IMAGE SEQUENCE DESCRIPTION USING SPATIOTEMPORAL FLOW CURVES: TOWARD MOTION-BASED RECOGNITION

By

MARK C. ALLMEN

A thesis submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN – MADISON

1991

Abstract

Recovering a hierarchical motion description of a long image sequence is one way to recognize objects and their motions. Intermediate-level and high-level motion analysis, i.e., recognizing a coordinated sequence of events such as walking and throwing, has been formulated previously as a process that follows high-level object recognition. This thesis develops an alternative approach to intermediate-level and high-level motion analysis. It does not depend on complex object descriptions and can therefore be computed prior to object recognition. Toward this end, a new computational framework for low and intermediate-level processing of long sequences of images is presented.

Our new computational framework uses spatiotemporal (ST) surface flow and ST flow curves. As contours move, their projections into the image also move. Over time, these projections sweep out ST surfaces. Thus, these surfaces are direct representations of object motion. ST surface flow is defined as the natural extension of optical flow to ST surfaces. For every point on an ST surface, the instantaneous velocity of that point on the surface is recovered. It is observed that arc length of a rigid contour does not change if that contour is moved in the direction of motion on the ST surface. Motivated by this observation, a function measuring arc length change is defined. The direction of motion of a contour undergoing motion parallel to the image plane is shown to be perpendicular to the gradient of this function.

ST surface flow is then used to recover ST flow curves. ST flow curves are defined such that the tangent at a point on the curve equals the ST surface flow at that point. ST flow curves are then grouped so that each cluster represents a temporally-coherent structure, i.e., structures that result from an object or surface in the scene undergoing motion. Using these clusters of ST flow curves, separate moving objects in the scene can be hypothesized and occlusion and disocclusion between them can be identified.

The problem of detecting cyclic motion, while recognized by the psychology community, has received very little attention in the computer vision community. In order to show the representational power of ST flow curves, cyclic motion is detected using ST flow curves without prior recovery of complex object descriptions.

Acknowledgements

Many thanks go to Professor Charles R. Dyer for guiding me through my years of learning as a graduate student. He first introduced me to the world of computer vision and due to his teaching I am the researcher I am today. Professor Dyer provided a very productive and pleasant work environment, including financial and intellectual support, but most importantly, freedom to pursue my personal research interests.

All the members the Vision Group were very helpful with their many insightful comments. Special thanks go to Ayse Unat for her comments on related work and pointers into the literature. I'll greatly miss her help.

I would like to thank all my friends who made my years in Madison a very enjoyable experience: Lester McCann, Donovan Schneider, Lisa Leutenegger, Scott Leutenegger, Steve Scott, Ganesh Mani, George Bier, Cheryl Bier, Colleen Moore and Michele Fábrega. Their friendship made my life special and provided a great balance between work and play.

In addition to exercising my mind by working on this thesis, I exercised my body to get away from this thesis. Thanks to all the people who kept me company while I kept in shape, in particular, Colleen Moore. Colleen and I made great swimming coaches for each other; workouts will not be the same without her.

Finally, many thanks to my family for all the encouragement and support they have given me my entire life. I would not be where I am today without them.

This work has been supported by the University of Wisconsin Graduate School under Project Number 910288 and the NSF under Grant Numbers IRI-8802436 and DCR-8521228.

Contents

Chapter 1

Introduction

Recovering a hierarchical motion description of an image sequence is one way to recognize objects undergoing motion. With many hierarchical representations, higher levels are recovered from lower levels. For example, in VISIONS [Hans78] the hierarchy is built in the following order with each level recovered from the previous level: vertices, segments, regions, surfaces, volumes, objects, and schemas. In the computational analysis of visual motion, an important unsolved research question is how a complete hierarchy of motion description should be represented and how it should be computed. We propose a new motion description hierarchy and claim that coordinated sequences of events such as walking and throwing can be recognized using this hierarchy *before* identifying the objects (e.g., legs and arms) performing the motion. By making this claim we address a common representation issue in computer vision—what is the proper sequence of representations from the image(s) to the final goal? In this thesis we investigate the low and intermediate-level aspects of one possible motion representation sequence from image frames to the final recognition of motions and objects.

Two other examples of the representation sequence issue are illustrated by approaches used for optical flow and texture analysis. Optical flow can be viewed as a correspondence problem and can be performed using two temporally-adjacent frames [Horn81]. When computed in this manner, the sequence of representations consists of the original images followed by an optical flow description. Alternatively, it is possible to compute optical flow after tokens, such as corners, have been identified based on the matching of these

tokens. In this case, the sequence of representations includes the original images, the final optical flow description, and object descriptions. Similarly, many approaches to texture and structure analysis first recover texels and then use them to recover object structure [Stev79]. Once a texel is known, its shape under projection can be computed. Using this information it is possible to recover the surface orientation of a surface containing the texel. Conversely, the structure of a surface can first be recovered or assumed, and then used in recovering the texels [Fu68, Fuku72, Tou74].

The issue of representation sequences in the human visual system has also been addressed in the psychology community. For example, it was once thought that the human visual system performed stereopsis at a high-level, after monocular object recognition took place. Julesz [Jule71] concluded from his studies with random dot stereograms that stereopsis can take place before monocular object recognition. When the human visual system views a single random dot stereogram monocularly, no structure is found in the image. But when stereograms are viewed together and the viewer fuses the images, structure is immediately perceived. Since no structure is perceived monocularly, it is clear that stereopsis is being performed before object recognition.

In addition to the question of when motion descriptions are calculated relative to when objects are recognized, there is another important issue. The question is *what* kinds of representations of motion are inferred? That is, motions can be described at many levels, from low-level, detecting that something in the scene has moved between frames, to high-level, recognition of a coordinated sequence of events. Most previous work has focused on low-level motion analysis. Frame differencing of temporally-adjacent frames is one common method of low-level motion analysis [Jain79b, Jain79a, Jain81, Jain83, Jaya83, Ande85, Lee88a]. Recovering orientation of a surface or the position within the scene of points on an object is commonly referred to as *structure-from-motion*. Structure-from-motion has been performed using correspondences between points [Ullm79] as well as without using point correspondences [Terz88].

Our approach is based on constructing a coordinated hierarchy of not only low-level motion description, but also intermediate-level and high-level motion description. High-level motion analysis, i.e., recognizing a coordinated sequence of events such as walking and throwing, has been formulated previously as a process that follows high-level object

recognition [Hogg83, Akit84, Leun87a, Leun87b]. Only recently have some researchers considered high-level motion analysis as a process that does not depend on complex object descriptions [Godd88b, Godd88a, Godd89, Goul89].

When the objects in the scene are common, complex objects such as people, it is clear what is meant by "object recognition." The object to be recognized is a person and the motion is walking, for example. However, for less abstract scene configurations, a close-up of a surface translating for example, the meaning of "object recognition" is not as well defined. In this case, the most that can be reported is that a surface is translating. The former example is more interesting from the point of view of this thesis, but cases such as the later are not excluded from consideration. When the motion in the scene is low level, such as a surface translating, then we say object recognition is reporting a surface translating. No higher-level description is possible. Throughout this thesis, object and motion recognition is defined as reporting the most abstract object and motion possible. And high-level object and motion refers to abstract configurations and sequences such as people and walking, respectively.

In most previous research, motion analysis consisted primarily of using a low-level motion description to recover rigid structure. Higher-level object recognition was then recovered from this structure. Discrete structure-from-motion [Brus83] algorithms use little more than the temporal disparity of a few points over a short interval leading to a very primitive description consisting of points' movements from one frame to the next. It is possible, however, to use the paths of points (or other features) over longer temporal intervals, and how the points move relative to one another, to recover a much richer hierarchy of motion description.

In this paper a new representation hierarchy consisting of three levels, a low-level, intermediate-level and high-level motion description, will be described. Low-level motion is primarily concerned with making spatiotemporal (ST) changes between image frames explicit. The intermediate-level explicitly represents ST paths and viewer-centered interactions, e.g., occlusion. High-level motion is concerned with recovering and representing coordinated sequences of events. The low and intermediate levels do not make any *a priori* assumptions about the domain or objects. This hierarchy of motion information can then be used to aid in segmentation, recognition, tracking or other object-dependent

tasks.

A significant computational issue for a complete theory of vision is whether to first identify the objects undergoing motion and then recognize the motion or, recognize the motion before identifying the objects. Goddard [Godd88a] noted that there are two ways that motion in a scene can be used. The first paradigm used motion information to first recover the rigid structure of the objects. The recovered structure was then used to index into models [Hogg83, Akit84, Leun87a, Leun87b]. The second paradigm used motion information to index into the models. Gould and Shah [Goul89] and Goddard [Godd88a] have asserted that it is possible to recognize an object and its high-level motion without first recovering its structure. The motion of a few representative points, rather than the low-level structure, was used to recognize the object [Goul89]. This implies high-level models that are rich in temporal information [Godd88a]. For example, the fundamental event in Goddard's system was a change of angular velocity of a joint. A high-level model consists of a sequence of angular velocity changes. A sequence of angular velocity changes recovered from the image sequence was used to recognize, or index into, the high-level motion models. Structure had no role in the indexing. We propose a variation of the second paradigm. Goddard used only low-level motion information to index into high-level motion models. We propose that intermediate-level and high-level motion descriptions should be recovered and used to index into the models.

Johansson [Joha73] showed that people can immediately recognize moving light displays of simple motions such as swinging pendulums and rolling wheels, as well more complex common motions such as walking and cycling. Since no structure exists in a single moving light display image, the human visual system is clearly capable of recovering a high-level motion description from the motion of only a few points. These results suggest that high-level motion description is performed before object recognition.

To test our theory that coordinated sequences of events do not have to be computed *after* a complete static description of the objects has been recovered, we develop in this thesis a new, specific computational theory and implement a system that tests it. We propose, as others have [Boll87, Jain88, Liou89], that an ST volume is the proper low-level representation of the input sequence. Further, we propose that spatiotemporal flow curves, which "flow" through an ST cube and represent short-range and long-range

image motion, is an appropriate intermediate-level representation because it represents the salient properties of image motion.

Given that ST flow curves are the intermediate-level representation, we must also specify what should be computed from the ST flow curves. In this thesis we propose other parts of a complete intermediate-level motion representation, specifically the relative times that rigid parts move, the relative positioning between solid sub-parts, and object interactions such as occlusion, disocclusion, and collision. We argue that this information can be recovered from the ST flow curves alone, and serves to organize and characterize coordinated sequences of events prior to recognizing the semantics of constituent components of motion.

By studying a possible sequence of representations for motion analysis, we enhance our ability to successfully automate the recognition of complex objects in real world scenes. Objects can be characterized and classified according to their dynamic geometric structure, not just static structure. A real world domain is full of deformable and articulated objects, as well as rigid objects, that any general vision system must understand. Motion of non-rigid objects depends on model-specific dynamic transformations that control the temporal motion dependencies of related parts of an object. For example, a multi-linked arm throws by a coordinated sequence of motions of each joint. By studying an alternative representation sequence, rather than a traditional representation sequence of a static model, we increase our ability to work in a real world domain. Note that the MLD results support the view that motion-based recognition is useful for real world objects.

The main result of this thesis is that intermediate-level object motion descriptions can be recovered prior to recognition of the objects undergoing motion. In order to show this result, ST flow curves, a powerful intermediate-level representation, are defined and used. In the remainder of the chapter we describe ST flow curves in more detail. With ST flow curves formally defined, the chapter concludes by describing what should be computed from ST flow curves. A review of related work in psychophysics and computer vision is then presented (Chapter 2). This review addresses work related to the problem of recognizing motion prior to object recognition. Work directly related to our approach, e.g., ST flow curves, is presented at appropriate places throughout the thesis. Chapter

3 shows how to compute ST surface flow, the instantaneous motion of each point in an ST cube, from which the ST flow curves are computed. Chapter 4 then shows how ST flow curves are recovered and used to compute intermediate-level properties such as occlusion and disocclusion in an image sequence. Finally, Chapter 5 shows how to use ST flow curves to compute another intermediate-level representation, namely cyclic motion. This serves to test the plausibility of ST-motion-based understanding and shows the computational power of ST flow curves.

1.1 Spatiotemporal Flow Curves

Temporal coherence is a powerful constraint for understanding visual motion. Just as the gray levels of pixels around a point in an image are not independent, i.e., spatial coherence exists, the temporal neighbors of a point are not independent. If a sequence of images is taken with a short enough time interval between the images, the inter-frame motion will be relatively small. This results in little change in the gray level at a point between frames, i.e., there is temporal coherence. Spatial coherence has been used innumerable times for image understanding. For example, edge points in a gray level image are defined as points where there exists a sudden change in gray level. Points where the gray level varies smoothly, i.e., there exists spatial coherence, are not edge points. Edge operators detect edge points based on this definition, recovering edge points only where spatial coherence does not exist [Cann86]. Temporal coherence, while used in short image sequences for the computation of optical flow [Agga88], ST surface flow [Allm90a] and simplifying the correspondence problem [Grzy89], has not been used previously as a constraint for motion understanding over long image sequences.

In addition to temporal and spatial coherence of intensity values, there is temporal and spatial coherence of motion. Except at depth discontinuities, the optical flow of an image does not vary randomly between pixels, i.e., there is spatial motion coherence. Similarly, the ST surface flow at a point does not change significantly between frames, i.e., there is temporal motion coherence. This thesis shows how temporal motion coherence over long image sequences can be used for motion understanding.

An ST cube, constructed by stacking a sequence of temporally-close images to-

gether, has temporal motion coherence (as well as temporal gray level coherence) and is therefore an ideal structure for studying image sequences using temporal coherence [Allm90a, Jain88, Bake88b]. A first step in understanding the motion in an ST cube is to determine the instantaneous motion of each point in the cube, called the *ST surface flow* (see Chapter 3). The next step is to compute the motion of each point through the entire ST cube, or until the point vanishes. An ST curve through an ST cube such that the tangent at a point on the curve equals the ST surface flow at that point is called an *ST flow curve*. ST flow curves are recovered from the ST surface flow and then curves with similar shape can be clustered together. Each cluster represents a temporally-coherent structure in the ST cube, i.e., structures that result from an object or surface in the image undergoing similar image motion over long intervals of time.

In order to recover 3D shape from a single image, Kanade used the heuristic of nonaccidental regularities [Kana81]:

> "Regularities observable in the picture are not by accident, but are some projection of real regularities."

to recover 3D shape from a single image." Similarly, we assume that similar image motion is the projection of similar scene motion. So a cluster of flow curves also corresponds to an object or surface in the scene undergoing similar scene motion over long intervals of time.

By using ST flow curves, the ST motion over an arbitrarily long image sequence is represented in a concise, well-defined form. Therefore, straightforward methods can be used to study them. For example, using clusters of ST flow curves, separate moving objects in the scene can be hypothesized, and occlusion and disocclusion between them can be identified by examining how clusters merge and split.

Figure 1.1 shows an example of how temporal motion coherence can be used for the interpretation of a long image sequence. Figure 1.1(a) shows a slice from an ST cube with one spatial and one temporal dimension. There are two objects, one accelerating to the right and one translating to the left. The ST surface flow is shown in Figure 1.1(b). As expected, the flow for the left object points toward the right, the flow for the right object points toward the left, and the flow for background points straight into time.

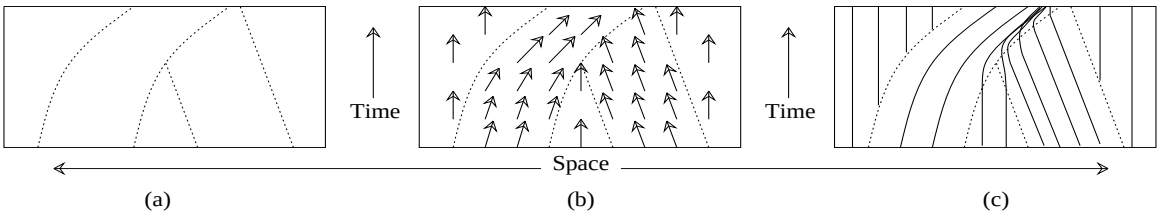


Figure 1.1: (a) A spatiotemporal image with two objects, one accelerating right and one translating left. (b) The ST surface flow. (c) ST flow curves.

