proposed Approach

An important impetus for the formulation of the algorithms described here was the nature of the video data used in this work. This video data came from two primary sources. The first was from a mobile platform with an attached video camera which captured video of the underside of a vehicle as the platform translated underneath the vehicle. The second source was from a video camera attached to a vehicle driven along straight roads as the camera was pointing at the roadside scene,


(a)

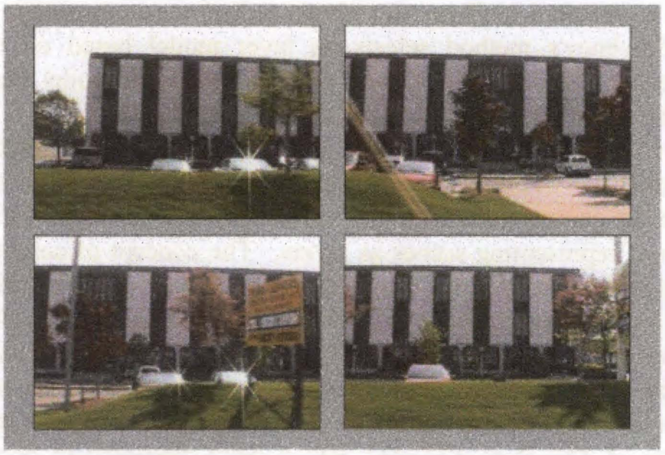

(b)

Figure 2.9 Test video sequences. Example frames for (a) undervehicle video and (b) roadside video.

perpendicular to the vehicle's motion. Example frames from these sequences are shown in Figure 2.9. The first thing we note is that due to the presence of motion parallax, the panoramic mosaicking techniques that register images using pure perspective transforms or that map images to spherical or cylindrical manifolds cannot be used, since these techniques require there be a one-to-one transformation relationship between adjacent images. Panoramic mosaicking techniques perform very well when it comes to mosaicking scenes with little or no motion parallax, but tend to perform poorly otherwise.

It is virtually impossible to capture roadside data without the presence of large motion parallax since we have chosen to use a moving platform to capture our data. If we are to capture video data of a large area without compromising the image resolution at any part of the scene of interest, then the platform must move from one part of the scene to another. Motion parallax is therefore a necessary obstacle that must be addressed in the mosaicking process.

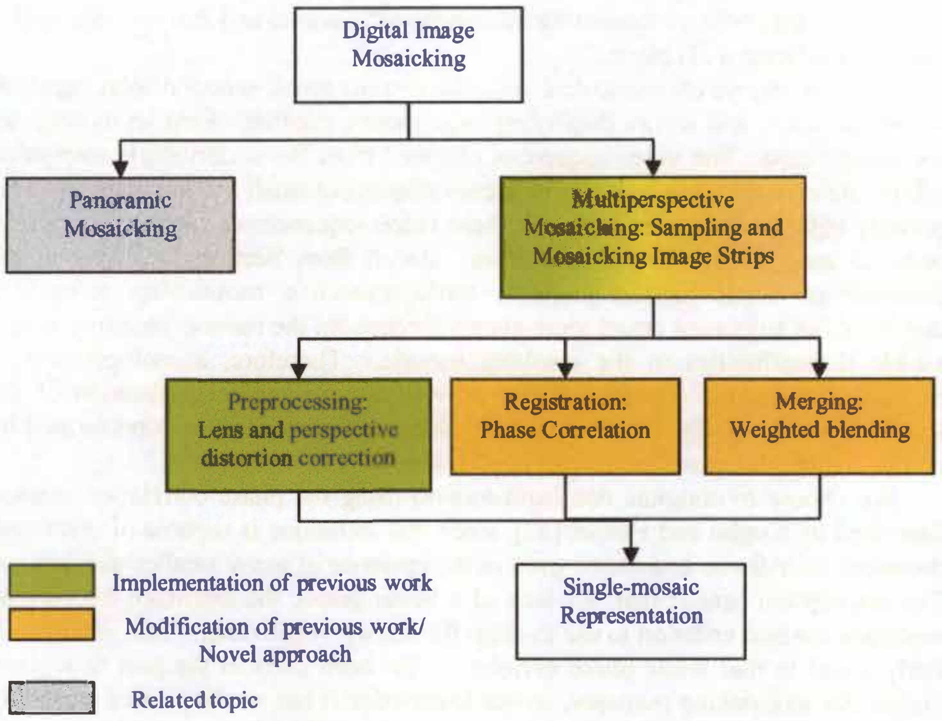
We use the paradigms associated with multiperspective mosaicking, as discussed by Peleg et al. [23], in order to form our algorithm. All our mosaics will be constructed from strips sampled from each frame in the video sequences. This technique is well suited for mosaicking dense video sequences such as the ones we are dealing with. Note that unlike Peleg's work with various projection manifolds, we assume, as Zhu et al. [22] do in their work with aerial mosaics, that the motion is restricted primarily to translating motion past the scene, and that the manifold is restricted to being a 2D plane.

In this work, we choose to deal with two distinct cases: scenes displaying small motion parallax, and scenes displaying large motion parallax. First let us consider the former case. The video sequences obtained from the undervehicle inspection efforts are considered to be cases of scenes displaying small motion parallax. Our primary objective here is to represent these video sequences as concise mosaics, in order to ease the process of inspection. Recall from Section 2.1.2 that in the presence of small motion parallax, multiperspective mosaicking techniques distribute the alignment errors more evenly throughout the mosaic, resulting in less visible discontinuities in the resulting mosaic. Therefore, a multiperspective mosaic of the video sequence would provide a wide-angle visualization of the sequence with few visible discontinuities. Registration of images is performed by computing the dominant motion between frames.

We choose to compute dominant motion using the phase correlation method described by Kuglin and Heines [32], since this technique is capable of extracting dominant inter-frame translation even in the presence of many smaller translations. The assumption here is that, for lack of a better guess, the dominant inter-frame motion is the best criterion to use to align frames with each other. This approach is fairly novel in that while phase correlation has been used in the past to register images for mosaicking purposes, to our knowledge it has not been used explicitly to perform strip-based multiperspective mosaicking. This dominant motion is used to determine the width of the strips sampled from each frame, as well as their correct alignment with one another in the final mosaic. Once the strips are aligned with one another, they are merged using a weighted blending scheme to create the final mosaic. We assume that the entire scene may be modeled as a flat plane, and that this entire plane may be represented by a single mosaic. We therefore refer to this representation as the ***single-mosaic representation.*** A summary of the efforts performed for the single-mosaic representation is shown in Figure 2.10.

We now address the case of scenes displaying large motion parallax. We treat the roadside video sequences as being cases of scenes with large motion parallax. We are interested in creating mosaics of elements in these road scenes for the purpose of texturing 3D models of these scenes, or simply as 2D representations of the scene for general visualization purposes. We have three objectives here: 1) to represent the video sequence as concisely as possible, 2) to represent all elements in the scene correctly with respect to their shape and size, and 3) to remove occlusions and thus produce more complete representations of occluded objects. In order to represent the scene concisely, we also use multiperspective mosaicking paradigms to form our mosaics; in other words, all mosaics are formed by combining strips sampled from each frame in the video sequences.

This is very similar to the work of Zheng [24, 25], who used multiperspective mosaicking techniques to create route maps of outdoor environments for robotic navigation. However, our intention is not just to create a route map that summarizes all elements in the scene in a single image, but to reconstruct, as best we can, objects of interest in the scene that are occluded by foreground elements, as well as sample each element according to the speed at which they move past the camera, so as to produce each element as they appear in the original sequence. Recall from Section 1.1, that a single multiperspective mosaic of a video sequence will distort objects in the scene according to their distance from the camera: objects



Digital Image Mosaicking
Panoramic Mosaicking
Multiperspective Mosaicking: Sampling and Mosaicking Image Strips
Preprocessing: Lens and perspective distortion correction
Registration: Phase Correlation
Merging: Weighted blending
Single-mosaic Representation
Implementation of previous work
Modification of previous work/ Novel approach
Related topic

Figure 2.10 Summary of efforts for single-mosaic representation. Note that panoramic mosaicking is listed here as a related topic, but no implementation of panoramic mosaicking techniques will be discussed in detail in this document.

close to the camera will appear truncated, while objects further away will appear elongated. This is because along any given strip sampled from a video frame, every element within that strip is sampled at the same rate, even though those elements may be moving past the camera's field of view at different speeds. If we intend to reproduce all elements correctly with respect to shape and size, we need to sample each element in the scene according to the speeds at which they move past the field of view. Therefore, we make a compromise to our requirement of representational conciseness: instead of creating one single mosaic to represent the entire scene, we create several mosaics, with each mosaic associated with a unique pixel-wise velocity.

This approach closely mirrors the layered representation methods discussed in Section 2.2. In fact, we derive many of our paradigms based on these methods in our solution formulation. Not every aspect of a proper layered representation is reproduced in this work, however. Referring again to Figure 2.8, in a typical layered representation of video, each layer consists of an intensity/color channel (which defines the appearance of the layer), an opacity channel (which determines occlusion relationships between layers), and a velocity channel (which determines how the layer is warped with time in order to reproduce the motion of the elements within that layer in the original video sequence). Since we aren't interested in reproducing a video sequence *per se*, only in reproducing the elements within that sequence as a series of 2D mosaics, we reduce the problem to computing the correct color/intensity channels for each layer.

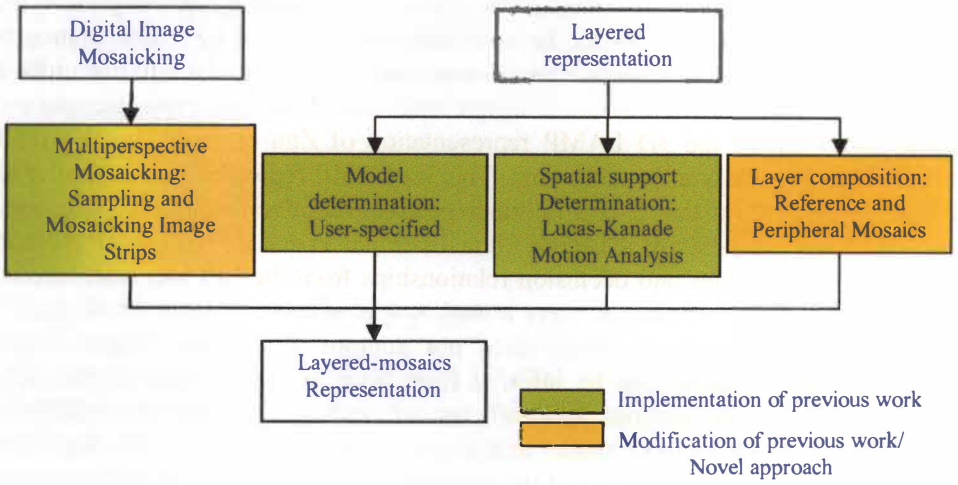

Digital Image Mosaicking
Layered representation
Multiperspective Mosaicking: Sampling and Mosaicking Image Strips
Model determination: User-specified
Spatial support Determination: Lucas-Kanade Motion Analysis
Layer composition: Reference and Peripheral Mosaics
Layered-mosaics Representation
Implementation of previous work
Modification of previous work/ Novel approach

Figure 2.11 Summary of efforts for layered-mosaics representation.

We still need to deal with the basic problems of layer model determination and spatial support determination. Model initialization is performed manually, just as Baker et al. [36] initially chose to do. Unlike Baker, however, our model initialization does not involve manually assigning spatial segments to the scene, but only involves determining the number of layers we'd like to use to represent the scene, and then assigning unique pixel-wise velocities to each of those layers. Spatial support determination is determined by computing the velocity for each pixel, and then assigning those pixels to each layer according to their computed velocities. This aspect of the layer extraction algorithm is closely related to our registration technique for mosaicking, which is discussed next.

In the single-mosaic representation, we proposed using phase correlation to perform registration for mosaicking, which only extracts one dominant inter-frame motion from a pair of adjacent frames. This is not a per-pixel velocity estimation, meaning, we do not acquire a dense flow field assigning vectors to each pixel in each video frame. We would like to have such a flow field for each frame, because this would not only provide us with velocity information with which we can sample mosaic strips, but also help us determine spatial support for each layer. In order to compute these dense flow fields, we use the same technique used by Wang and Adelson [35] to perform local motion estimation, which is a derivation of the Lucas-Kanade algorithm. Departing from Wang and Adelson's layer extraction method, we then segment each frame using the computed pixel velocities, according to the pre-initialized layer model.

From the segmented frames, we can now sample each element in the scene correctly, and place those elements into their own layer-specific mosaic. The final problem is dealing with occlusions. To do this we use a unique approach to mosaic composition. Instead of creating one mosaic for each layer, we first create multiple mosaics for each layer, with the strips used to form each mosaic sampled from different points in each video frame. Using knowledge of the spatial correspondences between these mosaics, we are capable of reconstructing each layer, minus occlusions. A summary of the efforts performed for the layered-mosaics representation is shown in Figure 2.11.

Because we intend to use mosaics as layers to represent a video sequence, we refer to this representation as the *layered-mosaics representation.* This approach aims to combine multiperspective mosaicking and layer extraction into one unified framework. To date, we only know of one other effort that explicitly attempts the same combination: the 3D LAMP representation of Zhu et al. [27]. We shall briefly compare Zhu's effort and ours. The 3D LAMP representation is similar to our representation in that it is a multiperspective, adaptive-resolution, occlusion-recovered representation of the scene. It is dissimilar in that the 3D LAMP method computes depth values and occlusion relationships from the EPI loci orientations, and uses those to determine its layer model, spatial support for each layer, and to remove occlusion. Our method does not attempt to compute depth values explicitly (though depth can be inferred from pixel velocity, which is basically what the EPI loci orientations indicate), nor are occlusion relationships explicitly determined. In our method, model initialization is performed manually, but from there, spatial support is determined using a derivation of the Lucas-Kanade motion analysis algorithm. Occlusions are removed by taking advantage of the peripheral visual information available in each video frame, and intelligently using this information to fill in the occluded areas wherever possible.

We now summarize the goals of our algorithms: for the single-mosaic representation, the input to our algorithm will be a dense sequence of images (usually taken from video), and the output will be a single mosaic summarizing the entire video sequence. For the layered-mosaics representation, the input will again be a dense sequence of images, and the output will be several mosaics, with each mosaic recreating a planar layer. All these layers may be used to recreate any segment of the original video sequence.

We have now completed the outline of our proposed algorithm. In the next chapter, we continue with a detailed description of the single-mosaic representation algorithm.

Chapter 3

Single-Mosaic Representation

This chapter describes the algorithms used to create a single-mosaic representation of a video sequence. In Section 3.1, we begin with a description of the general framework of our solution and describe in detail the problem we are attempting to solve. Next, in Section 3.2, we describe various image preprocessing steps that must be typically be performed before the video sequence may be mosaicked. The description of our solution proper is in Section 3.3, where we describe our registration, strip sampling, and strip merging algorithms. We end with remarks in Section 3.4.

3.1 General Framework

The methodology for mosaicking a video using a single-view representation is split into several tasks. These tasks were divided amongst several modules in order to ease the coding process. The methodology was developed based on several assumptions concerning the nature of the scene in the video sequence and the camera's movement. The specifics and implications of the aforementioned assumptions are discussed next, after which the various modules of the algorithm will be examined.

3.1.1 Data Constraints

Let us begin by briefly reviewing our goals: single-mosaic representation is used when a) motion parallax effects exhibited in the video sequence are small relative to the dominant motions between frames, and b) the purpose is to produce a single, large image as an overview of the entire scene. The undervehicle video sequences used in this work, for instance, display small amounts of motion parallax, and thus would be adequately represented by a single mosaic. Also, a multi-layered representation of these sequences would run counter to the purpose of simplifying the inspection process, since there would be several images to inspect as opposed to one wide-view image of the scene.

Now we state some of the assumptions and constraints we shall be using to formulate our solution. Since it is intended for the scene to be represented by a single mosaic, the first assumption is that the scene in the video sequence exists entirely on a single plane. Next, in order to simplify the mosaicking process, constraints are placed on the camera movement. It is assumed that the camera is translated solely on a single plane that is parallel to the plane of the scene. It is also assumed that the viewing plane of the camera is parallel to this plane of the scene. Finally, it is assumed that the camera does not rotate about its principal axis. The movement constraints imposed on the camera are meant to simplify the mosaicking process. These constraints may appear very limiting, but a systematic method of acquiring data of the scene would most likely obey these constraints. The data acquisition process is discussed in more detail in Section 5.1.

The collective effect of these constraints is that motion between frames is restricted to pure translational motion. This is simple to illustrate using principles of epipolar geometry [42]. Suppose we have two viewpoints of a point *M*, from two cameras with centroids *C* and *C'*. Assuming a pinhole camera model, if we know *m*, the projection of *M* on the viewing plane of one of the cameras, then the corresponding point *m'* on the viewing plane of other camera is constrained to lie on a line. This line is the epipolar line, and is the trace of the plane (*C*, *C'*, *M*). The formation of the epipolar lines (from points in the images *m* to *e* and *m'* to *e'*) are shown in Figure 3.1.

Now we assume the camera is bounded by the aforementioned movement constraints. Then, for two viewing planes corresponding to two consecutive frames in the video sequence, the epipolar lines are parallel because both viewing planes lie upon the same ground plane. We also assume the scene may be modeled as a plane that is orthogonal to the principal axes of the cameras, and a point *M* may only exist on this plane. Then the distance between the projections of any two points M_1 and M_2 on the viewing planes will be equal on both viewing planes (Figure 3.2).

If all scene and camera movement constraints are met, then all pixel movements between two consecutive frames are homogenous and purely translational. This greatly simplifies the motion analysis process, since a single horizontal-vertical translation may be used to define the homography between two adjacent frames.

An ideal video sequence would come from a camera moving in a constant direction while the camera's principal axis is kept orthogonal to the scene of interest. A camera placed on a mobile platform may be used for this purpose. The platform may then be moved in a straight line past the scene. If the scene is larger than the camera's vertical field of view, several straight line passes may be made to ensure the entire scene is captured. A single pass will produce one mosaic. Figure 3.3 illustrates a typical acquisition setup.

The speed of the platform need not be constant, though its direction should be as constant as possible to reduce inter-frame rotations in the video sequence. Once the video has been captured, the video sequence is decomposed into individual video frames. These frames are used as input for the mosaicking modules.

3.1.2 Processing Modules

The mosaicking process for single-mosaic representation reflects the general approach outlined in Figure 2.1. This process is outlined in Figure 3.4. The input images are the video sequence separated into individual frames. A preprocessing module is used to perform distortion correction on each input image. The registration module uses phase correlation to perform motion analysis in order to compute inter-frame motion vectors. A merging module performs strip selection and blending of the strips to form the mosaic, which is the output of the algorithm.

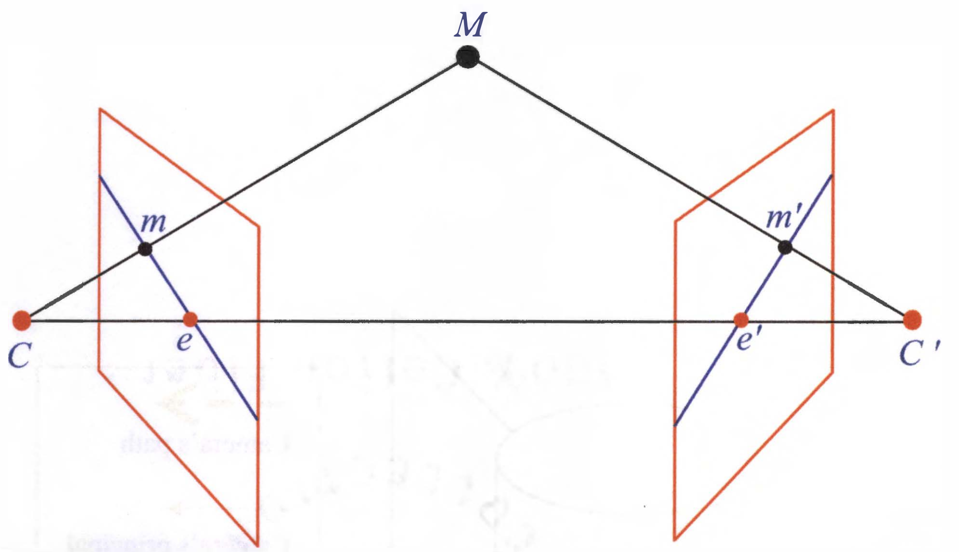


Figure 3.1 Epipolar geometry. Given m, the point *m'* has to lie on its epipolar line *m'-e'*.

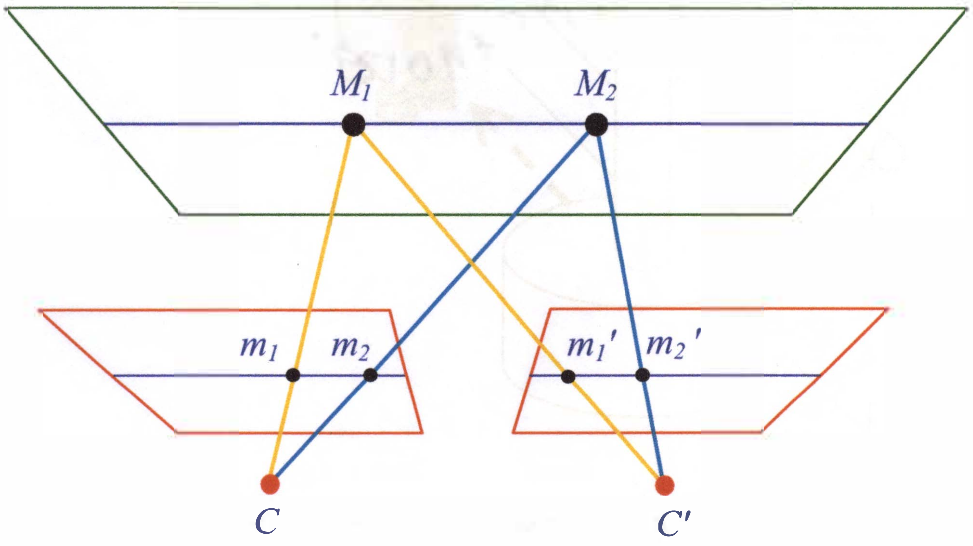


Figure 3.2 Epipolar geometry for parallel viewing planes. Note that the distances m_1-m_2 and m_1'-m_2' are equal.

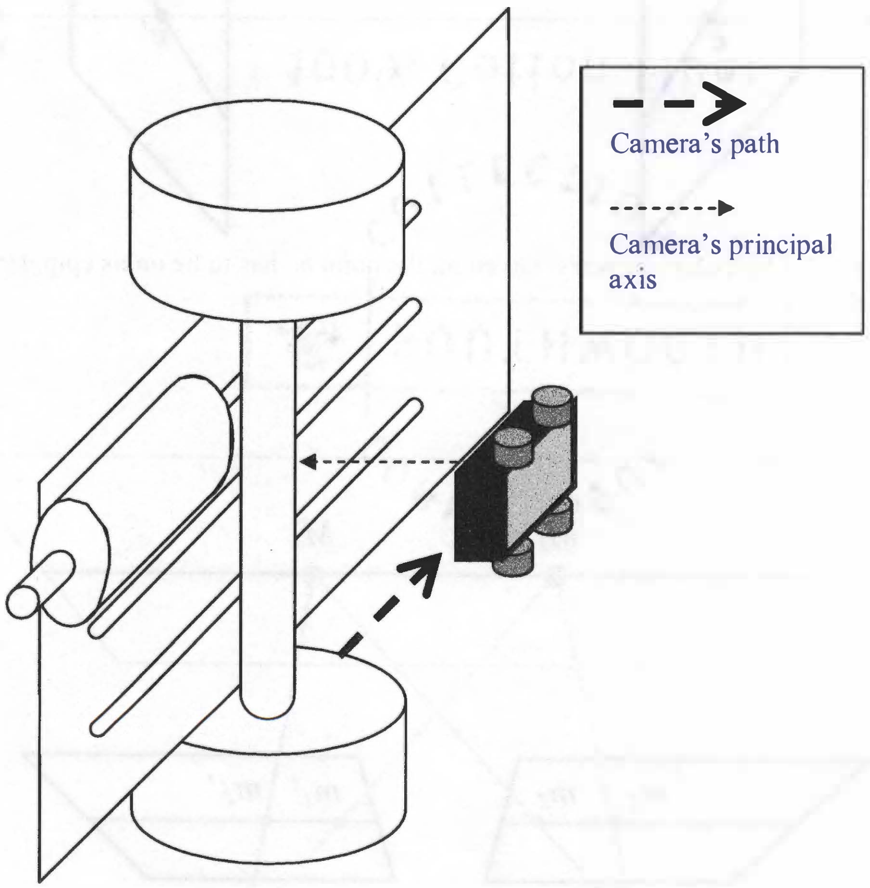


Figure 3.3 Video acquisition for undervehicle inspection.

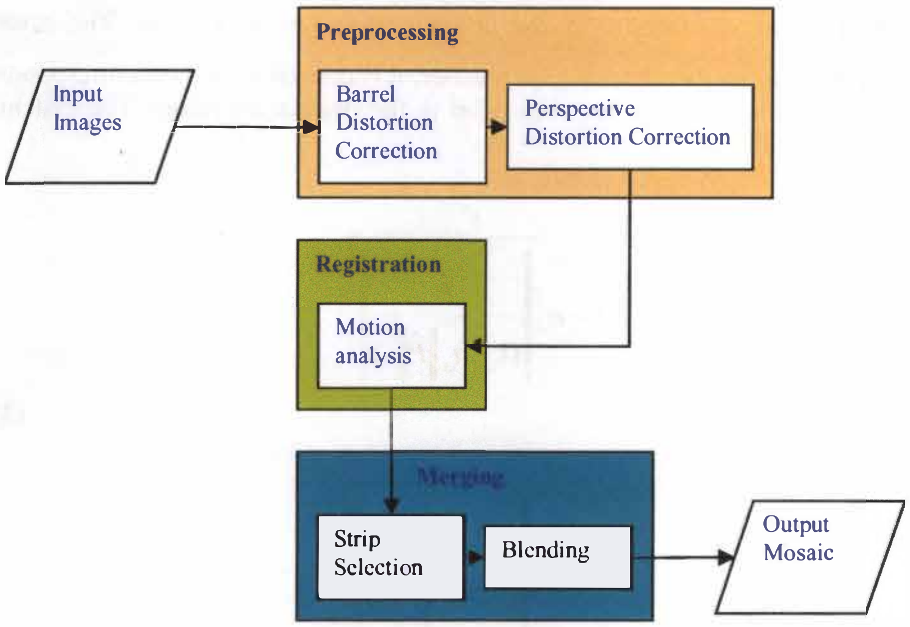


Figure 3.4 Mosaicking Process for Single-Mosaic Representation

3.2 Preprocessing

Before the sequence can be mosaicked, some preprocessing is required to correct each image for barrel distortion, a problem addressed extensively in the past [43, 44]. Barrel distortion is a common phenomenon associated with off-the-shelf camera lenses. Also, there are times when the camera views the plane of the scene at an angle during acquisition. Therefore, a projective transform is performed on each frame to ensure that the camera's viewing plane appears parallel to the plane of the scene. Barrel distortion and angle compensation are performed on each frame in the video sequence before the actual mosaicking is performed.

3.2.1 Barrel Distortion Correction

Given the lens projection factors, a_x and a_y, the horizontal and vertical coordinate vectors of the uncorrected image, x and y, and the horizontal and vertical coordinate vectors of the corrected image, x' and y', the equation that gives an approximate correction for barrel distortion is

$$\begin{aligned} x' &= x(1 - a_x \|\hat{P}\|^2) \\ y' &= y(1 - a_y \|\hat{P}\|^2) \end{aligned}, \tag{3.1}$$

where $\|\hat{P}\|$ is the modulus of the coordinate vector (x, y) [45]. The reverse transform is normally used, since in practice, it is desired to find the corresponding pixel in the source image for each pixel in the destination image. The matching reverse transform is

$$x = \frac{x'}{1 - a_x \left\| \frac{\hat{P}}{(1 - a_x \|\hat{P}\|^2)} \right\|^2}, \qquad (3.2)$$

$$y = \frac{y'}{1 - a_y \left\| \frac{\hat{P}}{(1 - a_y \|\hat{P}\|^2)} \right\|^2}$$

The equations above assume that the images are converted to a normalized (-1 to 1) coordinate system on both axes. The relevant conversions are as follows:

$$i = \frac{(x+1)width}{2}, \qquad (3.3)$$

$$j = \frac{(y+1)height}{2}$$

$$x = \frac{(2i - width)}{width}, \qquad (3.4)$$

$$y = \frac{(2j - height)}{height}$$

where *width* and *height* refer to the images' horizontal and vertical dimensions, in pixels, respectively, and *i* and *j* are coordinate vectors in the original, unnormalized images. The barrel distortion correction was performed by using the camera to take images of a calibration grid, and then manually adjusting a_x and a_y using the grid image as a reference.

3.2.2 Perspective Distortion Correction

The purpose of perspective distortion correction is to make it appear as though the scene's motion is orthogonal to the principal axis of the camera. A similar procedure is employed by Zhu et al. [22] as an 'image rectification' step. This procedure is required if the camera was viewing the scene of interest at an angle, which, as will be explained in Section 5.1, is true in our case. To perform perspective distortion correction, a projective warp is applied to each frame in the video sequence. Suppose we have the a point in the original image $m_1 = (x_1\ y_1\ z_1)^t$, and a point in the corrected image $m_2 = (x_2\ y_2\ z_2)^t$. Perspective distortion correction is performed using

$$m_2 = VRm_1, \tag{3.5}$$

where

$$V = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

and

$$R = \begin{bmatrix} \cos\phi\cos\kappa & \sin\omega\sin\phi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\phi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\phi\cos\kappa & -\sin\omega\sin\phi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\phi\sin\kappa + \sin\omega\cos\kappa \\ \sin\phi & -\sin\omega\cos\phi & \cos\omega\cos\phi \end{bmatrix} \tag{3.7}$$

are the focal length scaling and 3D rotation matrices, with ω, φ, and κ being the pan, tilt, and rotation angles of the image plane.

The warp parameters are determined manually, using visual cues in the scene in question. If the angle at which the camera was viewing the scene is known, this could be translated into the warp parameters as well. Resampling of the images is done using nearest-neighbor interpolation.

3.3 Mosaic Creation

In order to register the images for single-mosaic representation, motion analysis is performed using the phase correlation algorithm. The images are then merged by selecting and aligning strips sampled from the center of the images, according to the results of phase correlation. Finally, a weighted pixel blending scheme is used to reduce the visibility of seams between strips. These aspects of the registration and merging processes are discussed next.

3.3.1 Motion Analysis: Phase Correlation

Recall that when the aforementioned movement and scene constraints are met, all that is required to perform registration between images is to find their mutual vertical and horizontal translations. The phase correlation method, introduced earlier, is a technique that works in the frequency domain to acquire the horizontal and vertical displacements between two images in an image pair. There will still be inconsistencies in the uniformity of the motions obtained, due to small amounts of motion parallax, but we assume the dominant motions are sufficient to obtain a reasonable estimate of the motion at the center of the images, from which the mosaic strips are taken.

Of the horizontal and vertical displacements obtained, one will represent the dominant motion, which is the general direction the camera was moving in. This displacement will be referred to as the *primary motion.* The other displacement is caused mostly by camera jitter or slight changes in the mobile platform's direction. This displacement will be referred to as the *secondary motion.* In this work, in a motion vector (u,v) the primary motion always corresponds to v, the horizontal vector component, and the secondary motion always corresponds to u, the vertical vector component (As will be made clear in Section 3.3.2, this causes the resulting mosaics to always be horizontally oriented, as will be seen in Section 5.3.2.). If the primary motion in the original image sequence is oriented vertically, then the input images are rotated before motion analysis is performed.

Again, given the Fourier transforms of the two images, F_1 and F_2, the cross-power spectrum of these two images is given by Equation 2.7. In theory, the inverse Fourier transform of the cross power spectrum (ICPS) produces a function with an impulse at the displacement coordinates, (x_0, y_0) which correspond to (u,v). In practice, the function will not be a pure impulse, but there will be a visible peak at the coordinates corresponding to the most dominant motion (Fig. 2.7). For a scene displaying motion parallax, there may be several motions at slightly different velocities present between two adjacent frames. In this case, there shall be several peaks of varying heights, and the highest peak is chosen as representative of the overall motion.

In this implementation of the phase correlation algorithm, all input images are resized to 256x256 images. This is done to facilitate the use of the implementation of the Fourier transform used in this work, which requires that the pixel width and height of the input images be powers of 2. The resizing algorithm uses a simple nearest-neighbor resampling scheme. Once phase correlation is performed, it is straightforward to use the resize factors to obtain the correct translations. Suppose we use rescaling factors f_x and f_y for the vertical and horizontal dimensions, ***height*** and *width*, respectively. Then, we have

$$f_x = \frac{256}{height}, \qquad f_y = \frac{256}{width}, \tag{3.7}$$

Once phase correlation has been performed on the pair of 256x256 images, giving us (x_0, y_0), then the true displacement between the images in their original resolution is given by

$$u = \frac{x_0}{f_x}, \qquad v = \frac{y_0}{f_y}. \tag{3.8}$$

Resizing the original images does introduce some error to the motion analysis results. The horizontal and vertical errors, e_x and e_y, are directly related to the size of the original images:

$$|e_x| < \frac{1}{f_x}, \qquad |e_y| < \frac{1}{f_y}. \tag{3.9}$$

We do not address correcting for accumulative error in this work, and, in practice, the errors tend to be small compared to the magnitude of the recovered vectors.

The results of the phase correlation algorithm may be affected by a phenomenon called *Discrete Fourier Transform leakage*, or DFT leakage. DFT leakage occurs in most Fourier transforms of real-world images, and is caused by the discontinuities between the opposing edges of the original image. Although a real-world image is a finite and non-periodic set of data, the DFT algorithm assumes that the data is infinite and periodic. Hence the edge discontinuities present within the image (which is where the assumption of periodicity fails) tend to produce high axis components in the Fourier transforms of those images. This phenomenon is illustrated in Figure 3.5.