Figure 1.1(c) shows the resulting ST flow curves.

The flow curves for the left object all have similar shape, as do the flow curves for the right object. These curves can be clustered using properties of curves that measure the shape of the curve, e.g., curvature and torsion. However, the flow curves for the right object and the background have identical shape, i.e., they are straight. So in addition to examining the shape, for straight flow curves the slope of the curve is also required. A space curve is completely defined, up to a rigid translation and rotation, by its curvature and torsion [DoCa76]. By applying standard clustering techniques using curvature, torsion and slope, three distinct clusters are formed. Each cluster represents the motion of a single object or the background over a long period of time.

As shown in Figure 1.1(c), by definition the flow curves associated with the right object merge into the flow curves associated with the left object as the right object becomes occluded. Over time the clusters associated with the two objects merge to form one cluster. This results because the temporal coherence assumption is violated. Where occlusion events such as this occur, temporal motion coherence does not exist.

Temporal motion coherence can be interpreted by the following two intuitive yet powerful constraints that we call the *temporal uniqueness constraints*. First, a point in an image can only move to at most one point in the next image. Second, a point in an image can come from at most one point in the previous image. In areas where these constraints are not violated, i.e., temporal motion coherence exists, the flow curves will correspond to the motion of a single point, and therefore they can be clustered together to identify surfaces in motion in the scene. Where these constraints are violated, or it appears they are violated, occlusion or disocclusion between surfaces has occurred

resulting in clusters of flow curves splitting and merging (see Figure 1.1(c)).

The temporal uniqueness constraints can be interpreted formally as a mapping from one image to a subsequent image. Let $\mathbf{f}$ map a pixel from one frame to a subsequent frame. That is, $\mathbf{f} : \Re^2 \rightarrow \Re^2$, $\mathbf{f}(p_1) = p_2$, where p_1 and p_2 are pixels in a frame and a subsequent frame, respectively, and are projections of the same point in the scene. The temporal uniqueness constraints say that $\mathbf{f}$ is a bijection, i.e., 1-1 and onto.

Chapter 4 will describe in detail how ST flow curves are recovered and will further justify why ST flow curves are an appropriate intermediate-level representation.

1.2 Intermediate-Level Motion Representations

The hypothesis of this thesis is that it is not necessary to first recover the high-level structure of an object in an image sequence *before* recognizing any high-level motion of that object. As will be pointed out in Chapter 2, psychological models of the human visual system have shown that it is feasible to first recover the rigid parts of an object using motion and then use this rigid structure information to recognize the object [Joha73, Joha76, Bert84, Bert85]. All of the models presented in Section 2.2.2 also recover rigid parts and then use this information to index into the models of objects to be recognized.

Gould and Shah [Goul89] and Goddard [Godd88a] both have argued that it is possible to recognize an object and its high-level motion without first recovering its high-level geometric structure. The motion of a few representative points, rather than the low-level structure, e.g., rigidity, can be used to recognize an object [Goul89]. Their approaches require high-level models that are rich in temporal information [Godd88a]. The fundamental event in one of Goddard's *scenarios* is a change of angular velocity. A scenario becomes active when the correct sequence of these changes occur. In other words, the motions of joints is used to recognize, or index into, the high-level motion models. Geometric structure has no role in the indexing.

In order to index using motion information, obviously our models have to be able to be indexed using motion. As will be shown in Chapter 2, all the high-level structure models are indexed using structure rather than motion. Goddard [Godd88b, Godd88a, Godd89]

presented the only model to date that indexes directly from motion. But Goddard's model uses only angular velocity change of joints as the motion information for indexing.

Our goal is to first recover a high-level motion description starting with an image sequence. The image sequence can be represented by a low-level motion description such as a spatiotemporal volume. We propose that an intermediate-level motion representation is required to bridge the gap between low-level and high-level motion descriptions. We will construct this intermediate level non-purposively, i.e., without depending on the high-level models, analogous to the way Marr developed the primal sketch and $2\frac{1}{2}$ sketch [Marr82].

As in any vision system, the knowledge representation used has a great effect on the way the system performs. A vision system that uses motion information to recognize high-level motion must make three things explicit: 1) relative positioning between the rigid sub-parts, 2) the relative time that parts move, and 3) real and apparent object interaction. In the remainder of this section we argue why these components are necessary. As we discuss each we show how and why ST flow curves are suitable intermediate-level motion representations for recovering the three components.

1.2.1 Relative Spatial Information

The relative spatial locations of rigid parts is clearly important in a motion description because if the rigid parts are not positioned correctly relative to each other, the high-level motion is drastically altered. For example, consider a person with their left forearm undergoing a throwing motion and their right upper arm also undergoing the motion for throwing. Each arm part, considered separately, is not sufficient to determine that the high-level motion is throwing. But if considered together, while ignoring the incorrect relative spatial orientation, the two parts specify a throwing motion. In order to prevent throwing motion from being recognized, relative spatial information should also be recovered.

Cutting showed that relative spatial information is a necessary component of MLDs in order for the HVS to perceive walking [Cutt81a]. He presented subjects with a "walking" MLD, where the initial positions of the lights were altered while keeping their individual

absolute motions correct. None of the subjects detected the walking motion.

Note that relative spatial information is not only a dynamic property but a static property. In any single static image, this property must be met, but we also require it to be true across all frames.

This thesis proposes a computational approach in which high-level motion is computed without first recoving high-level structure. Relative spatial information is structure information, so should not be needed to recover high-level motion. However, we must make it one component of motion analysis to prevent sequences that contain the correct movements, but in incorrect relative positions, from being recognized as some high-level motion. Similar to Kanade's heuristic of nonaccidental regularities, if we assume that nature does not conspire against us by presenting these pathological cases, then the importance of this component is diminished.

To verify that object parts are in the correct relative positions during a temporal interval, we will examine the relative positions of the ST flow curves in the ST cube over that temporal interval.

1.2.2 Relative Temporal Information

The relative times that rigid parts move is of fundamental importance in a motion description since if the rigid parts do not move at the correct relative times, the interpretation of high-level motion is altered. For example, consider a baseball player throwing a ball and person throwing a ball "like a girl." The difference between these two motions is the time that the elbow joint opens. A small delay in the opening of the joint gives two distinct throwing motions.

In the previous section, an experiment involving a spatially anomalous "walking" MLD was described. A similar experiment involving a temporally anomalous walking MLD could be presented. In this case, each light would undergo the correct motion, but at the incorrect time. For example, each light could start its motion at different times. In this experiment it would be expected that subjects would not detect the walking motion, as in the previous experiment.

To detect that some motion is taking place at the correct relative time, one can compare ST flow curves. In the throwing example, one can recover the time that the ST flow curves created by the projection of the forearm start moving away from the ST flow curves created by the projection of the upper arm.

1.2.3 Object-Interaction Motion Features

Just as contours in a single image are indicative of structure in the scene, ST flow curves in an ST cube representing a long image sequence are indicative of object motion in the scene. In order to recognize high-level motions, we must take into account how both apparent and real inter-object interactions, e.g., collision, affect high-level motion and the resulting ST flow curves. For example, to recognize walking one must be able to handle the fact that walking can take place on many different types of terrain. Analogously, in order to recognize a static image of an articulated object, one must model the valid configurations of the object. In our case, until the effects of possible object interactions are modeled, any vision system is doomed to treating the same high-level motion interacting with different objects as separate, unrelated types of high-level motions. For example, walking up a hill and walking on a flat road would not be recognized as equivalent, although they differ only in how the walker interacts with the ground. Similarly, an object moving behind another object should be interpreted as only an apparent object interaction due to occlusion. Even though occlusion and collision, for example, are physically quite different, they are similar in the way they are sensed by the observer. That is, while some interactions are real interactions, e.g., objects colliding, and some are apparent, e.g., one object partially occluding another, all are manifested as viewer-centered motion features in the ST cube.

Rubin and Richards, and Engel [Rubi85, Enge86] proposed that there are four important motion boundaries: force impulses, starts, stops and pauses. These boundaries also describe some real, as opposed to apparent, interactions between objects. For example, when two rigid objects collide there is a force impulse applied to the objects. We propose that there are at least three ways to detect real object interactions. These three ways subsume the four motion boundaries. The three dimensions that an object can change

over time are: (1) an object changes shape (e.g., when a hand squeezes a rubber ball, the shape of the ball is altered), (2) an object changes its speed (e.g., a ball slows down as it rolls along the ground), and (3) an object changes its direction of movement (e.g., a ball changes its direction when a bat hits it). These interactions, and any combination of them, can be described without knowing anything about the objects involved.

We now have four different types of object interaction: one apparent, occlusion, and three real, object shape, speed and direction change. It can now be shown how each of these interactions manifests itself as a motion features. That is, each interaction is detectable by examining the shape and location of ST flow curves.

Figure 1.1 showed flow curves which result from one object moving in front of another. In this case flow curves associated from an object becoming occluded will "flow" into flow curves associated with the occluding object. The motion feature is the merging of the flow curves.

If an object changes its shape such that the shape change is preserved under projection, the spatial extent of the group of flow curves associated with the object will expand, shrink, or generally deform as the projection of the object changes. Note that an object moving in depth will also cause its projected group of flow curves to shrink or expand.

To detect a speed change of an object, we detect a change in the *slant* of a flow curve. The slant of an ST flow curve at a time is defined as the angle between the tangent of the flow curve and a vector parallel to the time axis. If an object changes its speed such that the projection of the object also changes its speed, then the slant of the flow curve will change at that time. If the speed increases, the slant increases. If the speed decreases, the slant decreases.

Finally, ST flow curves indicate the direction of movement of the projection of an object. If the tangent direction of a flow curve changes, the object must have changed its direction of movement in the image.

We have shown how each of the four types of object interactions appear as motion features, features that are detectable by examining ST flow curves. It was also shown how the other intermediate-level properties, relative spatial and relative temporal information, are detectable using ST flow curves. Object interaction, in particular occlusion, will be

examined in detail in this thesis. Relative spatial and relative temporal information are discussed in the final chapter.

Chapter 2

Related Work

Related work comes from both the psychological and the computer vision communities. In psychology, the work is primarily focused around *moving light displays* (MLD). Previous work on computational visual motion can be classified into four categories according to the type of information recovered from an image sequence. The four categories are: low-level structure, high-level structure, low-level motion and high-level motion. After examining the related work, it becomes apparent that an intermediate-level motion representation is lacking in the literature.

2.1 Related Work in Psychology

Clearly, there exists a large amount of psychological research that addresses motion perception in the human visual system (HVS). However, in this chapter we are specifically concerned with the use of motion to recognize an object or to recognize the motion of an object. We are not necessarily addressing the issue of perceptual organization of motion. That is, we are not primarily reviewing psychological models of low-level motion. After briefly describing experiments using MLD-like stimuli of low-level motion, we will review the MLD literature and show that an intermediate-level motion representation, namely relative temporal information, is lacking.

Lettvin, et al. [Lett59, Lett61], and Henn and Grüsser [Henn69] developed a mo-

tion detector such that proximate, similarly moving points are conceptually grouped. Bosche [Bosc67] produced images of random dots where some dots moved to the left and others moved to the right. This was perceived as rigid, transparent motion. Similarly, Ullman [Ullm79] presented two dotted, transparent cylinders, one inside the other, rotating in opposite directions. The correct perception of two counter-rotating cylinders was perceived.

Chapter 1 described high-level motion as abstract sequences such as walking. The experiments just presented are examples of structure-from-motion and not recovery of high-level motion, for there is no high-level motion present. Fortunately, most of the work dealing with MLDs does address the HVS's ability of motion-based recognition of high-level objects and motion. This results because most MLD experiments are of complex, real world objects, people for example.

Chapter 1 showed why relative temporal information is important in a model of motion-based recognition. However, it will be clear that none of the psychological models contain an explicit representation of time.

Johansson [Joha73] used MLDs consisting of ten lights to examine human performance at interpreting articulated objects. MLD's were used so that only point motion information at the joints was available. In the first demonstration adults watched an MLD of human walking, as viewed from the side of the walker. Johansson found that humans can almost instantaneously recognize the MLD's of walking. The second demonstration used moving light displays that were made with the walker moving toward the camera in directions between 80 and 45 degrees to the camera. The results with this display were similar to the results in the first demonstration. Johansson also tested displays for running, cycling, climbing, dancing in couples, various types of gymnastic motion, and others. In all cases adults spontaneously and correctly identified the motion.

Runeson [Rune81] made MLD's that had points of light not only on the joints of a person but also at the corners of a box that the person was lifting. Observers of this MLD were able to judge the weight of the box.

Other studies [Cutt81b, MacA83] found evidence that adults are capable of recognizing friends and the gender of a person from only the joint motions in MLD's.

There have been numerous psychological models to explain how the HVS is able to correctly interpret MLD's. Bertenthal [Bert85] presented three properties that must be perceived in order to perceive a moving light display as biomechanical motion. The three processes are: 1) connectivity: the lights are connected into arms, legs, etc.; 2) depth order: the lights are ordered in depth; 3) implicit contour: an implicit contour consistent with the edges of the human form is perceived.

Bertenthal, Proffitt and Cutting [Bert84] presented a model that consists of five levels: individual motions, organization of figural coherence, biomechanically appropriate motions, causal (kinetic) relations, and identification of human form. The individual motions level refers to the perception of the absolute motion of a single moving light. Consider, for example, a point on the ankle of a walking person. The complicated absolute motion caused by the combination of swinging and translation of the ankle is perceived. The relative motion of swinging relative to the hip, is not perceived. The organization of the figural coherence level recovers relative motions. For example, the ankle of a walking person is seen as a simple pendulum motion relative to the hip. At the third level, observers respond to motions that are biomechanically appropriate. For example, the upper and lower body components are perceived as having pendular motions. It is assumed that it is known by the perceiver that pendulum motion is a biomechanically-appropriate motion. For human motion, nested pendulum motions comprise biomechanically-appropriate motions. At the causal (kinetic) relations level, observers see motion as having a causal role. For example, motions in one part are seen as causing certain motion in other parts. At the final level, the identification of human form, observers perceive a human walking.

Johansson [Joha73] proposed three principles that may be used in order to perceive correctly MLDs of biomechanical motion. The three principles state: 1) the HVS groups moving points together, 2) moving points that keep the same relative positions are grouped into rigid parts, and 3) motions of multiple rigid parts are organized into a single common motion, such as walking, if possible. Later, Johansson [Joha76] presented an additional theory using a hierarchy of coordinate systems to explain this human performance.

None of these psychological theories contains an explicit temporal description of motion. Since the HVS requires the lights of an MLD to move at the correct relative times,

we would expect that the relative motions of rigid parts are central to a computational model. For example, if the limbs of a person swing at incorrect times, walking motion should not be perceived. These models explain the HVS's recovery of structure, but not how it uses relative temporal information for recognizing complex biomechanical motion such as walking.

The lack of an explicit representation of time in the recovery of walking motion and other types of abstract motion is common within the computer vision literature as well. A survey of related work in computational vision is presented in the next section. We will return to the psychological study of articulated objects when we examine the specific problem of recovering cyclic motion in Chapter 5.

2.2 Related Work in Computational Vision

Previous work on computational visual motion can be classified into four categories according to the type of information recovered from an image sequence. The four categories are: low-level structure, high-level structure, low-level motion and high-level motion. Low-level structure consists primarily of work that recovers rigid parts from image sequences; *structure-from-motion*, for example. High-level structure deals with recovering and representing non-rigid objects. Given some object in an image sequence, all these representations are best suited to match the scene structure with a model rather than match the image or scene motion with a model. On the other hand low-level motion is primarily concerned with combining image sequences so that viewer-centered changes between frames due to motion in the scene is made explicit; optical flow, for example. Work on high-level motion is concerned with using lower-level motion descriptions in order to recover a coordinated sequence of events that can then be used to index into models of high-level motion [Godd89].

Figure 2.1 shows a characterization of all work in computational vision. We can examine how the characterization of low-level and high-level structure and low-level and high-level motion fit into this characterization. The top node in Figure 2.1 represents all computational vision work. This work can be divided into that which deals with a single image, *Static*, and multiple images, *Dynamic*. The dynamic work typically com-

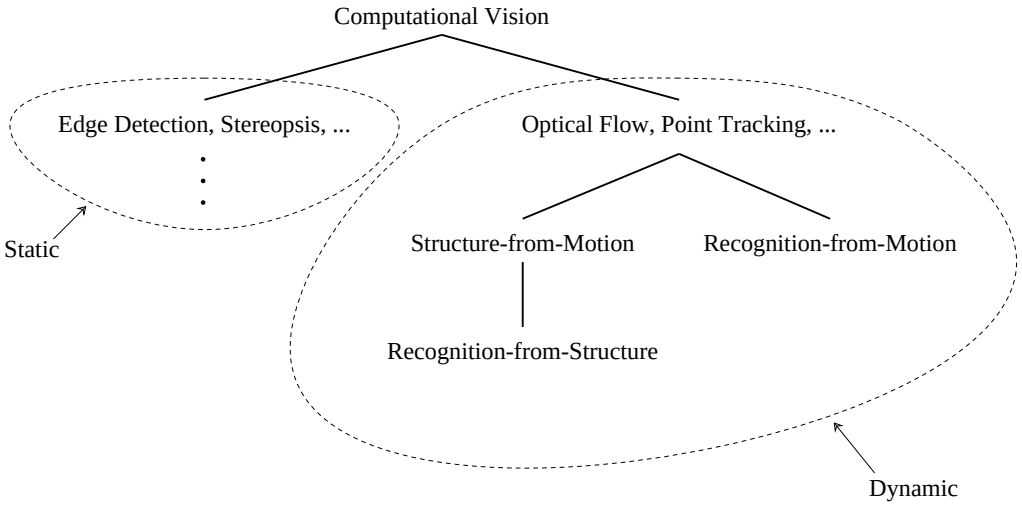


Figure 2.1: Characterization of computational vision research.

putes structure-from-motion and then recognizes objects using the recovered structure. This thesis puts forth the hypothesis that the structure-from-motion, recognition-from-structure path is not the only possibility; recognition-from-motion is also possible.

Our four categories are all on the dynamic side of Figure 2.1 since they deal with image sequences. Low-level structure consists of work that recovers rigid parts so it fits into the structure-from-motion node. High-level motion is concerned with representing non-rigid objects, e.g., people, and how they change over time while performing some motion, e.g., walking. However, it will be shown that all these models use structure, rather than motion, to recognize the object. Therefore, this work fits into the recognition-from-structure node. Low-level motion is concerned with making changes between frames explicit, optical flow for example. High-level motion work fits into the recognition-from-motion node since it is concerned with using motion for recognition.

Very little previous work has addressed the issue of computing high-level motion descriptions prior to recognizing objects. Most of the work that might be considered to address this issue is actually recovering high-level structure and is not trying to recognize the motion before recognizing objects. For example, questions such as the following are not answered: "How many objects are in the scene?" "Where are the objects?" "What

is the background?" "Where is (dis)occlusion taking place?" Only the high-level motion work is of direct relevance to this thesis. However, it is worth examining work in the other categories to see how and why high-level motion was not used for object recognition.

It is worth noting that there is currently no node between [Optical Flow, Point Tracking, ...] and recognition-from-motion. An intermediate-level motion node, defined in Chapter 1 as relative spatial and temporal changes and motion features, does not currently exist. But ST flow curves will provide a means to compute it.

2.2.1 Low-Level Structure

The *rigidity assumption* is frequently used as a heuristic for recognizing rigid parts. This assumption says that if a group of pixels, or some other low-level feature, can be interpreted as points on a rigid object, then assume that the pixels are in fact points on a rigid object.

Rashid [Rash80] used this assumption to recover rigid parts from an MLD. The correspondence of the points between frames was recovered and the 2D velocity of each point calculated. Every point in the MLD was then represented by the four-vector (x, y, v_x, v_y) where x and y specify the point's position in the image, and v_x and v_y its 2D velocity. A graph was then constructed with each four-vector represented as a node. Each node was connected to every other node and the Euclidean distance between nodes calculated. Next, the minimal spanning tree was found. The resulting tree appears as clusters of edges with the clusters connected by long edges. These long edges are *cut* and the resulting clusters represent all distinct, rigid parts or objects.

Webb and Aggarwal [Webb82] used a *fixed-axis assumption* that states: the motion of every rigid object consists of a translation plus a rotation about an axis that is fixed in direction for short periods of time. Any object satisfying the fixed-axis assumption, also satisfies the following property: the motion of a point relative to any other point on a rigid object traces out circles in planes normal to the fixed axis. Webb and Aggarwal used this property to recover the structure of rigid objects as well as the rigid parts of jointed objects.

Leung and Yang [Leun87a, Leun87b] developed a system for human body motion analysis. They used a model of the human body and divided the problem into three phases. The first phase separated moving body parts from the background. The second phase labelled the body parts. The third phased assigned motion verbs to the movement. Phases one and two, which correspond to recovering low-level structure, were addressed by the authors but the third phase, which is high-level motion recognition, was left as future work.

Most previous work on structure-from-motion [Ullm79, Webb82, Spet87, Brus83] can be classified as detecting low-level structure. That is, structure-from-motion work is usually concerned with finding the structure of rigid parts and not the composition of rigid parts that forms more complex objects.

2.2.2 High-Level Structure

High-level structure analysis is concerned with how to construct high-level structures of non-rigid objects. The most common high-level structure representations [Marr80, Hogg83] have complex objects stored as a hierarchy of rigid parts with constraints between the parts. Many representations also store how the parts move while performing some high-level motion such as walking [Marr80, Hogg83].

Marr and Vaina [Marr80] extended Marr and Nishihara's model [Marr78] so that the representation includes how rigid parts move in relation to each other. The new, augmented representation is object-centered as in Marr and Nishihara's original model [Marr78]. Also, the representation retains a hierarchical form by allowing the specification of how parts move throughout the hierarchy. Marr and Vaina [Marr80] also developed a *State-Motion-State* (SMS) representation to describe how articulated objects move over time. In the SMS representation a behavior is described as a sequence of *motion segments*. A *motion segment* of an object M consists of the interval between two adjacent, overall-rest states of M.

Hogg [Hogg83] used a model similar to Marr and Vaina's [Marr80] to recognize a walking person. Hogg's model of a human is hierarchical with constraints specifying how the parts can move relative to one another. High-level motions such as walking

were stored as sequences of instantaneous descriptions of the 3D structure of the human model. The system searched for a frame that best matched the human model. Once this was done, frames before and after this matched frame were matched to the model. This was done knowing that the human in the scene was performing the high-level motion of walking. The system successfully recognized and tracked a human in the image sequence. The system, however, had only one object model, a human, and walking is the only high-level motion that the system could track.

Akita [Akit84] also worked with a model of a human. His model was similar to Marr and Vaina's [Marr80] in that it used generalized cylinders to represent solid parts and it stored relations between the cylinders to represent the joints. High-level motions such as walking and throwing were represented as sequences of key frames. Each key frame represented a different phase of body posture from the point of view of occlusion. In other words, a key frame was stored for every topologically-distinct view of the body in motion. As in Hogg's work [Hogg83], Akita matched one frame of the sequence to the human model and then used the key frames to predict where occlusion would occur in other frames.

All the computational models presented in this section are more spatially based than temporally based. That is, they make spatial aspects of the model more explicit than temporal aspects. Given some object in an image sequence, all these models are better suited to match the structure of the object with a model rather than match the motion of the object with a model. This is demonstrated by the fact that a model is matched to one frame of a sequence [Hogg83, Akit84] before the motion of the object is used. It is because of this spatial over temporal emphasis that these models are classified as addressing high-level structure analysis rather than high-level motion.

2.2.3 Low-Level Motion

Liou and Jain [Liou89] presented past work on spatiotemporal representations that included a survey of filter-type spatiotemporal models. Typically, spatiotemporal filters work in the frequency domain and are sensitive to "motion energy" that is present in the data. The essential concepts of filter models are described by Liou and Jain, and Watson

and Ahumada [Liou89, Wats85]. Let $c(x, y, t)$ define the gray level at each point x, y, t over some interval. The Fourier transform is denoted by $\tilde{c}(u, v, w)$. With translation r_x and r_y, the transform becomes:

$$c(x - r_x t, y - r_y t, t) \quad \rightarrow_3 \quad \tilde{c}(u, v, w + r_x u + r_y v)$$

where $\rightarrow_3$ indicates the 3D Fourier transform. The psychophysical work includes that of Watson, Barlow, VanSanten, Fleet and Heeger [Heeg87, Wats85, Barl65, VanS85, Adel85, Flee84a, Flee84b]. Computational models include those developed by Marr, Buxton, and Liou [Marr81, Buxt83, Buxt84, Liou89].

According to Liou and Jain [Liou89], filter models suffer from several fundamental problems. First, most filter models assume constant velocity. Second, filter models suffer from the scale problem, i.e., they have problems separating events at different scales. Third, filter models are suitable when motion is in a steady state, i.e., the motion of objects does not change abruptly. They perform poorly if motion is not in a steady state.

Much of the temporal examination of image sequences consists of looking at only a few frames at a time. Frame differencing has been a popular method of temporal examination of two frames [Jain79b, Jain79a, Jain81, Jain83, Lee88b, Jaya83, Ande85]. With this method, the difference of two temporally-adjacent frames is calculated and where the difference is nonzero, there has been motion. Lee and Anderson [Lee88b, Ande85] constructed spatial pyramids from difference images in order to track objects. But this spatial pyramid must be rebuilt for every pair of frames. Many optical flow [Horn81] and structure-from-motion [Ullm79] algorithms are similar in that they also use only a few frames to accomplish their goal.

Recently, there has been a trend toward using more than a few frames to examine an image sequence [Jain88, Aloi88]. Jain notes: "Most techniques try to do too much using too few frames. In fact, there has been misplaced emphasis on solving problems, like structure from motion, using minimal information in a minimal number of frames."

This trend away from using only a few frames has lead to the use of spatiotemporal volumes. Spatiotemporal volumes are three-dimensional structures that are built by "stacking" a sequence of densely-sampled image frames. The spatiotemporal volume is a

viewer-centered, three-dimensional (x-y-time) description. If we sample densely enough in time and an edge operator is applied [Boll87], the spatiotemporal volume will contain surfaces and volumes created by the surfaces. These surfaces and volumes represent object motion swept out through time.

Liou and Jain [Liou89] considered using spatiotemporal filters but, because of the drawbacks, opted for a variation of the spatiotemporal surface model just presented. Their model used a hypersurface in a 4D volume. The four dimensions were: two spatial, one temporal, and another for the monge patch defined by the grey levels in each neighborhood of each space-time point. This hypersurface in 4D is the analog of the surface in 3D that is created by using the gray levels to define the "height" of the surface. They then defined a motion detector using these hypersurfaces.

Most work on spatiotemporal surfaces has used a 3D volume rather than the 4D space of Liou and Jain. The fourth dimension is eliminated by storing the gray level at each space-time point. Aloimonos [Aloi88] introduced *flow lines* to represent the trajectories of point features through the spatiotemporal volume. Peng and Medioni [Peng89] worked with 2D *slices* of a spatiotemporal volume. For every edge point in the first frame, four slices, centered at the point, were taken. Each slice had one spatial and one temporal dimension. Assuming objects are only translating, these slices contained lines that indicated the speed of movement of the point where the slice was centered. The greater the slope of the line in the slice, the slower the point was moving in the image. The slopes of the lines in each group of four slices were examined and the normal component of velocity of the point was computed.

Engel and Rubin [Enge86] presented a system that detects visual motion boundaries. This was an implementation of Rubin and Richards earlier work [Rubi85]. Rubin and Richards argued that force impulses, starts, stops and pauses, are important temporal events. Starts, stops, and pauses were detected by examining the velocity of a moving point. Assuming mass remains constant, a force impulse, or force discontinuity, was detected by examining the acceleration of a moving point.

Gould and Shah [Goul89] used motion information to categorize the motion of objects. They defined a trajectory primal sketch as a low-level representation for the movement of points. Using scale space, important events were detected and primitive movements

or trajectories, including translation, rotation, projectile and cycloid were described.

A different approach was taken by Terzopoulos, Witkin and Kass [Terz88]. They built a dynamic 3D model from image data. These models are dynamic since they change to best match the image data. A model can be conceptualized as a deformable 3D space curve spine with a deformable sheet rolled around the spine. The spine is aligned with the medial axis of the projection of an object in the image. The sheet is then deformed to best match the silhouette of the object in the image. Since the sheet can only match the silhouette of the object, it is assumed that objects are symmetric about their medial axis. At this point the system recovered a generalized-cylinder-like model. But it is unclear how this 3D model might be used to recognize the object. It is also unclear how changes of the model over time can be described. But since the spine and sheet deform as the image data changes, the model describes the image motion and deformations of the object over time.

2.2.4 High-Level Motion

Low-level motion is concerned with recovering relatively simple descriptions of the input sequence so that changes due to movement between frames are made explicit. High-level motion is concerned with using these low-level motion descriptions in order to recover a coordinated sequence of events that can then be used to index into models of high-level motion.

Using spatiotemporal surfaces, Bolles and Baker [Boll87, Bake88b] examined how to recover camera motion. Baker [Bake88a] showed how to construct these spatiotemporal surfaces. Most of this work has concentrated on recovering camera motion rather than a description of high-level motion in the scene.

Badler and Smoliar [Badl79] presented a survey of representations of human movement. They reviewed: 1) human movement notations which include Labanotation and Eshkol-Wachmann notation, 2) representations of the human body which include stick figures, surface models and volume models, and 3) representations of movement which include key frames, movement functions, goal-directed behavior, constraints and simulation. Badler and Smoliar concluded that animation by simulation is preferable because

simulation systems model the interactions between movements in an environment. It will be shown in the next section how object interactions, e.g., collisions, can be a useful intermediate-level motion description.

O'Rourke and Badler [O'Ro80] used constraints and a model of the human body to analyze human walking. The model is the same as the one developed by Badler and Toltzis [Badl79]. The model consists of overlapping spheres and the constraints arise from two sources: 1) the structure of the human body and 2) properties of the function that form images by projection of the model. The system was given the high-level motion being performed and then predicted and verified where the body parts would appear. In this respect this work is very similar to Hogg's work [Hogg83].

Goddard [Godd88b, Godd88a, Godd89] developed a general paradigm for recognizing high-level motion. He assumed that the scene contains an articulated stick figure moving parallel to the image plane. It is further assumed that the imaging system tracks the center of rotation of the trunk and that limbs rotate around an end of the trunk or around an end of another part. He used a connectionist network with units that were responsive to angular velocity changes of the joints. A coordinated sequence of events, or *scenario* as Goddard refers to it, consists of a sequence of changes of angular velocity of the joints. A scenario becomes active if the correct angular velocity changes are detected in the correct order.