In order to deal with DFT leakage, a mask is applied to each image prior to calculating its Fourier transform. A common suggestion is that this mask be based

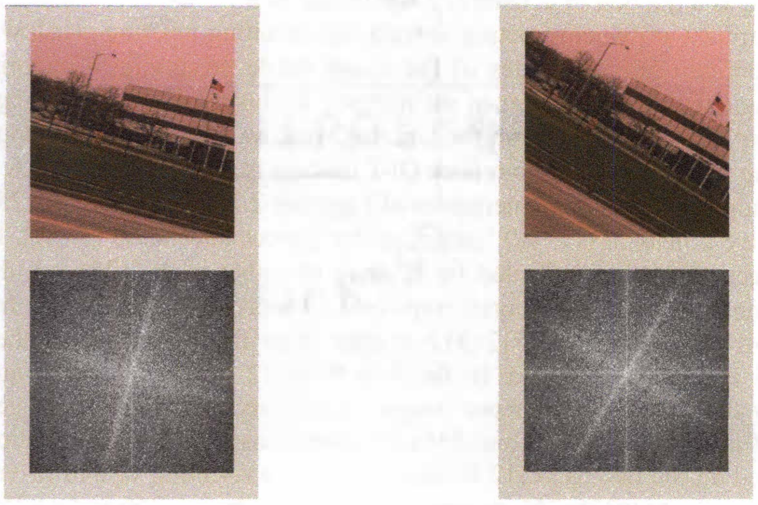

(a) (b)

Figure 3.5 DFT leakage. The top images of (a) and (b) are rotated versions of one another, and the bottom images are their Fourier transforms. We use these two images in order to show that the strong components which are present at the axes are clearly independent of the rotations of the images. These axis components are caused by DFT leakage, which is caused by the sharp disparity in intensity between the opposing edges of the images.

on the Hamming window [46], which is a tapering function that increasingly reduces the intensity values of the image pixels as they get further from the center of the image, producing a vignetting effect on the image. The equation for the 1-dimensional Hamming window, which would provide the 1D weights of the tapering window, is

$$H(x) = 0.54 + 0.46\cos\left(\frac{\pi x}{a}\right). \tag{3.10}$$

Another suggestion is to use a variant of the same function, called the Hanning window:

$$H(x) = 0.5 + 0.5\cos\left(\frac{\pi x}{a}\right), \tag{3.11}$$

Both functions are similar to the Gaussian function, which may be substituted readily:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-(x-\mu)^2/\left(2\sigma^2\right)\right\}, \tag{3.12}$$

where μ is the mean of the Gaussian and σ is the standard deviation. In order to produce the vignetting effect, μ corresponds to the center of the images, while σ is computed as a fraction of μ, usually close to half of μ.

Regardless of which function is used, the resulting tapering window removes the discontinuities at the sides of the image while preserving a majority of the information towards the center of the images. In this work, the Hamming window (Equation 3.10) is used to form the tapering window. An example of applying the Hamming window in order to reduce DFT leakage is shown in Figure 3.6.

A straightforward implementation of Equation 2.7 on a pair of 256x256 images would give a peak at (x_0', y_0') indicating the correct translation (x_0, y_0). However, the correct translation indicated by x_0' may be either x_0 or -(256-x_0), due to the symmetrical nature of the Fourier transform. The same is true for y_0'. In order to avoid this ambiguity, two 512x512 images (four times the size of the original 256x256 images) are created. In the first 512x512 image, its fourth quadrant is filled by the first 256x256 input image. In the second 512x512 image, its first quadrant is filled by the second 256x256 input image. Phase correlation is then performed on the two 512x512 images. Using the resulting x_0' and y_0' values, (x_0, y_0) between the two 256x256 images is calculated to be x_0' -256 and y_0'-256.

In order to decrease the chances of false registrations, and also to speed up the registration process, we use four parameters, u_{min}, u_{max}, v_{min} and v_{max}, to specify a search region within the ICPS function to which the search for the peak of the function is restricted. These parameters are of course converted using f_x and f_y before they are used to specify the search region within the ICPS function (whose dimensions are always based on the 256x256 images).

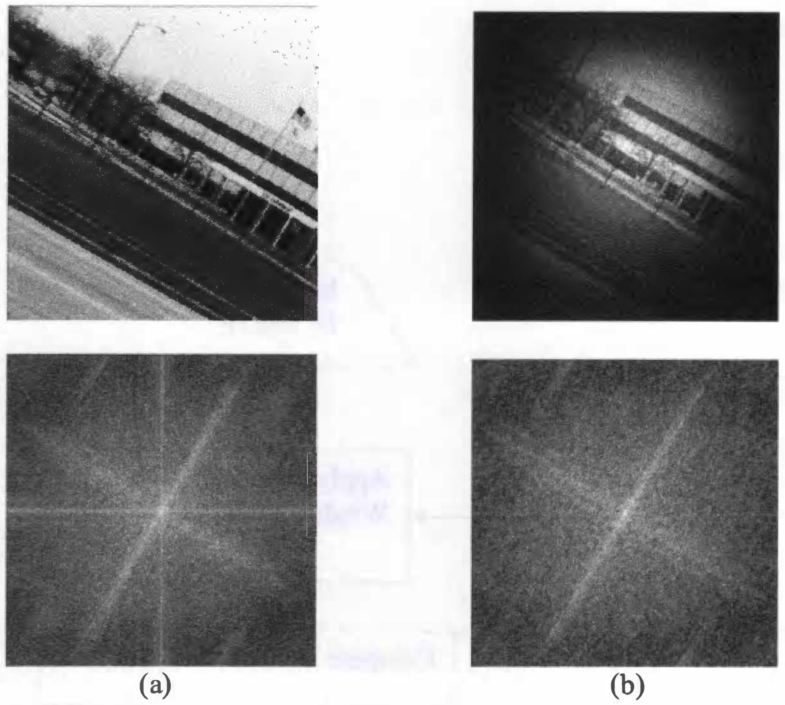


Figure 3.6 Eliminating DFT leakage. (a)A test image and its Fourier transform. (b) Result of applying the tapering window. Notice the reduction of axis components. Note that we use a tilted image so that the frequency components of the scene may be easily differentiated from the axis components caused by DFT leakage.

We summarize our phase correlation algorithm in Figure 3.7. There are two sets of parameters that are specified in the algorithm: one parameter for the tapering window and four parameters for the ICPS peak value search. For the tapering window, the parameter is *a*, which is the Hamming window parameter which determines the width of the curve of the Hamming function, which in turn determines the 1D weights of the tapering window. The four parameters, u_{min}, u_{max}, v_{min}, and v_{max} are upper and lower bounds limiting the search region when finding the peak value of the inverse cross power spectrum.

3.3.2 Strip Selection

Once the horizontal and vertical displacements between two images are known, strips are acquired from one of the images based on those displacements. For a pair of adjacent images, the strip is formed from the more recent image in the sequence. The principal behind strip selection for multiperspective mosaicking is to select strips that are perpendicular to the motion flow.

Recall from Section 3.3.1 that the horizontal motion of the sequence, v, corresponds to the primary motion of the sequence. Therefore, the horizontal dimension of a strip sampled from an image D_n is equal to v found using phase correlation performed using input images D_n and D_{n+1}. The vertical dimension of the strip will be y, the vertical dimension of D_n, (which should be the same for all images in the sequence). The strip is sampled from the center of D_n. Note that

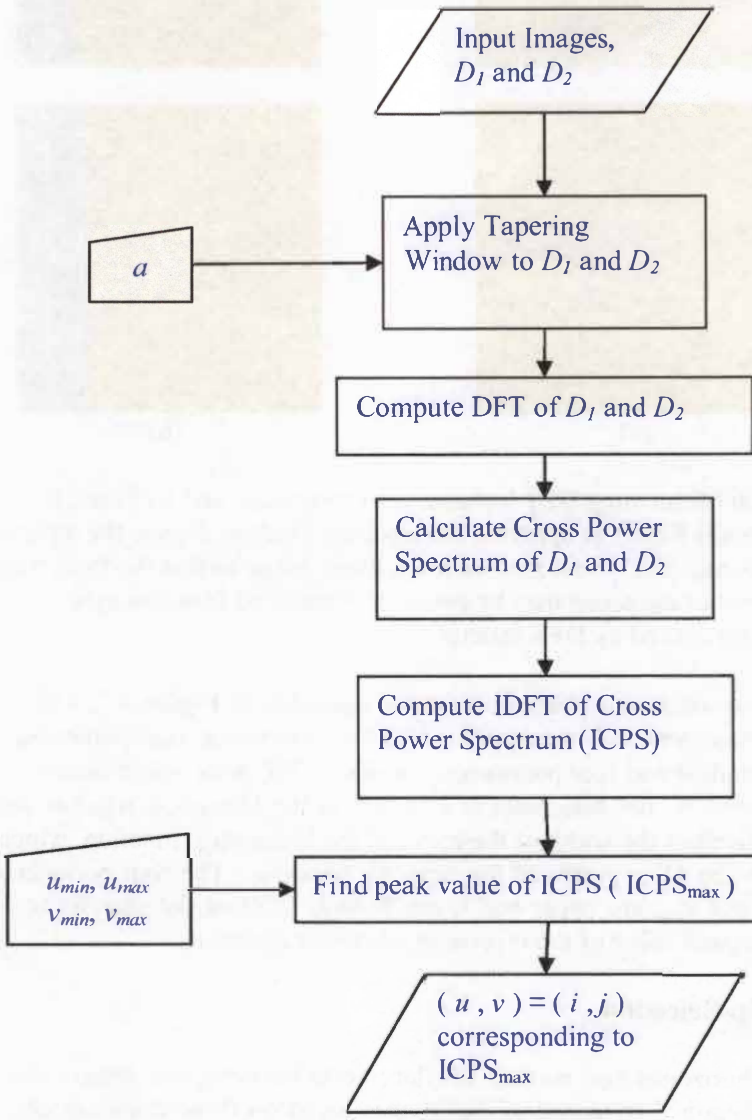

Input Images,
D_1 and D_2
Apply Tapering Window to D_1 and D_2
a
Compute DFT of D_1 and D_2
Calculate Cross Power Spectrum of D_1 and D_2
Compute IDFT of Cross Power Spectrum (ICPS)
u_{min}, u_{max}
v_{min}, v_{max}
Find peak value of ICPS ($ICPS_{max}$)
$(u, v) = (i, j)$ corresponding to $ICPS_{max}$

Figure 3.7 Phase Correlation Algorithm.

since the horizontal dimensions of strips correspond to the primary motion, and the mosaic strips are joined together along the strip edges perpendicular to the primary motion, the lengthwise orientation of the mosaics is horizontal. (The orientation of the mosaics is not important; the output images may be rotated easily.)

3.3.3 Merging of Strips

When combining two strips together, the secondary motion is used to align adjacent strips properly. However, although the strips may be properly aligned, seams may still be noticeable due to misalignments (caused by motion parallax or rotation) and inconsistent lighting. A simple blending scheme is used in order to reduce the visual discontinuity caused by seams. Suppose in the mosaic D_m, we have two strips sampled from two consecutive images, D_1 (the image on the left) and D_2 (the image on the right). The blending function is a one-dimensional function that is applied along a line orthogonal to the seam of the strips. For a coordinate i along this line, the intensity of its pixel in D_m is determined by

$$D_m(b-\frac{w}{2}+i)=\underbrace{(1-\frac{i}{w})}_{A_1}\underbrace{D_1(c_1+\frac{w_1}{2}-\frac{w}{2}+i)}_{B_1} \quad + \qquad\qquad (3.13)$$
$$\overbrace{(\frac{i}{w})}^{A_2}\overbrace{D_2(c_2-\frac{w_2}{2}-\frac{w}{2}+i)}^{B_2}, \qquad i=1\ldots w\ ,$$

where c_1 and c_2 are the coordinates corresponding to the centers of D_1 and D_2, respectively, w_1 and w_2 are the widths of the strips sampled from D_1 and D_2, $w=\min(w_1,w_2)$, and b is the mosaic coordinate corresponding to the boundary between the two strips. The terms A_1 and A_2 are weights for the pixel intensities for D_1 and D_2, while B_1 and B_2 are the pixel intensities themselves. For color images, this function is applied to the red, green, and blue channels of the image. At a seam, this function adds weighted pixel values from the images that intersect at the seam. The weights of each pixel in a strip is a function of the distance of the pixel from the intersecting seam; the weights increase as pixels get closer to the center of the strip from which they are sampled, and decrease as they get further. At the seam, the weights for pixels from both strips in an adjacent pair are equal, so that both adjacent images contribute equally to the pixel values at the seam. Note that, for a strip, more information is sampled from its source image than is specified by the pixel-wise primary motion computed for that image. The extra information sampled for a strip 'bleeds' into the adjacent strip in order to achieve a feathering effect. The amount of information sampled from two neighboring strips to perform the blending at their boundary corresponds to the pixel-wise width of the smaller of the two strips. Because the smaller width is used, there is no danger of non-adjacent strips being blended together. The blending process is illustrated in Figure 3.8.

After the blending is complete, the two strips have been successfully mosaicked. The process is then repeated for each subsequent frame in the video

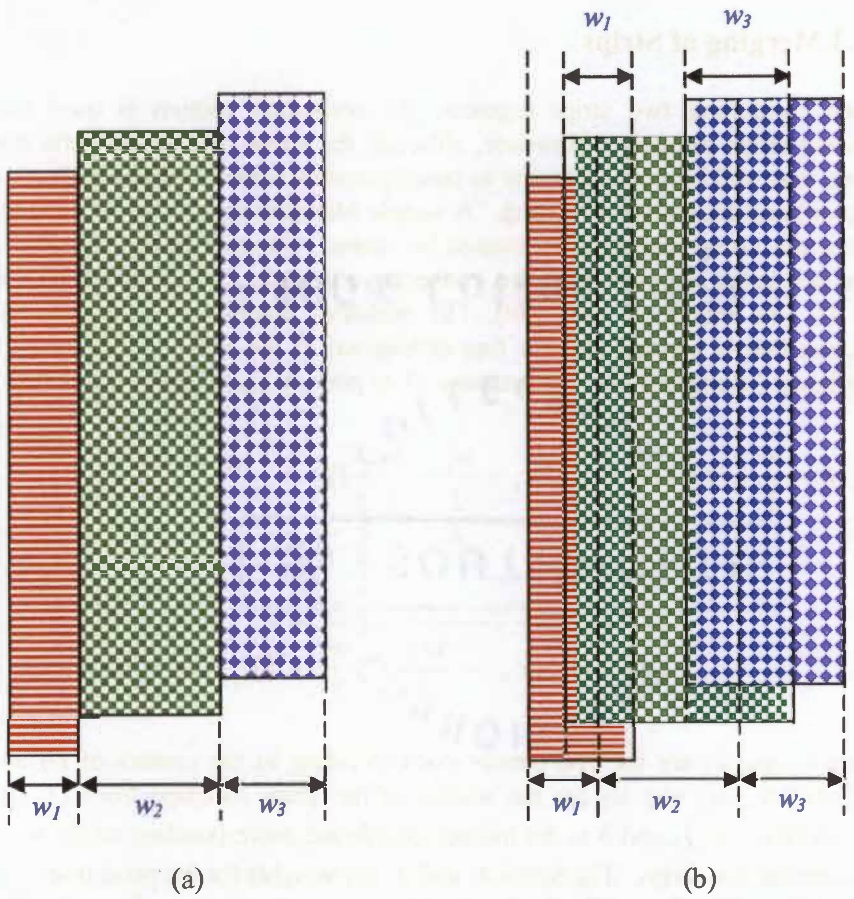


Figure 3.8 Blending of strips. a) Three consecutive strips without blending. Their widths correspond to the recovered primary motions of the images they were sampled from. b) The same three strips with blending. At the boundaries between strips, additional information is sampled from the frames corresponding to the strips to be used for the blending. The pixel-wise width of the additional information for both strips is always half the pixel-wise width of the smaller strip. In this way, only strips that are adjacent to one another are blended into one another, and there is no possibility of unintentional blending between strips that are not adjacent to one another. For instance, in the example shown above, the regions where blending occurs between the middle strip and its two neighbors has the same pixel-wise width of its neighbors. This is because the middle strip is larger than both of its neighbors; hence the pixel-wise widths of the blending regions are based on the pixel-wise widths of the neighboring strips.

sequence. After each cycle of the merging process, the vertical and horizontal displacement of the last strip in the mosaic is recorded, and this information is used as the anchor for the next strip in the mosaic. Once every frame in the video sequence has been processed, the mosaic is complete.

3.4 Remarks

It should be noted that the data constraints of our system are rarely rigorously met in any undervehicle video sequence, or most real-world video sequences for that matter. However, for a scene where there is no great disparity between elements in the scene, this technique suffices as long as the video sequence is sufficiently dense. One important consideration is the inter-frame motion of the sequence to be mosaicked. The effects of different video frame rates, or more specifically, different average inter-frame motions, are illustrated in Figure 3.9. These discontinuities happen because there is small motion parallax in our sequences, and hence different elements may move past the field of view at different speeds. Yet, within a single strip, all elements in that strip are being sampled at the same rate. This is what causes the types of discontinuities shown in Figure 3.9. Smaller inter-frame motions tend to produce less visible discontinuities in the resulting mosaic. Recall from Section 2.1.2 that Rousso et al. [26] used interpolated views between video frames to increase the sampling rate of the strips and hence improve continuity between strips. This is essentially the same as increasing the frame rate of the original video. In this work, we do not attempt to increase the frame rate by view interpolation. Our only suggestion is that the original video be captured at an acceptable frame rate relative to the motion of the platform used in the capture process. The influence of inter-frame motion, high or low, on the results will be seen in Chapter 5.

The selection of the registration parameter a (the tapering window parameter) and the search region parameters were not discussed here. There is some logic to how these parameters may be selected, but some experiments were also performed to verify that logic. The selection of these parameters and the experiments performed to support these selections are reserved for discussion in Chapter 5.

Finally, we note that the method of perspective and barrel distortion detailed here is somewhat simplistic in that the parameters are chosen manually, using input images as a visual reference to gauge the 'correctness' of the parameters. Our focus in this work was not to implement sophisticated perspective and barrel distortion correction algorithms; we only desired an approximate correction to reduce the extreme distortions that were clearly visible. For our current purposes, an approximate correction suffices, though possible improvements to our distortion correction methods will be discussed in Chapter 7.

We have now completed our description of the single-mosaic representation algorithm used in this work. Next, we will describe the layered-mosaics representation scheme used to process video of scenes displaying large motion parallax.

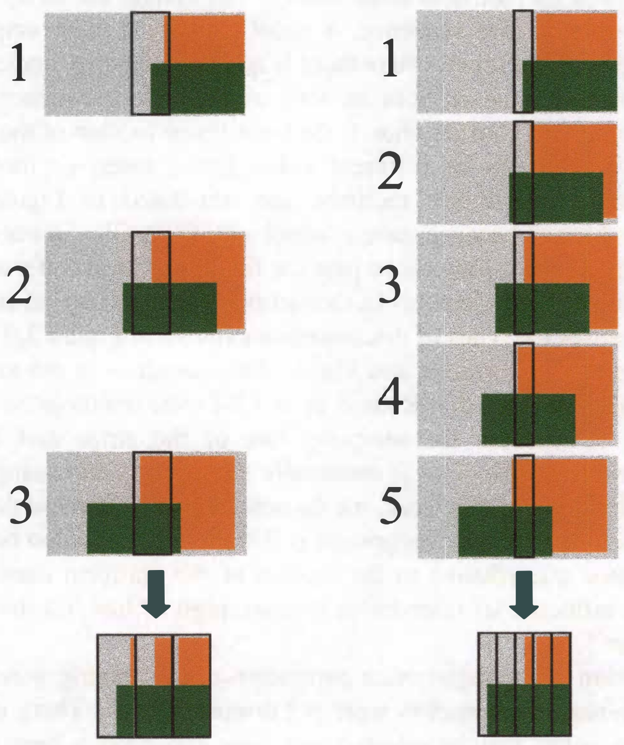


Figure 3.9 The effect of different inter-frame motions on the resulting mosaics. High inter-frame motions tend to result in large discontinuities in the mosaic, while low inter-frame motions reduce these discontinuities. Note that the strip width changes according to the inter-frame motion: a wide strip for large motion and a narrow strip for small motion.

Chapter 4

Layered-Mosaics Representation

This chapter describes the algorithms used to create a layered-mosaics representation of a video sequence. In Section 4.1, we begin with a description of the general framework of our solution and describe in detail the problem we are attempting to solve. Next, in Section 4.2, we describe the various processes involved in mosaicking a sequence into layers, including motion analysis, model initialization, spatial support determination, and layer composition. We end with remarks in Section 4.3.

4.1 General Framework

The principles used to create single-mosaic representation are now extended to the process of creating a layered-mosaics representation. Adjustments are made to the mosaicking process to facilitate the extraction of layers. The most important of these is the use of the Lucas-Kanade motion tracking algorithm to perform motion analysis. Again the process is divided up into several program modules. Before discussing these modules, the data constraints used earlier are revised for the case of a multi-layer planar scene.

4.1.1 Data Constraints

For the single-mosaic representation, it was assumed that the scene exists entirely on a single plane parallel to the viewing plane. The extension to layered-mosaics representation is straightforward: it is now assumed that the scene is composed of several planar layers that are at varying distances from and parallel to the viewing plane. Suppose we have three points M_1, M_2, and M_3 on three planes of the scene P_1, P_2, and P_3 respectively (Figure 4.1), and that these three points lie on a ground plane orthogonal to P_1, P_2, and P_3. It is observed that the distance between the points m_1 and m_2 and the distance between their corresponding points m_1' and m_2' are not equal. This is caused by the disparity in the normal distance of the planes P_1 and P_2 from the viewing planes. In a video sequence, this is observed as motion parallax; objects in the foreground move past the camera's field of view faster than objects in the distance. Also, it is observed that there is no projection of the point M_3 on the viewing plane of C, due the occluding plane P_2.

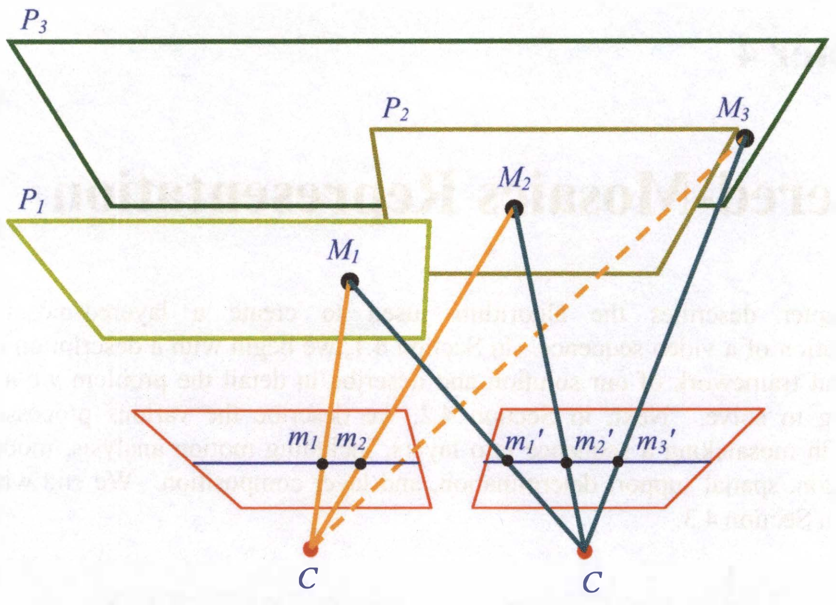


Figure 4.1 Multi-layered configuration of planar scenes. The distances m_1-m_2 and m_1'-m_2' are not equal, while there is no projection of M_3 on the viewing plane of C at all.

The disparity between the distances m_1-m_2 and m_1'-m_2' is directly related to the disparity in the normal distances of P_1 and P_2 from the viewing planes. Therefore, assuming the scene and camera's movement constraints are met, the spatial support for each layer may be inferred by obtaining the translation velocities of pixels between consecutive frames. Pixels exhibiting the same translation are assigned to the same layer.

Video acquisition for the layered-mosaics representation is similar to the acquisition method described in Section 3.1. The camera, placed on a mobile platform, moves in a straight line past the scene while the camera is pointed towards the scene. For reasons that will be made clear in Section 4.2.2, it is required that the speed of the moving platform remain fairly constant throughout the entire acquisition process. As before, the video obtained is decomposed into individual frames, which becomes the input to the program modules. Figure 4.2 illustrates a typical acquisition setup.

4.1.2 Processing Modules

The mosaicking process for layered-mosaic representation is similar to the single-mosaic process for each individual layer mosaic. The differences are a) motion analysis is now performed using the Lucas-Kanade method and b) pixels are divided amongst the mosaics according to their velocities during the merging process. In addition to the mosaicking modules, model initialization for the layer representation is performed manually beforehand, and occluded sections of the mosaics are filled in using a mosaic composition module. The preprocessing module is similar in every aspect to the one described in Section 3.2, and will not be discussed here again. A general outline of the algorithm is shown in Figure 4.3.

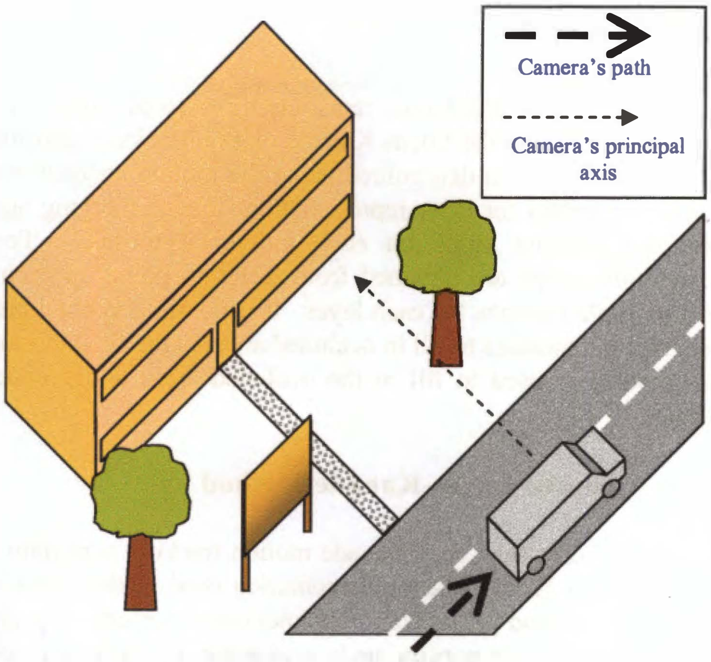


Figure 4.2 Video acquisition for outdoor / road scenes.

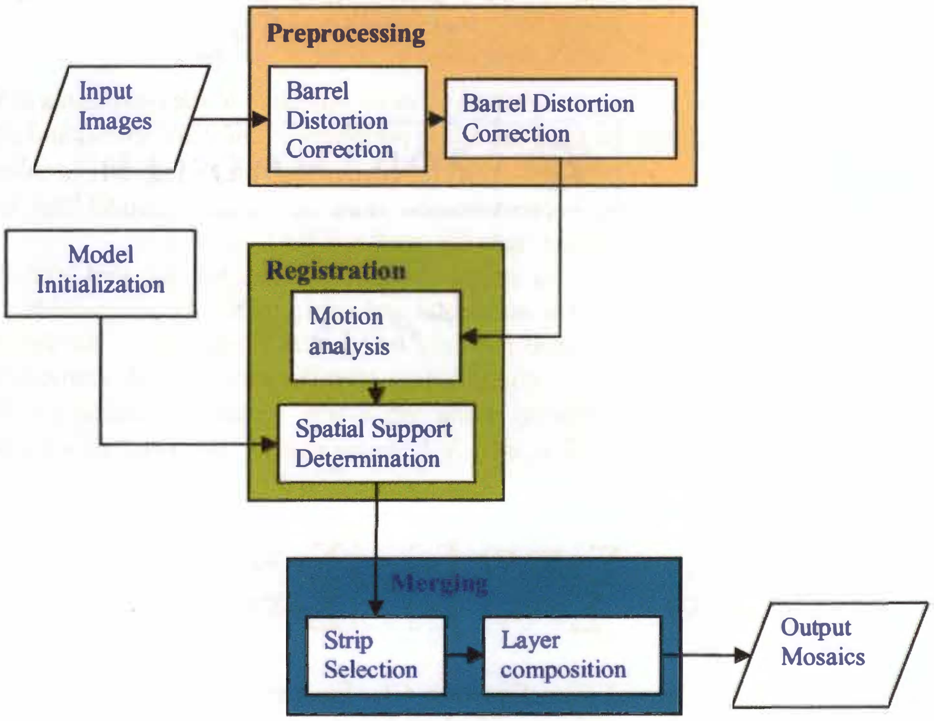


Figure 4.3 Mosaicking Process for Layered-Mosaic Representation.

4.2 Mosaic Creation

For the layered-mosaic representation, registration is also performed using motion analysis, but this time using the Lucas-Kanade motion tracking algorithm. Spatial support for each layer is then determined using the motion analysis results, based on a pre-initialized model for layer representation. Image merging again consists of selecting and aligning strips on each individual mosaic. To deal with occlusions, multiple strips are obtained from different points in each frame and used to form multiple mosaics for each layer. It is possible to combine the spatial data in these multiple mosaics to fill in occluded areas in the final mosaics. A layer composition module is used to fill in the occluded areas and produce the final layered mosaics.

4.2.1 Motion Analysis: Lucas-Kanade Method

A description of the original Lucas-Kanade motion tracking algorithm has already been given in Section 1.2.2. The implementation used in this work is based on those described in [28] and [47]. This implementation performs a weighted least-squares fit of local first-order constraints to a constant model for the velocity, V, in each small spatial neighborhood (denoted by Ω) by minimizing

$$\sum_{x\in\Omega} W^2(x)[\nabla I(x,t)\cdot V + I_t(x,t)]^2, \tag{4.1}$$

where $W(x)$ is a window function that gives more influence to the constraints at the center of the window than to the ones at the periphery, x and t are spatial and time variables, and I and ∇I are the pixel intensity and pixel intensity gradient, respectively. In short, this implementation finds the velocity model that best describes the spatial and temporal intensity gradients for a given pixel.

Suppose for each pixel in an image frame, the velocity associated with that pixel is (u,v), which describes the horizontal and vertical velocity components. To compute these velocities, we need not only the current image frame, but the two image frames before and the two image frames after the current image frame in the sequence. The intensity gradients along the x-axis, y-axis, and along the five consecutive frames are ∇I_x, ∇I_y, and ∇I_t, respectively. We need to solve the linear system

$$\begin{bmatrix} \sum w\nabla I_x^{\,2} & \sum w\nabla I_x\nabla I_y \\ \sum w\nabla I_x\nabla I_y & \sum w\nabla I_y^{\,2} \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum w\nabla I_x\nabla I_t \\ \sum w\nabla I_y\nabla I_t \end{bmatrix}, \tag{4.2}$$

which is a solution derived from Equation 4.1. Each element in the summations are the smoothed versions of the specified gradients.

Before the gradients are calculated, the five image frames are smoothed spatially and temporally to reduce the effects of noise on the gradient calculations. Spatial smoothing is performed by convolving the image with the following kernel:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

This kernel merely computes the average of the pixel and its 8 neighboring pixels. Temporal smoothing for the intensity of the current pixel, I_m, is computed using a Gaussian mask convolved with the values of current pixel and its corresponding pixels in the last 6 frames. The equation for this operation is

$$I_m = \sum_{n=1}^{6} \left(\frac{1}{\sigma\sqrt{2\pi}} \mathbf{exp}\left\{ {(1+n-\mu)^2}\Big/{2\sigma^2} \right\} \right) I_{m+n-6} \ , \tag{4.3}$$

where μ is the mean of the Gaussian (6 in this case, since we want the most weight given to the current pixel) and σ is the standard deviation of the Gaussian (set to 1 in this work).

Once spatiotemporal smoothing is complete, the intensity gradients ∇I_x, ∇I_y, and ∇I_t are calculated for each pixel in the current image frame (and only the current image frame) using the following convolution kernel:

$-\frac{1}{12}$	$\frac{8}{12}$	0	$-\frac{8}{12}$	$\frac{1}{12}$

which is the same convolution kernel used in [28] and [47] to compute the intensity gradient. The image is convolved with this kernel along the x-axis to calculate ∇I_x, along the y-axis to calculate ∇I_y, and along the five frames to calculate ∇I_t. These values are then used to calculate ∇I_x^2, $\nabla I_x \nabla I_y$, ∇I_y^2, $\nabla I_x \nabla I_t$, and $\nabla I_y \nabla I_t$ for each pixel. To obtain the summation elements in Equation 4.2, these gradients are smoothed using a separable, isotropic 5x5 kernel, with effective 1D weights (0.0625, 0.25, 0.375, 0.25. 0.0625). After the smoothed gradients have been obtained, they are used to solve for u and v in Equation 4.2. Once these have been calculated for each pixel in the image, the result is a flow field with velocity information for each pixel in the image.

The Lucas-Kanade implementation in this work is only accurate for pixel speeds of up to 2 pixels/frame. Higher speeds tend to produce inaccurate computation results. In order to deal with the case of higher pixel speeds, a simple multi-resolution motion analysis scheme is used. First, two rescaling factors, f_1 and f_2, are specified manually. These factors are used to decrease the size of the input images so that pixel movements within the sequence are smaller, and can be detected with more accuracy using our Lucas-Kanade implementation. For a single input image, two rescaled versions of that input image are created using f_1 and f_2. The new dimensions of the rescaled versions would be calculated as

$$\begin{aligned} height_1' &= f_1 \times height_1, \qquad & width_1' &= f_1 \times width_1 \ . \\ height_2' &= f_2 \times height_2, \qquad & width_2' &= f_2 \times width_2 \end{aligned} \tag{4.4}$$

A nearest-neighbor sampling scheme is used to sample the pixels of the rescaled images. Note that $0<f_1<1$, $0<f_2<1$, and $f_1<f_2$. Two rescaled versions of the input image are needed because the image sequence is processed twice, at two different resolutions. We do this so that we can detect the higher velocities at the lower resolution, but still retain accurate measurements of the lower velocities at a higher resolution. First, we rescale the image sequence using f_1. We then compute a (u',v') vector for each pixel in image using the implementation of the Lucas-Kanade algorithm discussed above, creating a flow field at f_1-resolution. We then rescale the flow field to original resolution, recomputing each (u',v') vector using

$$u = \frac{u'}{f_1}, \qquad v = \frac{v'}{f_1}. \tag{4.5}$$

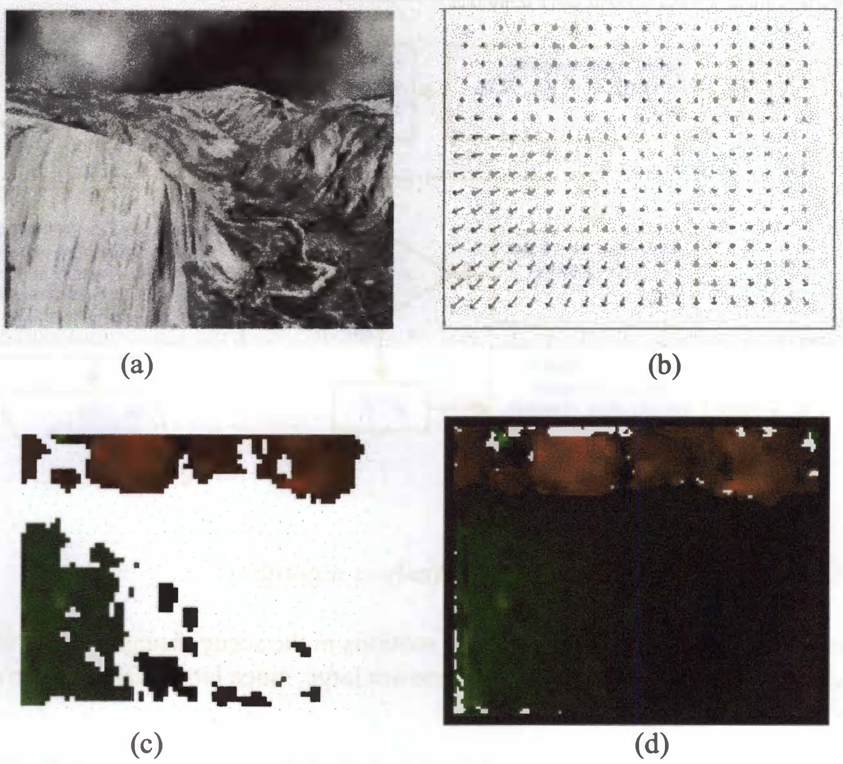
We use nearest-neighbor sampling to assign vectors to pixels with unspecified vectors when restoring the rescaling flow field to original resolution. Then, for each pixel in the flow field, we keep its vector only if the vector's magnitude, $|V|$, is above a threshold v_{lim}. This threshold is determined by the velocity magnitude that is equal to 2 at f_2-resolution. This is calculated using

$$v_{lim} = 2f_2. \tag{4.6}$$

Any velocity with $|V| < v_{lim}$ in the flow field is discarded. Then, the image sequence (not the flow field) is rescaled (from the original resolution, not f_1-resolution) using f_2. The (u',v') vectors are calculated for each pixel, creating a second flow field at f_2-resolution. As before, this second flow field is rescaled to original resolution, and the (u',v') vectors are again rescaled using Equation 4.5, substituting f_2 for f_1. Finally, we compare the second flow field with the first flow field, and for pixels in the first flow field whose vectors we discarded earlier, we substitute the corresponding (u,v) vectors from the second flow field. This process is demonstrated in Figure 4.4 using the Yosemite video sequence which was created by Lynn Quam at SRI [48] using f_1=0.25 and f_2=1.0 (original resolution).