Very little previous work has addressed the issue of computing high-level motion descriptions prior to recognizing objects. Most of the work that might be considered to address this issue is actually recovering high-level structure and is not trying to recognize the motion before recognizing objects. For example, questions such as the following are not answered: "How many objects are in the scene?" "Where are the objects?" "What is the background?" "Where is (dis)occlusion taking place?" In the next chapter, our approach to low-level motion analysis using ST surface flow is defined, followed by our approach to intermediate-level motion analysis using ST flow curves. ST flow curves are defined such that they are equivalent to the paths of lights in Johansson's MLDs. By doing this we expect that ST flow curves represent the salient properties of image motion.

Chapter 3

Spatiotemporal Surface Flow

Image motion over an arbitrarily long sequence generates a spatiotemporal (ST) cube of image data. Since many of the motions we are concerned with only become apparent over long sequences, an ST cube is the basic structure for analysis. A first step in understanding the motion in an ST cube is to determine the instantaneous motion of each point in the cube, called the *ST surface flow*.

Rather than use the partial derivatives of the surface as done in most gradient-based optical flow methods, the gradient of a function on the ST surface is used. It is observed that arc length of a contour does not change if that contour is moved in the direction of motion on the surface. Motivated by this observation, a function measuring arc length change is defined. The direction of motion of a contour undergoing motion parallel to the image plane is shown to be perpendicular to the gradient of this function. This gradient approximates the direction of motion when object motion in the scene is not parallel to the image plane or when perspective projection is used.

Motivation of ST surface flow is presented in the next section. Related work in differential geometry and ST surfaces is presented in Section 3.2. Section 3.3 shows that the direction of motion is perpendicular to the gradient of the arc length change function. This is shown using ST 2-surfaces composed from edge points and is then extended to ST 3-surfaces from a sequence of gray level images. Results are presented in Section 3.4. Finally, the relationship between our approach to computing ST surface flow and

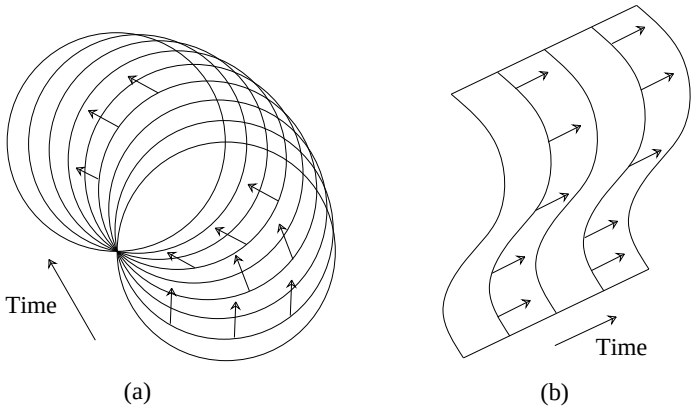


Figure 3.1: The spatiotemporal surfaces resulting from a circle rotating about a point on the circle (a) and a contour translating (b). The arrows show the local structure of the surfaces.

the gradient-based optical flow approach is discussed in Section 3.5. The behavior of our algorithm in cases where the aperture problem exists is also discussed in Section 3.5.

3.1 Innovations Over Traditional Flow

The structure of ST surfaces reveals a great deal of information about contours in a scene and their motion. As contours move, their projection into the image also moves. Over time, these projections sweep out spatiotemporal surfaces and volumes created by the surfaces. Thus, these surfaces and volumes are direct representations of object motion. If the images are edge maps, ST surfaces are 2-surfaces of edge points in a three-dimensional volume. Alternatively, gray level images can be used resulting in a four dimensional volume (x-y-time-gray level) containing 3-surfaces. In both cases, the local structure of the surfaces gives information about contours and the contours' motion.

The interpretation of ST surfaces must begin with the recovery of the motion that generated the surface. Figure 3.1a shows an ST surface formed under orthograhic projection of a circle rotating about a point on the circle. At any point on the surface there is a "twisting" shape as a result of the circular motion. The arrows on the surface indicate the motion of the contour at a point in space-time. If a contour is translating under

orthographic projection, as in Figure 3.1b, the direction of motion remains the same over time and a cylindrical ST surface results. The arrows show the *flow*, or local motion, of points on the contour that generated the ST surface. The flow at a point on an ST surface is called the *spatiotemporal surface flow*.

ST surfaces represent low-level motion, the motion of a point, as well as high-level motion, the motion of objects and camera motion. This can also be viewed as representing local and global motion. The motion of a point is an example of local motion. The motion of an object is more global but is limited to a region in an image. Camera motion manifests itself in ST surfaces by all points on all ST surfaces flowing in accordance with the camera motion. In this paper we are concerned with using local methods to recover low-level motion of points on ST surfaces, i.e., the ST surface flow. ST surface flow is a first, low-level step to motion understanding. The flow of an ST surface can be examined spatially and temporally about a point in order to recover information about the contours that generated the ST surface and how the flow changes over time. By combining this information over time and space we can begin to understand the global structure of ST surfaces and the motion they represent.

The examples in Figure 3.1 show that the structure of ST surfaces is a result of the motion of contours in time. One approach to quantifying this structure would be to use partial derivatives to examine how the surface changes. This is the approach taken in many optical flow algorithms [Agga88]. If $E(x,\ y,\ t)$ is the image intensity at point $(x,\ y)$ in the image at time t, then assuming that the intensity of a point does not change as it moves, we have

$$E(x,\ y,\ t) \quad = \quad E(x\ +\ \delta x,\ y\ +\ \delta y,\ t\ +\ \delta t) \tag{3.1}$$

Approximating the right-hand side by a truncated Taylor series about the point $(x,\ y,\ t)$ we obtain

$$E(x\ +\ \delta x,\ y\ +\ \delta y,\ t\ +\ \delta t) \quad = \quad E(x,\ y,\ t) \quad + \quad \delta x\frac{\partial E}{\partial x} \quad + \quad \delta y\frac{\partial E}{\partial y} \quad + \quad \delta t\frac{\partial E}{\partial t} \tag{3.2}$$

Substituting Eq. (3.1) in Eq. (3.2) and dividing through by δt we have

$$\frac{\delta x}{\delta t}\frac{\partial E}{\partial x} \quad + \quad \frac{\delta y}{\delta t}\frac{\partial E}{\partial y} \quad + \quad \frac{\partial E}{\partial t} \quad = \quad 0$$

In the limit as $\delta \rightarrow 0$

$$\frac{\mathrm{d}x}{\mathrm{d}t}\frac{\partial E}{\partial x} + \frac{\mathrm{d}y}{\mathrm{d}t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0 \tag{3.3}$$

This is known as the optical flow constraint equation and, along with additional constraints [Agga88], has been widely used to compute optical flow.

The optical flow constraint equation contains two unknowns, $\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$, and is therefore insufficient to uniquely determine the motion. The system is under constrained since δE is affected by motion along both axes. Motion along the x axis affects $\frac{\partial E}{\partial y}$ and motion along the y axis affects $\frac{\partial E}{\partial x}$. If this were not the case, as assumed by Limb and Murphy [Limb75], the motion could be solved using just the simplified constraint equations

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial E/\partial t}{\partial E/\partial x} \qquad\qquad \frac{\mathrm{d}y}{\mathrm{d}t} = \frac{\partial E/\partial t}{\partial E/\partial y}$$

Not only do the changes of the ST surface along x and y affect each other, but they do not have the uniformity we desire. For example, with the origin defined as the center of the image, consider an ST surface defined by $f(x,\ y,\ t) = (x+t)^2 + y^2$ such that each image has a paraboloid gray level structure and this structure translates along the x axis linearly with t. Since this surface results from a constant motion of a rigid gray level structure, the motion at every point in the image is the same. But the partial derivatives, $f_x(x,\ y,\ t) = 2(x+t)$, $f_t(x,\ y,\ t) = 2(x+t)$ and $f_y = 2y$, vary depending upon the point on the surface. Even though the surface was generated by a rigid structure undergoing constant translational motion, the *shape* of the surface, at least shape described by its partial derivatives, does not give the information necessary to recover the motion.

Rather than examining the shape of ST surfaces directly, we propose using a measure that does not vary with the parameters that define the surface. We use the gradient direction of a function on the surface rather than, say, the partial derivatives or gradient of the ST surface. This function measures the rate of change of the length of a contour as the contour moves along the surface. The directional derivative of this function gives information about the ST surface. Specifically, the direction of motion for a point is perpendicular to the gradient direction on the surface of this function.

Optical flow is equivalent to a temporal slice of the ST surface flow projected into the spatial plane. ST surfaces extend arbitrarily far back in time, so while any particular

time slice gives the optical flow, there is a temporal coherence to the ST surface flow that does not exist when optical flow is independently and repeatedly computed at successive times. A complete description of ST surfaces includes global as well as local information. That is, not only is the flow at each point needed, but higher level descriptions such as long-range periodic motion are also desired [Allm91, Allm90b]. In order to recover more global descriptions of ST surfaces, temporal as well as spatial coherence is needed. Optical flow uses spatial coherence, but the traditional approach which solves for the flow field at one instant of time, does not maintain temporal coherence. By recovering the ST surface flow of each point on an ST surface we can begin to understand the global structure of the ST surface and the physical behavior of the generating contours.

3.2 Related Work

Related work falls into two categories: related work with differential geometry and surface descriptors, and related work with ST surfaces. The former is concerned with using ideas from differential geometry to describe different types of surfaces. For example, if a range image is used to define a surface where the value at each pixel is the "height" of the surface, the curvature of the surface can be used for segmenting the image. There has been relatively little work using differential geometry for ST surface interpretation.

There have been numerous suggestions as to which measures are appropriate to describe a surface. Medioni and Navatia [Medi84] suggested using Gaussian curvature and the maximum principle curvature. Brady *et al.* [Brad85] used principle curvature directions and lines of curvature to describe surfaces. Besl and Jain [Besl86] suggested using Gaussian and mean curvature together as surface descriptors. Haralik [Hara83] *et al.* proposed the topographic primal sketch which labels each pixel in a range image as one of ten possible topographic labels. Gradients, Hessians, and first and second directional derivatives were used.

Since we are interested in the structure of ST surfaces and the structure is a result of the motion of contours, a measure that gives direction of motion is desirable. Many of these measures, with the exception of lines of curvature and directional derivatives, are directionless. Lines of curvature of an ST surface give little information about the contour

or the motion of the contour that generated the ST surface. In the restricted case where the motion is translational with constant speed, the lines of curvature associated with the minimum principle curvature correspond to the direction of motion. But this case is too restrictive to be of general interest. It will be shown that directional derivatives can be useful; not the directional derivative of an ST surface, rather the directional derivative of a function on an ST surface. Similar to lines of curvature, the directional derivative of an ST surface gives incomplete information about a contour and its motion except in very restrictive cases.

Goldgof, Lee and Huang [Gold88] used a curvature-based approach to analyze the motion of nonrigid surfaces. But their approach works with the actual 3D structure of the surfaces, i.e., the depth of each point of the surface has to be recovered *a priori*. We are concerned with recovering an interpretation of ST surfaces which can give information about motion in the scene without first recovering depth information.

Baker and Bolles [Boll87, Bake89] examined ST surfaces in order to recover camera motion but objects in the scene were stationary.

Jain [Jain88] described a spatiotemporal volume segmentation approach using the signs of the three principle curvatures of the ST 3-surface. While using curvature values is suitable for segmenting an ST volume, it does not give complete information about the structure, i.e., direction of motion, of the ST surfaces in the volume.

Peng and Medioni [Peng89] presented an algorithm which takes oriented slices of an ST volume. The features in the slices are then examined to recover the normal component of motion. Since only slices of the ST volume are used, there is no coherent treatment of the volume. Liou and Jain [Liou90] defined a motion detector based on the angle between the gradient direction in an ST volume and the time axis. Both these methods suffer from being able to detect only the normal component of motion even when the true motion is recoverable. Peng and Medioni addressed this problem with a second level of processing. For Liou and Jain this was not as great a problem since the goal was motion detection. Recovering only the normal component is sufficient if one is only interested in detecting where there was motion in an image sequence.

3.3 Local Spatiotemporal Surface Interpretation

In this section we show how to recover the instantaneous direction of motion of a point on the generating contour of an ST surface. We use a parameterization of the surface called the model parameterization which requires the motion of the generating contour. Since this parameterization assumes knowledge of the motion, we also show how the surface can be reparameterized as a Monge patch such that the results for the model parameterization still hold and this reparameterization is computable without knowing the motion.

3.3.1 Differential Geometry Preliminaries

Before presenting our results, we briefly review necessary definitions from differential geometry. See [DoCa76], for example, for more details. Subscripts on mappings indicate the partial derivative with respect to the subscript. $< \mathbf{a}, \mathbf{b} >_p$ indicates the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$ evaluated at the point p. The inner product is also indicated by $\mathbf{a}\cdot\mathbf{b}$. Throughout this thesis curves and surfaces are described parametrically rather than implicitly. For example, a parameterized circle is given by

$$\alpha(l) \;=\; (\cos(l),\ \sin(l)) \qquad\qquad \alpha:\ (0, 2\pi) \;\rightarrow\; \Re^2$$

Formally, a parameterized curve in $\Re^3$ is a map α: I $\rightarrow \Re^3$ of an open interval I = (a, b) of the real line $\Re$ into $\Re^3$.

Similarly, parameterized 2-surfaces and 3-surfaces are defined as, respectively, (see Figure 3.2):

$$\mathbf{X} : U \subset \Re^2 \;\rightarrow\; \Re^3 \qquad\qquad \mathbf{X} : U \subset \Re^3 \;\rightarrow\; \Re^4$$

For example,

$$\mathbf{X}(u,\ v) \;=\; (u,\ \sqrt{1 \;-\; (u^2 \;+\; v^2)},\ v)$$

parameterizes the top half of a sphere.

The tangent plane at a point p on a surface S is denoted by $T_p(S)$. For a 3-surface this plane is a hyperplane.

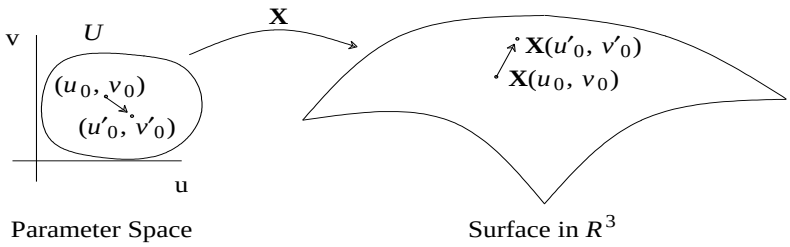


Figure 3.2: The definition of a surface. $\mathbf{X}(u', v')$ shows the change of the point $\mathbf{X}(p)$ on the surface for the change (u', v') of p in parameter space.

For any parameterized surface, the first fundamental form measures the movement of a point on the surface given a movement of the point in parameter space. For example, let (u_0, v_0) be a point in parameter space and $\mathbf{X}(u_0, v_0)$ be the point mapped to the surface. As u_0 and v_0 change, the first fundamental form measures how $\mathbf{X}(u_0, v_0)$ changes (see Figure 3.2). We express the first fundamental form in the basis $\{\mathbf{X}_u, \mathbf{X}_v\}$ associated with parameterization $\mathbf{X}(u, v)$ at p. For movement of (u', v') in parameter space, the first fundamental form at point p is

$$I_p \;=\; E\,(u')^2 \;+\; 2\,F\,u'\,v' \;+\; G\,(v')^2 \tag{3.4}$$

where

$$E \;=\; <\mathbf{X}_u, \mathbf{X}_u> \qquad F \;=\; <\mathbf{X}_u, \mathbf{X}_v> \qquad G \;=\; <\mathbf{X}_v, \mathbf{X}_v> \tag{3.5}$$

For example, let $\mathbf{X}(u, v) = (cos(u), sin(u), v)$. Then

$$\mathbf{X}_u \;=\; (-\sin(u),\ \cos(u),\ 0) \qquad \mathbf{X}_v \;=\; (0,\ 0,\ 1)$$

and

$$E \;=\; 1 \qquad F \;=\; 0 \qquad G \;=\; 1 \tag{3.6}$$

Substituting Eq. (3.6) in Eq. (3.4) gives

$$I_p \;=\; (u')^2 \;+\; (v')^2$$

Let $p = (0, 1)$ and the change in parameter space be $(u', v') = (0, 1)$. Then $I_p = 1$. So movement in parameter space results in movement of equal length on the surface. See Figure 3.3.

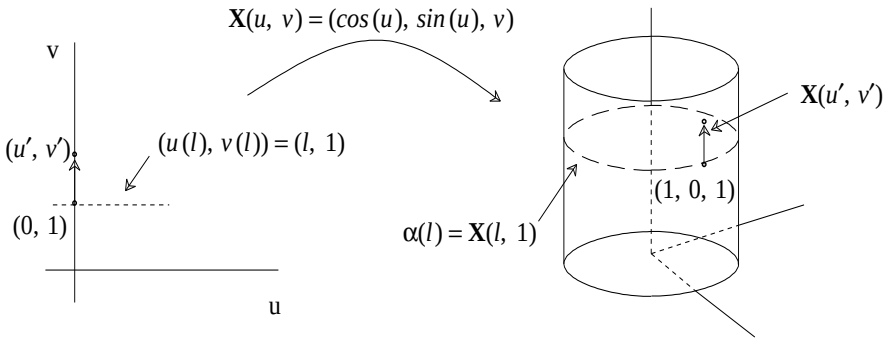


Figure 3.3: A change of (u', v') in parameter space results in a change of $\mathbf{X}$ (u', v') on the surface. A curve in parameter space mapped to a surface.

The first fundamental form is important because it can be used to answer metric questions about a surface. For example, the first fundamental form can be used to find the arc length of a curve on a surface. The arc length s of a parameterized curve $\alpha{:}I \rightarrow S$ is given by

$$s(l) \quad = \quad \int_0^l \; |\alpha'(l)\,|\,dl \quad = \quad \int_0^l \; \sqrt{I(\alpha'(l))}\; dl$$

In particular, if $\alpha(l) = \mathbf{X}(u(l), v(l))$ is contained in a coordinate neighborhood corresponding to the parameterization $\mathbf{X}(u, v)$, we can compute the arc length of α between, say, 0 and l by

$$s(l) \quad = \quad \int_0^l \; \sqrt{E\,\left(\frac{du}{dl}\right)^2 \; + \; 2\,F\,\frac{du}{dl}\frac{dv}{dl} \; + \; G\,\left(\frac{dv}{dl}\right)^2}\; dl \tag{3.7}$$

Continuing the example, let $(u(l),\ v(l)) \ = \ (l,\ 1)$ parameterize a curve in the parameter space of the surface as shown in Figure 3.3. Consequently, $\mathbf{X}(u(l),\ v(l)) \ = \ \mathbf{X}(l,\ 1)$ is a curve that goes around the cylinder. Then $\frac{du}{dl} \ = \ 1$ and $\frac{dv}{dl} \ = \ 0$. Substituting into Eq. (3.7) gives

$$\int_0^l \; dl \quad = \quad l$$

It is common for Eq. (3.7) to be rewritten as

$$\left(\frac{ds}{dl}\right)^2 \quad = \quad E\,\left(\frac{du}{dl}\right)^2 \; + \; 2\,F\,\left(\frac{du}{dl}\,\frac{dv}{dl}\right) \; + \; G\,\left(\frac{dv}{dl}\right)^2$$

giving the rate of change of arc length of α. In the example we have $\left(\frac{ds}{dl}\right)^2 \ = \ 1$, indicating that the parameterizing variable l and the arc length vary at the same rate.

3.3.2 The Model Parameterization of Spatiotemporal Surfaces

ST surfaces are formed as a result of the movement of objects in the scene. The movement of objects results in movement of the contours that make up the projection of the object in the image plane. The resulting ST surfaces can be parameterized based on the type of motion the object and its contours undergo in the scene.

Let S be an ST surface recovered from an image sequence. (How this is done will be addressed later.) If a temporal slice is taken from this ST surface, the generating contour will be a curve in this slice. Let $\alpha(s) = (s, h(s))$ parameterize the generating contour. Consider the case where the contour is translating parallel to the image plane with speed $V^s(t)$ and $V^h(t)$ in the s and h directions, respectively. The resulting ST surfaces can be parameterized using s and time, t , as the two parameters. s parameterizes the contour while t parameterizes the motion. An ST surface generated under orthographic projection of a translating contour is given by

$$\mathbf{X}\,(s,\ t) \quad = \quad (s\ +\ V^s(t),\ h(s)\ +\ V^h(t),\ t) \tag{3.8}$$

Similarly, an ST surface generated as a result of a contour rotating parallel to the image plane is given by

$$\begin{aligned} \mathbf{X}\,(s,\ t) \quad = \quad & (s\ \cos(\theta(t))\ +\ h(s)\ \sin(\theta(t)), \\ & -s\ \sin(\theta(t))\ +\ h(s)\ \cos(\theta(t)), \\ & t) \end{aligned} \tag{3.9}$$

Again, s parameterizes the contour while t parameterizes the motion. This parameterization results from rotating the contour as a function of time. $\theta(t)$ determines the rate of rotation. Rotation is assumed to be around the origin since, without loss of generality, we can always reparameterize the contour so that the origin is defined as the center of rotation. $\theta(t)$, $V^s(t)$ and $V^t(t)$ can be arbitrarily complex in t. For example, they can contain quadratic terms to allow for acceleration.

Combining Eqs. (3.8) and (3.9) we arrive at the following parameterized ST surface for an arbitrary rigid contour undergoing translation and rotation in the image plane:

$$\begin{aligned}\mathbf{X}\,(s,\,t) \;=\; & (s\,\cos(\theta(t)) \;+\; h(s)\,\sin(\theta(t)) \;+\; V^s(t),\\ & -s\,\sin(\theta(t)) \;+\; h(s)\,\cos(\theta(t)) \;+\; V^h(t),\\ & t)\end{aligned} \tag{3.10}$$

Given these restricted types of motion, any recovered ST surface can be described by these model parameterizations. However, since these parameterizations require knowing the motion of the contour, it is not possible to actually use them unless the motion is known *a priori*, which is generally not the case. Despite this, we can show properties of these ST surfaces that are invariant to the parameterization of the surface. For example, it can be shown with the parameterizations above that an ST surface generated from a contour undergoing constant translational motion has Gaussian curvature equal to 0 at every point on the ST surface. Thus, if an ST surface that was generated from a contour moving with constant translational motion is recovered from an image sequence, then the Gaussian curvature must be 0 on this ST surface. This is true no matter how the ST surface is parameterized. In the next section it will be shown how to recover the direction of motion of a point on a contour using the model parameterization above. It will then be shown how to reparameterize the surface such that the motion can be computed without knowing the motion of the contour.

3.3.3 Recovering Point Motion on Spatiotemporal Surfaces

The generating contour at time t is obtained by taking a temporal slice of the ST surface at time t. For any point on the contour we would like to recover the instantaneous motion of the point. First, consider a small segment of the contour which includes the point. Figure 3.4 shows the contour and segment at two times, t_0 and t_1. The long dashed lines show the true movement of the contour segment. Obviously, the arc length of the segment in temporal slice t_0 is the same as in temporal slice t_1. But suppose it is hypothesized that the motion of the contour is as shown by the short dashed lines. Since the contour is rigid, the end points of the segment undergo the same movement. The result of moving the end points in the same direction to the next temporal slice requires a change of arc length of the segment. *Thus, given any small segment of a contour in a temporal slice, the correct movement of the end points must result in no change of arc length in the next*

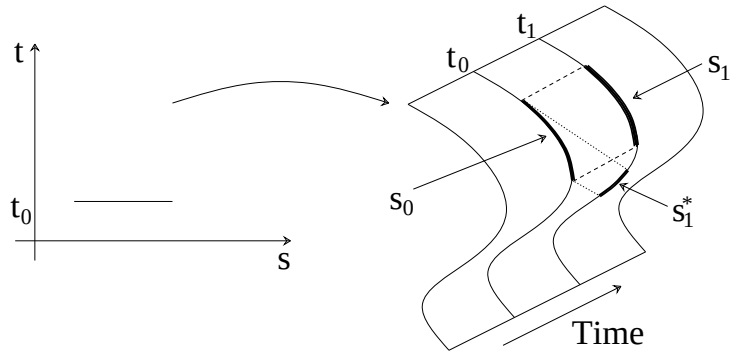


Figure 3.4: Arc length changes if the segment s_0 is moved in the wrong direction. s_1 preserves arc length, but s_1^* does not.

temporal slice. It is this observation that motivates our method. However, the small segment will be replaced by a point and this intuitive observation will be made formal.

Using the model parameterization of ST surfaces, the generating contour is parameterized on the ST surface by keeping t constant in the parameter space of the surface. For example, for an ST surface generated by translational motion, the following curve on the surface is the same as the generating contour:

$$\begin{aligned} \alpha(l) &= \mathbf{X}(u(l),\ v(l)) &= \mathbf{X}(l,\ t_0) \\ &= \mathbf{X}(l\ +\ V^s(t_0),\ h(l)\ +\ V^h(t_0),\ t_0) \end{aligned}$$

In general

$$u(l) = l \qquad\qquad v(l) = t_0$$

Earlier we defined the rate of change of arc length of a curve on a surface to be

$$\left(\frac{ds}{dl}\right)^2 = E\left(\frac{du}{dl}\right)^2 + 2F\left(\frac{du}{dl}\frac{dv}{dl}\right) + G\left(\frac{dv}{dl}\right)^2 \tag{3.11}$$

For the generating curve contour we know

$$\frac{du}{dl} = 1 \qquad\qquad \frac{dv}{dl} = 0 \tag{3.12}$$

Substituting Eq. (3.12) in Eq. (3.11), the square of the instantaneous change of arc length is given by

$$\left(\frac{ds}{dl}\right)^2 = E$$

E is defined at every point on an ST surface. For any given point, if one moves along the surface in the direction the contour is moving, then the change in E is 0. This fact will be proved below. Consequently, given a point on an ST surface, the direction of motion is the direction such that the change in E is 0. Equivalently, the direction of motion is the direction such that the directional derivative of E is 0. This direction will be shown to be perpendicular to the gradient direction of E in the tangent plane.

At this point we have an ST surface S and a function E defined at every point on the surface. The gradient direction of a differentiable function $f : S \rightarrow \Re$ is a differential map $\nabla f : S \rightarrow \Re^3$ which assigns to each point $p \in S$ a vector $\nabla f(p) \in T_p(S) \subset \Re^3$. ∇f is given by [DoCa76]:

$$\nabla f \;=\; \frac{f_s G \;-\; f_t F}{EG \;-\; F^2}\,\mathbf{X}_s \quad + \quad \frac{f_t E \;-\; f_s F}{EG \;-\; F^2}\,\mathbf{X}_t \tag{3.13}$$

Since we are only concerned with the direction and not the magnitude of the gradient, the denominators can be eliminated in Eq. (3.13).

Let $f = E$. Recall that $E = \mathbf{X}_s \cdot \mathbf{X}_s$. Therefore, from Eq. (3.10), for any combination of translational and rotational motion we have

$$\begin{aligned}\mathbf{X}_s(x,t) \;=\;& (\cos(\theta(t)) \;+\; h_s(s,\;t)\;\sin(\theta(t)),\\ & -\;\sin(\theta(t)) \;+\; h_s(s,\;t)\;\cos(\theta(t)),\\ & 0)\end{aligned}$$

$$E \;=\; \mathbf{X}_s \;\cdot\; \mathbf{X}_s \;=\; 1 \;+\; h_s^2(s)$$

The partial derivatives of E are then given as

$$f_s \;=\; E_s \;=\; 2\;h_s(s)h_{ss}(s) \qquad f_t \;=\; E_t \;=\; 2\;h_s(s)h_{st}(s) \;=\; 0 \tag{3.14}$$

Substituting $E_t \;=\; 0$ in Eq. (3.13), the gradient of E on an ST surface is defined as

$$\nabla E \;=\; (E_s\;G)\;\mathbf{X}_s \;-\; (E_s\;F)\;\mathbf{X}_t$$

Note that $\mathbf{X}_t$ is in the direction of motion. So if ∇E is perpendicular to the direction of motion, the inner product of ∇E and $\mathbf{X}_t$ should be 0. This is in fact the case:

$$
\begin{aligned}
\nabla E \cdot \mathbf{X}_t &= [(E_s\ G)\ \mathbf{X}_s \ - \ (E_s\ F)\ \mathbf{X}_t] \ \cdot \ \mathbf{X}_t \\
&= [\ E_s\ (\mathbf{X}_t \cdot \mathbf{X}_t)\ \mathbf{X}_s \ - \ E_s\ (\mathbf{X}_s \cdot \mathbf{X}_t)\ \mathbf{X}_t] \ \cdot \ \mathbf{X}_t \\
&= E_s\ (\mathbf{X}_t \cdot \mathbf{X}_t)\ (\mathbf{X}_s \cdot \mathbf{X}_t) \ - \ E_s\ (\mathbf{X}_s \cdot \mathbf{X}_t)\ (\mathbf{X}_t \cdot \mathbf{X}_t) \\
&= 0
\end{aligned}
$$

To summarize, we have shown using the model parameterization of an ST surface, ∇E is perpendicular to the direction of motion in the tangent plane. Unfortunately, this parameterization cannot be used directly because it assumes prior knowledge of the motion of the contour. Consequently, we must reparameterize the surface such that E is the same but knowledge of the motion is not required.

3.3.4 The Monge Patch Parameterization of Spatiotemporal Surfaces

While the model parameterization used in the previous section is useful for showing properties of ST surfaces, it is not useful for parameterizing ST surfaces when the motion is not known. In this section ST surfaces recovered from an image sequence will be represented as a set of Monge patches. For every point on an ST surface, a patch will be recovered and parameterized.

The result of the previous section says that the direction of motion is perpendicular to the gradient of E in the tangent plane. This remains true when the surface is reparameterized so long as E in the reparameterization equals E in the model parameterization. The Monge patches recovered at each point on an ST surface will be parameterized such that E in a parameterized Monge patch equals E in the model parameterizations. Therefore the result of the previous section will apply.

Any parameterized surface can be represented locally by a function [DoCa76]. The map

$$\mathbf{X}\ (s,\ t) \ = \ (s,\ t,\ f(s,\ t))$$

is called a Monge patch and f is the "height" as a function of s and t. See Figure 3.5. See [Besl86] for a complete discussion of Monge patches.

For each point in an ST volume, a quadratic patch is fit using that point and the

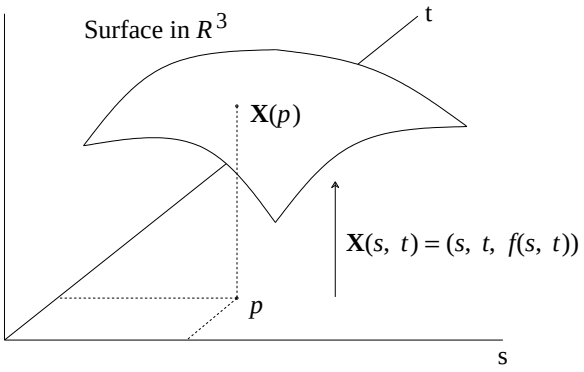


Figure 3.5: A surface defined locally as a height function.

temporal and spatial neighboring edge points. This quadratic patch defines f. The partial derivatives of the patch are equal to:

$$\mathbf{X}_s = (1,\ 0,\ f_s(s,\ t)) \qquad \mathbf{X}_t = (0,\ 1,\ f_t(s,\ t)) \tag{3.15}$$

Therefore

$$E = \mathbf{X}_s \cdot \mathbf{X}_s = 1 + f_s^2(s,\ t) \tag{3.16}$$

$$E_s = 2f_s(s,\ t)f_{ss}(s,\ t) \qquad E_t = 2f_s(s,\ t)f_{st}(s,\ t) \tag{3.17}$$

$f_s(s,\ t)$, $f_t(s,\ t)$, $f_{ss}(s,\ t)$ and $f_{st}(s,\ t)$ are easily computable [Besl86].