So far, we've described the Lucas-Kanade motion analysis algorithm with respect to I, the pixel intensity only. However, we are using color images, defined by the three R,G and B channels. The Lucas-Kanade algorithm is applied to all three channels separately. Different velocity measurements may be obtained for each channel. We pick the highest velocity estimate among the three as the correct estimate, with the reasoning that intensity changes due to motion may be less apparent in one or two channels, but if there is actual intensity change due to motion, at least one of the channels will exhibit a sharp change, resulting in a high velocity estimate.

We summarize our implementation of the Lucas-Kanade motion analysis algorithm in Figure 4.5. Although we list four parameters in the flow chart, the Gaussian function parameters for the temporal smoothing operation are set constant (μ = 6, σ = 2). So there are really only two free parameters in our implementation: f_1 and f_2, the rescaling factors for multi-resolution motion analysis.



● -5 pixels/frame	● -5 pixels/frame
● -4 pixels/frame	● -4 pixels/frame
● -3 pixels/frame	● -3 pixels/frame
● -2 pixels/frame	● -2 pixels/frame
● -1 pixels/frame	● -1 pixels/frame

Figure 4.4 Lucas-Kanade Motion Analysis. (a) Sample frame from the Yosemite sequence [48]. (b) The correct flow field for the sequence. (c) Results of motion analysis at original 1/4 resolution. Green regions denote pixels moving left, and red pixels denote regions moving right (for simplicity vertical motion is excluded in these visualizations). Color intensity is directly proportional to velocity maginitude (refer to the legend for specific intensity-magnitude correlations). The white regions denote velocities that were computed to be less than v_{lim}, and are thus discarded. (d) Results of motion analysis at original resolution. The white regions in (c) are filled in using the velocities recovered at original resolution.

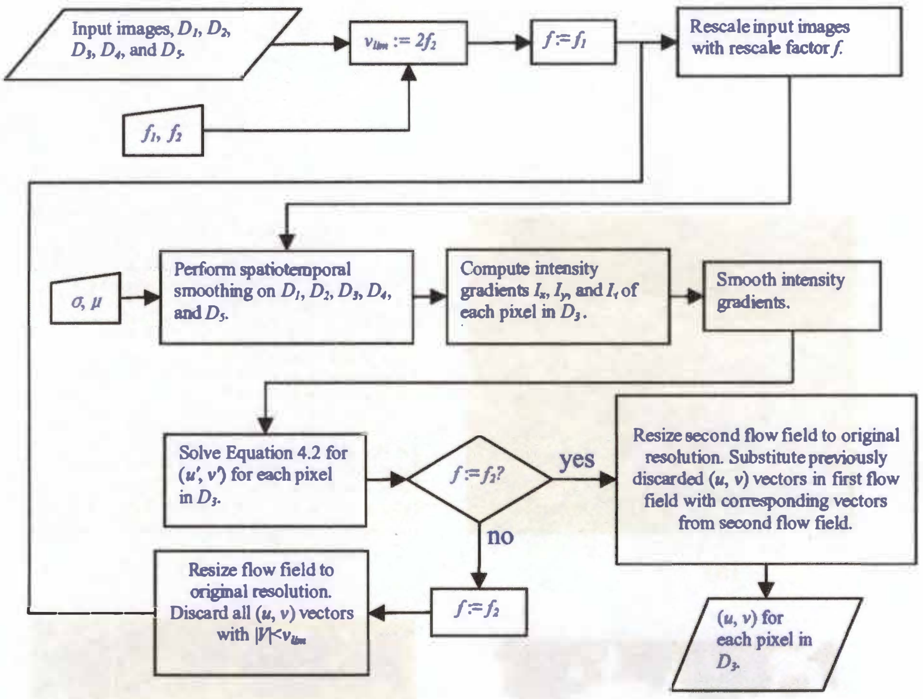

Input images, D_1, D_2, D_3, D_4, and D_5.
$v_{lim} := 2f_2$
$f := f_1$
Rescale input images with rescale factor f.
f_1, f_2
σ, μ
Perform spatiotemporal smoothing on D_1, D_2, D_3, D_4, and D_5.
Compute intensity gradients I_x, I_y, and I_t of each pixel in D_3.
Smooth intensity gradients.
Solve Equation 4.2 for (u', v') for each pixel in D_3.
$f := f_2$?
yes
no
Resize second flow field to original resolution. Substitute previously discarded (u, v) vectors in first flow field with corresponding vectors from second flow field.
Resize flow field to original resolution. Discard all (u, v) vectors with $|V| < v_{lim}$
$f := f_2$
(u, v) for each pixel in D_3.

Figure 4.5 Lucas-Kanade Motion Analysis algorithm.

These change as the magnitude of the motions in the scene change - f_1, for instance, tends to be small if motions in the scene are large, since larger motions can only be detected at smaller resolutions.

4.2.2 Model Initialization and Spatial Support Determination

Layer extraction in this work is split into two distinct processes: model initialization and spatial support determination. Spatial support determination is performed based on the parameters specified in the model initialization. These parameters are a) number of layers in the scene, and b) velocities associated with each layer. Since it is assumed that the scene and camera movement obeys the constraints defined in Section 4.1.1, all layers are assumed to follow the same motion model, which is purely translational motion of a rigid plane. Therefore, it is not required to specify separate motion models for each layer.

Determination of the model initialization parameters is performed manually by the user. The video sequence is observed to choose a number of layers that would adequately represent the scene. An estimate of the inter-frame motion for each layer is also obtained through observation of the video, and these estimates are used as the layer velocities. For a layer P_n, a velocity (u_n, v_n) is associated with it, with the component representing secondary motion (as defined in Section 2.4.2) typically set to zero. Model initialization using two frames as a reference is illustrated in Figure 4.6.

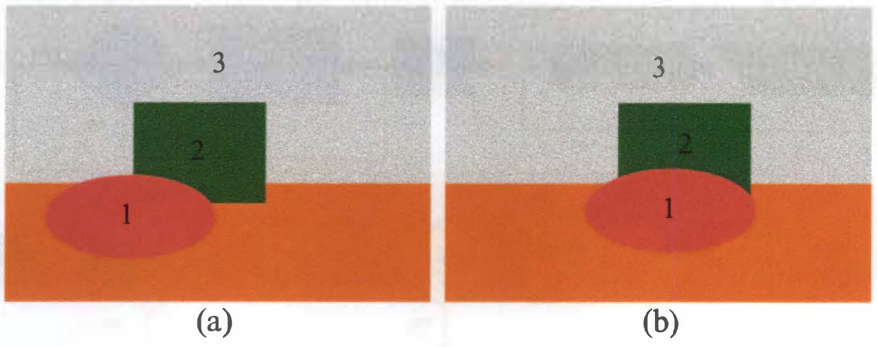


Figure 4.6 Model initialization of layers. Two mock video frames, (a) and (b), are used as a visual reference to perform model initialization. In this scene, a natural choice would be to designate separate layers to the plane of the object labeled 1, the object as labeled 2, and the background labeled as 3. The surface on which objects 1 and 2 lie on will most likely display non-translational affine motion, or, if the entire surface is homogenous, no apparent motion at all. No layer is initialized to represent this surface. Note that if the surface is homogenous, the spatial support determination process will most likely assign pixels corresponding to the surface to the background layer 3 if it also displays no apparent motion. Therefore, in this example, the scene is modeled as being represented by three layers.

The layer representation model may be initialized at any point before spatial support is determined. In this work, model initialization was performed before any other processing of the video frames. Once motion analysis of the frames has been performed, as described in section 4.3.1, we may determine spatial support for each layer. For a pixel in a given image, the Euclidean distance between its motion vector, (x, y), in 2D space and each of the layer-assigned motion vectors $(u_0, v_0 \ldots u_N, v_N)$, with N being the number of layers, is calculated. The shortest distance found indicates the layer that pixel is assigned to. In this manner, each frame is segmented according to the spatial support for each layer. This is repeated until each frame in the video sequence has been processed.

Note that we do not update the layer model after it has been initialized, and recall that one of the constraints placed on the camera movement was that the speed of the camera must remain fairly constant throughout the entire sequence. Because we do not update the layer model or any of the motion vectors associated with each layer during the spatial support determination process, the speed of the camera should not vary greatly, so that each layer displays the same motion properties throughout the entire sequence. If the motion of the camera varies throughout the sequence, the algorithm will lose track of many of the initialized layers, which will result in incorrect layer assignments during the spatial support determination process.

In a given frame, the collection of pixels assigned as belonging to a layer is henceforth referred to as that layer's *support region* within that frame. Motion analysis has provided an estimate for each layer's support region in each frame in

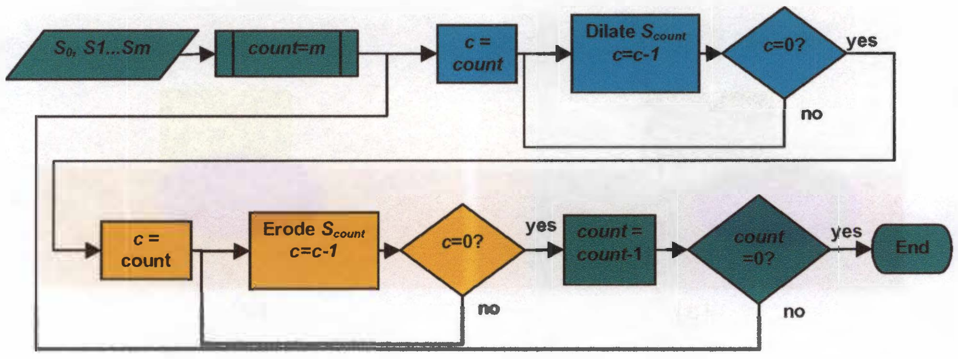


Figure 4.7 Hierarchical binary morphological operator. N is the number of layers and $S_0, S_1... S_n$ are the support regions for each layer. The order in which the regions are processed is arranged according to the magnitude of their velocities, with the S_0 indicating the layer with the smallest velocity and S_n indicating the layer with the largest. Beginning with S_n and ending with S_0 , the dilation and erosion operations are performed on each region in turn.

the video sequence. However, there may still be noticeable errors present in these support regions, due to inaccurate motion estimates. For layers whose velocities are relatively low, these errors tend to be small or nonexistent. Layers with higher velocities, however, tend to have large gaps in their support regions, where pixels have been assigned incorrectly to other layers. In order to reduce these incorrect assignments in a given frame, a hierarchical morphological operator was developed to close the gaps in each region.

The algorithm for this operator is shown in Figure 4.7. The dilation and erosion operations are performed as binary operations on each layer's support region in turn. Each region is dilated and eroded by the structuring element shown in Figure 4.8.

Dilation and erosion are usually performed in that order to form a closing operator [49]. The algorithm of Figure 4.7 is similar to a closing operator in that it fuses narrow breaks, eliminates small holes, and fills gaps in the regions' contours. This implementation performs the operations in a hierarchical manner. Starting with the region of highest velocity and ending with the region of smallest velocity, the extent to which the closing operation is performed is gradually lessened. This is done so that while closing the small gaps in the regions of small velocities, the results of closing the large gaps in the support regions of higher velocities is still preserved. If the closing operation were applied to each layer to an equal extent, the closing for regions of higher velocity, which typically display larger gaps, would be negated.

Once the morphological operations have been applied to each frame in the video sequence, the process of determining spatial support for each layer is complete. This information may now be used to form the layered mosaics.

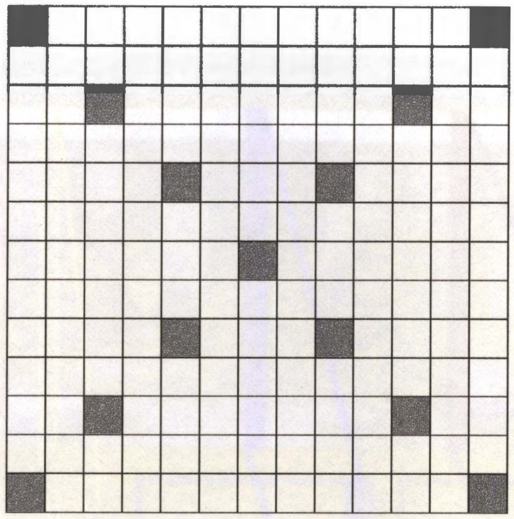


Figure 4.8 Structuring element for the hierarchical morphological operator. The element's form is comparable to a subsampled 13x13 grid, with the samples taken along the horizontal and vertical axes. This structuring element was chosen based on heuristic observation of the results.

4.2.3 Composition of Layered Mosaics

The challenge of representing partially occluded background elements in their entirety is dealt with using our layer composition method. To explain how this is done, we discuss the composition of a mosaic for a given planar layer, P_n, with partial occlusion. Again, strips are sampled from each frame from the video sequence, as was done for the single-mosaic representation. This time, however, there is no longer one global motion associated which each frame. Instead, each frame has been segmented according to the spatial support determination for each layer. So for a layer P_n, only those pixels that have been assigned to P_n, using the spatial support determination algorithm of Section 4.3.2, are referenced. For a given image frame, we wish to determine (x_0, y_0), the primary and secondary motions, which will determine the width and alignment of the strip. For each pixel assigned to P_n, the vector computed for that pixel is (x, y). To find (x_0, y_0), the average value of (x, y) for all pixels assigned to P_n are calculated. Hence,

$$x_0 = \frac{1}{d}\sum x, \quad x \in P_n, \qquad y_0 = \frac{1}{d}\sum y, \quad y \in P_n, \tag{4.7}$$

where d is the number of pixels in the given frame assigned to P_n. Strips are sampled from the frame according to (x_0, y_0), again with the width of the strip corresponding to the width of the primary motion. As it was in the single-mosaic representation, images are oriented so that the primary motion corresponds to y_0, and images are rotated accordingly if needed prior to processing. Only the intensity information of pixels belonging to the layer P_n is retrieved, while

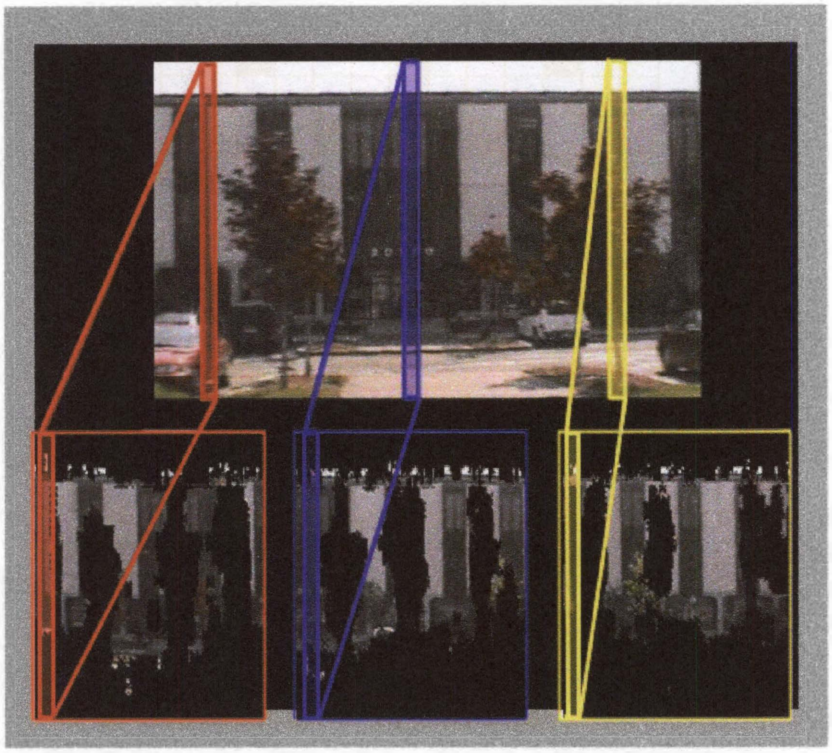


Figure 4.9 Creation of a reference mosaic and peripheral mosaics by sampling strips from different points in a frame.

information from pixels belonging to other layers is ignored. This will result in mosaics that have 'gaps' where there were occluding or background elements that did not belong to the layer P_n. Figure 4.9 illustrates this process.

The discussion of strip sampling above does not address one possible scenario: what if, for a particular layer, there are parts of the sequence that do not clearly exhibit the motion associated with that layer? In other words, layers containing disparate elements such as signboards and trees may not have elements representative of its motion at some point in the sequence. However, we still need strips to build the mosaic representing this layer, or the distances between these elements within a mosaic of that layer would be inaccurate. Currently, in this work, we do not attempt to accurately determine this distance, but instead use the most recently computed value of (x_0,y_0) for that layer if there are no vectors associated to a layer with which to compute (x_0,y_0). As it happens, in our current implementation, there is never an occasion when there are no vectors associated with a particular layer, since all vectors are assigned based on minimum distance to the layer vectors, not distance within a threshold.

In order to acquire a more complete representation of elements in the layer P_n, we create more than one mosaic of that layer. Each mosaic is created from strips sampled at different points in each frame. These strips are spaced apart evenly, and the pixel-wise distances of each strip from one another is known. Therefore, for P_n, we now have several mosaics M_1, M_2... M_k, where k is the number of mosaics that will be used in order to compose P_n. One of these mosaics, typically the mosaic composed from strips sampled closest to the center of each frame, (usually $M_{k/2}$) is used as a *reference mosaic* for composing P_n. The rest of the mosaics, because they are formed from strips sampled from either side of the center strip of each image frame, are referred to here as *peripheral mosaics*. Three parameters are used to determine how the strips for the peripheral and reference mosaics are sampled. The first parameter is k, the number of mosaics used to compose the layer. The other parameter is *dist*, the pixel-wise distance between the corresponding edges of the strips. The strips are always sampled with the reference strip close to the center of the image. Given the horizontal dimension of an image frame, *width*, the horizontal position of the edge of the first strip, y_a, is determined by

$$y_a = \frac{width - ((k-1)\times dist)}{2}, \tag{4.8}$$

after which consecutive strips are sampled at intervals of *dist* pixels. Figure 4.10 illustrates how these parameters would be used to sample strips from an image frame, using $k = 5$ as an example.

After the reference and peripheral mosaics have been created, there will still be noticeable 'noise' in the resulting mosaics, where local incorrect assignments of pixels will produce inconsistencies in the layers. To reduce these inconsistencies, we perform a simple morphological closing operation on each mosaic, using the

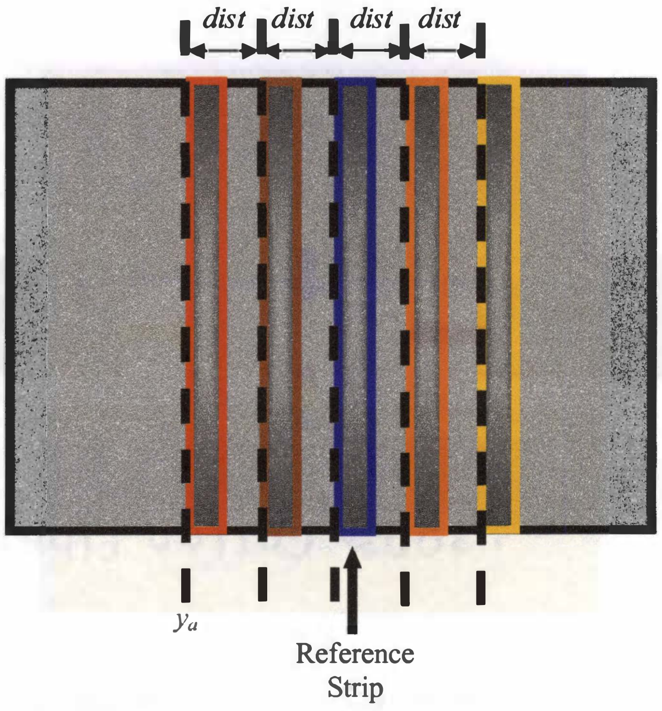
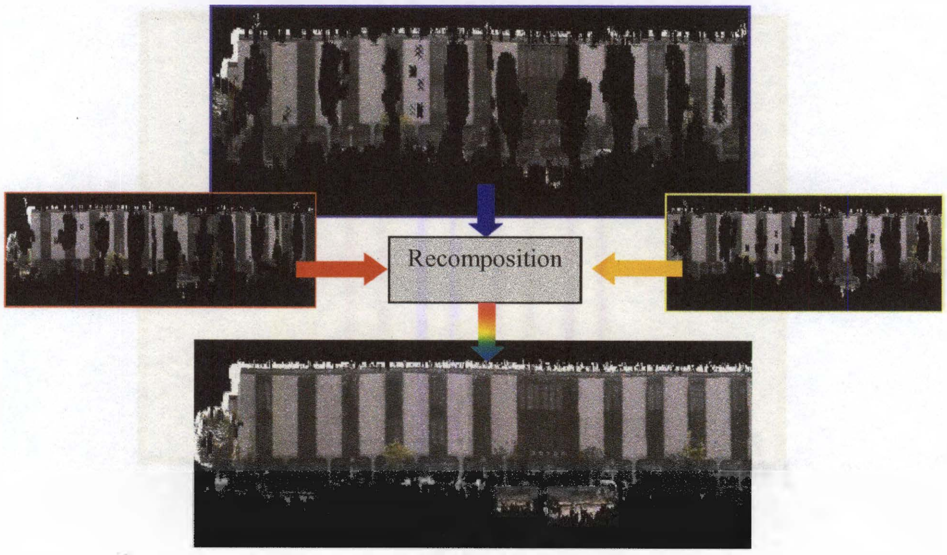


Figure 4.10 Sampling five strips from an image frame. Note that y_a is positioned so that the reference strip is close to (though not exactly at) the center of the image frame.

same structuring element of Figure 4.8. This time, however, the operation is performed on the *null* regions of each mosaic, i.e. the regions that were assigned as *not* belonging to that layer. The closing operation is only performed once, without the hierarchical looping of the morphological operator described in Section 4.2.2. The resulting, noise reduced mosaics are then used to perform the actual composition of the layer mosaic.

Now, since *dist*, the pixel-wise distance separating the strips sampled from each frame, is known, it is also known how the peripheral mosaics spatially correspond to the reference mosaic. This knowledge is used to fill in the 'gaps' in the reference mosaic, by gathering pixel intensity information from the peripheral mosaics that were created. First, the peripheral mosaics are ordered by the pixel-wise distance of their strips from the strips of the reference mosaic, from the smallest distance to the largest distance. Since the strips were sampled at equal distances apart, there will be two mosaics created from strips at the same pixel-wise distance from the reference strip; it does not matter which mosaic comes first in this order. Then, starting with the first peripheral mosaic, its pixel information is used to fill in the gaps of our reference mosaic. In most cases, the gaps in this mosaic will overlap with the gaps in our reference mosaic, so once all available pixel information has been obtained, the process is repeated for the next peripheral mosaic, and so on until all available pixel information from all the peripheral mosaics have been referenced. If the occlusions were not too large, and a sufficient number of mosaics were used, then the reference mosaic should now have all its gaps filled, making it a complete representation of our object of interest. Figure 4.11 illustrates this process.



(b)

Figure 4.11 Recomposing the reference mosaic using pixel data from the peripheral mosaics.

4.3 Remarks

In one aspect, we have loosened the constraints placed on the data in this algorithm, as opposed to the restrictions placed on the data of the single-mosaic representation: we no longer require that motion parallax in the scene be small. However, we have placed a constraint on the data for this algorithm that was not present before, which is the constraint that the speed of the moving platform does not vary greatly throughout the video sequence. Zhu's 3D LAMP representation [27] places a similar constraint on the data for their algorithm. The reason in both cases is the same: to simplify the tracking of layers throughout the entire sequence. If the speed of the platform were to vary greatly throughout the sequence, a more advanced feature tracking algorithm would have to implemented, as opposed to the straightforward motion analysis performed here, in conjunction with some framework for updating the motion models for each layer. As it stands, we have not addressed this problem yet in our implementation, though possible directions for improving the system in this respect will be discussed in Chapter 7.

One question that may arise is, why do we not use the layered-mosaics representation to process the undervehicle data , and therefore have just one unified method of dealing with both cases? The short answer is that it is possible to use the layered-mosaics representation to process the undervehicle data, but because of the nature of that data and the purpose of those mosaics, it is not efficient to do so. We do not require a layered representation of the underside of a vehicle because there are very few occlusions that can be removed to any meaningful degree, because motion parallax in the sequences is small. We only require a single overview of the scene for inspection purposes, and any objects hidden behind large undervehicle components cannot be detected in the visible spectrum. Also, creating a single mosaic between frames is much faster than attempting to compute spatial support from several layers. If we wish to extend the system to real time use in order to inspect several vehicles, say, in a parking lot, then the speed of the algorithm becomes an issue.

On the other hand, why aren't we applying the registration methods developed for the single-mosaic representation to the layered-mosaics representation? The largest difference between the two techniques lies in the registration method: phase correlation only gives us global motion estimates, whereas the Lucas-Kanade algorithm gives us local motion estimates. With phase correlation, we cannot directly infer layer assignments; some additional processing steps, including perhaps a block-based matching algorithm, are required to acquire layer assignments. Lucas-Kanade gives us layer assignment estimates right from the beginning, and the only challenge left is to refine those estimates.

We have now completed our description of the layered-mosaics representation algorithm used in this work. Next, we will describe the results obtained using the single-mosaic representation scheme.

Chapter 5

Single-Mosaic Representation Results

This chapter presents the experimental results for the single-mosaic representation algorithms proposed in Chapter 3. We will begin with the details of the data capture process and the data sets used in these experiments, and outline the parameters used for preprocessing and motion analysis of each of these data sets in Section 5.1. Then, we will discuss the experiments performed to determine the parameters used to perform phase correlation in Section 5.2. Finally, the resulting mosaics obtained for each data sequence are shown in Section 5.3.

5.1 Experiment Setup

Two image capture modalities were used to capture the data used in this work: standard (visible-spectrum) color video, and infrared video. The standard color video sequences for the undervehicle inspection efforts used in this work were taken using a Polaris Wp-300c Lipstick video camera mounted on a mobile platform. Infrared video was taken using a Raytheon PalmIR PRO thermal camera mounted on the same platform. The Lipstick camera has a focal length of 3.6mm, a 1/3" interline transfer CCD with 525-line interlace and 400-line horizontal resolution. The Raytheon thermal camera has a minimum 25mm focal length (36° horizontal and 27° vertical field-of-view) and produces images in several viewing modes with different color schemes. In this work, the viewing mode was set to a purple/blue/cyan/green/yellow/orange red/white color scheme, with each color representing different levels in the infrared spectrum.

Due to the size of the cameras and concerns with limited vehicle ground clearance, both cameras were not pointed directly at the scene during acquisition. Instead, the cameras were pointed at a mirror mounted on the same mobile platform that was set approximately to a 45° angle. This is why the perspective distortion correction step described in Section 3.2.2 is necessary. Using this configuration, the platform was moved along straight paths beneath a test vehicle. Several passes were made to cover the entire underside of the vehicle. The approximate distance of the cameras from the underside of the test vehicle was 6.5 inches. The mobile platform used to capture the data is shown in Figure 5.1.

All video sequences were processed using Adobe Premier 6.0. This video processing software is capable of extracting video from both cameras and writing to several file formats, including MPEG, AVI, and as bitmap sequences. In this work, the extracted video sequences were written out to 24-bit bitmap sequences. The video sequences used in this work are divided into individual data sets, so that

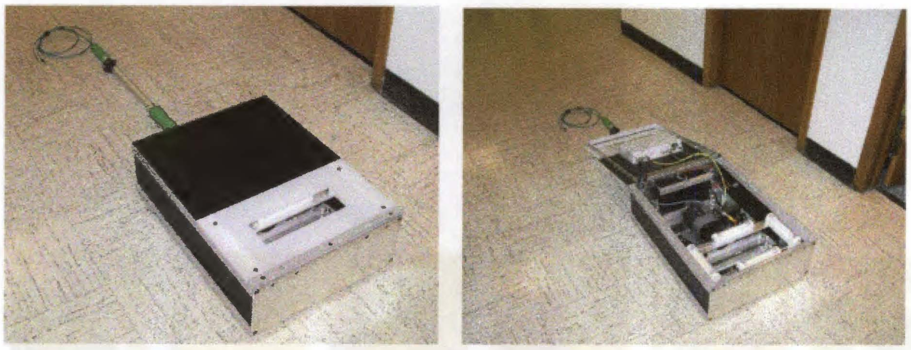


Figure 5.1 Mobile platform for undervehicle inspection. The Polaris Wp-300c Lipstick video camera and the Raytheon PalmIR PRO thermal camera are both mounted on the platform, and are pointed at a mirror tilted at a 45° angle.

the input and the results may be compared conveniently. Table 5.1 shows example frames from these data sets, the number of frames in each set, the size of each image, and the names assigned to each set. We also show sample *consecutive* frames from UVl and IRl in Figures 5.2 and 5.3, respectively, in order to give a better idea of the inter-frame motion within each sequence.

A note about the video extraction process: the frame rate for the video sequences was 30 frames-per-second. It was observed that the inter-frame motions tended to be on the order of 1-2 pixels for the standard color video sequences, and 5-10 pixels for the infrared color video sequences. Since phase correlation is capable of detecting fairly large translational motion, it was not desired that the frame rate be as high as the original frame rate. For our experiments, the sequences used here were undersampled so that the inter-frame primary motions were within the order of approximately 50-100 pixel-frames for the standard color video sequences, and 10-50 pixel-frames for the infrared sequence. This was done in Adobe Premier 6.0 during the video extraction process, where the original videos were undersampled to 2 fps while writing to bitmap sequences. The number of frames in the test sequences of Table 5.1 are the number of frames used *after* undersampling the video.

A note about the frame sizes: the original standard color video frames were at 480x640 resolution. However, the frame sizes shown in Table 5.1 are slightly smaller. This is because the frames have been cropped in order to remove some details at the edges that were not part of the underside of the vehicle. These details were the edges of the mirror used to reflect the underside of the vehicle. Since these areas are constant throughout a sequence, and provide no useful information for the registration process, they were cropped out of the video sequences. No alterations were made to the infrared video sequence, IR1.

Table 5.2 lists the various parameters used in the preprocessing and registration stages of the algorithm for each data set. Summaries of each parameter and appropriate comments are as follows:

Barrel Distortion Correction Parameters: a_x and a_y, the vertical and horizontal distortion correction factors, as explained in Section 3.2.1. These parameters were estimated manually, using images captured of a calibration grid using the Polaris Lipstick camera as a visual reference for the estimation. Note that no barrel

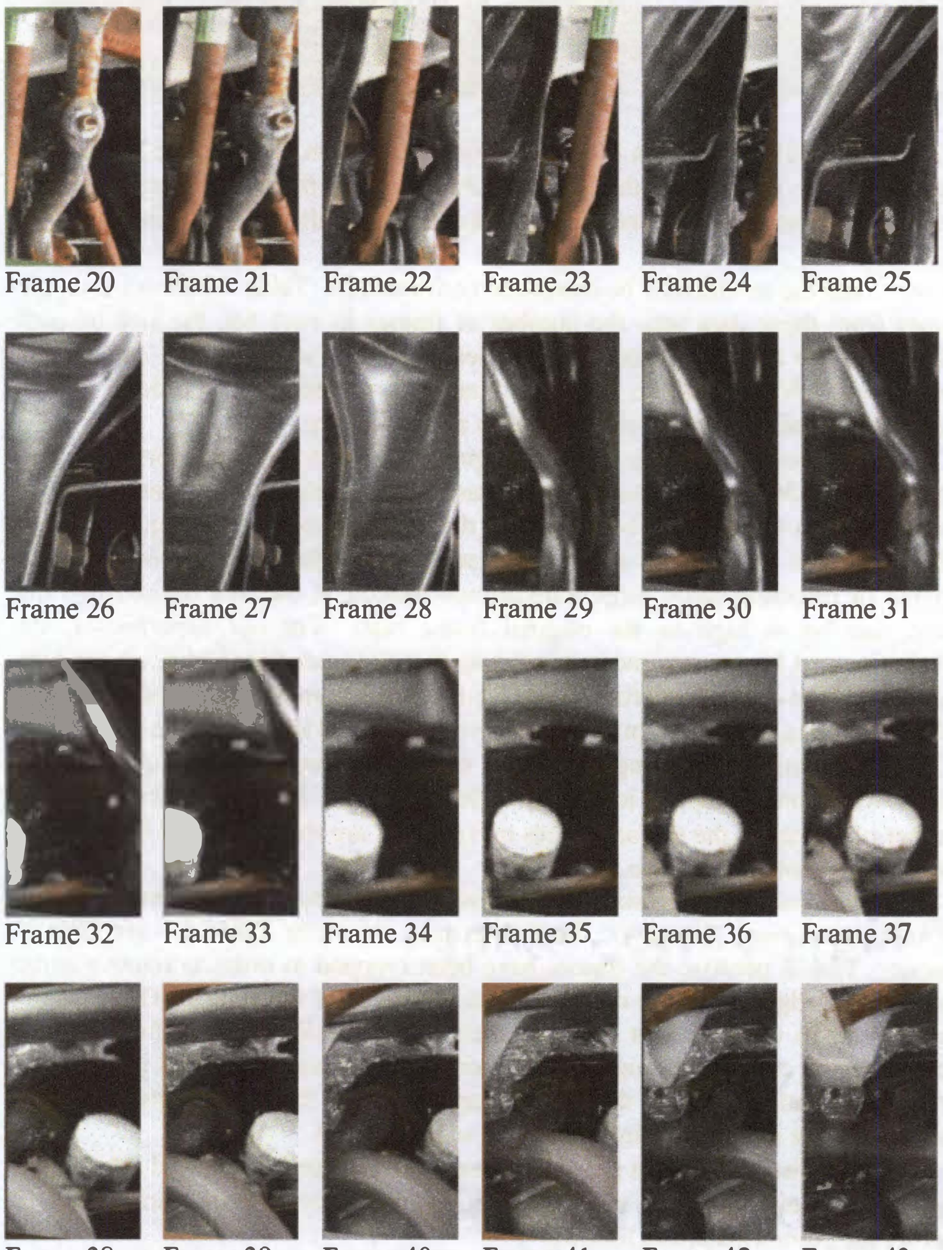


Figure 5.2 Frames 20-43 from sequence UV1

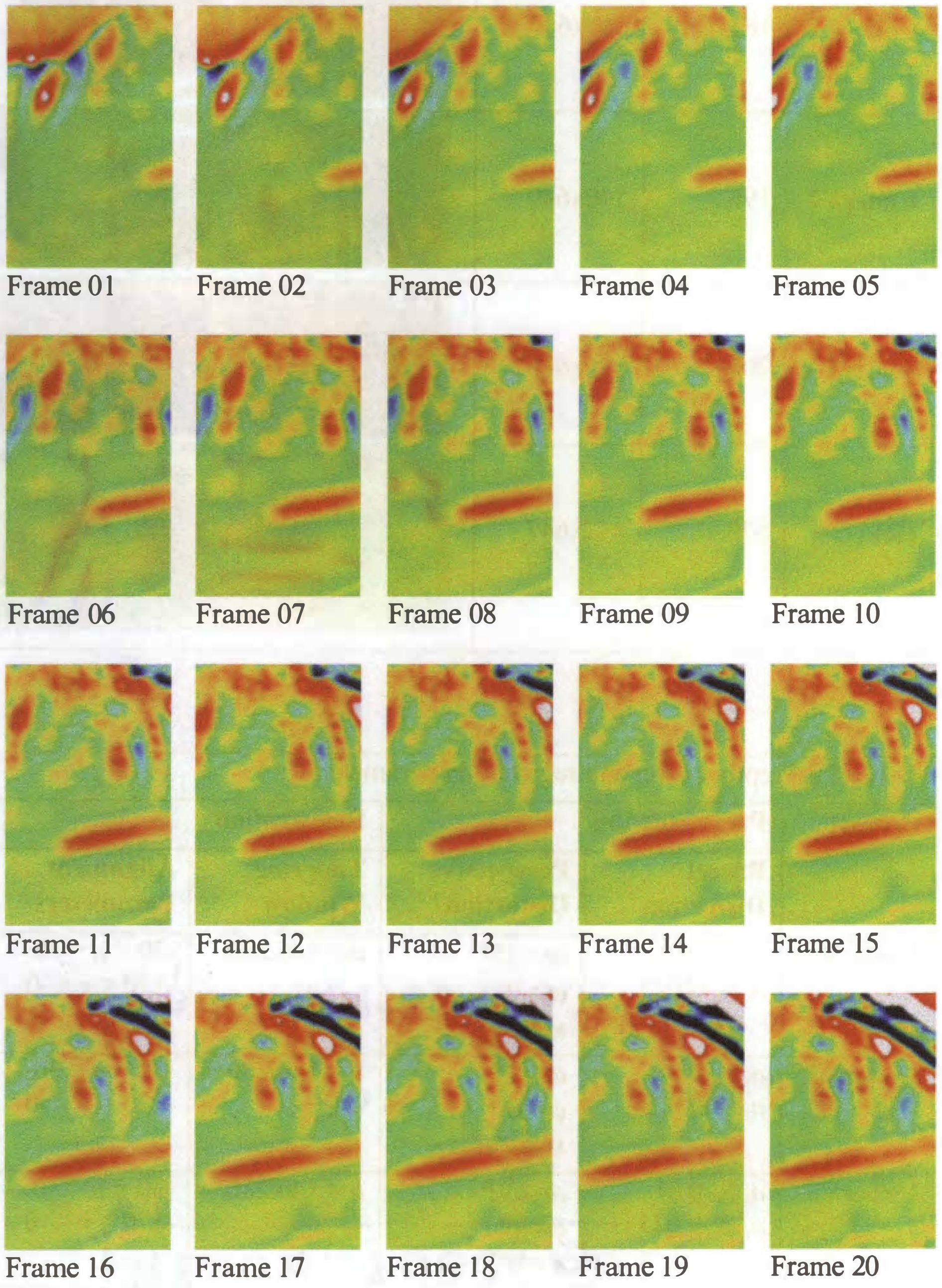


Figure 5.3 Frames 1-20 from sequence IR1

Table 5.1 Data sets from undervehicle inspection video

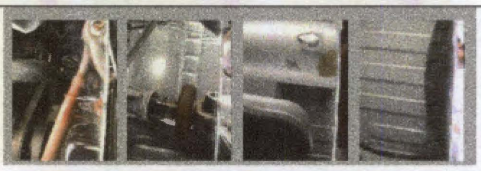
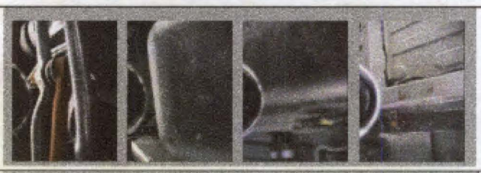
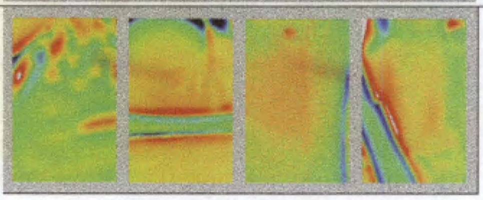
Data Set	Frames	Frame Size	Sample Frames
UnderV1	183	340x640	
UnderV2	196	340x640	
UnderV3	200	380x640	
IR1	679	412x647	

Table 5.2 Preprocessing and registration parameters

Dataset	**Preprocessing**		**Registration**	
	Barrel Distortion	**Perspective Distortion**	**Tapering Window**	**Alignment Parameters**
UnderV1	$a_x = -0.01$ $a_y = -0.025$	$\omega = 15^\circ$ $\varphi = 0^\circ$ $\kappa = 0^\circ$	$a = 146.286$	$-30 < u < 30$ $-170 < v < -0$
UnderV2	$a_x = -0.01$ $a_y = -0.025$	$\omega = 15^\circ$ $\varphi = 0^\circ$ $\kappa = 0^\circ$	$a = 146.286$	$-30 < u < 30$ $-170 < v < -0$
UnderV3	$a_x = -0.01$ $a_y = -0.025$	$\omega = 30^\circ$ $\varphi = 0^\circ$ $\kappa = 0^\circ$	$a = 146.286$	$-30 < u < 30$ $-170 < v < -0$
IR1	$a_x = 0$ $a_y = 0$	$\omega = 0^\circ$ $\varphi = 0^\circ$ $\kappa = 0^\circ$	$a = 146.286$	$-30 < u < 30$ $-100 < v < -0$

distortion correction was performed for the IR1 sequence, because it was difficult to gauge the effectiveness of any particular set of parameters for the infrared images.

Perspective Distortion Parameters: ω, φ, and κ, the pan, tilt, and rotation angles, as explained in Section 3.2.2. These parameters were set manually, using the video frames as a visual reference. Recall that the mirror used to reflect the underside of the test vehicle was tilted to approximately 45°. Therefore only the tilt angle φ was manipulated to correct for perspective distortion. Note that, again, no perspective distortion correction was performed for the IR1 sequence, for the same reasons as above.

Tapering Window Parameters: a, the Hamming window parameter, as explained in Section 3.3.1. To completely eliminate DFT leakage, we would use $a = 256/2 = 128$, since the size of the Fourier images is 256x256 pixels. However, this would also reduce much of the detail towards the center of the images. We use $a = 256/1.75 = 146.286$ as a compromise.

Search Region Parameters: u_{min}, u_{max}, v_{min} and v_{max}, the horizontal and vertical translation bounds for the phase correlation search region, as explained in Section 3.3.1.

These summaries only briefly describe the process of selecting the tapering window parameter a and search region parameters. Section 5.2 describes several experiments performed using various combinations of these parameters and provides an analysis of the results, providing a rationale for the parameters chosen here.

The mosaicking process was performed offline, after all image acquisition had been completed. All program code was written in the C++ programming language, using Borland C++ Builder 4. The code was run using an interactive GUI which was used to specify parameters and filenames.

5.2 Selection of Registration Parameters

Several experiments were performed in order to select parameters that produce satisfactory registration results from the standard color video sequences. We briefly review the parameters of our implementation of the phase correlation algorithm as discussed in Section 3.2.1. There are two sets of parameters that are specified in the algorithm: one parameter for the tapering window and four parameters for the ICPS peak value search. The tapering window parameter is a, which is the Hamming window parameter, and the search region parameters are u_{min}, u_{max}, v_{min}, and v_{max}, the bounds limiting the ICPS search region.

Although it would seem natural to experiment on synthetic images first in order to determine the characteristics of our implementation, we choose instead to obtain our parameters by experimenting with real video sequences. This is because phase correlation tends to work well for synthetic sequences that do not exhibit the common problems associated with real-word video sequences, i.e., inconsistent lighting, reflectance, motion blur, and so on. Results from such experiments would generally be meaningless. However, a key problem is that we lack ground truths by which to evaluate the results of using phase correlation on real video sequences. No external measurements of the camera's velocity were taken during the video capture process. Therefore, we will qualitatively evaluate the results based on observation of the motions in our video sequences, and select a set of parameters that give reasonable results for the sequences we are dealing with, and describe the rationale behind our selections.

We now describe the process of selecting our parameters. The initial selection of the four search region parameters was straightforward. The primary motion, v, was restricted to motions of 170 pixel-frames (about half the horizontal resolution of each video frame) or less in the negative direction, while the secondary motion was restricted to $\pm$ 30 pixel frames. From the video sequence, all exhibited inter-frame motions appeared to be well within these bounds. Therefore the parameters were initially set as

$$u_{\min} = -30$$
$$u_{\max} = 30$$
$$v_{\min} = -170$$
$$v_{\max} = 0$$

for all test video sequences.

Determining a suitable value for a, the Hamming window parameter, required more experimentation. We know that a should never be assigned a value more than 2, since this is the value at which the function is minimum at the edges of the images, and begins to increase again in a sinusoidal fashion if a is higher than 2. Therefore we experimented for several values of a: 256/0.5=512, 256/1=256, 256/1.5=170.667, 256/1.75=146.286, and 256/2=128. The result of using tapering windows with these values on a sample image frame is shown in Figure 5.4. The resulting u and v displacement values recovered from segments of the UnderV1 sequence, using different values of a, are shown in Figure 5.5.

Before we interpret the results, we shall describe, in general, the motions observed in the UnderV1 sequence. For most of the sequence, the observable motion is between 30 to 100 pixel-frames in the negative direction for the primary motion, and between -10 to 20 pixel-frames for the secondary motion. The exceptions are from frames 128 to 130, where the camera stops completely. Frames 29-40 are blurry due to some focusing problems that occurred during acquisition.

What do these results of Figure 5.5 say about how the parameter a affects registration? Firstly, it should be noted that the results for the u-motion (the secondary motion) do not immediately tell us anything about the correctness of the registration; a general observation of the plots tell us any of them may well represent a 'correct' set of u-motions, since these motion tends to be somewhat erratic. However, we do know from observing the video sequence that the camera is almost always constantly exhibiting a large amount of v-motion (the primary motion). This observation automatically disqualifies a = 512, a = 256, or a = 170.667, as these values result in too many instances of zero-motion. The large number of misregistrations are caused by the high axis components created by DFT leakage, which in turn creates high peaks at the origin in the Inverse Cross Power Spectrum (ICPS) of two images. Therefore, we narrow down the a values to 128

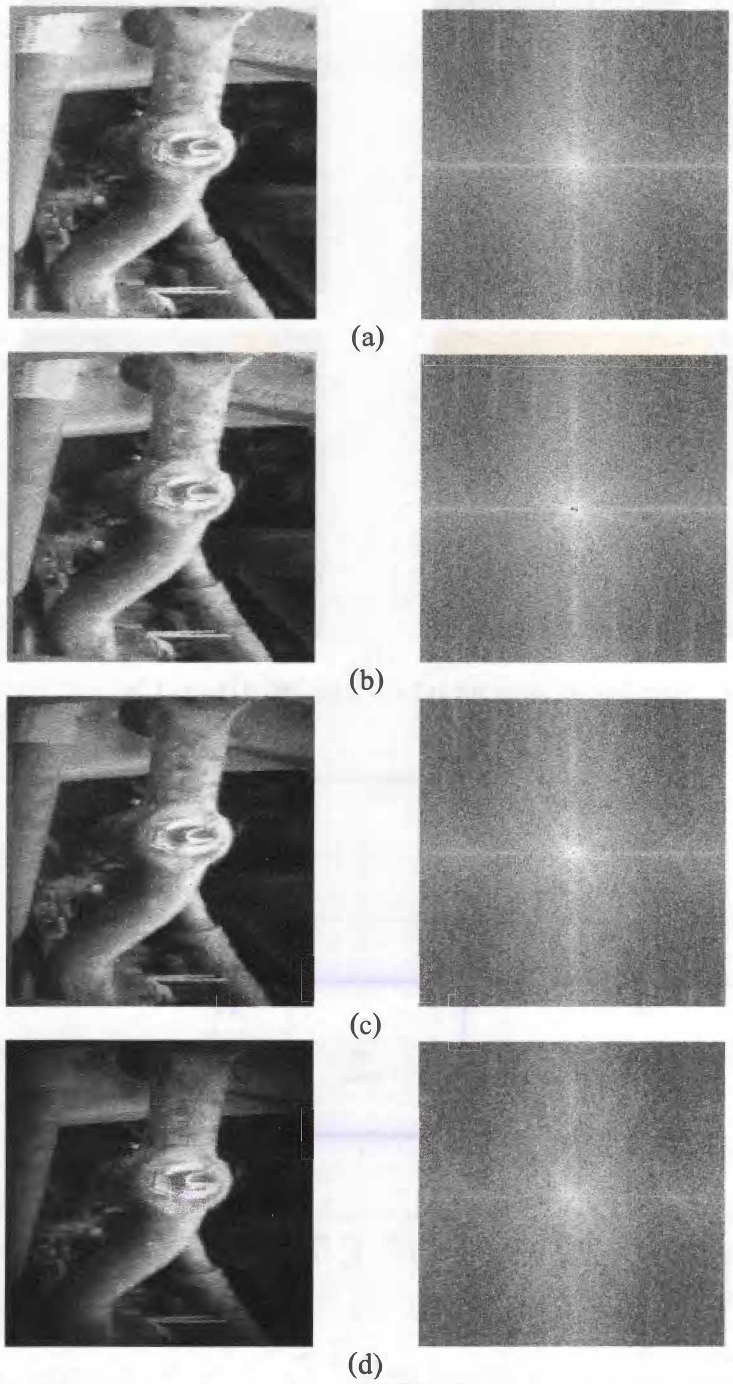
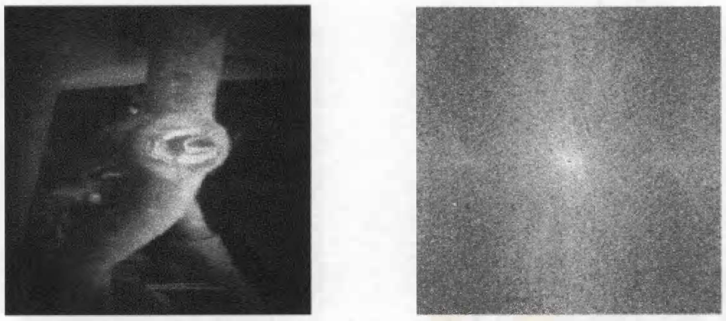


Figure 5.4 Effect of tapering window on FFTs. Input images are on the left, and the resulting FFTs of those images are on the right. Results are shown for (a) no tapering window (b) $a = 512$ (c) $a = 256$ (d) $a = 170.667$.


(e)

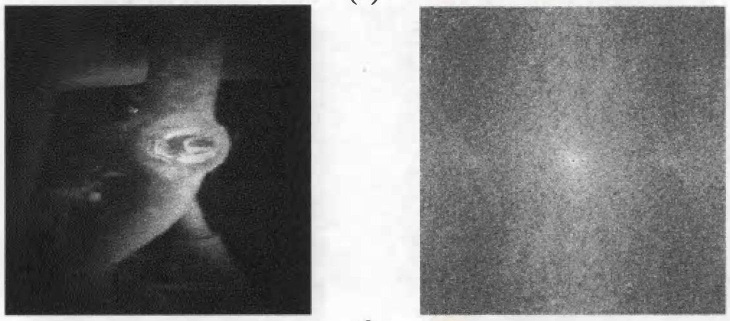
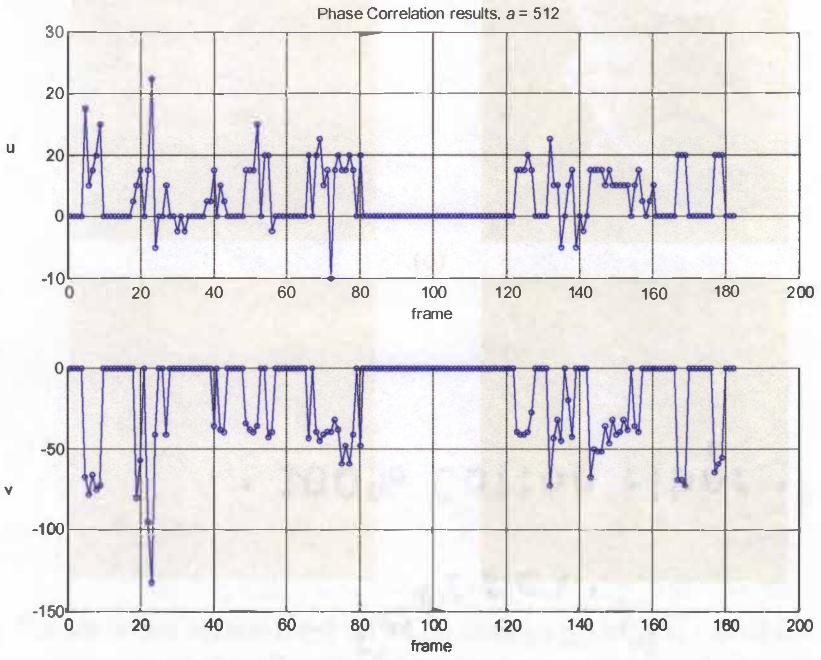
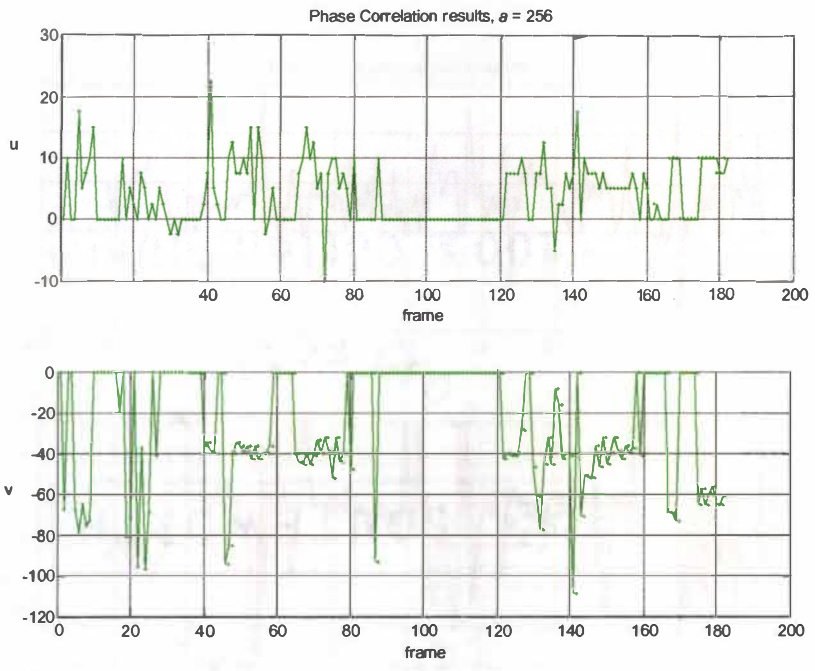

(f)

Figure 5.4 Continued. Results for (e) *a* = 146.286 (f) *a* = 128

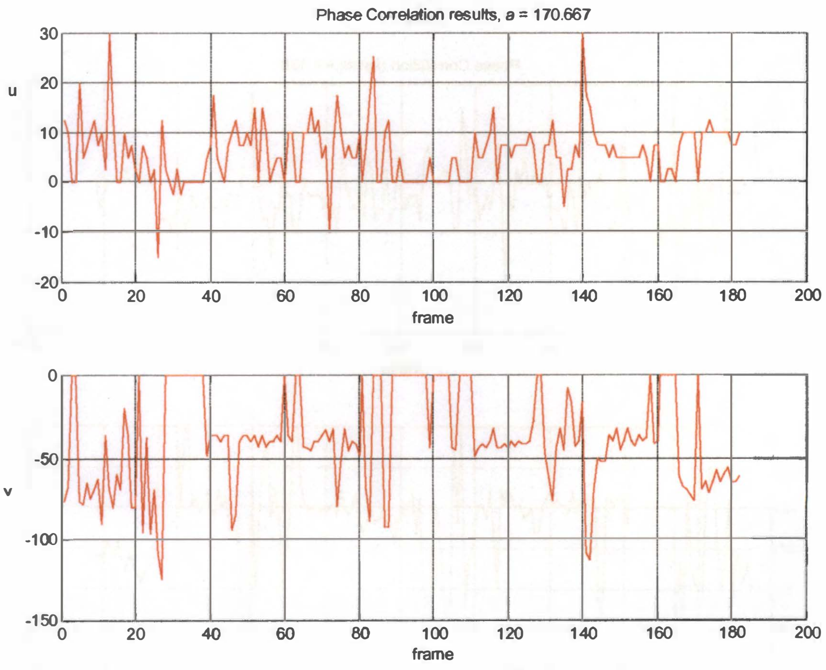


(a)

Figure 5.5 The vertical and horizontal translation vectors. (*u*,*v*) obtained for different values of the Hamming window parameter, *a*, plotted for all 182 frames in the UnderV1 video sequence. Results are shown for (a) *a* = 512.

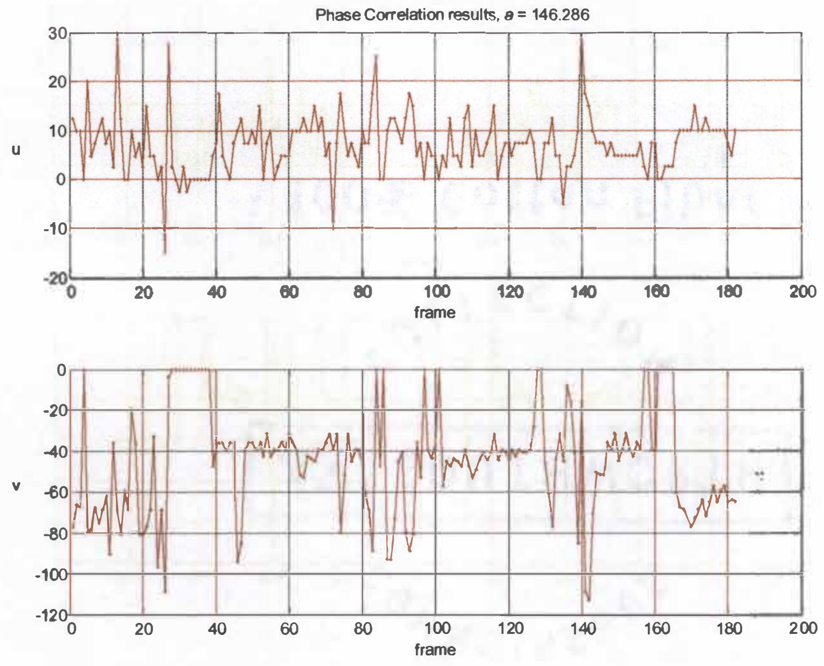


(b)

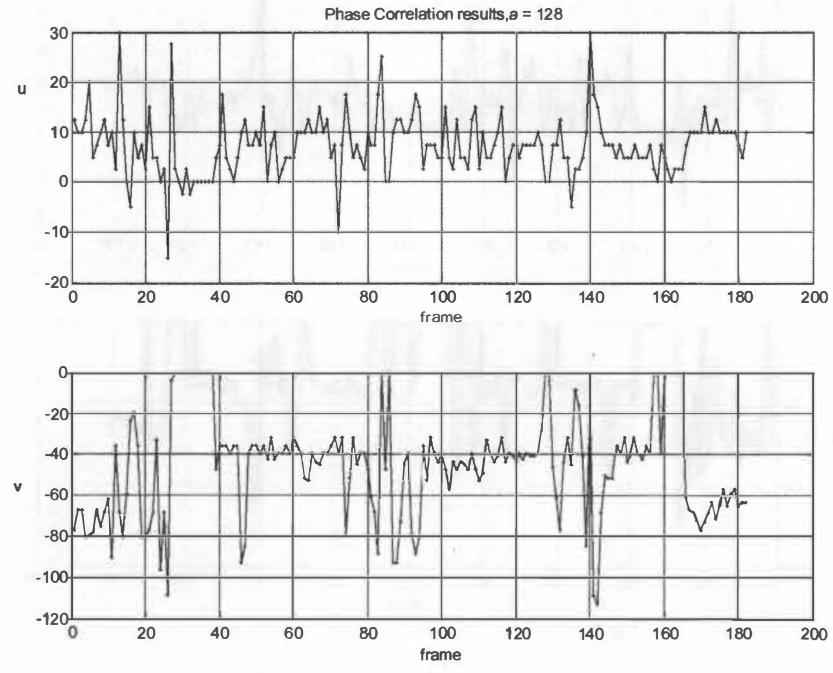


(c)

Figure 5.5 Continued. Results for (b) $a = 256$ (c) $a = 170.667$.



(d)



(e)

Figure 5.5 Continued. Results are shown for (d) $a = 146.286$ (e) $a = 128$.

and 146.286. These observations also confirm our suspicions that DFT leakage does affect registration: using a value of a twice as large as the image dimensions (a = 512 for 256x256 images) is almost equivalent to not using a tapering window at all (compare Figure 5.4(a) to Figure 5.4(b)), which results in a large number of misregistrations.

However, in the plots of v for a = 146.286 and a = 128, we still observe many zero-motion registrations (frames 25-39 and frames 158-165 in particular). These are mostly caused by instances in the camera sequence when there was lack of detail in the scene (most of the visible details are at the edges of the image) or excessive blurring caused by the camera refocusing, again lending more strength to the peak at the origin of the ICPS, rather than the peak at the correct coordinates. The ICPS functions of these image pairs, examples of which are shown in Figure 5.6 and Figure 5.7, tend to have a local peak at the origin, which may become higher than the peak at the coordinates corresponding to the correct motion vector. This is not always the case, however, as there are times when there really is zero inter-frame motion (frames 128-130), where the camera actually stops for a short time. To decrease the chances of zero-motion misregistration, whenever the inter-frame motion between two images is found to be (0,0), we calculate the sum of intensity errors between the images using equation 2.1. If this sum is below a threshold, the registration at (0,0) is kept. If it is not, then we modify the search region boundaries, changing v_{min} from 0 pixel-frames to -10 pixel-frames, and perform phase correlation again, and use the new vector obtained as the correct registration. The result of using these modifications is shown in Figure 5.8.

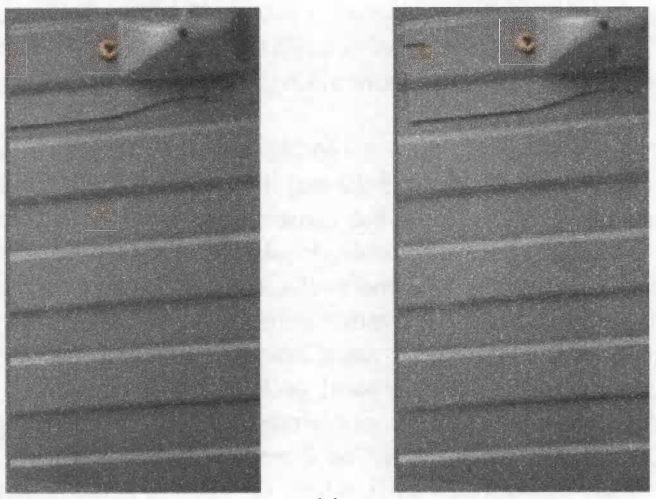
We observe that for a = 146.286 and a = 128, there is little noticeable difference in the vectors obtained. To obtain the results shown in Section 5.3.2, we used a = 146.286, observing from the FFT image of Figure 5.4(e) that this value eliminates most of the effects of DFT leakage while preserving much of the detail of the original images.

The approach used to overcome this problem of misregistration (due to the peak at the origin of the ICPS) is somewhat ad hoc. A more general formulation of the problem was not pursued, however, since misregistrations due to the local peak at the origin of the ICPS appear to be the most common cause of misregistration (another observed cause of misregistration are small rotations of the camera, which will be discussed in Section 5.3.2.). Using these parameters, we expect that this implementation will be fairly robust towards blurring and lack of detail in the images.

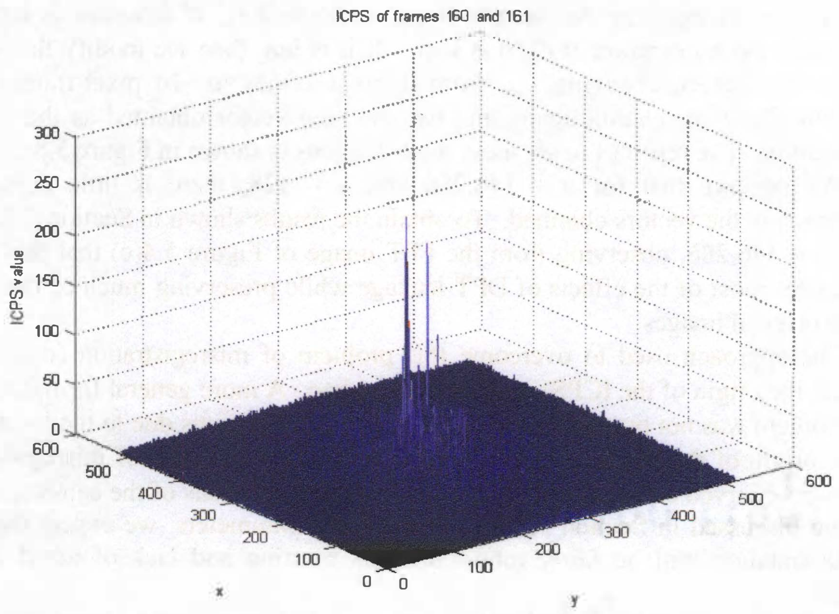
Note that for the infrared video sequence IR1, we merely used the same tapering window and search region parameters used for the standard color video sequences.

5.3 Results of Algorithms

In this section, we will discuss some of the results obtained using our single-mosaic representation algorithms with the UnderV1, UnderV2, and UnderV3 video sequences. First, the results of the distortion correction algorithms will be discussed in Section 5.3.1. Then, the results of the blending scheme and the final

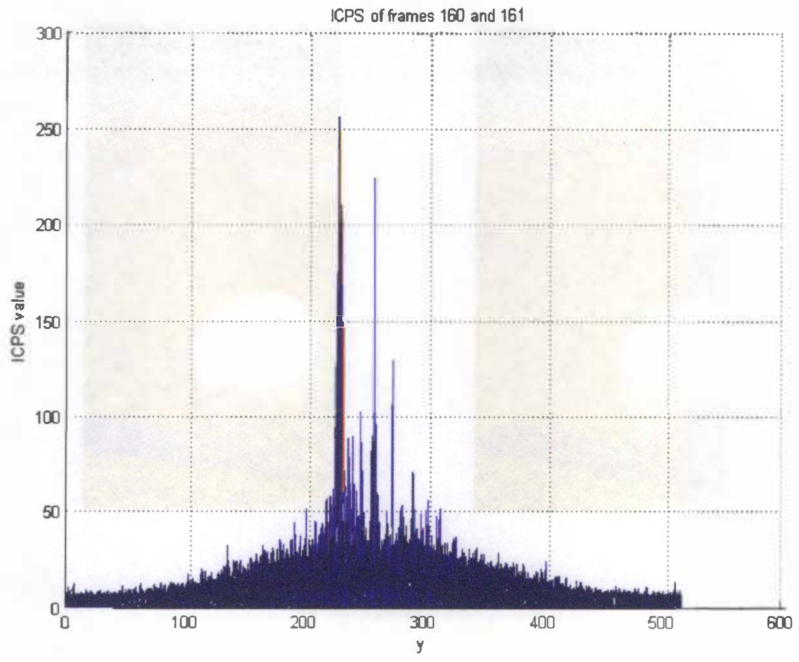


(a)



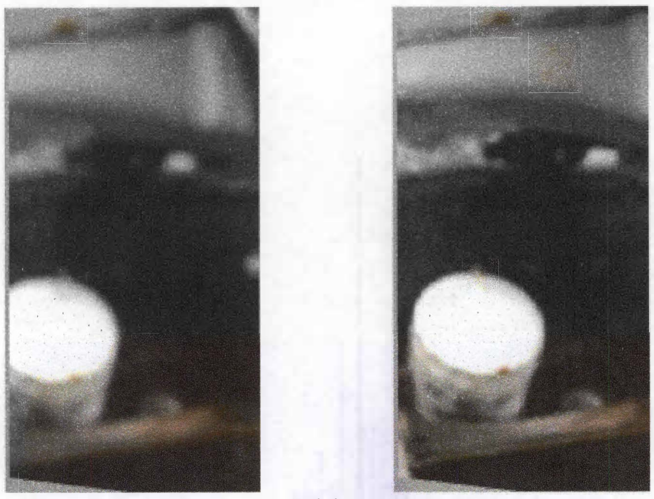
(b)

Figure 5.6 False peaks at the ICPS origin for frames 160-161. (a) Frame 160-161 from sequence UnderV1. (b) The resulting 3D plot of the ICPS function.