We will now show that E in a Monge patch parameterized surface is equal to E in the model parameterization of that surface. Since the surface is the same and the function E on the surfaces are equal, the perpendicular direction to the gradient of E will be equal in the two parameterizations.

Recall that $E = \mathbf{X}_s \cdot \mathbf{X}_s$. Therefore $E = 1 + h_s^2(s)$ in the model parameterization and $E = 1 + f_s^2(s,t)$ in the Monge patch parameterization. To show that the E's in the two parameterizations are equal we need to show that $h_s(s) = f_s(s,\ t)$. In the Monge patch parameterization, the s and t directions can be chosen arbitrarily as long as they are not parallel. Therefore we can choose them to correspond with s and t in the model parameterization. t is perpendicular to the image plane and s is perpendicular to t. $h_s(s)$ in the model parameterization is the slope of the generating contour. Let p be a point on the generating contour at time t_0 and $h_s(s)$ the slope of the generating

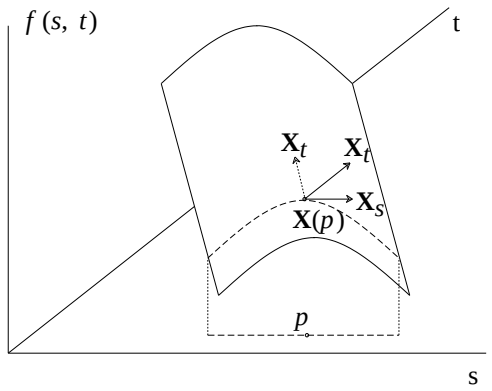


Figure 3.6: The basis vectors in the model and Monge patch parameterizations. The short dashed arrow indicates the model parameterization and the solid arrows indicate the Monge patch parameterization. $\mathbf{X}_s$ is the same in both. The long dashed line shows the generating contour which is translating left.

contour at t_0. Consider a Monge patch centered at p. The partial derivative of f with respect to s, $f_s(s,\ t)$, is the same as the derivative of the curve of intersection formed by the surface and a plane perpendicular to the t axis. This curve of intersection is simply the generating contour. Therefore $f_s(s,\ t)$ must equal $h_s(s)$ and E is the same in both parameterizations. Figure 3.6 shows the relative positions of the basis vectors in each parameterization.

Since the functions E defined on the ST surface are equal, and reparameterization of a surface does not affect the gradient of a function on the surface, we have shown that ∇E in the model parameterization must equal ∇E in the Monge patch parameterization. Earlier we showed that ∇E in the model parameterization is perpendicular, in the tangent plane, to the direction of motion, so ∇E in the Monge patch parameterization must also be perpendicular to the direction of motion.

An Algorithm

An exact procedure for recovering the motion of a point on a contour using the Monge patch parameterization can now be given:

- Detect points on ST surfaces in a given ST volume using a spatiotemporal edge operator, e.g., three-dimensional zero-crossings of the Laplacian
- Fit a quadratic surface to the edge points [Besl86]
- Compute $\mathbf{X}_s$, $\mathbf{X}_t$ and E, F and G of the patch (Eqs. (3.5), (3.15) and (3.16)) [Besl86]
- Compute E_s and E_t (Eq. (3.17))
- Compute ∇E on the patch (Eq. (3.13))
- Compute the direction in the tangent plane of the point that is perpendicular to the gradient direction

In order to find the direction perpendicular to the gradient, the following procedure is used. The cross product of the two basis vectors, $\mathbf{X}_s$ and $\mathbf{X}_t$, gives a vector normal to the tangent plane. The cross product of this normal with the vector in the gradient direction gives a vector in the tangent plane which is perpendicular to the gradient direction.

3.3.5 Gray Level Images and Spatiotemporal 3-Surfaces

All the ideas presented for ST surfaces defined by the edge points in an ST volume can be extended to work with gray level images. Using edge maps resulted in a 2-surface embedded in 3-space. Using gray level images results in a 3-surface embedded in 4-space. The basic ideas and procedures are the same but the dimension is increased. This results in slightly different definitions which are presented below. The "contour" is now two dimensional, parameterized by r and s. The gray level at each pixel defines the height of the contour. The model parameterization, again used to derive properties of the surface, for a contour undergoing translation and a contour undering rotation are, respectively,

$$\mathbf{X}(r,\ s,\ t) \ = \ (r \ + \ V^r(t),\ s \ + \ V^s(t),\ h(r,\ s),\ t) \tag{3.18}$$

$$\begin{aligned} \mathbf{X}(r,\ s,\ t) \ = \ & (r\ \cos(\theta(t)) \ + \ s\ \sin(\theta(t)), \\ & -r\ \sin(\theta(t)) \ + \ s\ \cos(\theta(t)), \\ & h(r,\ s),\ t) \end{aligned} \tag{3.19}$$

Combining Eqs. (3.18) and (3.19) gives the parameterization for a contour undergoing translation and rotation. The first fundamental form coefficients are

$$A = \mathbf{X}_r \cdot \mathbf{X}_r \qquad B = \mathbf{X}_s \cdot \mathbf{X}_s \qquad C = \mathbf{X}_t \cdot \mathbf{X}_t$$

$$D = \mathbf{X}_r \cdot \mathbf{X}_t \qquad E = \mathbf{X}_s \cdot \mathbf{X}_t \qquad F = \mathbf{X}_r \cdot \mathbf{X}_s$$

The gradient direction of a function, f, on a 3-surface is given by:

$$\nabla f = \begin{bmatrix} BC - EE & DE - CF & EF - BD \\ DE - CF & AC - DD & DF - AE \\ EF - BD & DF - AE & AB - FF \end{bmatrix} \begin{bmatrix} f_r \\ f_s \\ f_t \end{bmatrix} \tag{3.20}$$

in the basis $\{\mathbf{X}_r,\ \mathbf{X}_s,\ \mathbf{X}_t\}$.

Since the contour is now parameterized by two variables, there are two functions that can be defined on the contour. One gradient direction is not sufficient to uniquely determine the direction of motion since there are an infinite number of vectors in 3-space perpendicular to the gradient direction. Therefore, the gradient directions of two functions are used. The cross product of the two gradient directions is perpendicular and in the direction of motion. These two functions are the analogs of E in the first fundamental form for a 2-surface and are defined as

$$f_1 = A \qquad f_2 = B$$

where A and B are the coefficients from the first fundamental form for 3-surfaces. The gradient of each function is found separately. Using the model parameterization for translation and rotation of gray level images (the combination of Eqs. (3.18) and (3.19)) we find that A and B and their partial derivatives are equal to the following:

$$\begin{aligned} \mathrm{A} &= 1 + h_r^2\ (\mathrm{r,\ s}) & \mathrm{B} &= 1 + h_s^2\ (\mathrm{r,\ s}) \\ A_r &= 2\ h_r\ (\mathrm{r,\ s})\ \ h_{rr}\ (\mathrm{r,\ s}) & B_r &= 2\ h_s\ (\mathrm{r,\ s})\ \ h_{gr}\ (\mathrm{r,\ s}) \\ A_s &= 2\ h_r\ (\mathrm{r,\ s})\ \ h_{rs}\ (\mathrm{r,\ s}) & B_s &= 2\ h_s\ (\mathrm{r,\ s})\ \ h_{ss}\ (\mathrm{r,\ s}) \\ A_t &= 2\ h_r\ (\mathrm{r,\ s})\ \ h_{rt}\ (\mathrm{r,\ s}) = 0 & B_t &= 2\ h_s\ (\mathrm{r,\ s})\ \ h_{st}\ (\mathrm{r,\ s}) = 0 \end{aligned}$$

Substituting A_r, A_s and A_t into Eq. (3.20) we get a vector in the basis $\{\mathbf{X}_r,\ \mathbf{X}_s,\ \mathbf{X}_t\}$ which is perpendicular to the direction of motion. This is proved by showing that the inner product of the resulting vector with $\mathbf{X}_t$ is 0. A second vector perpendicular to the

direction of motion is obtained by replacing A with B. The cross product of these two vectors then gives the direction of motion.

The procedure to compute the gradient directions and the direction of motion is very similar to the procedure when the ST surface is a 2-surface. A Monge patch and its partial derivatives are now defined as

$$\begin{aligned}
\mathbf{X}(r,\ s,\ t) &= (r,\ s,\ t,\ f(r,\ s,\ t)) \\
\mathbf{X}_r(r,\ s,\ t) &= (1,\ 0,\ 0,\ f_s(r,\ s,\ t)) \\
\mathbf{X}_s(r,\ s,\ t) &= (0,\ 1,\ 0,\ f_g(r,\ s,\ t)) \\
\mathbf{X}_t(r,\ s,\ t) &= (0,\ 0,\ 1,\ f_t(r,\ s,\ t))
\end{aligned}$$

The quadratic surface fit described by Besl and Jain [Besl86] can easily be extended to describe 3-surfaces and used to compute the Monge patch partial derivatives. It is no longer necessary to detect ST surfaces using an edge operator. All the temporal and spatial neighbors of a point in the ST volume are used to fit the quadratic surface.

The gradient direction of the functions A and B each define a vector in 4-space. The direction of motion on the ST 3-surface is perpendicular to this vector. Note that we are only interested in the projection of this perpendicular direction into 3-space. But the perpendicular vector in 4-space must be found before the projection into 3-space. To find the perpendicular direction we use the same procedure as used for ST 2-surfaces. The cross product of the three basis vectors $\mathbf{X}_r$, $\mathbf{X}_s$ and $\mathbf{X}_t$ is computed. The resulting vector is perpendicular to the tangent hyperplane at the point. The cross product of this vector and the two gradient direction vectors gives a vector in the tangent hyperplane which is perpendicular to both gradient vectors. This final vector can then be projected into 3-space to give the direction of motion.

3.4 Results

Figures 3.7 to 3.12 show the results of our method on five image sequences. The sequences include motion of edge images in Figures 3.7 and 3.8, and gray level images in Figures 3.9 to 3.12. The motion in Figures 3.8 and 3.12 was produced by physically moving the

objects in front of the camera. The motion in Figures 3.7, 3.10 and 3.11 was generated synthetically by transforming each image in a static sequence of frames. All gray level images in a sequence were smoothed using a 7×7×7 Gaussian-weighted kernel. The kernel was convolved twice with each ST volume.

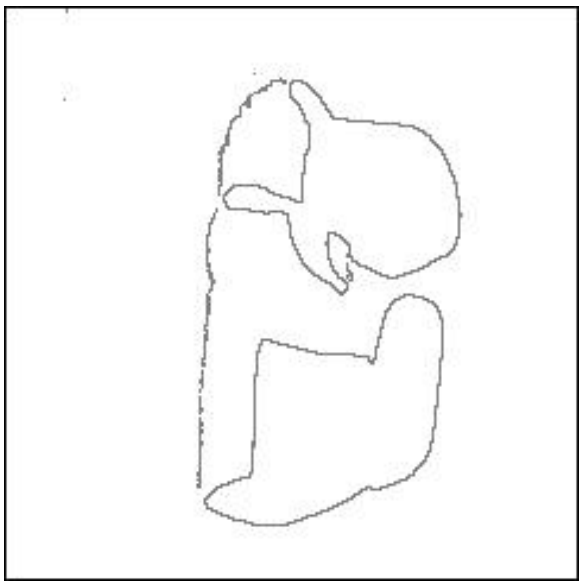
The results show a single temporal slice from the middle of each ST surface flow field. *Time* is into the page so the vectors shown are the result of projecting the ST surface flow vectors into the spatial plane. A vector showing no motion, i.e., no spatial component, is oriented straight into time and appears as a dot. The greater the speed of a point, the greater the spatial component of the vector and hence, the longer the vector appears in the spatial projection.

We found that the edge images worked better when there was a relatively large amount of movement between frames, i.e., greater than 1 pixel/frame, whereas the gray level images worked better with small movements, i.e., less than 1 pixel/frame.

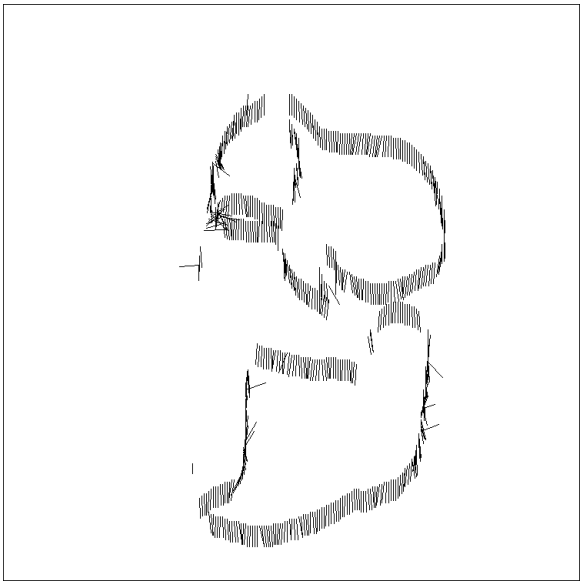
With the gray level images there was a graceful degradation in results as the number of frames in the sequence was reduced. This was also the case with the edge images but a significantly longer sequence was required to achieve good results.

Flow vectors were not found for the edge images at the top and back of the object in Figure 3.7 and the top of the object in Figure 3.8 because an insufficient number of edge points were found in these regions. For each edge point, the edge points within a neighborhood around that point were found and used for fitting a Monge patch. When edge points were missing in some frames, or the neighborhood size was so large that edge points could not be found for each position in the neighborhood, a Monge patch was not fit to the data.

There are regions of incorrect results in the lower-left part of the left object and the top of the right object in Figure 3.8. These errors were caused because edge points from separate ST surfaces were recovered as one surface. In general, the problem of segmenting edge points into coherent ST surfaces is a difficult problem [Bake88a] which we have not focused on in this research. Gray level images are preferable in this respect since the intermediate steps of detecting edge points and segmenting them into ST surfaces are avoided.

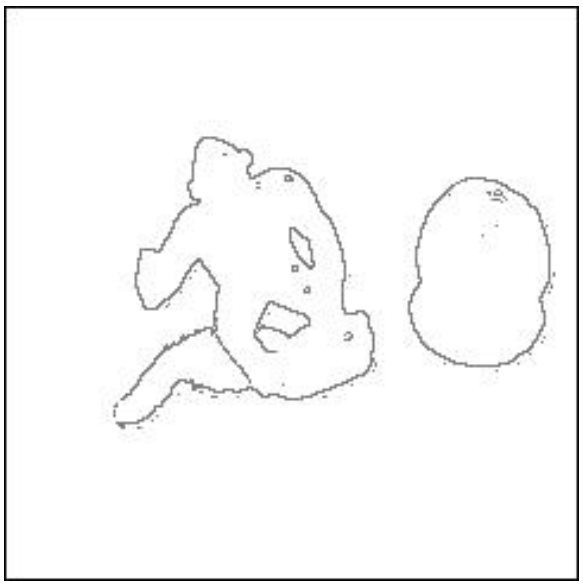


(a)

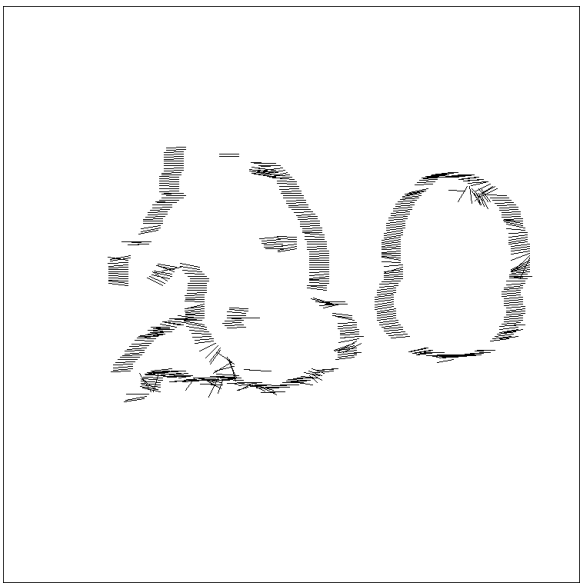


(b)

Figure 3.7: (a) Middle frame from a 9 frame sequence of a doll translating down at 2 pixels/frame. (b) ST surface flow for the middle frame. The final flow was smoothed using a 3×3 kernel. The motion was synthetically produced by translating each static image.