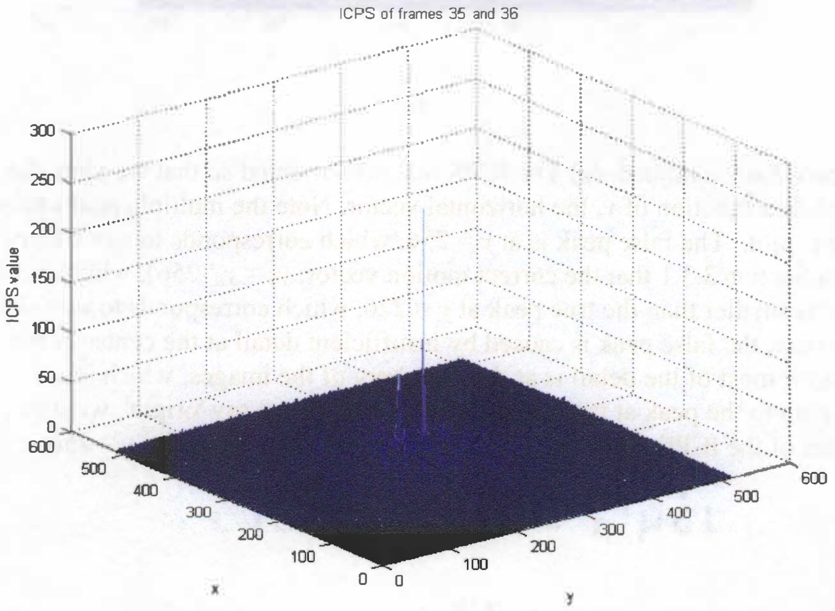


(c)

Figure 5.6 Continued. (c) The ICPS function oriented so that we view the ICPS as a function of *y*, the horizontal vector. Note the multiple peaks present in this plot. The false peak is at $y = 256$, which corresponds to $y_0 = 0$ (recall from Section 3.3.1 that the correct motion vector, $y_0 = y_0'-256$), which in this case is smaller than the true peak at $y = 226$, which corresponds to $y_0 = -30$. In this case, the false peak is caused by insufficient detail at the center of the images; most of the detail is at the periphery of the images, which lends strength to the peak at the origin (Note that when we say 'origin', we mean the center of the ICPS function, which corresponds to $x = 256$ and $y = 256$).

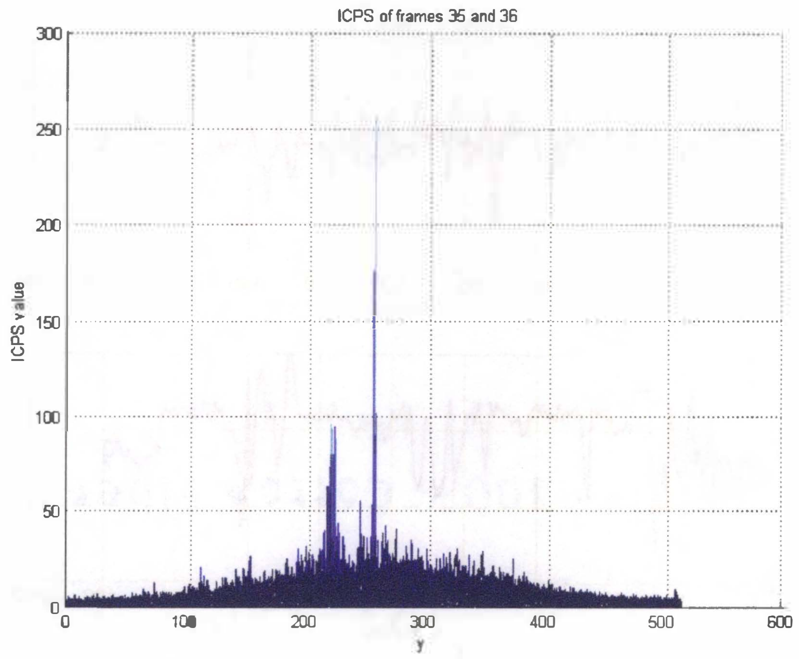


(a)



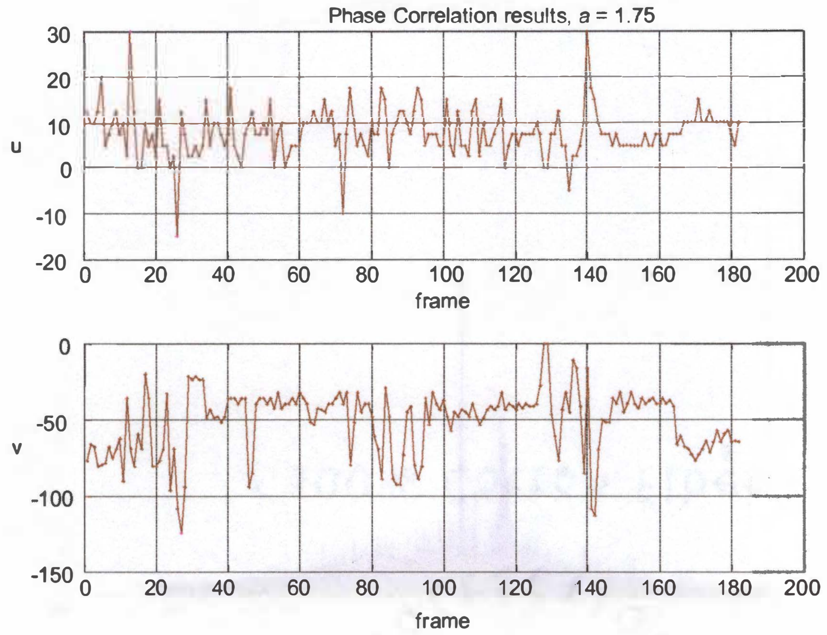
(b)

Figure 5.7 False peaks at the ICPS origin for frames 35-36. (a) Frame 35-36 from sequence UnderV1. (b) The resulting 3D plot of the ICPS function.

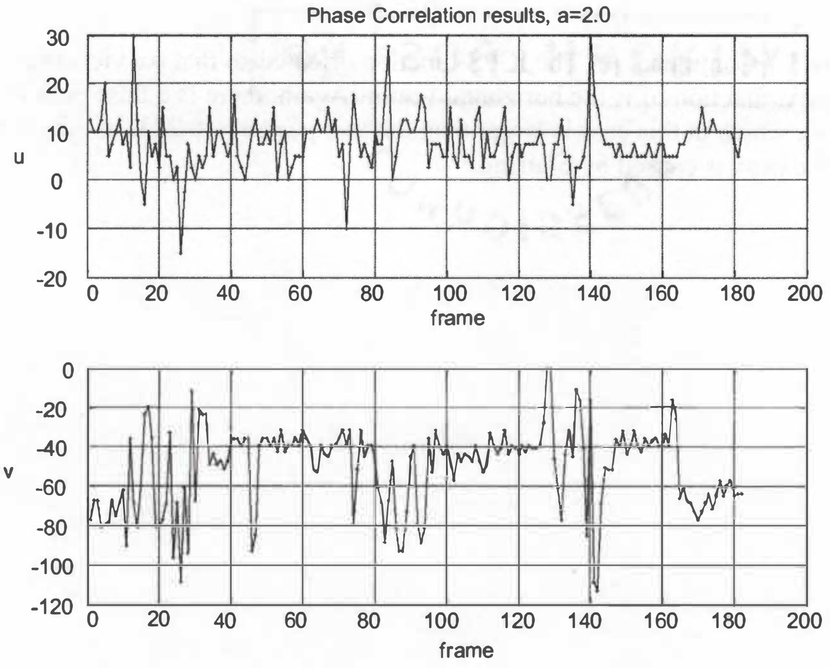


(c)

Figure 5.7 Continued. (c) The ICPS function oriented so that we view the ICPS as a function of *y*, the horizontal vector. Again, there is a false peak is at *y* = 256, which in this case is larger than the true peak at *y* = 223. In this case, the false peak is caused by blurring.



(a)



(b)

Figure 5.8 The vertical and horizontal translation vectors, (u,v), obtained for different values of the Hamming window parameter, a, with modifications to compensate for zero motion. Results are shown for (a) $a = 1.75$ (b) $a = 2$.

mosaics will be shown in Section 5.3.2. We reserve a detailed discussion of the results of phase correlation for Chapter 7.

5.3.1 Results of Distortion Correction

In Figures 5.9 and 5.10, we show some of the results of our lens and perspective distortion correction algorithms using the parameters specified in Table 5.2. The results for the UnderV1 and UnderV2 sequences are similar, since they both use the same correction parameters, so we only show results for frames from UnderV1. Again, the purpose of perspective distortion correction is to make it appear as though the plane of the scene was orthogonal to the camera's principal axis. Note that in the original images, lines that ought to be parallel appear instead to intersect, and this is rectified once distortion correction has been implemented.

It should be acknowledged that one set of distortion correction parameters will not correct the perspective of every element in the video sequence, since the sequences contain elements of slightly varying distances from the camera (again, the assumption that the scene exists entirely on a single plane is only an approximation of the nature of the scene). We chose a set of parameters that corrected for perspective distortion where it was most visible, and these parameters appeared to work well for all other parts of the sequences as well. Since distortion correction was not the focus of our work, we chose not to pursue this matter any further.

5.3.2 Mosaics of Undervehicle Video

After the distortion correction steps have been completed, the inter-frame motion vectors for each video sequence is computed using the parameters listed in Table 5.2. Using these vectors, strips are sampled and aligned accordingly to form the mosaics, as described in Section 3.3.2. Figure 5.11(a) shows a section of a mosaic formed in this manner. Note the seams that are visible at the boundaries that separate strips from different frames. Using the blending scheme described in Section 3.3.3, the result is shown in Figure 5.11(b). The result of using this blending scheme for UnderV1, UnderV2, UnderV3, and IR1 are shown in Figure 5.12, 5.13, 5.14 and 5.16, respectively. In Figure 5.15, the mosaics created from the UnderV1 and UnderV2 sequences are compared side-by-side with some photos taken using a still camera of the underside of the vehicle (which was raised in order to obtain the field of view seen in these photos).

From these results, it is observed that our algorithm works well for most sections of the undervehicle video. There are still several obvious visual discontinuities in the mosaics, however, and we will now discuss these discontinuities and their causes. Figures 5.17 to 5.20 show several cases of discontinuities observable from the mosaics. Before we look at these examples, though, we will discuss the most obvious discontinuity in Figure 5.16, the IR1 mosaic. There is a section of the mosaic where the secondary motion recovered is obviously far more erratic than anywhere else in the mosaic. This part of the video sequence had little or no detail for which to detect motion, and hence the motion vectors recovered for this part of the video sequence were probably just random peaks within the ICPS search region specified in Table 5.2.

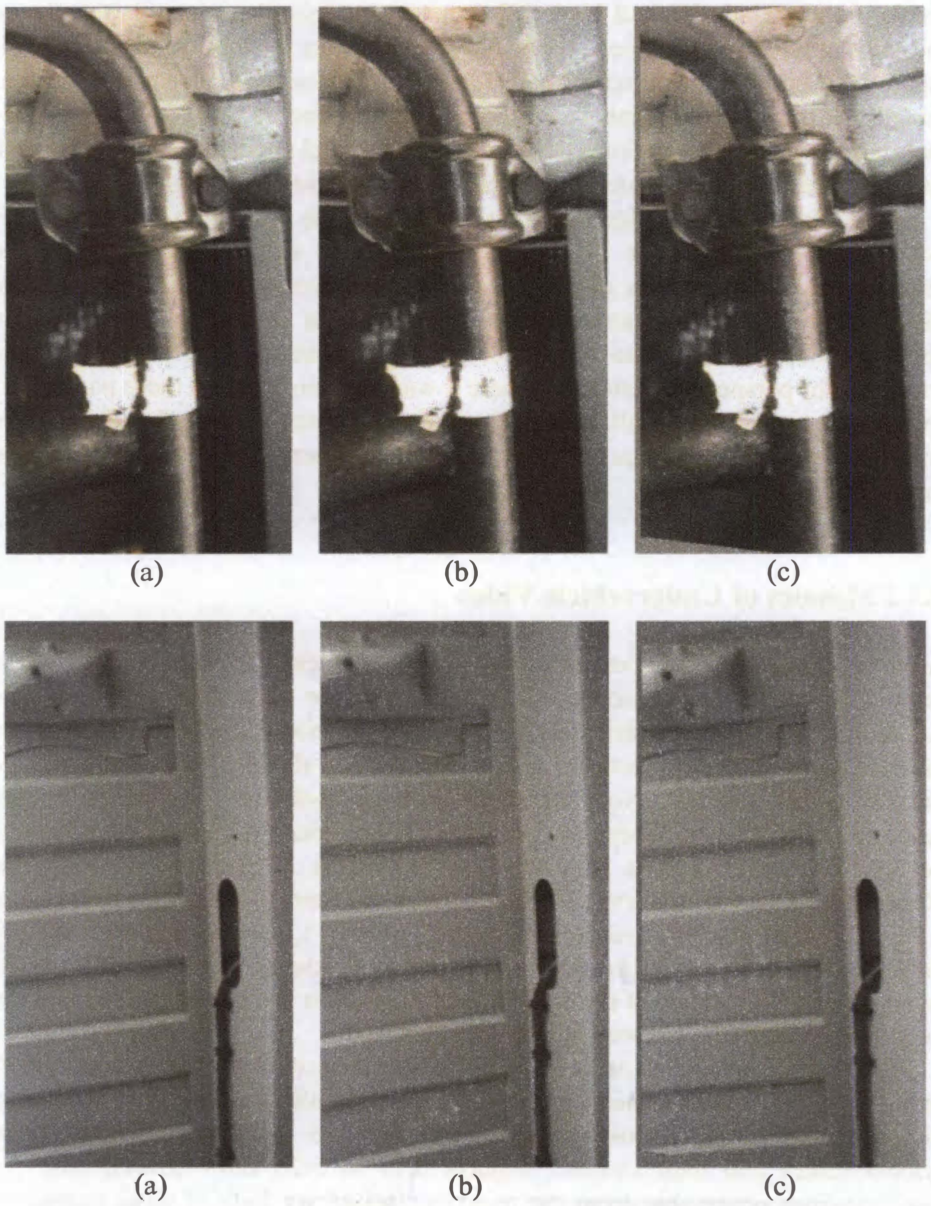


Figure 5.9 Perspective distortion correction for two frames from sequence UnderV1 using a_x = -0.01, a_y = -0.025, ω = 15°, φ = 0°, κ = 0° : (a) original image (b) with barrel-distortion correction and (c) with perspective distortion correction.

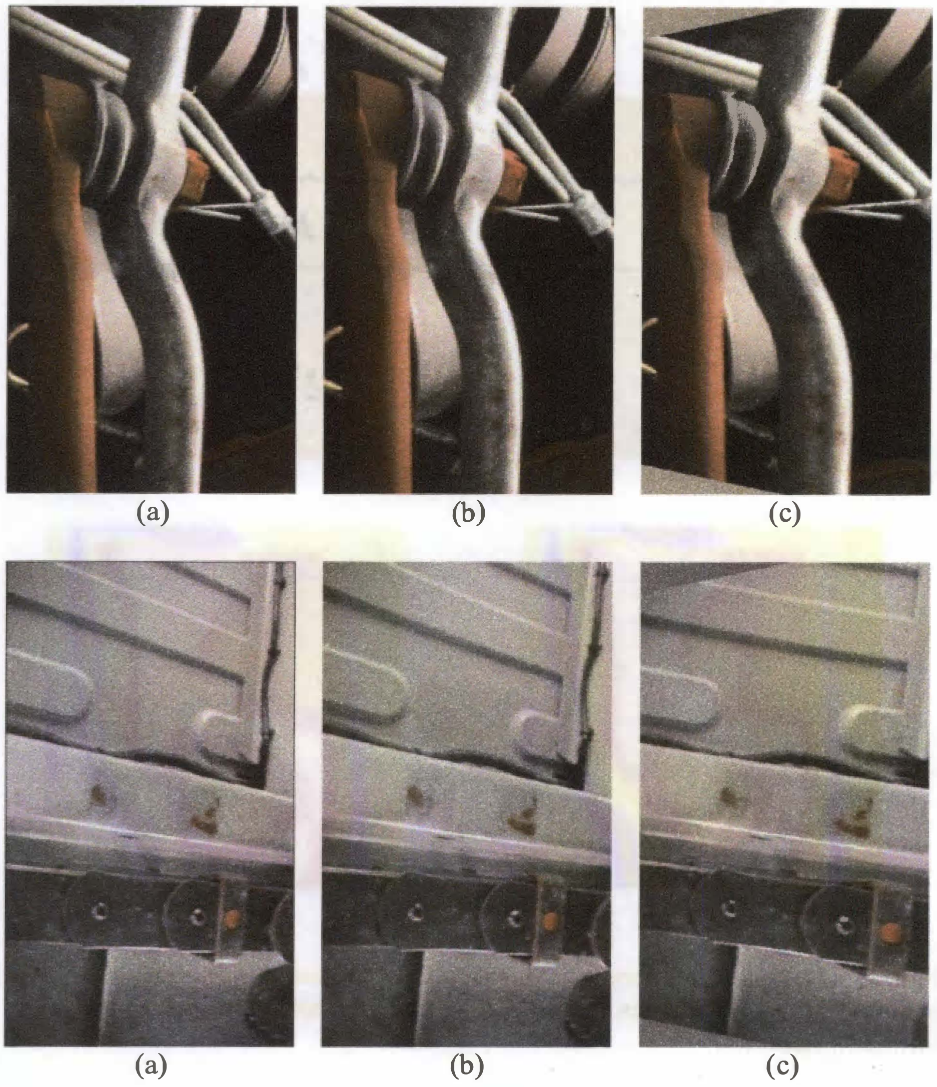
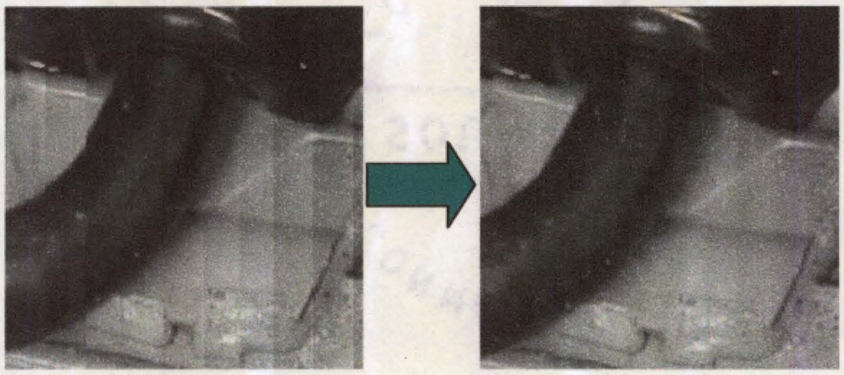
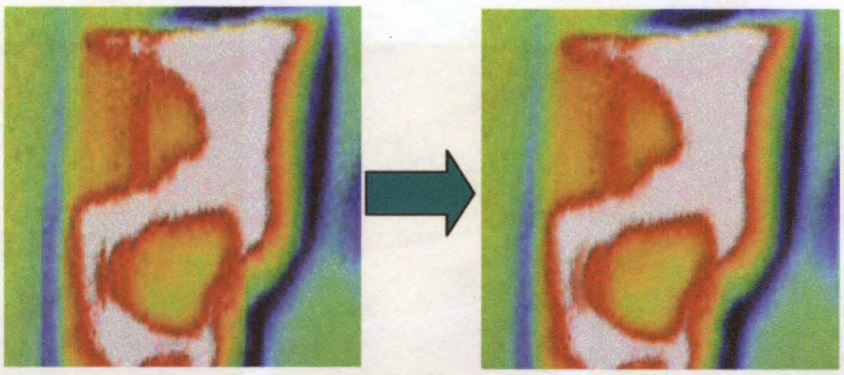


Figure 5.10 Barrel distortion correction. Two frames from sequence UnderV3 processed using $a_x = -0.01$, $a_y = -0.025$, $\omega = 30°$, $\varphi = 0°$, $\kappa = 0°$: (a) original image (b) with barrel-distortion correction and (c) with perspective distortion correction.



(a)



(b)

Figure 5.11 Results of blending. Mosaic without blending (left) and with blending (right). Examples are shown for (a)visible spectrum video and (b) infrared video.

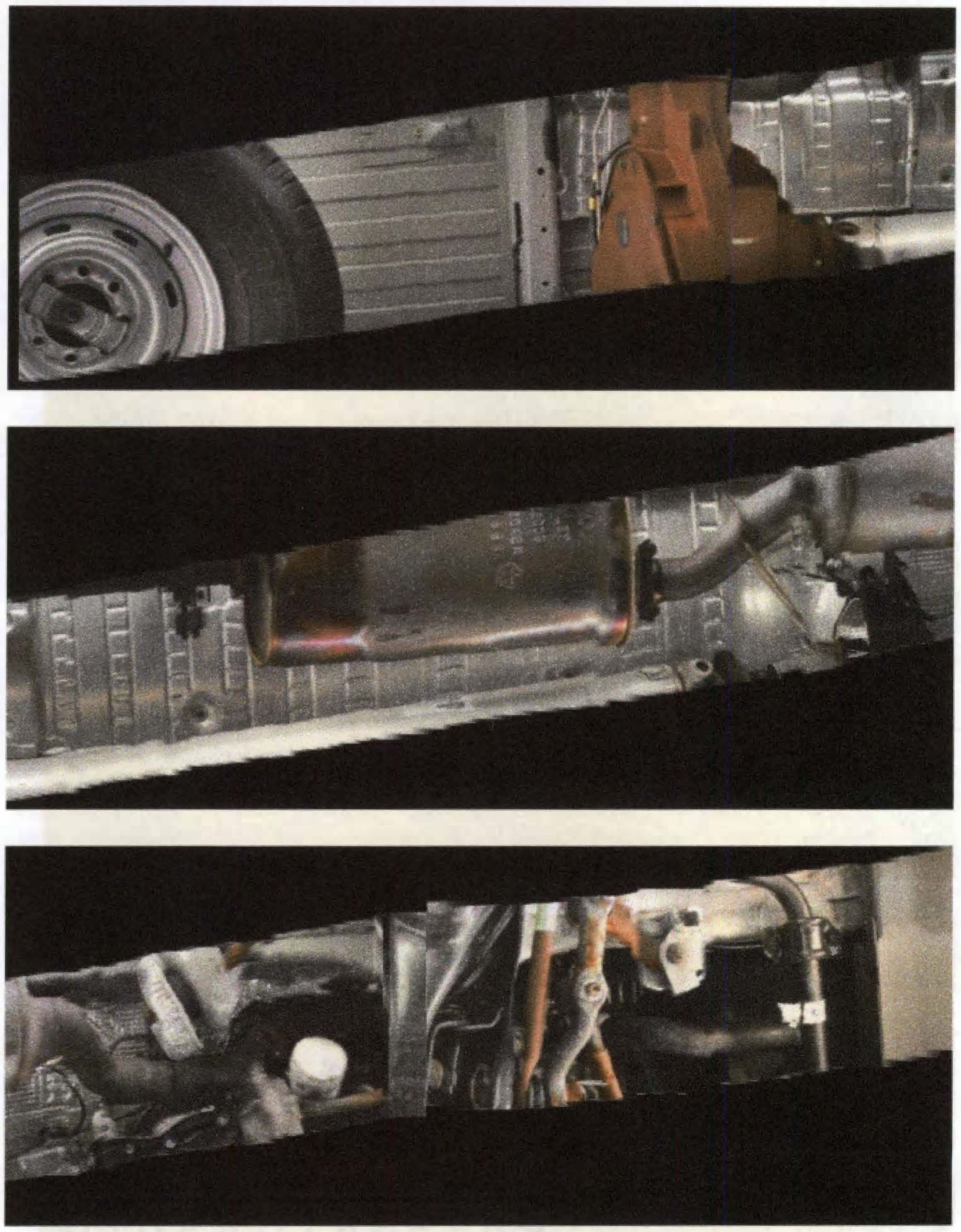


Figure 5.12 Mosaic of sequence UnderV1. The mosaic has been split into three sections so that the detail in the mosaic can be seen here. The original mosaic proceeded from left to right with the top image being the leftmost section and the bottom mosaic being the rightmost section.

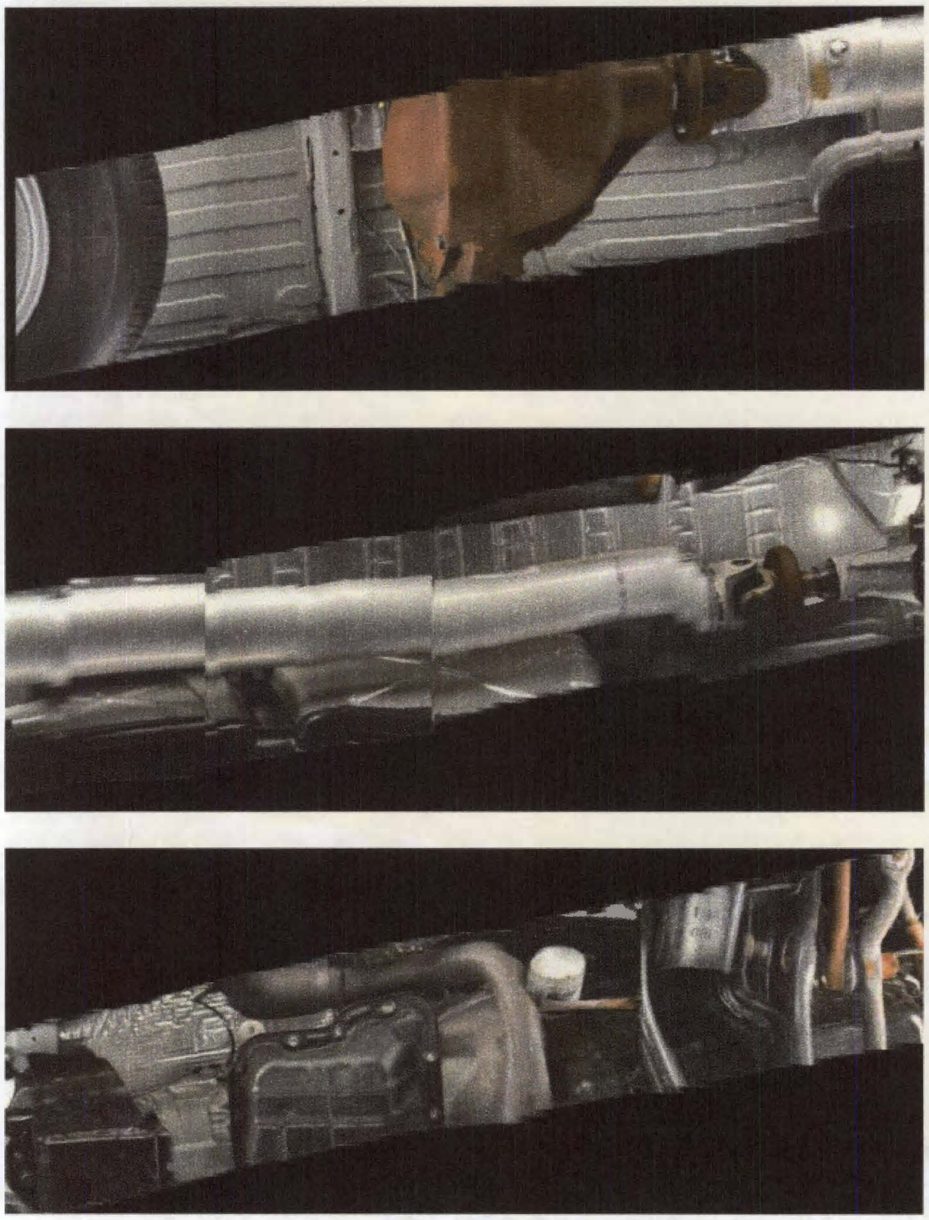


Figure 5.13 Mosaic of sequence UnderV2. The mosaic has been split into three sections so that the detail in the mosaic can be seen here. The top image is the leftmost section of the mosaic and the bottom mosaic is the rightmost section.

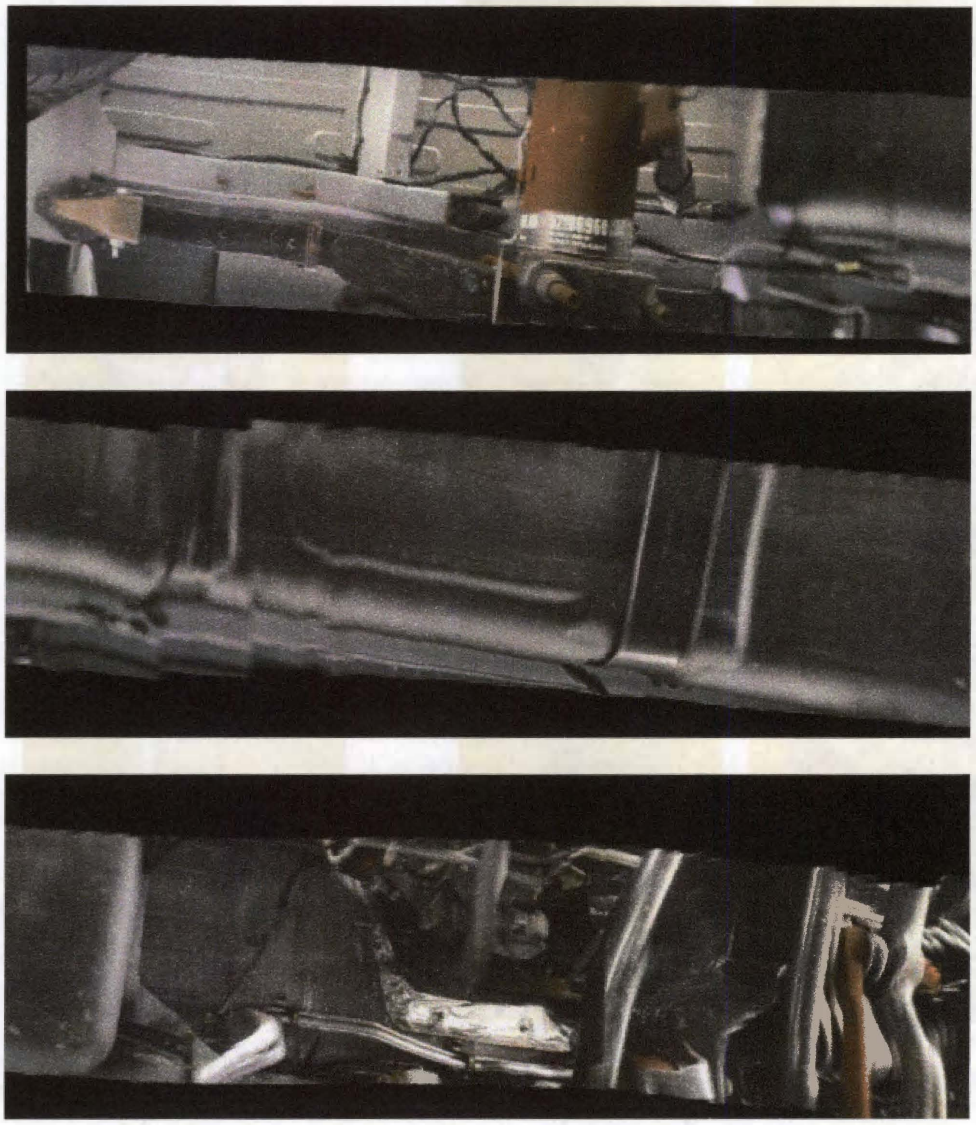


Figure 5.14 Mosaic of sequence UnderV3. The mosaic has been split into three sections so that the detail in the mosaic can be seen here. The top image is the leftmost section of the mosaic and the bottom mosaic is the rightmost section.

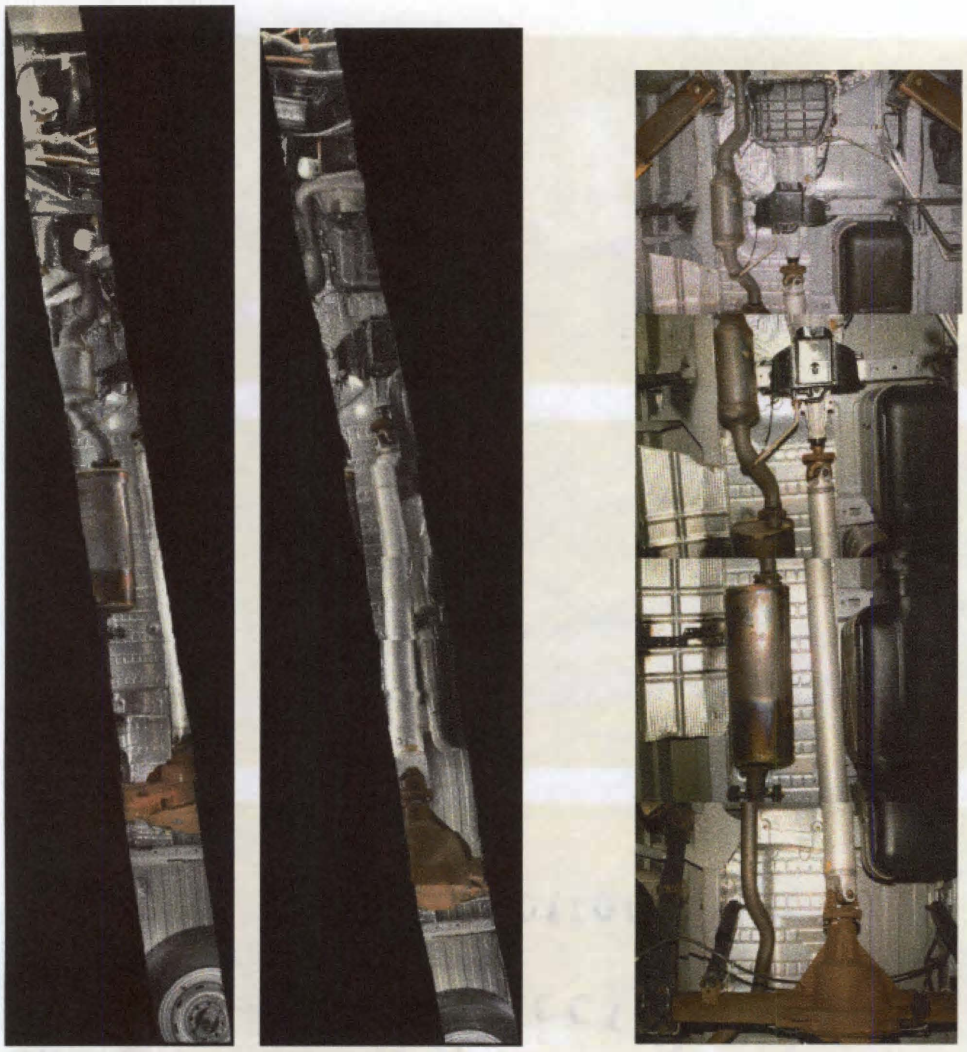


Figure 5.15 Comparison of mosaics with still shots of the underside of the vehicle. (a) Mosaic created from the UnderV1 sequence, (b) Mosaic created from the UnderV2 sequence, and (c) 4 still photos of the underside of the vehicle.

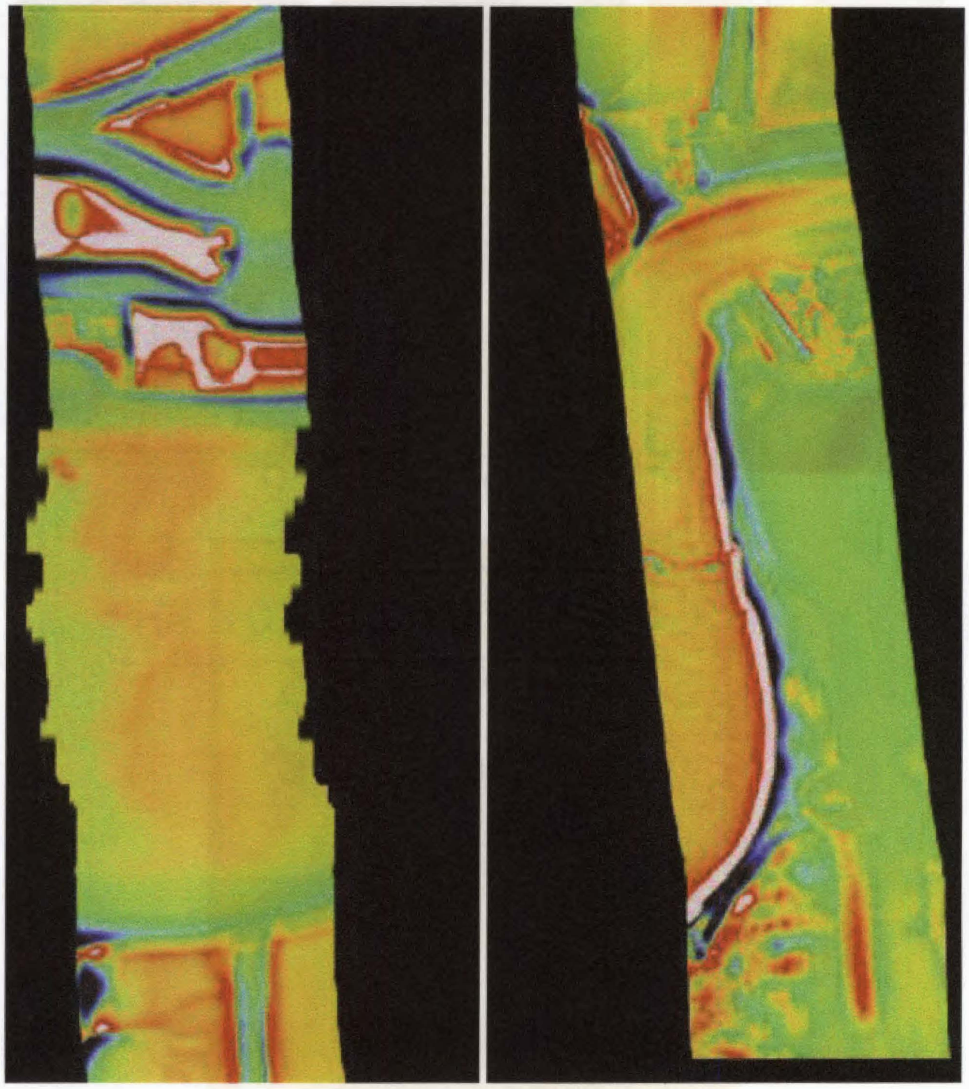
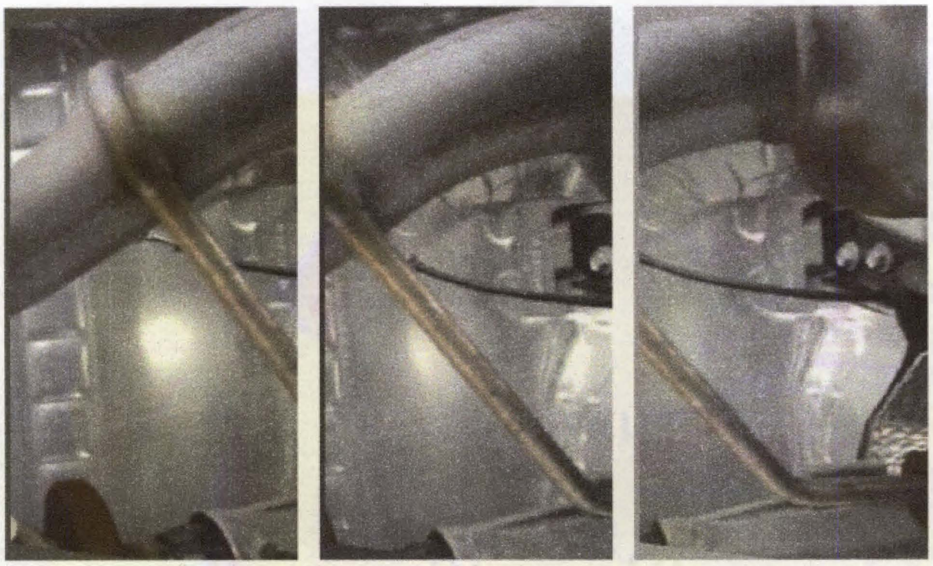


Figure 5.16 Mosaic of infrared video. The mosaic has been split into two sections so that the detail in the mosaic can be seen here. The image on the left is the top section of the mosaic and the image on the right is the bottom section of the mosaic. Note that a large part of the top section of the mosaic exhibits erratic secondary motion, where there was little or no detail with which to detect motion.


(a)

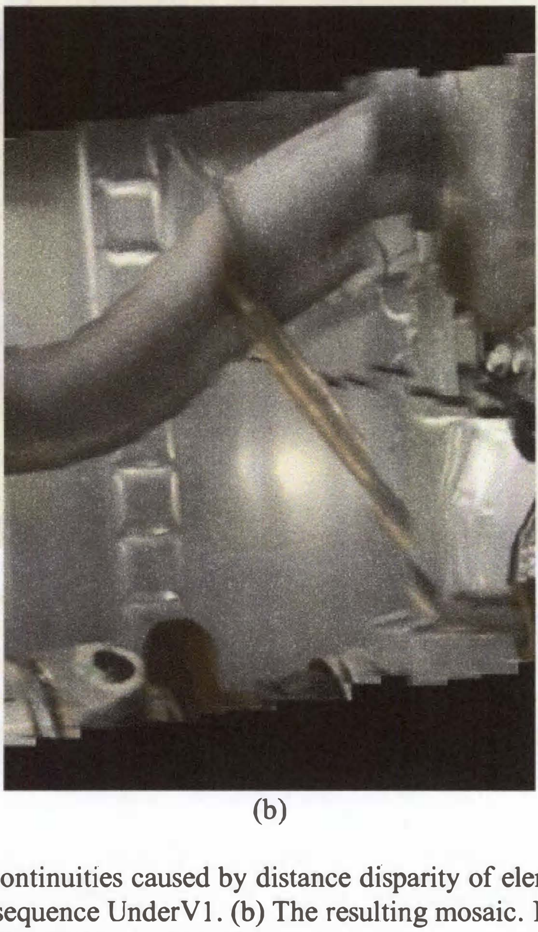
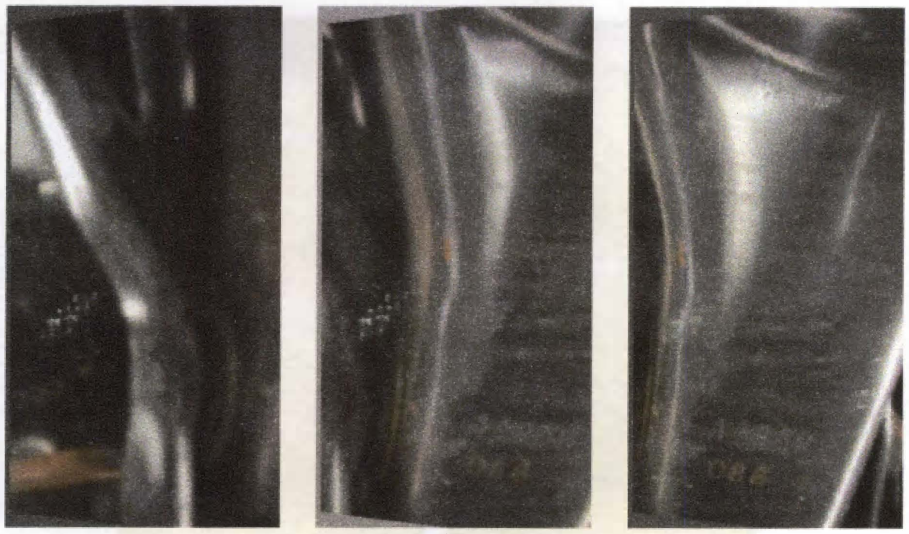

(b)

Figure 5.17 Discontinuities caused by distance disparity of elements. (a) Three frames from the sequence UnderV1. (b) The resulting mosaic. Note the 'shearing' of the diagonal bar and the large pipe.


(a)

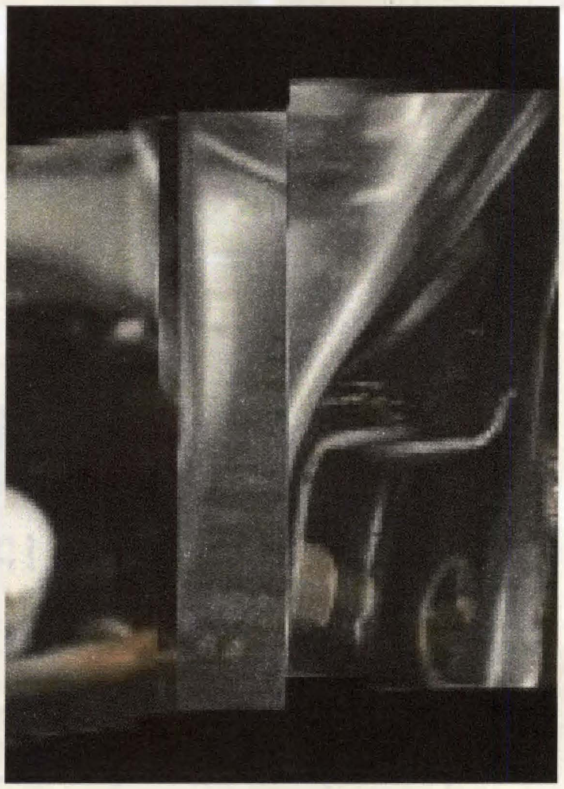
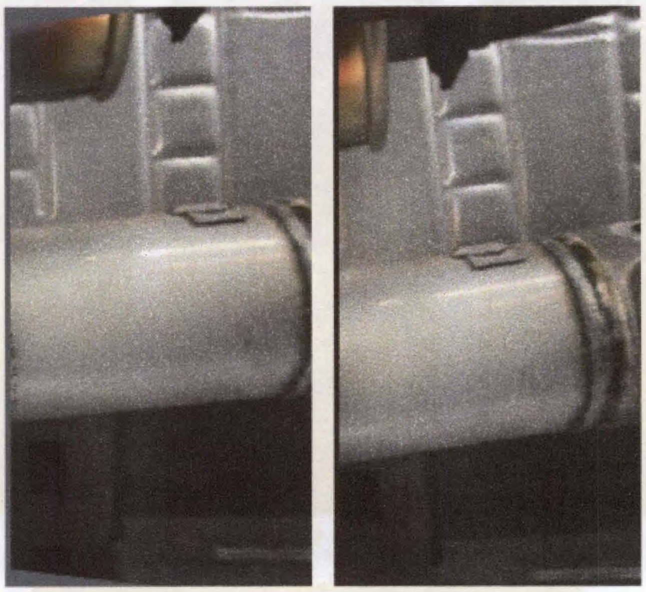

(b)

Figure 5.18 Discontinuities caused by low frame rate. (a) Three frames from the sequence UnderV1. (b) The resulting mosaic. The three frames shown are consecutive in the sequence; the large translation between the first two frames caused problems with registration.


(a)

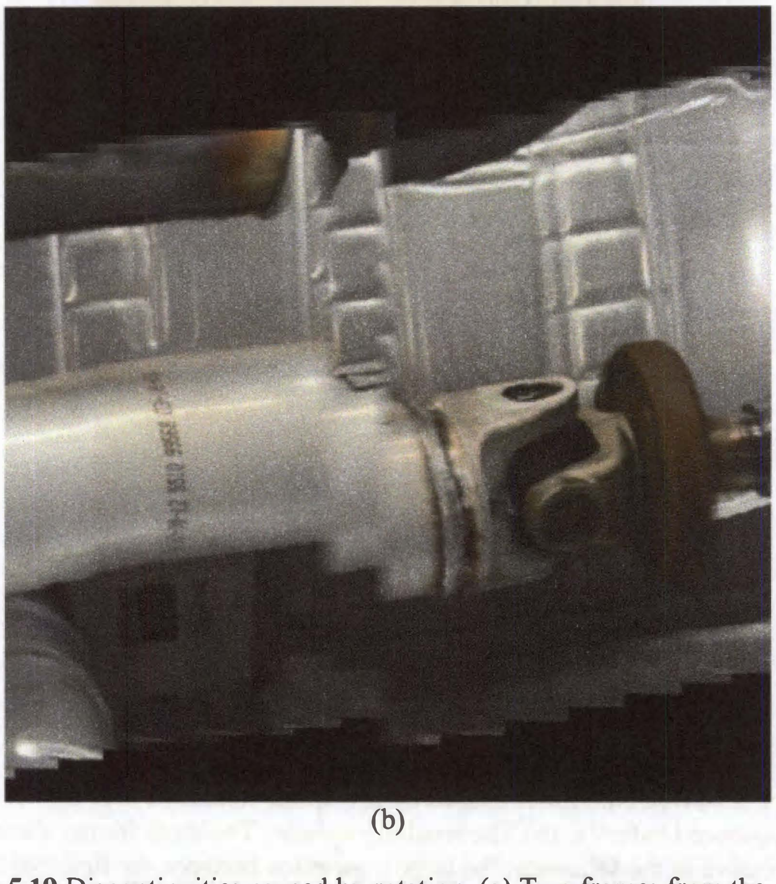
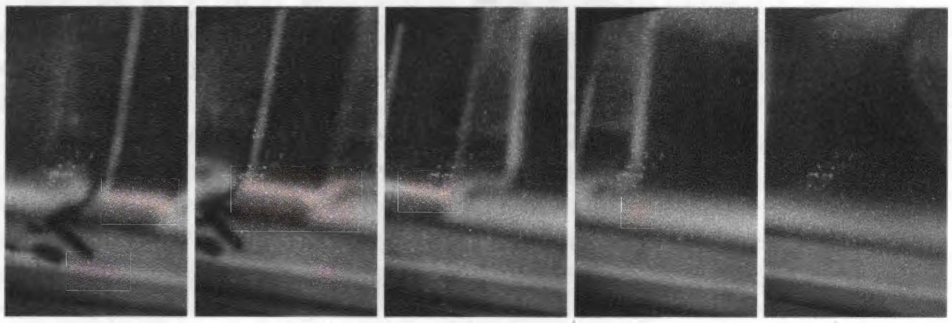

(b)

Figure 5.19 Discontinuities caused by rotation. (a) Two frames from the sequence UnderV2. (b) The resulting mosaic. The rotation angle between the two input frames is interpreted as a vertical translation by the registration algorithm.


(a)

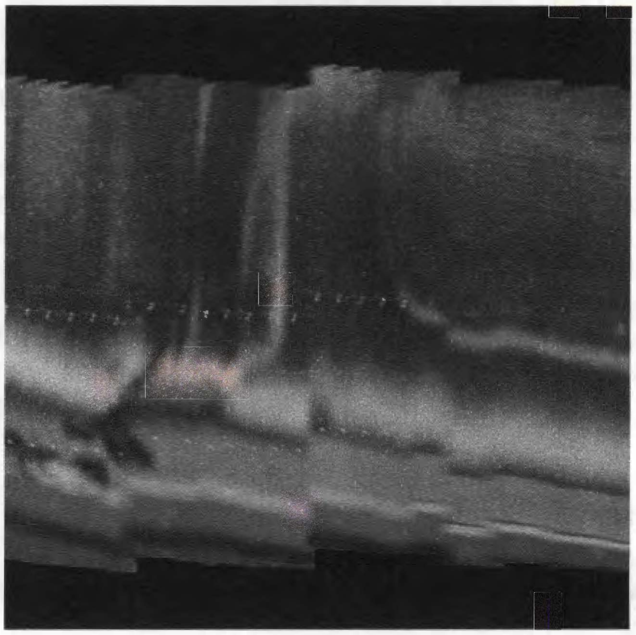

(b)

Figure 5.20 Discontinuities caused by image blur. (a) Five frames from the sequence UnderV3. (b) The resulting mosaic. The camera's auto-focus function caused this part of the sequence to appear blurred, complicating the registration process.

In Figure 5.17, a diagonally oriented metal bar and a large pipe both appear to be 'sheared' in the mosaic. This is because the two elements are at different distances from the camera, and are thus moving past the camera's field-of-view at different rates. The shearing is caused by the fact that no single translation vector can correctly compensate for the translational movement of both elements. Hence, a strip in the mosaic may be aligned properly for one element, but not for another, creating the shearing effect. This is simply a case where the limits of our assumption, that the entire scene exists on a single plane, is plainly observable, and tends to be noticeable wherever there is substantial disparity in the relative distances of elements from the camera. Recall from our discussion of the effects of inter-frame motion in Section 3.4 that if we had sampled the video at a faster frame rate, these effects would be less noticeable, since the misalignments then become distributed throughout the additional strips, creating a smoother-looking mosaic.

In Figure 5.18, a discontinuity due to incorrect motion estimation is shown. An element close to the camera moved past the camera's field-of-view at more than 100 pixel-frames. This large inter-frame motion caused a poor estimation of the camera movement. In the mosaic, a large visual discontinuity is apparent at the center of the element. Again, the solution lies in sampling the video sequence so that the inter-frame motions are within an acceptable range.

In Figure 5.19, the discontinuity is cause by camera rotation. Notice how, between the two input frames where the discontinuity occurs, there is a noticeable change in the orientation of the drive-train. Since we do not recover for rotation effects in this implementation, the registration algorithm models the rotation as a vertical translation, causing the shearing of the drive-train in the mosaic. Some preliminary tests were performed using Reddy and Chatterji's [34] rotation-invariant phase correlation technique, which was discussed in Section 2.1.3, but these tests did not produce correct results for the two images shown here. We did not attempt to address the problem any further, and instead leave it as future work.

Our final example is shown in Figure 5.20, where the discontinuities are simple misregistration due to image blur. This blur was caused by the camera refocusing during the acquisition process. This problem is easily solved by turning off all auto-focus functions on the camera prior to acquisition. Nevertheless, it is interesting to see how the algorithm performs in the presence of image blur.

5.4 Remarks

From our results, some conclusions may be inferred about the conditions under which our algorithm performs well. Obviously, a video sequence should display no out-of-focus frames, and minimal jerking (rotating) motion. Also, the overall quality of the mosaic will be determined by the magnitude of the inter-frame motions (a property related to the speed of the moving platform and the frame rate). For the standard color video sequences, which had inter-frame motions on the order of 50-100 pixel-frames, the discontinuities tend to be somewhat severe in several parts of the mosaic, whereas there are virtually no visual discontinuities discernible in the infrared sequence (except for the part of the mosaic where there is little discernible camera motion), which had inter-frame motions on the order of 10-50 pixel-frames. Granted it is more difficult to gauge the quality of the infrared mosaic due to the nature of the infrared sequence, but it is reasonable to assume that if the inter-frame motions of the color video sequences were within the order of 10-50 pixel-frames, the quality of those mosaics would be improved. Again, it should be noted that the inter-frame motions of a video sequence are affected by both the speed of the camera capturing the sequence as well as the frame rate at which we sample the video sequence.

We have now completed our discussion of the results of our single-mosaic registration algorithm. Now we move on to a discussion of the results obtained using the layered-mosaic representation algorithm.

Chapter 6

Layered-Mosaics Representation Results

This chapter presents the experimental results for the layered-mosaic representation algorithms proposed in Chapter 4. We will begin with the details of the data capture process and the data sets used in these experiments in Section 6.1. We then outline the processing parameters used for each of these data sets, and discuss the rational behind these parameters in Section 6.2. Finally, the resulting registration parameters and mosaics obtained for each data sequence are shown in Section 6.3.

6.1 Experiment Setup

We will first discuss the method used to capture the video sequences used in this work, as well as the various parameters used to process each sequence in Section 6.1.1. Then, we will discuss the rationale behind the model initialization parameters used for each video sequence in Section 6.1.2.

6.1.1 Processing Parameters and Test Data

The video sequences used in this work were captured using a Sony DCR-TRV730 Digital 8 Camcorder, which uses a 1/4" 1.07 mega pixel color CCD. Video was recorded on 8mm tapes in 480x720 resolution. The camcorder was mounted on the roof of a van during acquisition and was pointed towards the side of the van. The vehicle used as the mobile platform used to capture the data is shown in Figure 6.1.

The video sequences were processed using the Adobe Premier 6.0 video editing software, which was briefly described in Section 5.1. As before, the extracted video sequences were written out to 24-bit bitmap sequences. The video sequences used in this work are divided into individual data sets, so that the input and the results may be compared conveniently. Table 6.1 shows example frames from these data sets, the number of frames in each set, the size of each image, and the names assigned to each set.

Unlike the video sequences used for the single-mosaic representation experiments, we did not undersample these video sequences. This is because the Lucas-Kanade algorithm works better, to a point, with video sequences at higher frame rates. Recall from Section 4.2.1 that our Lucas-Kanade implementation (without taking into account our multi-scale approach) is accurate for pixel velocities of approximately 2 pixels. This is why it is desirable to maintain as high a frame rate as possible.

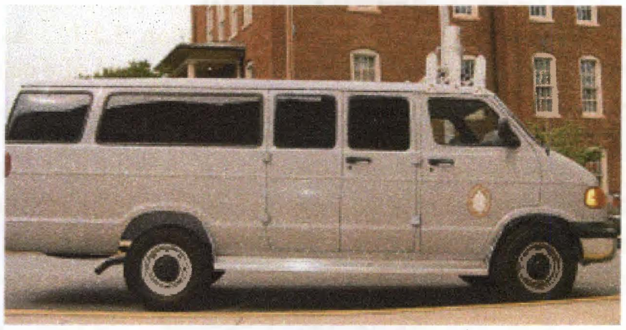


Figure 6.1 The vehicle used as the mobile platform for roadside data acquisition. A rig was placed on the roof of the van to hold the scanning equipment.

Table 6.1 Data sets from undervehicle inspection video

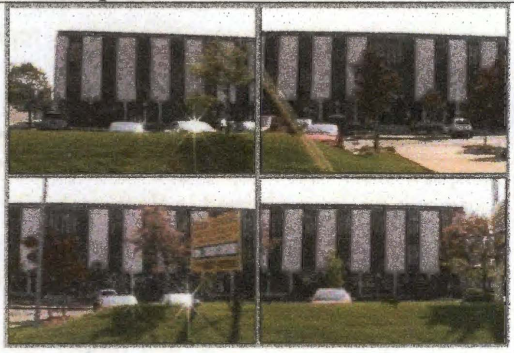
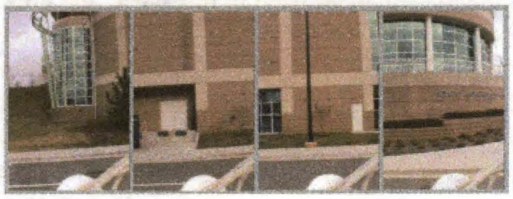
Data Set	Frames	Frame Size	Sample Frames
Warren	504	720x480	
BBHall	914	480x720	

Table 6.2 lists the various parameters used in the preprocessing and registration stages of the algorithm for each data set. Summaries of each parameter and appropriate comments are as follows:

Barrel Distortion Correction Parameters: a_x and a_y, the vertical and horizontal distortion correction factors, as explained in Section 3.2.1. As before, these parameters were estimated manually, using images captured of a calibration grid as a visual reference.

Perspective Distortion Parameters: ω, φ, and κ, the pan, tilt, and rotation angles, as explained in Section 3.2.2. These parameters were set manually, using the video frames as a visual reference. In most cases, only the rotation angle, κ, was used to correct for minor rotations in the camera orientation. In some cases, the tilt angle φ was also used to make corrections.

Temporal Smoothing Parameters: μ and σ, the Gaussian function parameters, as explained in Section 4.2.1 These parameters are always set to 6 and 2, respectively, but we include them here for completeness.

Multi-scale Factors: f_1 and f_2, the rescaling factors used for multi-scale motion analysis, as explained in Section 4.2.1.

Model Initialization Parameters: u_n and v_n, the vertical and horizontal vectors associated with each layer, as discussed in Section 4.2.2. We do not explicitly list the number of layers used to represent each video sequence. Instead, the number of (u_n, v_n) vectors listed indicates the number of layers. The elements in each video sequence that each layer is meant to represent is discussed in further detail in Section 6.1.2.

Layer Composition Parameters: *dist*, the pixel-wise distance between strips sampled from an image, and *k*, the number of mosaics used to perform layer composition, as explained in Section 4.2.3.

6.1.2 Layer Assignments for Model Initialization

Recall from Section 4.2.2 that model initialization is performed manually, using estimates of the inter-frame motions of elements in the video sequence. In the video sequences used in this work, the object of interests were building facades that were obscured by various foreground objects, i.e., signboards, lamp posts, trees, etc. The layers were chosen with the intent of separating the building facades from the foreground objects. For the both sequences, layer P_2 was assigned as the facade layer, and layer P_3 was assigned as the foreground layer. A background layer, P_1, was assigned to represent the plane at infinity - the plane where there is no discernible pixel motion. In a real video sequence, this is usually used to represent sky, though there may be large regions of moving elements that are homogenous, and therefore appear to have no inter-frame motion. The BBHall sequence, in particular, has very little visible sky, but many homogenous regions (road, grass, etc) with little or no discernible local motion. Parts of these regions tend to be misregistered as belonging to layer P_1. In the video sequences used in

Table 6.2 Processing parameters. (a) Preprocessing and registration parameters

Dataset	Preprocessing		Registration	
	Barrel Distortion	Perspective Distortion	Temporal Smoothing	Multi-scale factors
Warren	$a_x = -0.026$ $a_y = -0.012$	$\omega = 0°$ $\varphi = -5°$ $\kappa = 1.5°$	$\mu = 6$ $\sigma = 2$	$f_1 = 6$ $f_2 = 4$
BBHall	$a_x = -0.012$ $a_y = -0.026$	$\omega = 0°$ $\varphi = 0°$ $\kappa = -2.5°$	$\mu = 6$ $\sigma = 2$	$f_1 = 6$ $f_2 = 4$

Table 6.2 (b) Model initialization and layer composition parameters

Dataset	Model Initialization	Layer Composition
Warren	P_1: $u_1 = 0$, $v_1 = 0$ P_2: $u_2 = 0$, $v_2 = -2.5$ P_3: $u_3 = 0$, $v_3 = -6$	$dist = 45$ $k = 9$
BBHall	P_1: $u_1 = 0$, $v_1 = 0$ P_2: $u_2 = 0$, $v_2 = -4$ P_3: $u_3 = 0$, $v_3 = -6$	$dist = 20$ $k = 5$

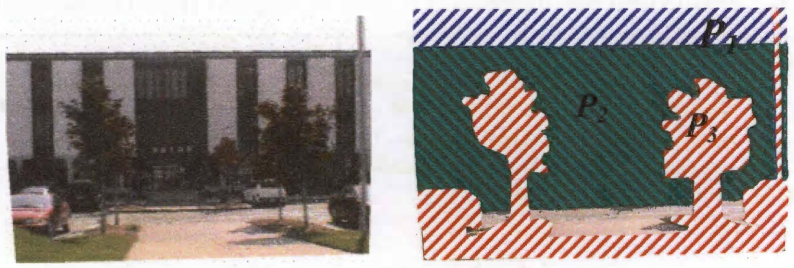
this work, the effects of large homogenous regions on our results are reduced by processing the by-layer segmentations using the hierarchical morphological operator described in Section 4.2.2, as will be made clear in Section 6.2.

In Figure 6.2, the approximate layer assignments of various elements (the layers we are *attempting* to assign those elements to, using the parameters of Table 6.2(a)) for each video sequence are shown, using sample frames from the video sequences. These are only the *intended* layer assignments, not the actual results of the algorithm. Note that there are various regions in the images whose intended layer assignments are not explicitly shown; these are elements that were not necessarily of interest, and it does not matter to which layer they are assigned. Most of these 'unimportant' elements tend to be the large, homogenous regions discussed above.

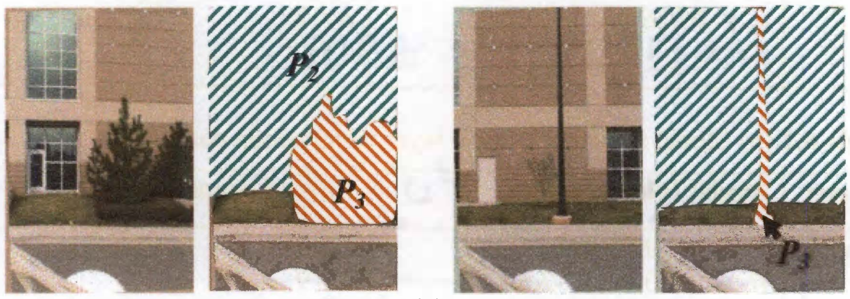
This ends our description of the experiment setup. Now we move on to examine the intermediate results of our method, using the parameters just described.

6.2 Results of Algorithms

In this section, we will first examine the results of using our spatial support determination algorithms, which include our motion analysis and segmentation-by-motion algorithms. The result of using these algorithms for sample frames from the test video sequences are shown and discussed in Section 6.2.1. Then the



(a)



(b)

Figure 6.2 Approximate layer assignments for model initialization. Shown for (a) The 'Warren' test sequence, and (b) the 'BBHall' test sequence.

resulting reference and peripheral mosaics, as well as the final composite mosaics are shown and discussed in Section 6.2.2.

6.2.1 Results of Spatial Support Determination Algorithms

We shall use several sample frames from the two test sequences to illustrate the results of processing the sequences at the various stages of our motion analysis and spatial support determination algorithms. Results for individual frames are shown in Figure 6.3 for images from the Warren sequence, and Figure 6.4 for the BBHall sequence. The results for each stage in the algorithm are shown side-by-side, in sequence, so that the results from the different stages of the algorithm may easily be compared. In order, each sequence shows a) the results of performing Lucas-Kanade motion analysis (Section 4.2.1), the initial segmentation by motion (Section 4.2.2), and the refined segmentation after using the hierarchical morphological operator (Section 4.2.2). For purposes of comparison, the resulting segmented images are shown alongside the intended layer assignments in Figure 6.5.

The results shown in Figures 6.3 and 6.4 still show several incorrectly assigned regions in the final segmentations. However, even though the spatial support determination for individual frames may not be entirely correct (and for most real-world sequences, never will be, at least with our current implementation), it should be noted that over the course of many frames, assignment errors do not tend to be propagated. As will become apparent in Section 6.2.2, our layer composition method is capable of correcting for most of these incorrectly assigned regions.

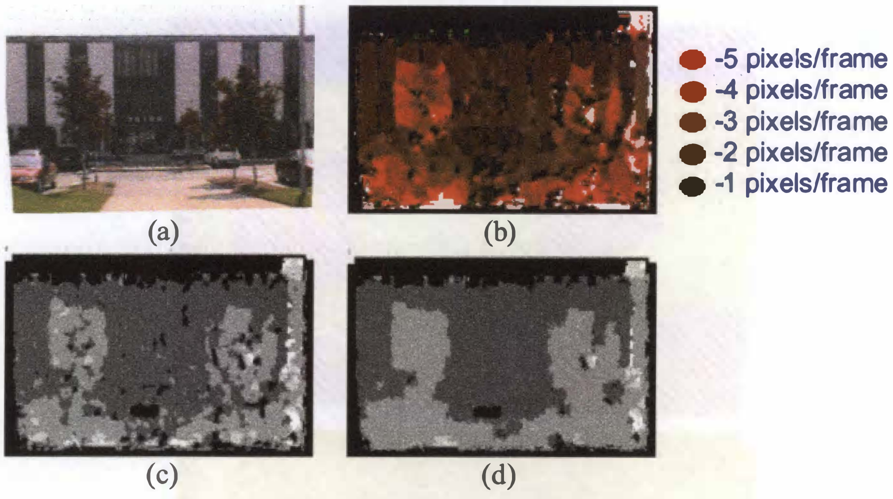


Figure 6.3 Results at different stages of spatial support determination.
(a) A sample frame from the Warren sequence, (b) the result of Lucas-Kanade motion analysis, (c) the results of segmentation by motion, and (d) the results of hierarchical morphological operation.

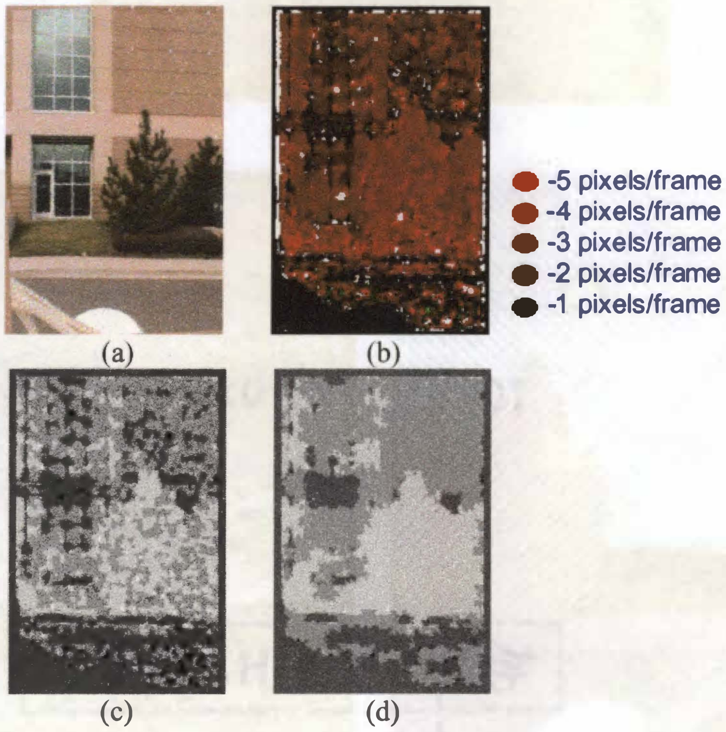
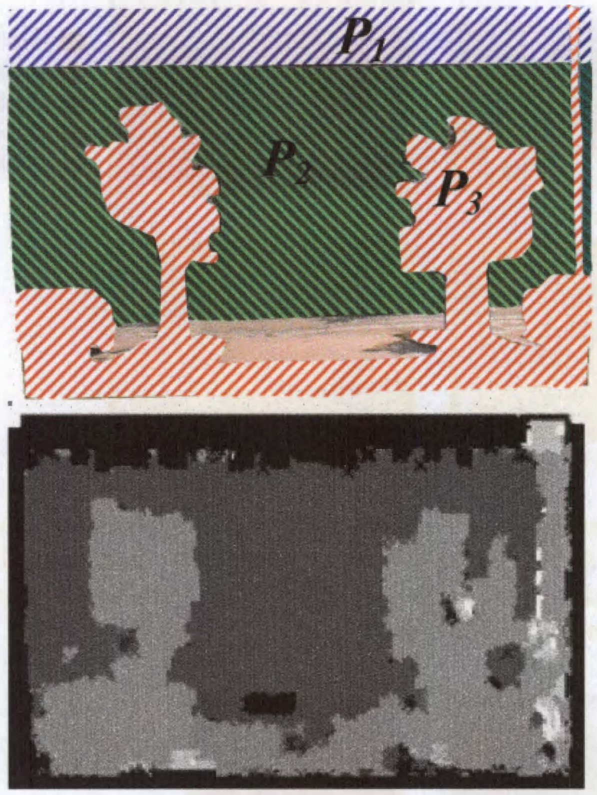
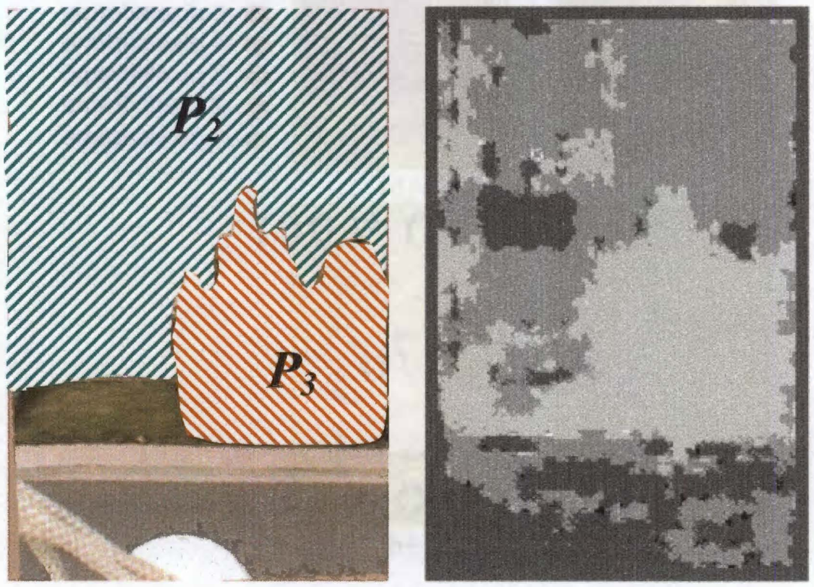


Figure 6.4 Results at different stages of spatial support determination.
(a) A sample frame from the BBHall sequence, (b) the result of Lucas-Kanade motion analysis, (c) the results of segmentation by motion, and (d) the results of hierarchical morphological operation.