(a)



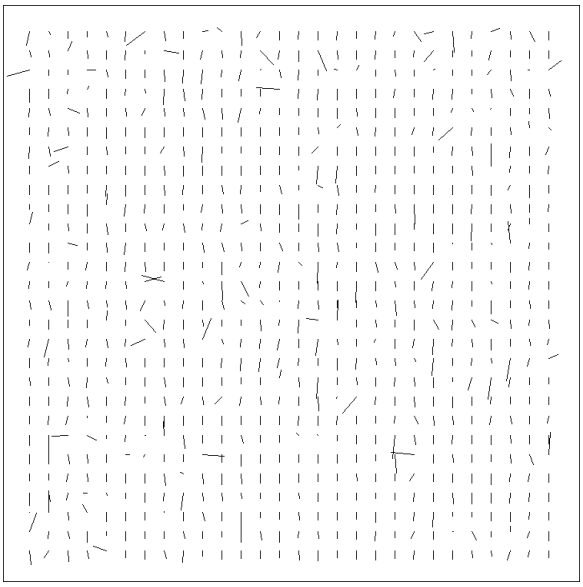
(b)

Figure 3.8: (a) Middle frame from a 13 frame sequence of a doll and an orange translating left at approximately 1.3 pixels/frame. (b) ST surface flow for the middle frame. The final flow was smoothed using a 3×3 kernel. The motion was produced by physically moving the objects in front of the camera.

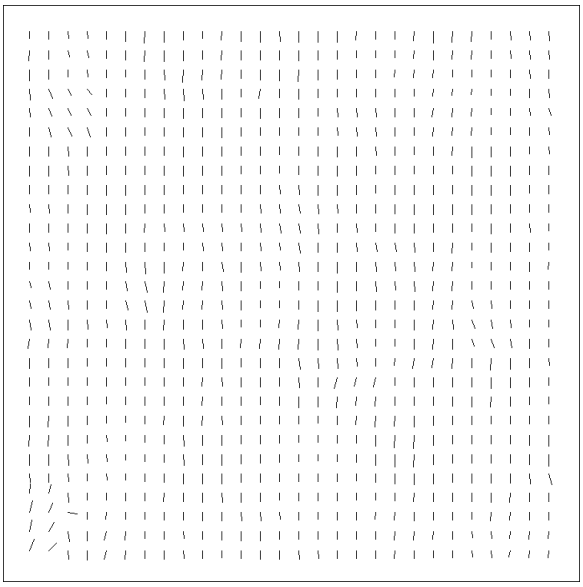
(a)

(b)

Figure 3.9: Two gray level images used in subsequent figures. (a) A page from a phone book; one frame from a 7 frame sequence in which no motion occurred. (b) A circuit board from a 7 frame sequence of images. The board was translated left by physically moving the object in front of the camera at approximately 1.3 pixels/frame.

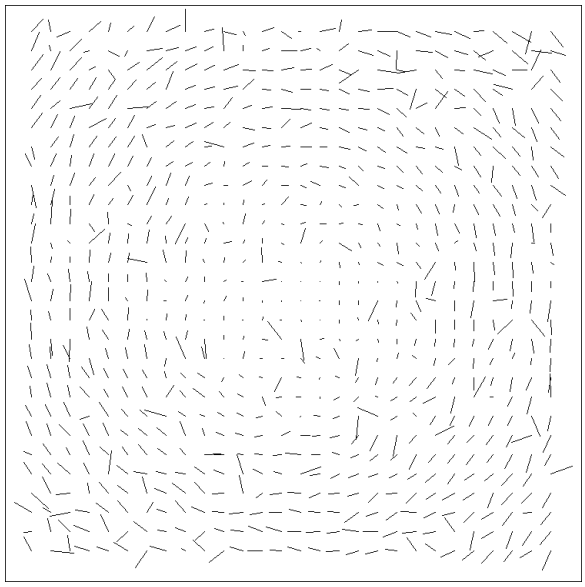


(a)

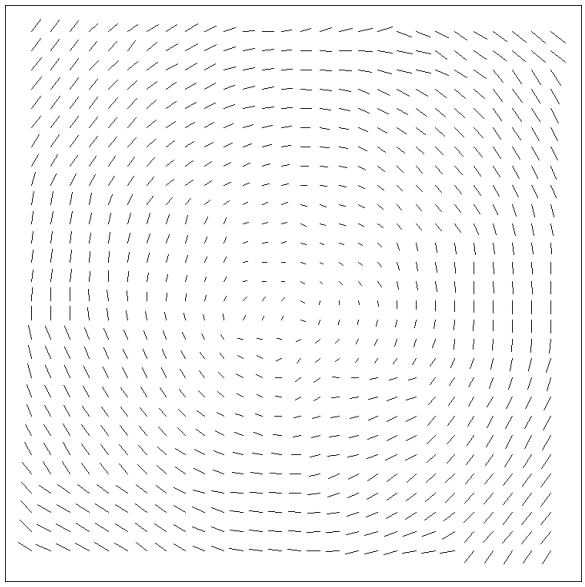


(b)

Figure 3.10: (a) ST surface flow for the middle frame of the phone book sequence, where motion was synthetically produced by translating each frame down 0.5 pixels/frame. (b) The flow in (a) smoothed using a 3×3 kernel.

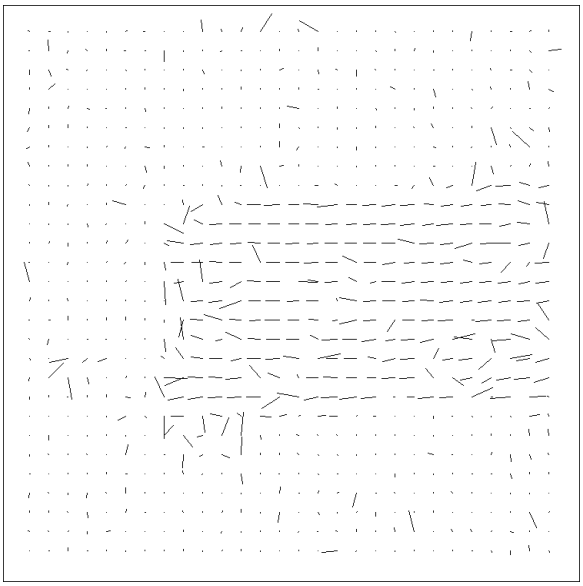


(a)

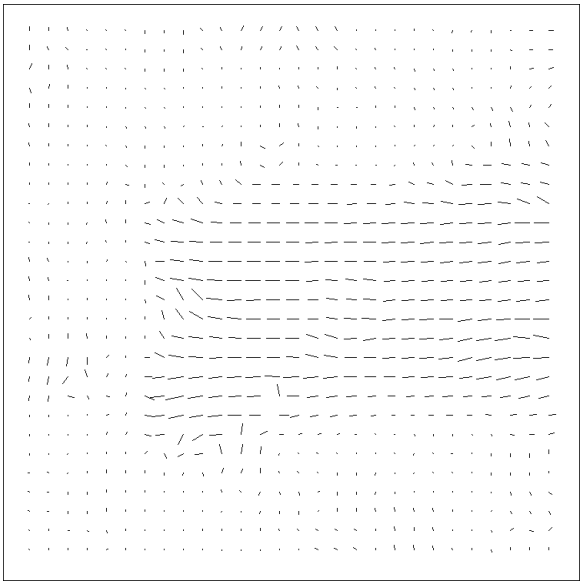


(b)

Figure 3.11: (a) ST surface flow for the middle frame of the phone book sequence, where motion was synthetically produced by rotating about the center at 0.008 radians/frame. (b) The flow in (a) smoothed using a 5×5 kernel.



(a)



(b)

Figure 3.12: (a) ST surface flow for the middle frame of the circuit board sequence. (b) The flow in (a) smoothed using a 3×3 kernel.

Below we present more quantitative results of our method to compute ST surface flow in gray-level image sequences. Thus far, the computed flow contained few errors. Here we present empirical evidence that shows that those results are representative. Further, it will be shown how the size of the surface patch used for fitting an ST surface affects the result and how the amount of smoothing of the input sequence affects the result. Finally, we will show the robustness of our method.

Clearly, the quantitative results given below are dependent upon the image sequences used. However, the qualitative form of the results should not change with other image sequences. And by examining the qualitative results in this section, the relative accuracy of ST surface flow computation under different conditions can be better understood.

In order to measure the correctness of out results we need a method of measuring errors. In the sequences produced with synthetic motion, the correct flow is known. By comparing the computed ST surface flow vectors with the correct ST surface flow vectors, the error can be measured. Given a computed unit vector and the correct unit vector in the ST cube, the angle between the two vectors is the error. The error in a slice of the computed flow is the mean error at all pixels in the slice. The error is always computed prior to smoothing the flow vectors. This was done so that only the ST surface flow computation was evaluated and not the effects of smoothing the computed flow.

To show that the results on the slices shown above are representative, we show that the ST surface flow in all slices of the ST cube have similar mean error. Only sequences with synthetic motion were used since those are the only sequences where the correct flow is known exactly. Table 3.1 shows the standard deviation of the mean errors for slices of the ST cube. That is, the mean error is computed for many slices of an ST cube and the standard deviation of these means is computed. The low standard deviation indicates that the mean errors of slices of the ST surface flow are similar throughout the ST cube. Therefore, the results shown above are representative of the results throughout the ST cube.

The results above all used the same ST surface patch size for computing partial derivatives and smoothed the image sequence by the same amount. This was done for consistency. However, the results are relatively insensitive to the choice of patch size and the amount of smoothing. That is, the mean error does not vary greatly as these

Image Sequence	Mean Error Through the Sequence (Radians)	Standard Deviation of Mean Error
Page Translating Down (Figures 3.9 and 3.10)	0.21	0.067
Page Rotating Parallel to Image Plane (Figures 3.9 and 3.11)	0.25	0.037
Halfdome Rotating in Depth (Figures 3.20 and 3.21)	0.11	0.024
Zoom into Krakow (Figures 3.20 and 3.22)	0.38	0.0051

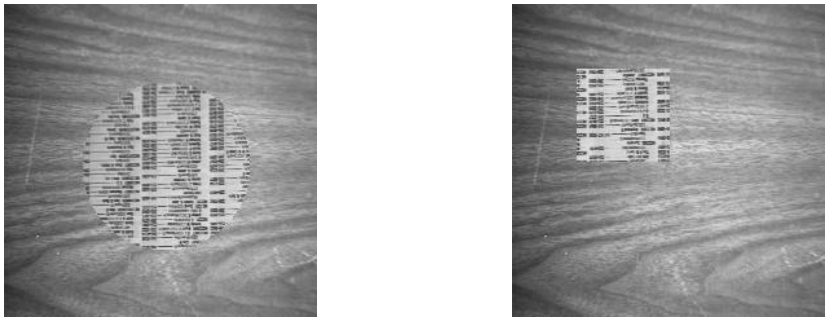
Table 3.1: The mean error of eight slices of an ST cube and the standard deviation of these mean errors. The low standard deviation shows that all slices of the ST cube have similar error in the computed ST surface flow.

parameters are changed.

Figures 3.14 and 3.15 show the mean error as a function of the amount of smoothing and the size of the surface patch used to fit the ST surface. The 3D Gaussian weighted smoothing kernel varied from a standard deviation of 0 pixels to 3 pixels, with each of the three dimensions of the kernel identical. The patch size varied from $3\times3\times3$ to $19\times19\times19$ with each of the three dimensions always of equal size.

The sequences shown in Figure 3.9(a) were used for generating the error surfaces in Figure 3.14. In the first sequence the page rotated parallel to the image plane at 0.008 radians/frame. In the second sequence the page translated down and right at 0.4 pixels/frame. In both cases the phone book page covered the entire image so there were no occlusion boundaries.

The sequences shown in Figure 3.13 were used to generate the error surfaces in Figure 3.15. In the first sequence the disk rotated parallel to the image plane at 0.008 radians/frame against a static background. In the second sequence a square patch translated down and right at 0.4 pixels/frame against a static background. The motion was synthetically produced in both sequences, guaranteeing no camera noise and producing



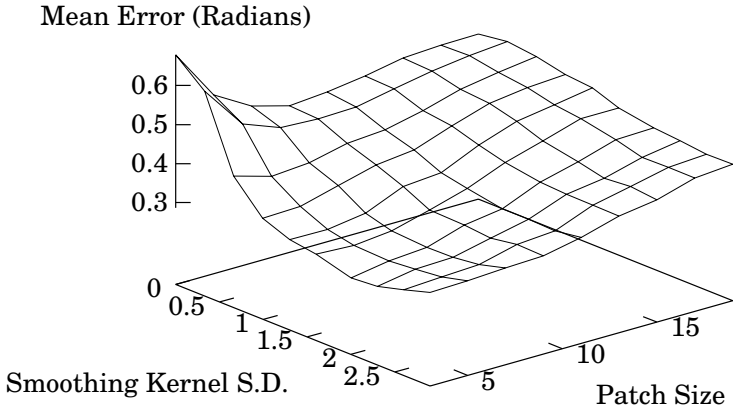
(a) (b)

Figure 3.13: (a) One frame of a sequence where a circular disk rotates parallel to the image plane at 0.008 radians/frame against a static background. (b) One frame of a sequence where a square patch translates down and right at 0.4 pixels/frame against a static background.

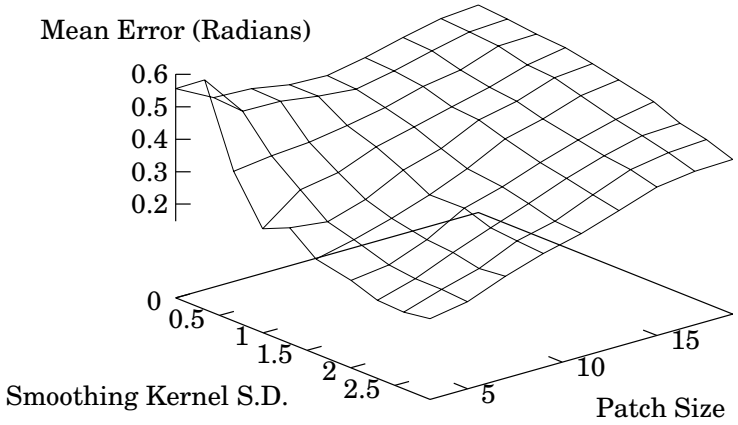
exact motion.