(a)



(b)

Figure 6.5 Comparison of intended layer assignments and the spatial support determination of our algorithm. Shown for sample frames from (a) the Warren sequence, and (b) the BBHall sequence.

6.2.2 Mosaics of Road Video Sequences

After the spatial support for each layer has been determined for each frame in the video sequences, the process of forming the layered mosaics may begin. First, the reference and peripheral mosaics for each layer are formed, using the method discussed in Section 4.2.3. The resulting mosaics for the P_2 layer are shown in Figure 6.6 for the Warren sequence, and Figure 6.8 for the BBHall sequence. These mosaics are further processed using the closing operator described in Section 4.2.3, and the results for the Warren and BBHall sequences are shown in Figures 6.7 and 6.9, respectively. Finally, using the layer composition method described in Section 4.2.3, the final composite mosaics for the Warren sequence and the P_3 layer are shown in Figure 6.10, and the same for the BBHall sequence are shown in Figure 6.11. Note that the P_3 layers were formed without having to use the layer composition method, because these layers were not occluded. We merely show the mosaics formed from the strips sampled closest to the center of each video frame. We do not show results for P_1, the plane at infinity, since there is little or no relevant information in these mosaics.

It is difficult to perform a quantitative comparison due to the lack of a ground truth, but these comparisons provide some heuristic confirmation of the accuracy of our results. Figures 6.12 and 6.13 show examples of recombining the layers taken from the Warren and BBHall sequence into one composite image. Nevertheless, from visual comparison between our results and the original video frames, it can be seen that our composite mosaics have recreated the objects of interest in both sequences quite well. In both cases, the foreground layer pixels occlude the background layer pixels, except for the pixels in the foreground layer with unspecified values.

From these results, it can be seen that our mosaicking and layer composition removes most of the occluding elements that were present in the original sequence when composing the facade layer, P_2. However, there are still various aspects of the results that should be examined in detail. Firstly, there were parts of the Warren and BBHall P_2 layer mosaics that could not be recovered fully, simply because the occluding elements were too large. The most visible examples are the tree at the end of the Warren sequence and the large bush at one end of the BBHall sequence. Both of these examples are shown in Figure 6.14 and Figure 6.15, along with accompanying sample frames from the original sequence showing these elements as they originally appeared in the video sequence.

There are also parts of the composite mosaics where the building facade was incorrectly recovered due to the non-planar structure of the building. When parts of a building are markedly closer or further away from the camera, our assumption that the entire building exists on a single plane fails, and this may cause these sections of the building to be incorrectly recovered or assigned to layers aside from the facade layer. An example of this is clearly visible in the BBHall mosaic, where a section of the building that curves away from the camera is incorrectly recovered. This example, which is visible in the example from Figure 6.15, is shown with emphasis in Figure 6.16.

A final note on the foreground (P_3) mosaic for the Warren sequence: in the original sequence, the trees were actually closer to the camera than the cars parked behind the grassy mound, and were hence moving past the camera's field of view faster than the cars. However, we assigned both the trees and cars to the same

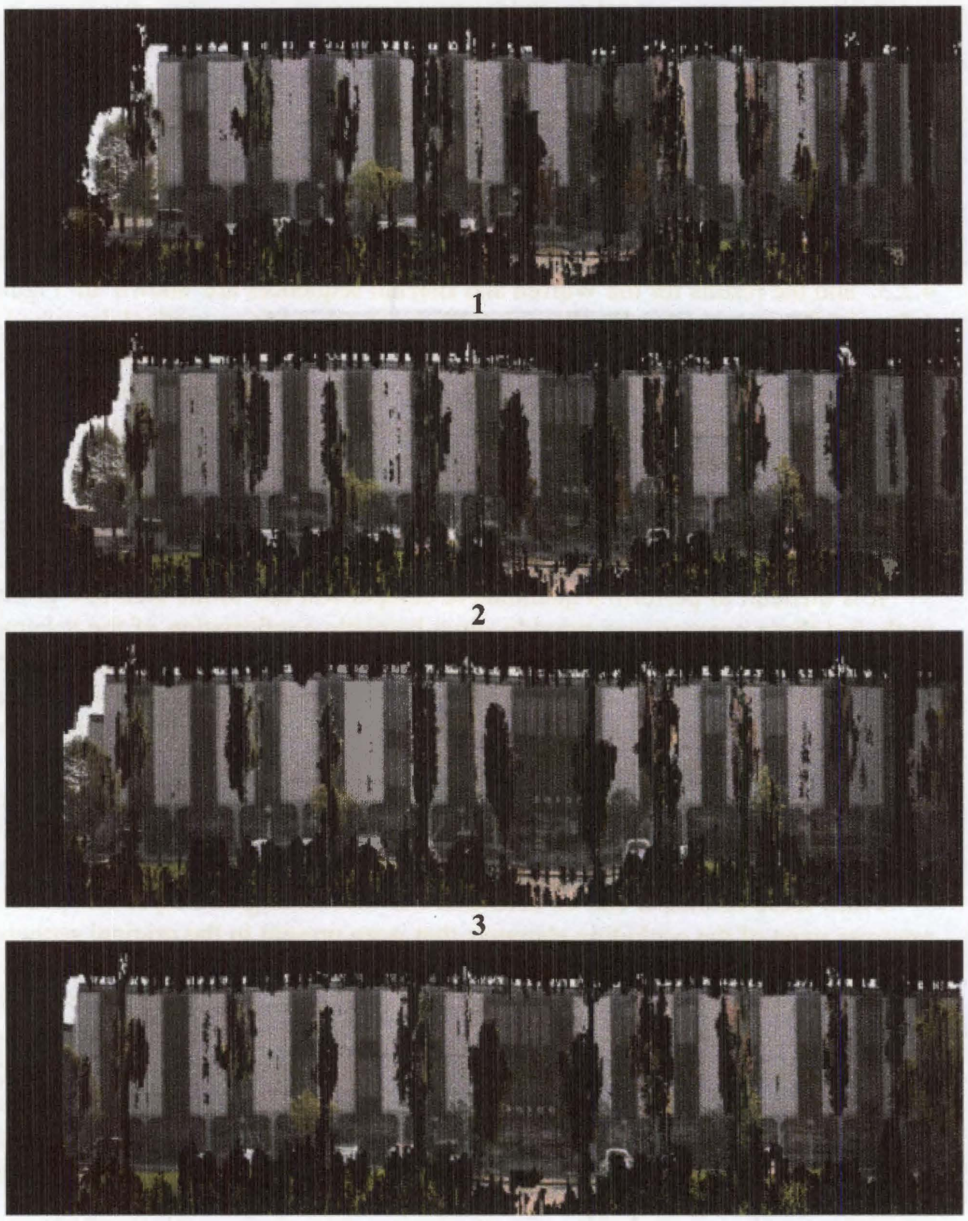


Figure 6.6 Mosaics 1 through 4 of the Warren sequence created for the P_2 layer. Nine mosaics were created to represent this layer. The reference mosaic is the fifth mosaic (according to the order the mosaics are shown). The rest are the peripheral mosaics, spatially offset from the reference mosaics by *m* x *dist* pixels, with *m* determined by the order of each mosaic relative to the reference mosaic. The black gaps in these mosaics denote areas that do not belong to this layer (the sky in the background, occluding objects such as trees, signboards, etc). These gaps are still noisy, however, and a morphological closing operator must be used to completely remove the elements that do not belong to the P_2 layer.

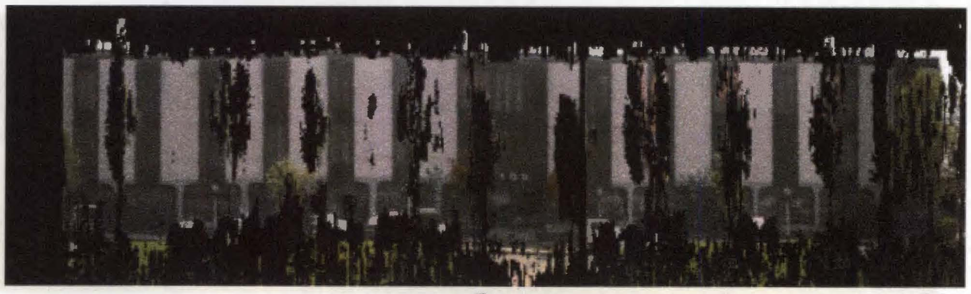
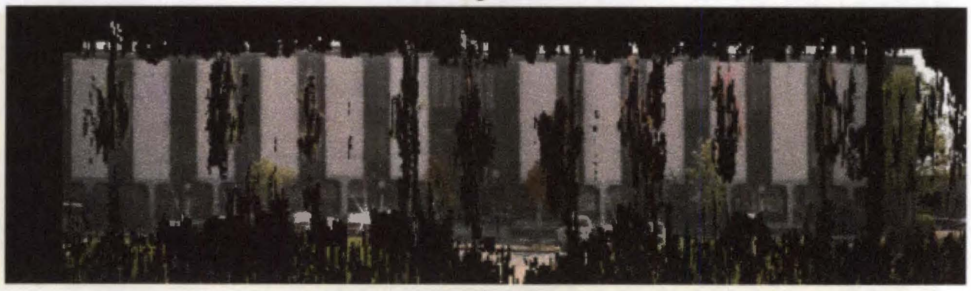
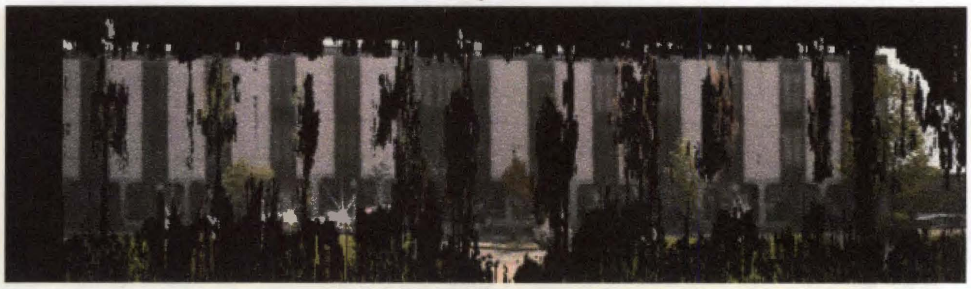
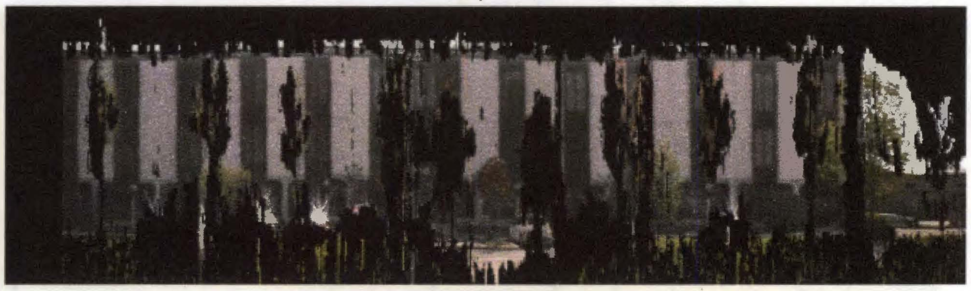
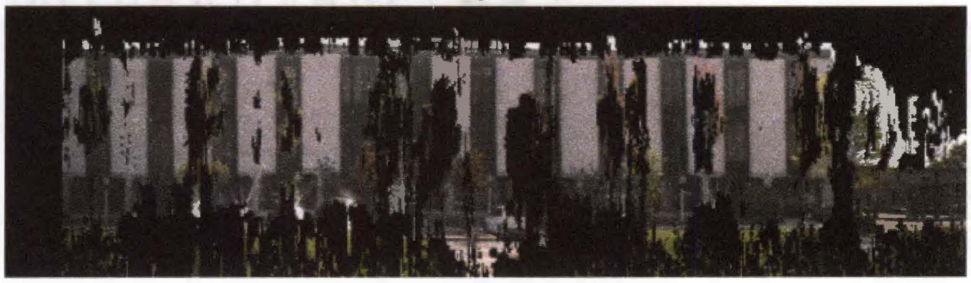
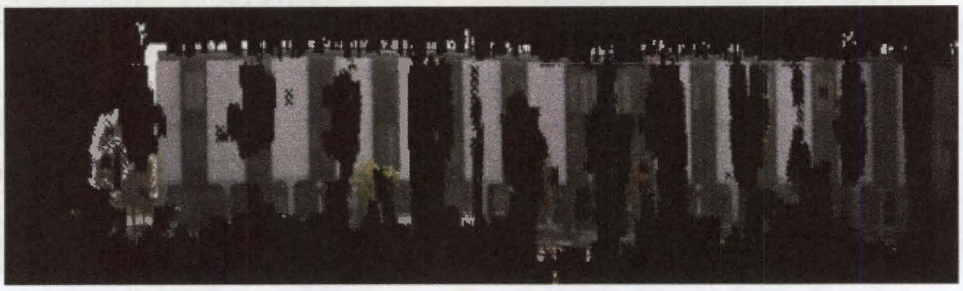
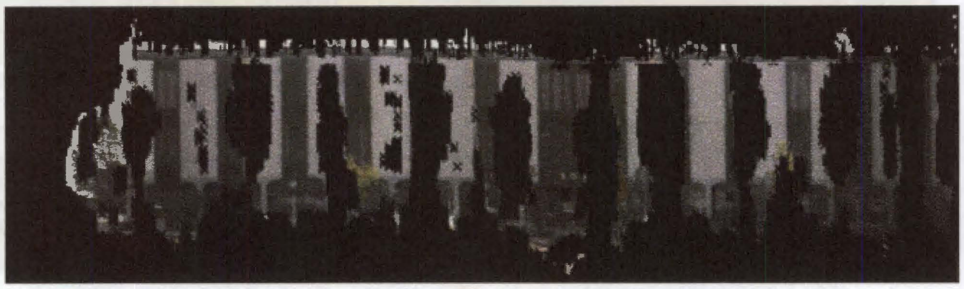

5

6

7

8

9

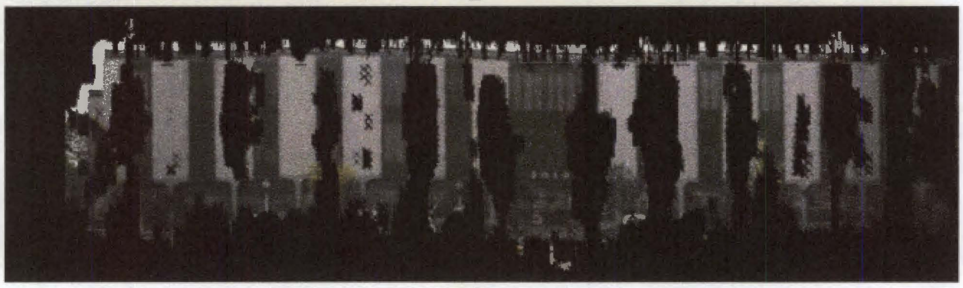
Figure 6.6 Continued. Mosaics 5 through 9 of the Warren sequence created for the P2 layer.



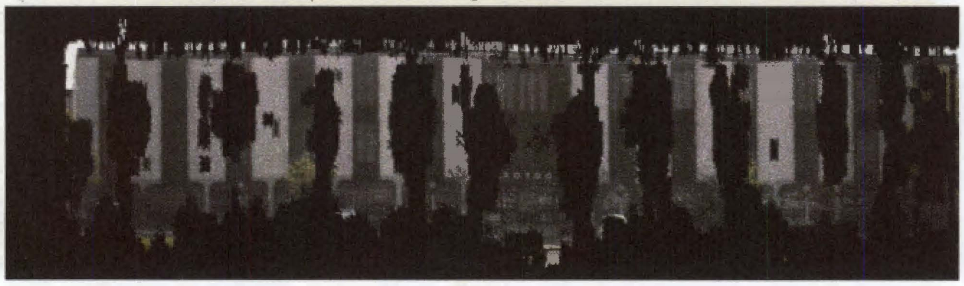
1



2

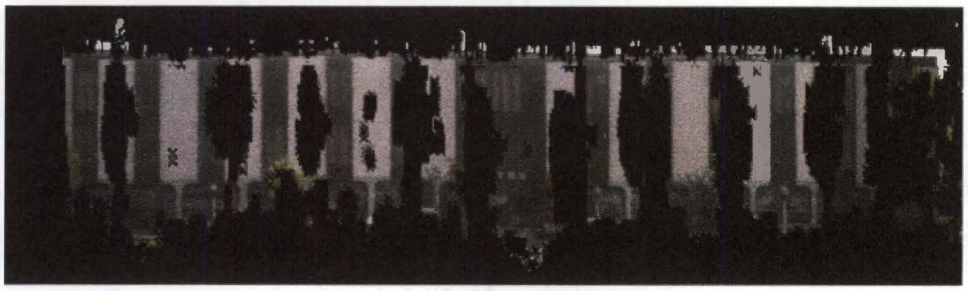


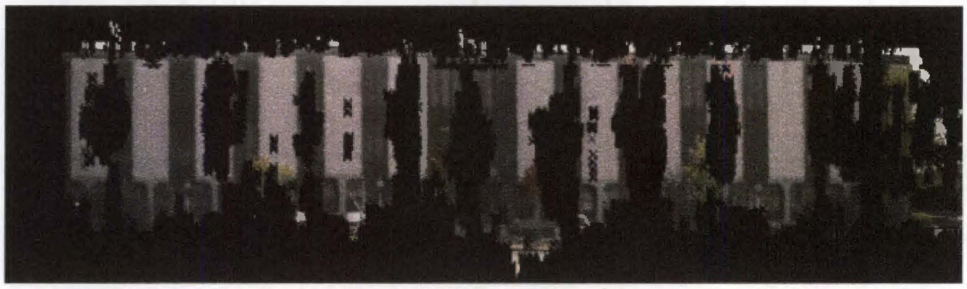
3

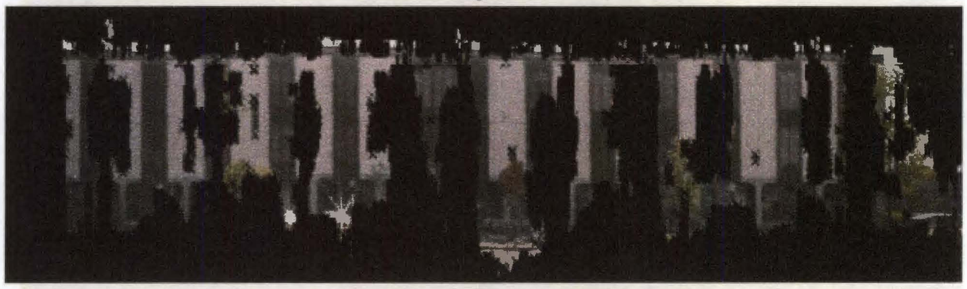


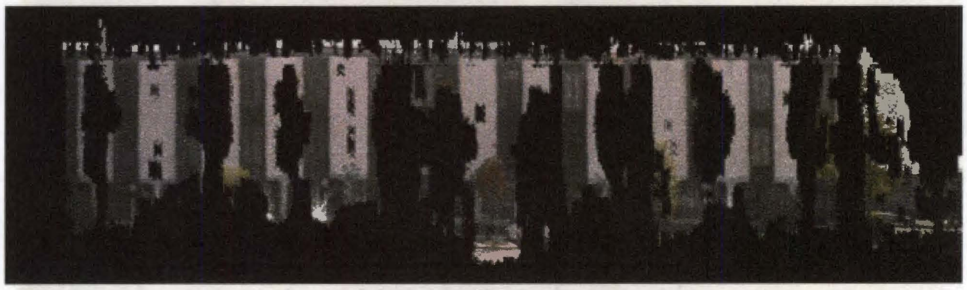
4

Figure 6.7 Mosaics 1 through 4 of the Warren sequence created for the P2 layer, after closing operation. With the non-layer regions better defined, the reference and peripheral mosaics may now be used to create the final composite mosaic seen in Figure 6.10.


5


6


7


8

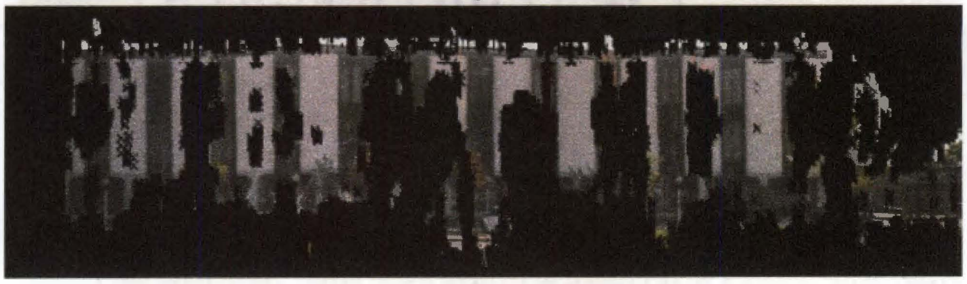

9

Figure 6.7 Continued. Mosaics 5 through 9 of the Warren sequence created for the P_2 layer, after closing operation.

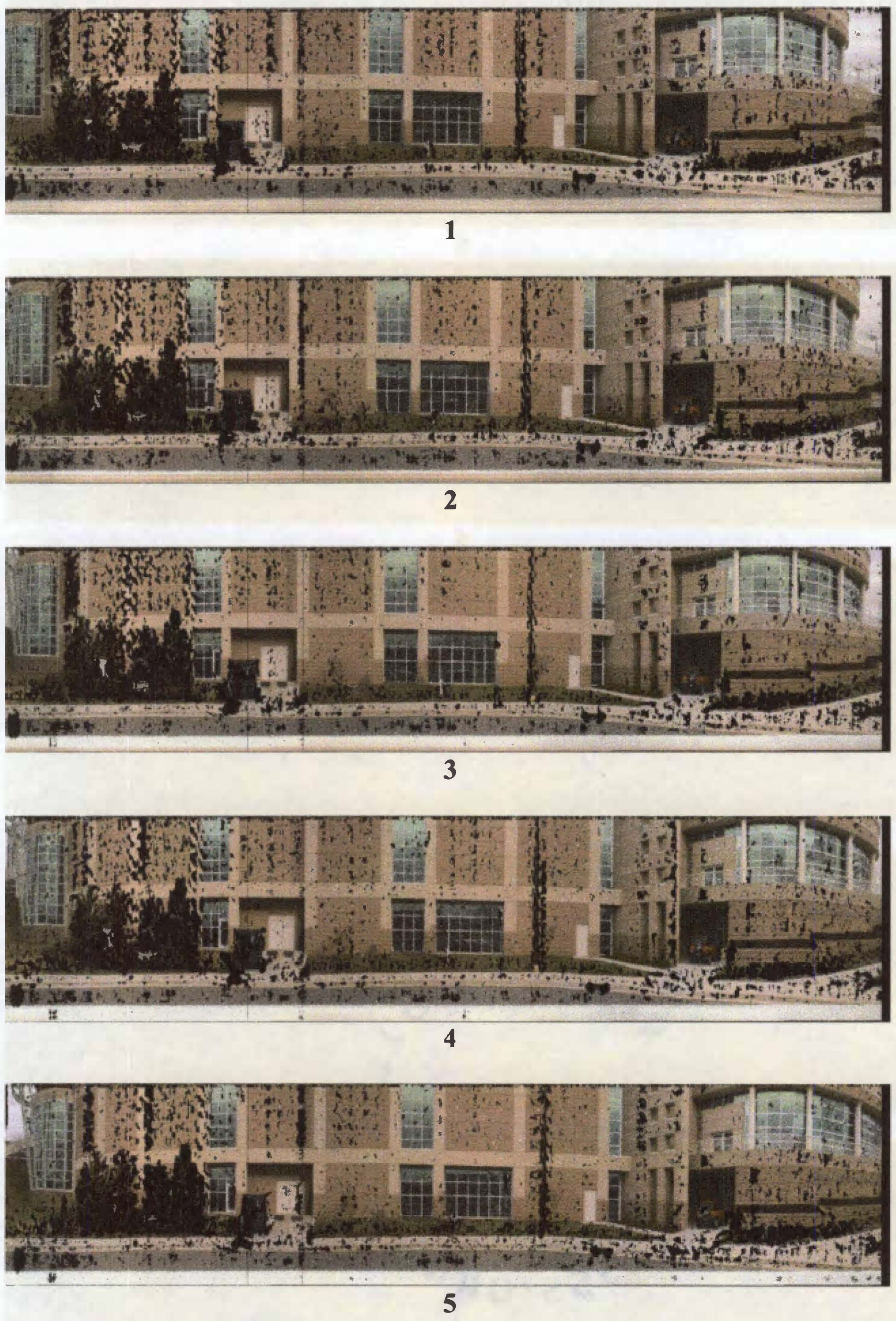
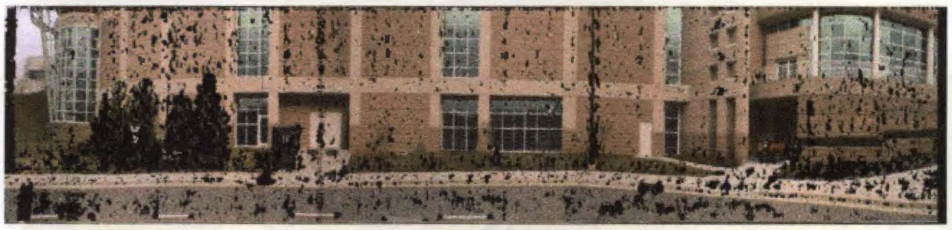
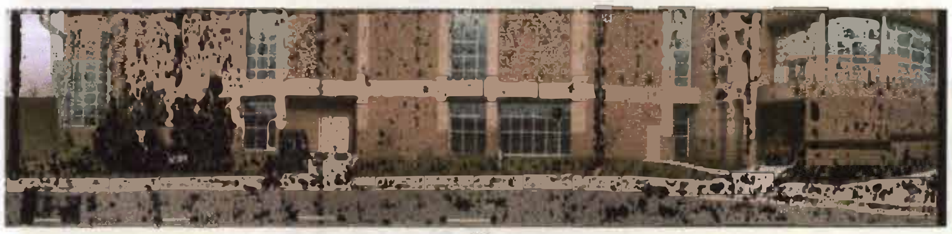


Figure 6.8 Mosaics 1 through 5 of the BBHall sequence created for the P_2 layer. Nine mosaics were created to represent this layer. As before, the reference mosaic is the fifth mosaic. Again, the black gaps in these mosaics denote areas that do not belong to the P_2 layer, and the morphological closing operator will again be applied to close the gaps in these regions.

6


7


8

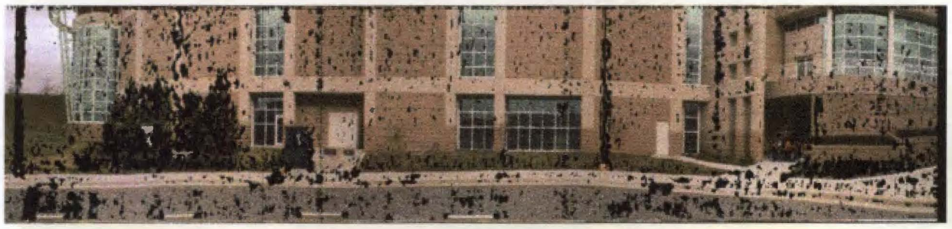

9

Figure 6.8 Continued. Mosaics 6 through 9 of the BBHall sequence created for the P_2 layer.

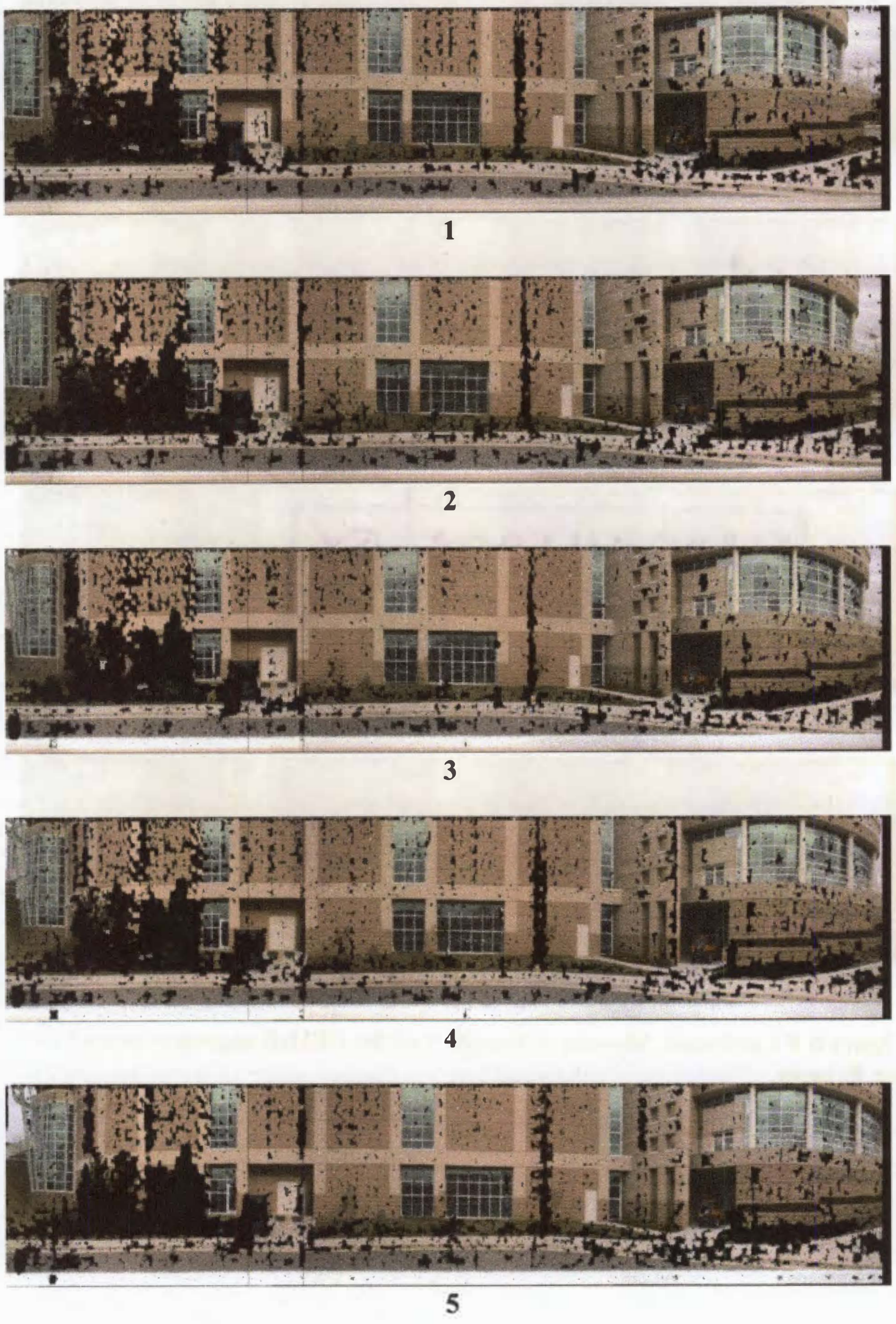


Figure 6.9 Mosaics 1 through 5 of the BBHall sequence created for the P2 layer, after closing operation. The reference and peripheral mosaics may now be used to create the composite mosaic of Figure 6.10.

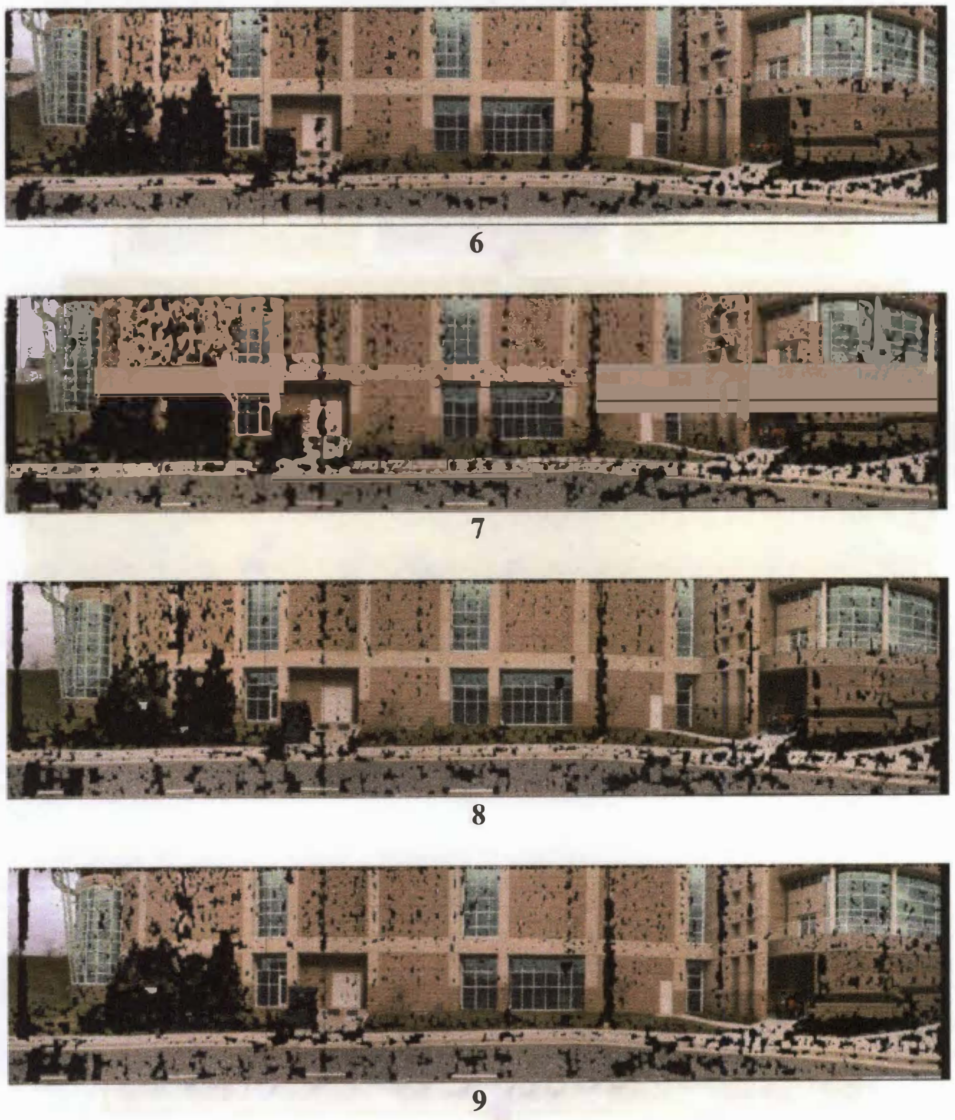


Figure 6.9 Continued. Mosaics 6 through 9 of the BBHall sequence created for the P_2 layer, after closing operation.

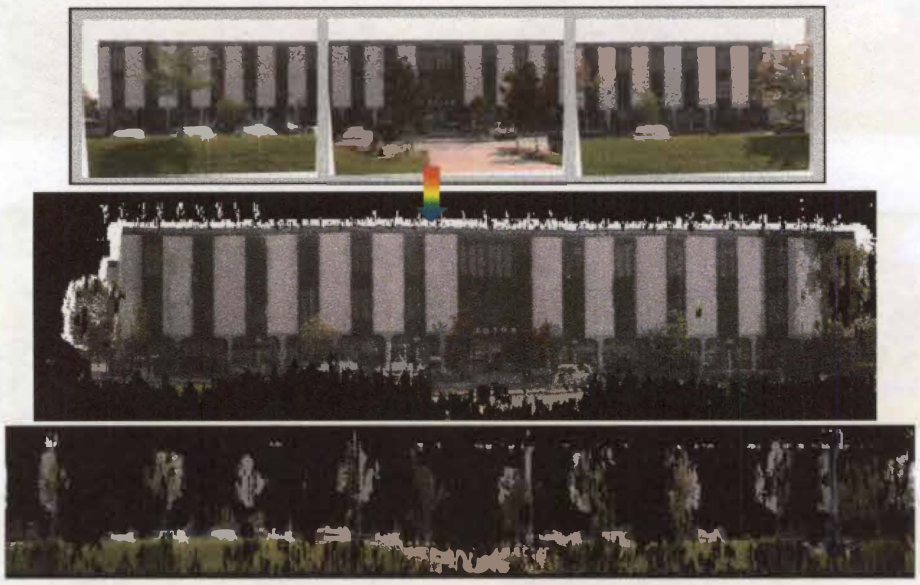


Figure 6.10 Comparison between output layer mosaics and original frames from the Warren sequence.

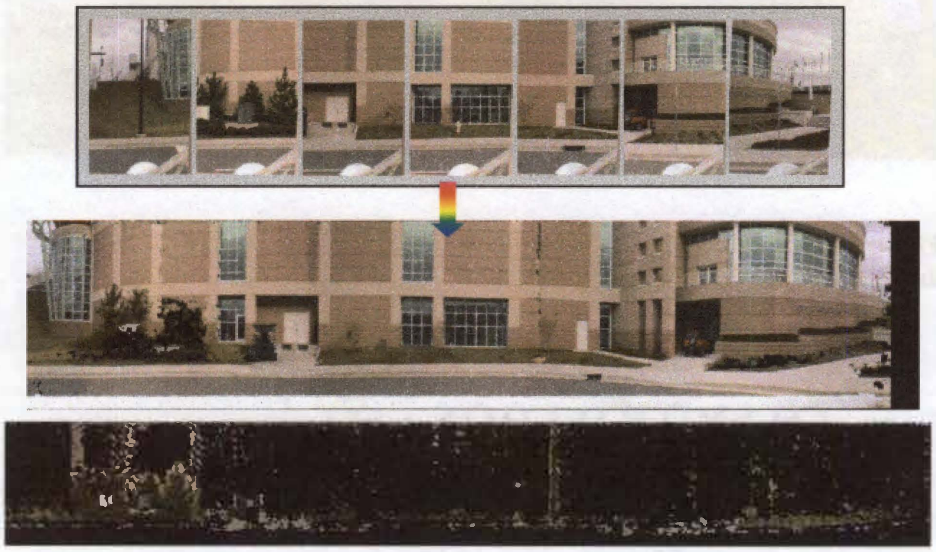


Figure 6.11 Comparison between output layer mosaics and original frames from the BBHall sequence.

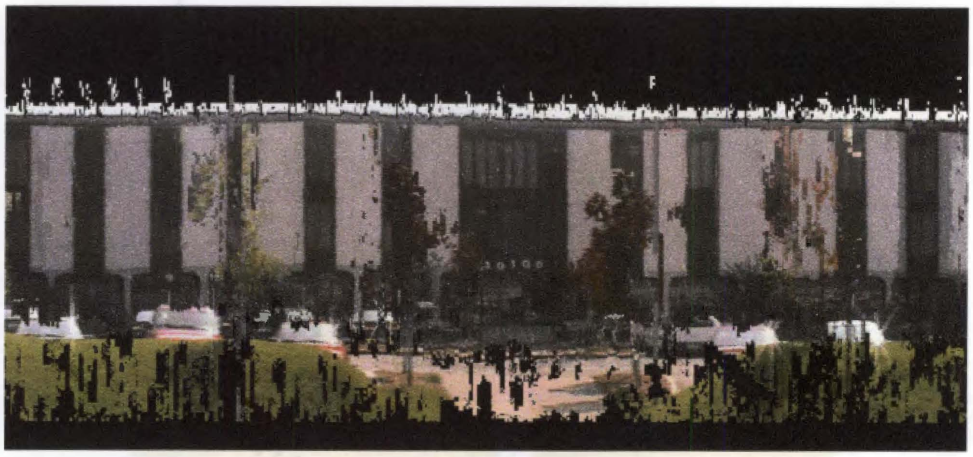


Figure 6.12 Composite of facade and foreground layers created from the Warren sequence. Note that since the foreground layer extracted for the Warren sequence had a wider pixel-wise width than the background layer, it is no 'correct' overall fit between the foreground and facade layers in their entirety. Here we only focus on recreating the center of the building, fitting the cars and trees from the foreground layer that occluded this part of the building in the original sequence.

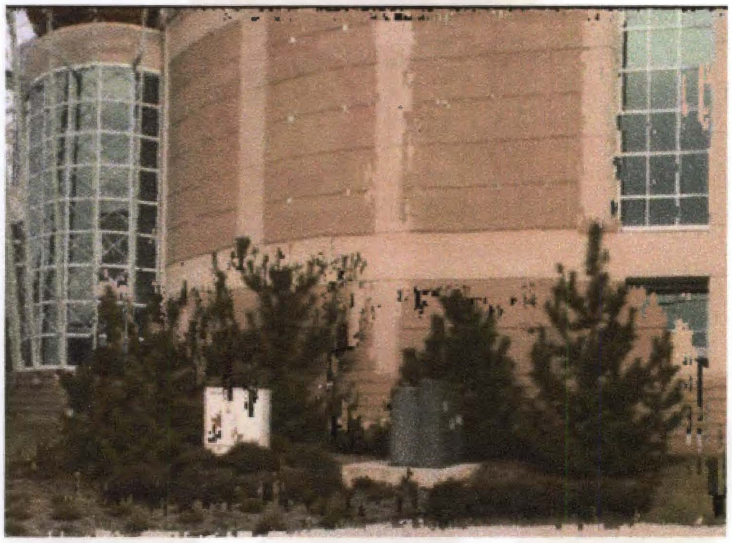
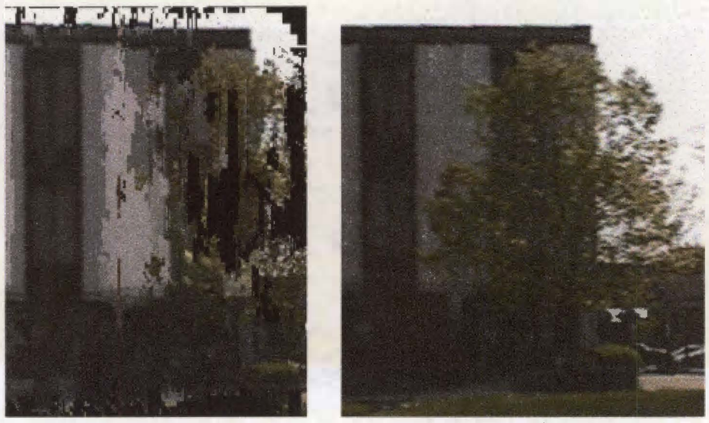
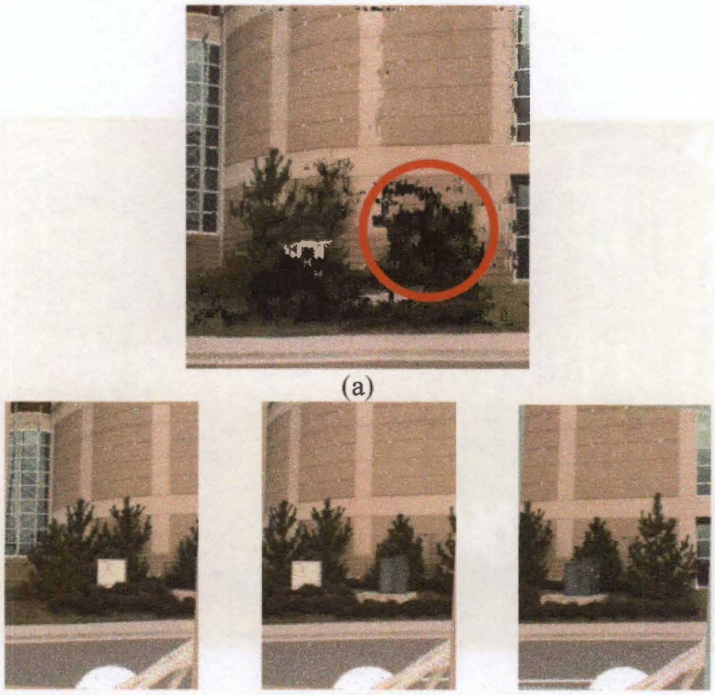


Figure 6.13 Composite of facade and foreground layers created from the BBHall sequence. Note that since the foreground layer extracted for the BBHall sequence had a wider pixel-wise width than the facade layer, there is no one 'correct' overall fit between the foreground and facade layers in their entirety. Here we only focus on recreating the curved part of the building occluded by the large bushes in the foreground layer, fitting them to approximate their appearance in the original sequence.



(a) (b)

Figure 6.14 Example of unrecovered region, due to large occlusions, from the Warren sequence. (a) A section of the composite mosaic (b) Sample frame corresponding to the mosaic section. The tree occludes a large area of the edge of the building, making this region difficult to recover accurately.



(a)

(b)

Figure 6.15 Example of unrecovered region, due to large occlusions, from the BBHall sequence. (a) A section of the composite mosaic (b) Sample frames corresponding to the mosaic section. The large bush at this end of the building makes a large part of the building impossible to recover.

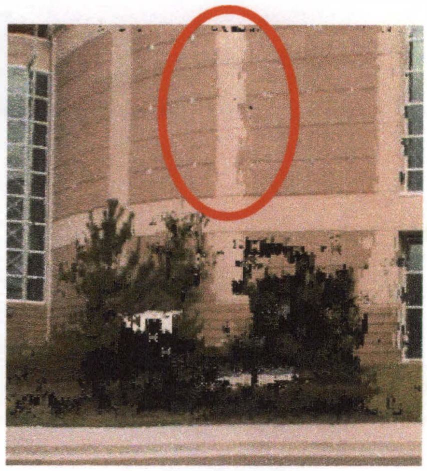


Figure 6.16 Example of an incorrectly recovered region due to the non-planar structure of the building facade.

layer, since we were only interested in recovering the building facade correctly. The trees are therefore not recovered correctly, and are all slightly truncated from their original sizes in the video sequence. The cars have been correctly recovered, since the layer vector corresponds to the speed at which the cars moved past the camera's field of view.

6.3 Remarks

The video sequences used in our experiments in this work showed a sufficient amount of detail and the speed of the camera in both sequences was relatively constant, which helped to ensure reasonable layer assignments throughout the entire sequence. Hence, the results for these sequences were somewhat satisfactory. Still, the various composition errors discussed earlier shows that the algorithm can be improved.

This concludes our discussion of the results of our layered-mosaics representation algorithm. We shall conclude with a discussion of possible improvements to the algorithms described in this document.

Chapter 7

Conclusions

In this final chapter, we first provide a summary of the work that we have presented in this document in Section 7.1. We then conclude with directions for future improvements to the work we have presented here in Section 7.2.

7.1 Summary

We have presented two separate but related methods of mosaicking dense video sequences. Both methods use a multiperspective mosaicking framework to take advantage of the large amount of video data available in order to create visually smooth and continuous mosaics. Two modes of representation for video scenes were presented: the single-mosaic representation that is used to mosaic scenes with little or no motion parallax, and the layered-mosaics representation, which is used to represent scenes displaying large motion parallax.

To our knowledge, this is the first time phase correlation has been applied as a primary registration method for performing multiperspective mosaicking, and some of the advantages and disadvantages of using phase correlation have been presented here. The literature reviewed in this work which addressed multiperspective mosaicking used gradient-based local motion detection or intensity-matching methods to perform registration. One advantage of using phase correlation is that we can quickly acquire a dominant inter-frame motion, even if there is small motion parallax, which can be used to perform registration. The drawback is that we do not acquire local motion estimates, which is used in some of the works reviewed to perform local correction for misregistrations due to motion parallax. However, this problem may be alleviated by using denser video sequences (such that the inter-frame motions are on the order of 10-40 pixel-frames), which results in local misregistrations being distributed more evenly throughout the mosaic, resulting in a smoother mosaic. It has been shown that this technique is robust with respect to imaging modalities: it is capable of mosaicking both normal visible-spectrum video sequences as well as video taken in the infrared spectrum. It may be concluded that, provided the motion constraints of the camera are adhered to, the video is sufficiently dense, and proper constraints on the ICPS search region are provided, that the phase correlation registration method provides good results for video sequences of scenes with small motion parallax.

Only one other published effort we are aware of, by Zhu et al. [27], attempts to represent a video sequence of roadside scenes as a series of layered mosaics, which was summarized in Section 2.1.2, and briefly compared with our effort in Section 2.3. Another effort, by Zheng et al. [24, 25], also attempts to create mosaics of roadside sequences, but since their primary purpose is robotic navigation, not recreation of specific scene elements, they make no attempt to create a layered, spatially-accurate representation of the elements in their mosaics. Our method attempts to recreate all elements in the scene in their correct spatial proportions by sampling them according to the speed at which they move past the camera's field of view and mosaicking them into motion-specific layer mosaics. Our method requires that the layer model be manually initialized, after which a derivation of the Lucas-Kanade algorithm, which is a gradient-based local motion detection method, is used to determine spatial support for each layer within each frame of the video sequence. A hierarchical morphological operator was devised to compensate for the 'noisy' initial segmentations. Mosaics are created for each layer using these segmented frames. We devised a layer composition scheme that uses reference and peripheral mosaics, created from strips sampled at different points in each video frame, in order to recover occluded regions of a layer. This allows us to compensate for occlusion without explicitly determining occlusion relationships. The method works well as the elements in the scene adhere well to our planar-layer assumptions.

The results obtained in this work point towards many directions for improvements for our algorithm. Next, we shall discuss these possible future directions for research.

7.2 Future Work

In keeping with the overall structure of our document, we shall first discuss possible improvements that may be made to our single-mosaic representation algorithm, and finish with a discussion of suggested improvements to the layered-mosaics representation algorithm. Improvements to the single-mosaic representation algorithm are discussed in Section 7.2.1, and improvements to the layered-mosaics algorithm are discussed in Section 7.2.2.

7.2.1 Single-Mosaic Representation

The current implementation of the single-mosaic representation still uses offline processing, and has not yet been modified to form mosaics in real-time. Optimization of the mosaicking code or a hardware implementation of the code would increase the practicality of the algorithms developed here, especially with regards to undervehicle inspection applications.

There are some aspects of the algorithm that could be automated to increase its robustness. Recall from Section 3.2 that the preprocessing parameters used to perform lens distortion and perspective distortion correction are set manually by the user. Several intelligent algorithms for correcting lens distortion have been proposed, and there are camera calibration tools widely available on the internet. Many of these options would most likely provide more accurate lens distortion results compared to our implementation.

Intelligent selection of perspective distortion parameters, however, is not as widely addressed. In our work, perspective distortion correction was performed by adjusting roll, pitch, and yaw angles that represented the angles at which the camera was viewing the scene, using visual cues in the images as reference. However, the validity of the results are based purely on heuristics; we have no ground truth by which to gauge the validity of the results. If we did have a reference image that was 'correct', then the problem would be reduced to matching corresponding points between the uncorrected and reference image. Unfortunately, no such reference images exist for the data we would typically deal with. What possible solutions are there to this problem? One possible solution would be to detect patterns of converging lines in an image, and associating a projective transformation that would make these lines parallel. Another would be to simply experimentally determine suitable parameter values for various vehicle ground clearances, and have the user select a ground clearance level that would in turn determine the correction parameters.

The selection of the tapering window parameter *a*, and the ICPS search window parameters was described in Section 5.2 as a trial-and-error process. The results of these experiments point towards a way of adjusting these parameters automatically. One may begin by assuming that *a* is within the range of *dim*/2 to *dim*/1.5 (where *dim* is the pixel dimension the square input image), and assign it a value within this range by default (for example, a reasonable suggestion, for an 256x256 pixel image would be *a* = *dim*/1.75 = 256/1.75 = 146.286, as was used in this work). Then, to update *a* and the search window parameters, we may use the sum of intensity errors to gauge the correctness of each registration, and modify the ICPS search window if the sum of intensity errors is above a threshold. This approach would be similar to what was implemented in this work, except the check using the sum of intensity errors is used for each registration.

One problem that was faced in evaluating the results of our algorithm was that we lacked a ground truth by which to gauge the accuracy of our results. The correctness of the mosaicking algorithm is judged purely by observation. Also, we lack any method of compensating for accumulated error in the mosaics. For the purposes of inspecting the underside of vehicles, it may not have been necessary to achieve a 100% accurate reconstruction of the scene. Still, a logical extension of our work would be to somehow eliminate or compensate for the accumulated error in our mosaics. A hardware-oriented approach to addressing this problem might involve using a wide angle-lens to capture the entire width of the vehicle so that a single mosaic of this wide-angle video would completely cover the entire underside of the vehicle. This might of course entail some more complex preprocessing to correct for the extreme distortion of wide-angle lenses prior to mosaicking. However, this approach runs somewhat counter to our original goal of maximizing the pixel resolution of our results. An alternative would be to use multiple cameras on a wider platform, with the entire width of the vehicle covered by the span of the platform.

Another approach would be to investigate the integration of external velocity measurements (again, using hardware) from the moving platform to support and enhance our results. Using such measurements, it will be possible to eliminate or reduce the accumulated errors in our mosaics, which result in mosaics such as those seen in Figure 5.15, which cannot be aligned correctly, because they do not perfectly correspond to one another spatially. What are the advantages of aligning the mosaics with one another spatially? Spatially aligned mosaics would help the inspection process when viewing the entire underside of the vehicle. Spatially aligned mosaics would also assist in another process: if pixel correspondences between the mosaics were to be determined, then that information could be used to infer 3D information using epipolar geometry. This idea may also be applied to the multiple-camera configuration described above, since the different mosaics may be aligned easily by assuming each camera was moved past the camera at equal speeds. A rough 3D representation of the scene would give viewers more degrees of freedom while viewing the scene, further facilitating the inspection process.

We have discussed some of the logical extensions of our work regarding the single-mosaic representation. Now we shall move on to discuss improvements to the layered-mosaics representation.

7.2.2 Layered-Mosaics Representation

In this work, we chose not to address the problem of automating the model initialization process. Many other works discussed in Section 2.2 already discuss layer model determination, either as part of or before the process of spatial support determination. In our case, since we perform spatial support determination based on the local velocity of pixels, it would make sense to determine the layers in a scene by detecting clusters of motion vectors in the vector space. Each cluster would of course correspond to a layer, with the centroid of the cluster representing the motion vector associated with that layer. The challenges associated with this approach would be a) identifying a suitable clustering algorithm, and b) determining the cluster size/properties that would identify a cluster as representing a valid layer. There is one other consideration: there may be layers whose elements are not present throughout the entire sequence. The layer model may therefore have to be updated as the sequence is processed, when elements of a new layer are detected that no longer 'fit' into any of the existing layers. With this approach, we would no longer be initializing the layer model, but updating the model concurrently as we are determining spatial support for each layer.

Another aspect of our algorithm that could be improved upon is the motion analysis algorithm used to perform spatial support determination. Currently we use a derivation of the Lucas-Kanade algorithm to perform motion analysis. This algorithm performs well if its constraints are met and when there is sufficient detail in the scene, but oftentimes we encounter video sequences that are difficult for a local motion estimator to deal with. First, the smoothness constraints associated with the Lucas-Kanade method are not always strictly adhered to; the moving platform capturing the data may exhibit changes in speed, and lighting conditions may vary from frame to frame. Second, building facades in a video sequence usually exhibit large homogenous areas where there may be little or no detail to be detected. Local motion estimators would therefore compute no motion in these regions, even though it is clear from observing the borders of these regions that they are moving.

What are some possible solutions to these problems? One solution would be to use a spatiotemporal segmentation technique that associates spatial regions in each video frame with a velocity, as opposed to purely local estimates of velocity. Acquiring a spatiotemporal segmentation of each frame gives us three advantages: a) we would be less restricted by the smoothness constraints of local motion estimators, b) we would be able to match large homogenous regions to their appropriate velocity estimates, and c) we now have a method of tracking the

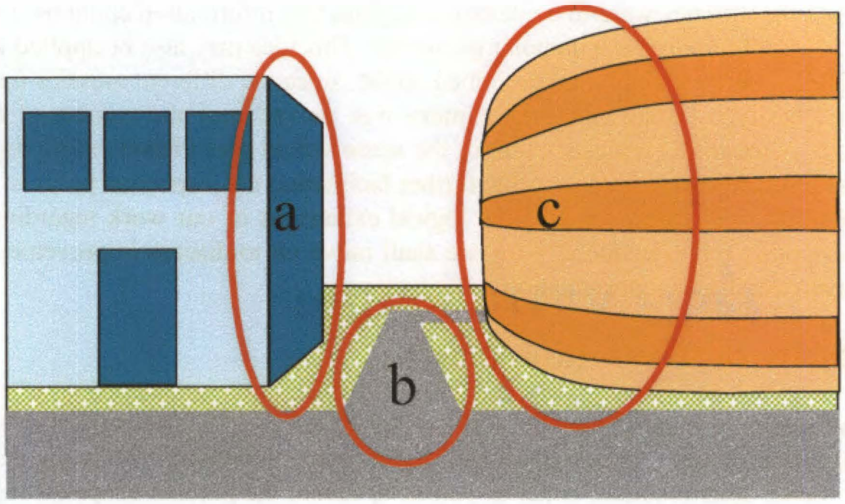


Figure 7.1 Examples of common elements that do not obey the constraints of our algorithm. a) sides of buildings, b) roads/paths c) curved facades.

movements of spatial regions, even if the speed of the platform changes during the video sequence, hence keeping all initial layer assignments intact. Our algorithm currently requires that the platform used to perform image capture moves at a uniform speed throughout the sequence. A proper spatiotemporal segmentation algorithm would help free us from this restriction.

Currently our algorithm is only capable of recreating layers consisting of planar elements, and only if those planes are orthogonal to the camera's principal axis. Any parts of the scene that do not obey these restrictions tend to be recreated poorly, or their layer assignments are incorrect. Some examples of scenes that are not represented well by our algorithm are shown in Figure 7.1. To account for these elements, we could follow Wang and Adelson's lead, and attempt to match regions of each image to affine motion models using linear regression, and then clustering those regions that display similar motion parameters. Each of these clusters would form a layer with the corresponding affine motion parameters. However, what about the case where a single building facade exhibits more than one motion model, as the curved facade in our example does? Would we want that building to be represented by one layer, or more? If we wish to associate only one layer with that facade, then how do we intelligently associate all the different motion models exhibited by the building facade with the same layer? Finally, do we wish to continue representing such facades, which are no longer planar, as 2D mosaics, or do we wish to better represent the curved aspect of the facade using a 3D representation, as Zhu's 3D LAMP technique does? All these questions, related by the same problem, should be investigated in tandem, since their individual solutions most likely influence the entire framework.

Our final suggestion for future work again involves integrating external hardware measurements of vehicle velocity in order to correct for accumulated error. This extension would be similar to the one previously discussed for the single-mosaic representation. In addition to compensating for accumulated error, hardware velocity measurements provide an additional solution to the problem of varying vehicle speed discussed earlier. Such measurements would help us keep track of the layer assignments of elements even when the speed of the vehicle changes drastically throughout the video sequence, since we can initialize our layer assignments at the beginning and modify the motion model associated with each layer according to the hardware velocity measurements.

7.3 Closing Remarks

In this document, we have presented the efforts made to combine and implement several paradigms and techniques used in digital image mosaicking and layer extraction to support the task of scene visualization. Two closely related solutions were tailored to the specific needs for which the data was acquired. For the undervehicle inspection effort, a single-mosaic representation was devised to ease the process of inspection, and for the outdoor roadside scanning effort, a layered-mosaics representation was devised to remove occlusions from objects of interest and recreate elements in the presence of motion parallax. It is hoped that these solutions may be improved upon and extended, as well as applied to scene visualization applications aside from those documented here.

References

[1]M. Kirk, D. Denning, "Eyes and Eye Evolution," *http://ebiomedia.com/gall/eyes/eye1.*

[2]R. Benosman, S.B. Kang, *Panoramic Vision*, Springer-Verlag, New York, 2001.

[3]R. Naughton, "Adventures in Cybersound, " *http://www.acmi.net.au/AIC/PANORAMA.html.*

[4] G. R. Crack, "Bayeux Tapestry, " *http://hastings1066.com/.*

[5] Reading Borough Council (Reading Museum Service), Berkshire, UK, "The Reading Bayeux Tapestry," *http://www.bayeuxtapestry.org.uk.*

[6]M. Irani, P. Anandan, J. Bergen, R. Kumar, S. Hsu, "Efficient representations of video sequences and their applications, " *Signal Processing : Image Communication*, 8:pp. 327-351, 1996.

[7]Y. Chen, "Image Stitching-Comparisons and New Techniques," *Center for Image Technology and Robotics Technical Reports*, CITR-TR-30, NewZealand: Department of Computer Science, The University of Auckland, 1998.

[8]R. Szeliski, H. Y. Shum, "Creating full view panoramic image mosaics and environment maps," *Proceedings of SIGGRAPH*, pp.252-258, 1997.

[9]R. I. Hartley, "Self-calibration from multiple views of a rotating camera," *Third European Conference on Computer Vision*, 1:pp. 471-478, May 1994.

[10]G. Stein, "Accurate internal camera calibration using rotation, with analysis of sources of error," *Fifth International Conference on Computer Vision*, pp. 230-236, June 1995.

[11]R. Szeliski, "Image mosaicking for tele-reality applications," *IEEE Workshop on Applications of Computer Vision*, pp. 44-53, December 1994.

[12]M. Irani, P. Anandan, S. Hsu, "Mosaic bases representations of video sequences and their applications," *Fifth International Conference on Computer Vision*, pp. 605-611, June 1995.

[13]R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, pp. 22-30, March 1996.

[14]C. J. Harris, M. Stephens, "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference,* pp. 147-151, 1988.

[15]I. Zoghlami, O. Faugeras, R. Deriche, "Using geometric corners to build a 2D mosaic from a set of images," *Proceedings of the IEEE Conference on Computer Vision and Pattern Registration*, pp. 420-425, 1997.

[16]G. Y. Tian, D. Gledhill, D. Taylor, "Comprehensive interest points based imaging mosaic," *Pattern Recognition Letters*, 24:pp. 1171-1179, 2003.

[17] S. Peleg, J. Herman, "Panoramic Mosaics by Manifold Projection," *IEEE Proc. of Conference on Computer Vision and Pattern Recognition*, pp. 338-343, June 1997.

[18]F. Odone, A. Fusiello, "Applications of 2D image registration," Research Memorandum RM/99/15, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, UK, 1999.

[19]J. Davis, "Mosaics of scenes with moving objects," *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 354-360, 1998.

[20]S. E. Chen, "Quicktime VR - An image-based approach to virtual environment navigation, " *Proceedings of SIGGRAPH*, pp. 29-38, 1995.

[21]S. Peleg, M. Ben-Ezra, "Stereo panorama with a single camera, " *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 395-401, June 1999.

[22]Z. Zhu, A. R. Hanson, H. Schultz, F. Stolle, E. M. Riseman, "Stereo mosaics from a moving camera for environmental monitoring," *International Workshop on Digital and Computational Video*, pp. 45-54, 1999

[23]S. Peleg, B. Rousso, A. Rav-Acha, A. Zomet, "Mosaicking on Adaptive Manifolds," *IEEE Pattern Analysis and Machine Intelligence*, 22:10:pp. 1144-1154, 2000.

[24]J.Y. Zheng and S. Tsuji, "Panoramic Representation for Route Recognition by a Mobile Robot," *International Journal of Computer Vision*, 9:1:pp. 56-76, 1992.

[25]M. Shi and J.Y. Zheng, "Spatial Resolution Analysis of Route Panorama," *Proceedings of 2003 International Conference on Image Processing*, 2:pp. 311-315, Sept 2003.

[26]B. Rousso, S. Peleg, I. Finci, and A. Rav-Acha "Universal mosaicking using pipe projection," *Proceedings of the International Conference on Computer Vision*, pp. 945-952, January 1998.

[27]Z. Zhu, A. R. Hanson, "3D LAMP: a New Layered Panoramic Representation," *The Eighth IEEE International Conference on Computer Vision*, 2:pp. 723-730 July 2001.

[28]J. Barron, D. Fleet, S. S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, 12(1):pp. 43-77, 1994.

[29]B. Lucas, T. Kanade, "An Iterative Image Registration Technique With An Application To Stereo Vision," *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674-679, 1981.

[30]J.A. Rodriguez, C. Urdiales, A. Bandera, F. Sandoval, "A multiresolution spatiotemporal motion segmentation technique for video sequences based on pyramidal structures," *Pattern Recognition Letters*, 23:pp. 1461-1469, 2002.

[31]G. Valencia, J. A. Rodriguez, A. Bandera, F. Sandoval, "Spatiotemporal video segmentation and motion estimation through irregular pyramids," *Pattern Recognition Letters*, 36:pp. 1445-1447, 2003.

[32]C. D. Kuglin, D. C. Hines, "The phase correlation image alignment method," *Proceedings IEEE International Conference on "Cybernetics and Society"*, pp. 163-165, 1975.

[33]L. Hill, "Phase Correlation for Motion Measurement," *http://home.freeuk.com/lyndonhill/pc.html.*

[34] B. S. Reddy, B. N. Chatterji, "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration," *IEEE Transactions on Image Processing*, 5(8):pp. 1266-1271, August 1996.

[35]E. A. Adelson, J.Y.A. Wang, "Representing Moving Images With Layers," *IEEE Transactions on Image Processing*, 3:pp. 625-638, Sept 1994.

[36]S. Baker , R. Szeliski, P. Anandan, "A Layered Approach to Stereo Reconstruction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.434-441, June 1998.

[37]P. H. S. Torr, R. Szeliski, P. Anandan, "An Integrated Bayesian Approach to Layer Extraction from Image Sequences," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, pp. 297-303, March 2001.

[38]S. Ayer, H. Sawhney, "Layered Representation of Motion Video," *Proceedings of the International Conference on Computer Vision*, pp.777-784, 1995.

[39]Q. Ke, T. Kanade, "A Robust Subspace Approach to Layer Extraction," *IEEE Workshop on Motion and Video Computing*, pp. 37-43, December 2002.

[40]J. Shi, J. Malik, "Motion segmentation and tracking using normalized cuts," *IEEE International Conference on Computer Vision*, pp. 1154-1160, January 1998.

[41]R. Kumar, P. Anandan, M. Irani, J. Bergen, K. J. Hanna, "Representation of scenes from collections of images," *IEEE Workshop on Representations of Visual Scenes*, pp. 10-17, June 1995.

[42]O. Faugeras, Q. T. Luong, *The Geometry of Multiple Images*, The MIT Press, 2001.

[43]H. Farid, A. C. Popescu, "*Blind removal of lens distortion*," Journal of the Optical Society of America, 18:pp. 2072-2078, September 2001.

[44]R. Swarninathan, S. K. Nayar, "*Non-metric calibration of wide-angle lenses and polycameras*," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2:pp. 419, 1999.

[45]P. Bourke, "Lens Correction and Distortion, " *http://astronomy.swin.edu.au/~pbourke/projection/lenscorrection.*

[46]Eric W. Weiss , "Eric Weisstein's World of Mathematics," *http://mathworld.wolfram.com/HammingFunction.html.*

[47]C. Neustaedter, "An Evaluation of Optical Flow using Lucas and Kanade's Algorithm," Calgary, Canada: Department of Computer Science, University of Calgary.

[48]R. Fisher, S. Perkins, A. Walker, E. Wolfart, "Image Processing Learning Resources," *http://www.dai.ed.ac.uk/HIPR2/libmot.htm.*

[49]R. Gonzalez, R. Woods, *Digital Image Processing*, Addison Wesley Publishing Company, Reading, Massachusetts, 1992.

Vita

Jin Choon Ng was born in Seremban, Negeri Sembilan, Malaysia on the 24th of March, 1978. He began attending Metropolitan College in Subang Jaya, Selangor, Malaysia in 1995 and transferred to the University of Tennessee, Knoxville in 1997, where he received a B.S in electrical engineering in 2000, and an M.S. in electrical engineering in 2003.