For the image sequences used in Figure 3.14 it is expected that large surface patches and large smoothing kernels do not adversely affect the result. When a surface patch overlaps a gray-level discontinuity, e.g. at an occlusion boundary, however, error is introduced into the patch because of the discontinuity of gray-levels. This results in errors at those discontinuities in the computed ST surface flow. Similarly, when a smoothing kernel overlaps an occlusion boundary, errors are introduced because the image sequence is smoothed across a gray-level discontinuity. If occlusion boundaries exist, large surface patches and large smoothing kernels will overlap the boundary more often, resulting in greater mean error. The sequences used in Figure 3.14 had no occlusion boundaries, so this error source was not present.

Even though the same speed and direction of motion was used to generate the sequences associated with Figures 3.14 and 3.15, the error in Figure 3.15 is less than in Figure 3.14. In Figure 3.15, relatively little of the image is occupied by occlusion boundaries so little error is introduced by the smoothing kernel and surface patch overlapping the boundary. Also contributing to the low mean error is the fact that the computed flow is always correct in the motionless background. Since the same image was used to generate the entire sequence, the temporal partial derivative is exactly zero, resulting in the exact ST surface flow being computed. However, in image sequences of scenes that

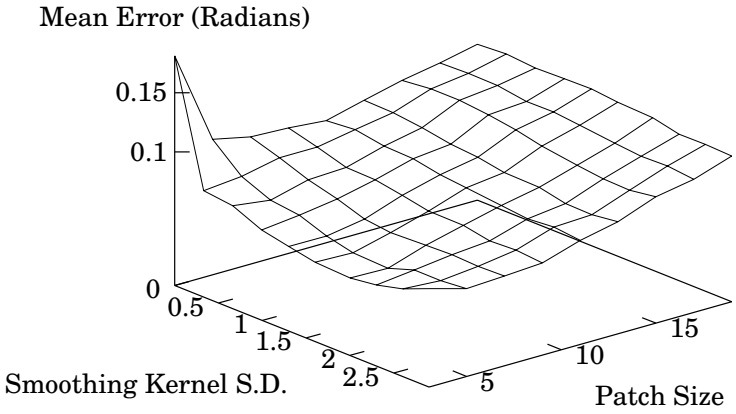


(a)

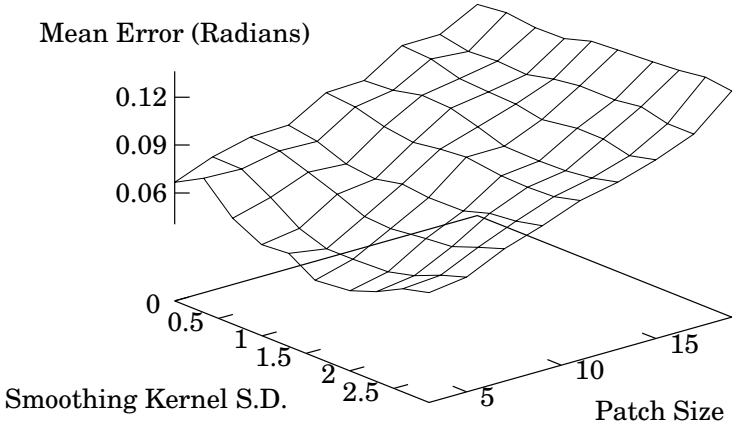


(b)

Figure 3.14: Mean error as a function of the ST surface patch size and the amount of smoothing. (a) The error surface resulting from a sequence consisting of a planar surface (occupying the entire image) rotating parallel to the image plane at 0.008 radians/frame (Figure 3.9(a)). (b) The error surface resulting from a sequence consisting of a planar surface (occupying the entire image) translating parallel to the image plane at 0.4 pixels/frame (Figure 3.9(a)). "Smoothing Kernel S.D." is the standard deviation of the Gaussian-weighted kernel.



(a)



(b)

Figure 3.15: Mean error as a function of the ST surface patch size and the amount of smoothing. (a) The error surface resulting from a sequence consisting of a planar surface rotating parallel to the image plane against a static background (Figure 3.13(a)). (b) The error surface resulting from a sequence consisting of a square patch translating parallel to the image plane against a static background (Figure 3.13(b)). "Smoothing Kernel S.D." is the standard deviation of the Gaussian-weighted kernel.

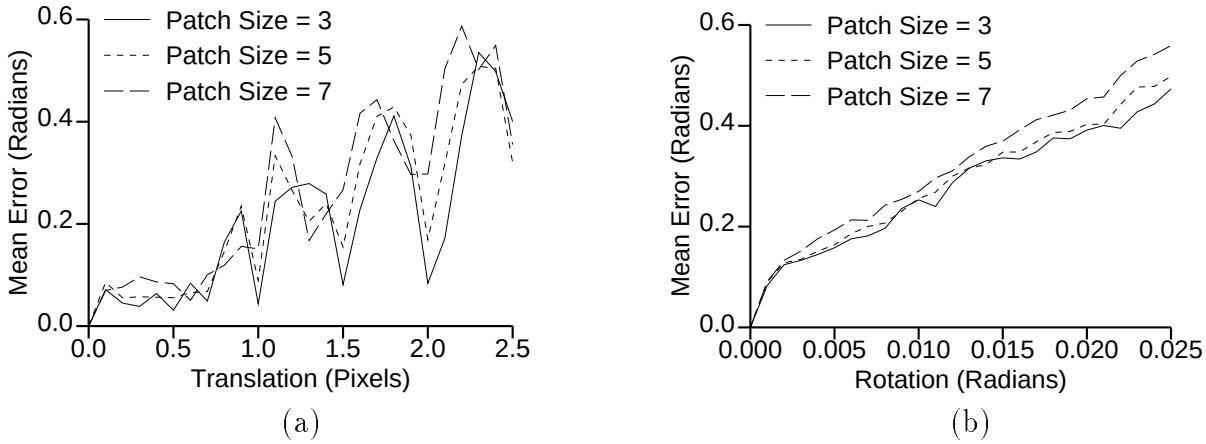


Figure 3.16: Mean error as a function of the amount of motion. (a) Sequence consisted of a planar surface translating parallel to the image plane. (b) Sequence consisted of a planar surface rotating parallel to the image plane.

contain many small moving objects, many occlusion boundaries result, giving larger error for large surface patch sizes and smoothing kernels.

Regardless of the amount of occlusion in the image sequence, for the image sequences used, the mean error increases with surface patch size. When computing a patch, a neighborhood is centered at a point and all points within the neighborhood are used to compute the best fitting patch. As the patch size increases, points more distant from the center point are used to compute the patch. While the patch with the least error is computed, using points farther away may result in greater error at the center. This results if the points farther away are on a differently shaped surface than the point at the center of the neighborhood. This occurs with highly textured images, as was the case in our image sequences. So a smaller patch can allow the data near the center of the neighborhood point to be fit well, whereas a large patch must also fit the data points farther way, thereby increasing the error of the fit near the center.

Surface patch size and the amount of smoothing are the only variable parameters in our method. However, there are other factors that affect the computation of ST surface flow, the speed of objects in the scene, for example. As the motion in the scene increases, the mean error is expected to increase. This occurs because the fit of the surface patch to the data is worse as the speed increases.

Figure 3.16 shows the mean error as a function of the amount of motion. In the synthetic sequence for Figure 3.16(a) the surface was translating to the right, and in the sequence for Figure 3.16(b) the surface was rotating parallel to the image plane. For the case of rotation, the error increases gradually. But in the case of a translating surface, the error has local minima and maxima because of sampling errors. Synthetic motion is produced by computing how each frame of the sequence would appear given a continuous image. This continuous image is then sampled into an n×n image. For example, in a sequence where a planar patch is translating to the right at 0.6 pixels per frame, the gray-level at pixel (x, y) in the first frame will be at location $(x + 0.6, y)$ in the next frame. Since images are sampled at integer coordinates only, pixel (x, y) in the second frame must be assigned an interpolated value. Using zero-order interpolation means pixel (x, y) is assigned the gray-level from pixel (x, y) in the previous frame. In the third frame, the gray-level is at $(x + 1.2, y)$ in the continuous image and at $(x + 1, y)$ in the sampled image. Synthetic continuous image sequences could have been used in our experiments, however real cameras produce digital images so we chose to use digital images.

Note that the correct flow used in the computing error is the flow that would result if continuous images were available. That is, it is the projection of the actual flow of objects in the scene. This is done because it is the correct flow resulting from a scene that we wish to compare our results to.

Figure 3.16(a) shows local minima at integer pixel motions. This results because there are no sampling errors in these cases. Local minima also result at half-pixel points in addition to other locations that depend upon the size of the surface patch used. This results because the motion causes the image object to line up on pixel boundaries on a regular basis, reducing the sampling error. Similar local minima do not occur in the rotating page sequence (Figure 3.16(a)) because the amount of motion varies throughout the image. The gradual increase in the error shown in Figure 3.16 indicates the method gracefully degrades as the amount of motion increases.

There are a number of potential sources of noise in an image sequence, including: error in gray-level quantization, i.e., incorrect gray-level values, temporal sampling error and discontinuous variations of the scene motion or image motion. The robustness of our method of computing ST surface flow to each of these sources of noise will now be

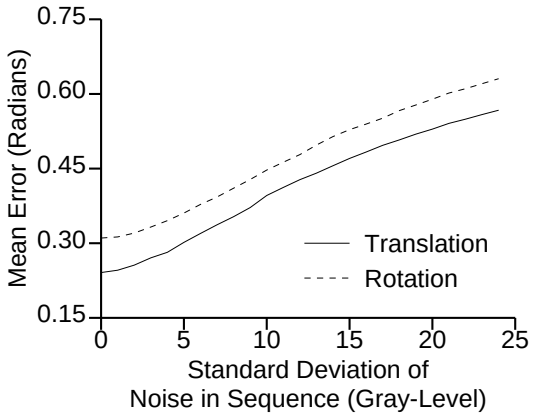


Figure 3.17: Mean error as a function of the amount of normally distributed white noise in the image sequence. The image shown in Figure 3.9(a) was translated and rotated in two separate sequences.

examined.

The amount of noise due to the gray-level quantization of the camera can be measured in a static sequence by examining how the gray-level varies at each spatial position through the sequence. This noise can be artificially increased by altering gray-levels through the sequence. The degradation of results can then be examined as this noise increases.

Figure 3.17 shows the mean error as a function of the amount of noise due to the camera. The rotation sequence consisted of a surface (Figure 3.9(a)) rotating parallel to the image plane at 0.008 radians/frame. The translation sequence consisted of the same image translating parallel to the image plane. The same image was used throughout the sequence but the gray-level at each pixel in every frame was altered by a random amount. The gradual increase in the error shown in Figure 3.17 indicates the graceful degradation of results as the amount of noise due to the camera increases.

In an ideal scene the motion of objects is smooth. However, object motion may not be completely smooth and/or the camera may not be absolutely stationary. The amount of noise due to these sources can be artificially controlled. In our test sequences generated synthetically, the motion was exact. This is equivalent to smooth motion with exact temporal sampling, i.e., the time between frames is known exactly. Noise can be

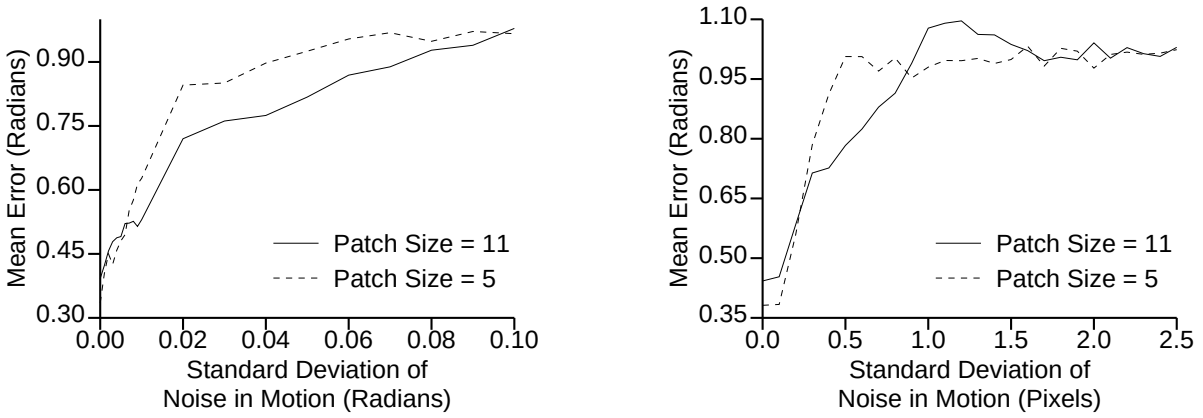


Figure 3.18: Mean error as a function of error in the motion. (a) Sequence consisted of a planar surface rotating parallel to the image plane. (b) Sequence consisted of a planar surface translating parallel to the image plane. Normally distributed noise was added to the synthetically produced motion. The results are shown for two different surface patch sizes.

introduced by altering the object motion a small random amount through the sequence and varying the time between frames.

The graphs in Figure 3.18 resulted from a phone book page translating (Figure 3.18(a)) and rotating (Figure 3.18(b)) parallel to the image plane. As before, the same image is used throughout the sequences. The mean error is shown as a function of the amount of noise in the object motion. As shown in Figure 3.18 the effect of noise in the motion can be counter-acted by using large surface patches. This effectively smooths the input, thereby reducing the effect of the noise in the motion.

In addition to noise from non-perfect motion in the scene, noise also occurs due to inexact in the temporal sampling. In the test sequences generated synthetically, the temporal sampling was exact. The mean errors shown in Figure 3.19 were generated from a sequence with noise added to the temporal sampling rate. The image sequence again consisted of a surface rotating parallel to the image plane (producing Figure 3.19(a)) and translating parallel to the image plane (producing Figure 3.19(b)). The same image was used throughout the sequences. As in the case of noisy object motion, the effect of this type of noise can be counter-acted by using large surface patches.

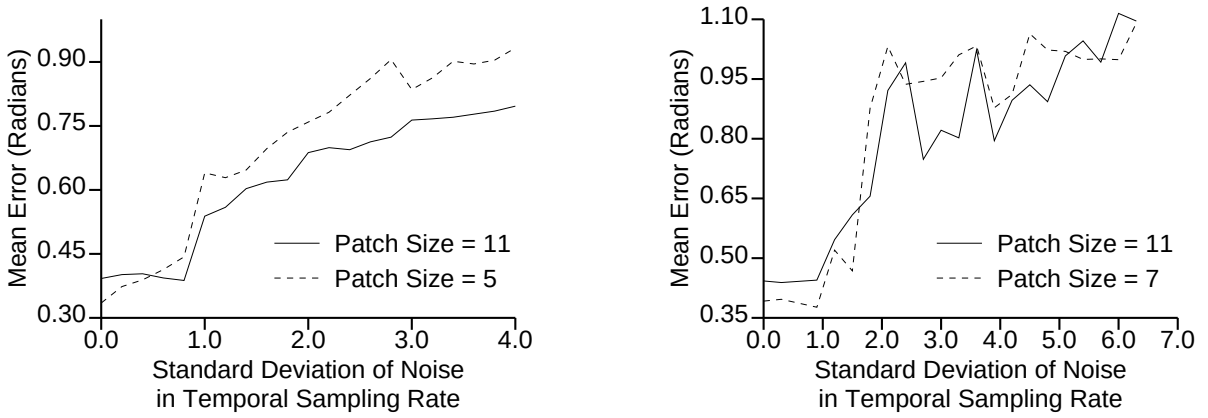


Figure 3.19: Mean error as a function of error in the temporal sampling. (a) Sequence consisted of a planar surface rotating parallel to the image plane. (b) Sequence consisted of a planar surface translating parallel to the image plane. Normally distributed noise was added to the time between frames. The results are shown for two different surface patch sizes.

In conclusion, in this section we have shown that the mean error of the computed ST surface flow does not vary greatly as the surface patch size or the amount of smoothing varies. It was also shown that the results first presented in this section are representative of slices of ST surface flow throughout the ST cube. Finally, we demonstrated how our method behaves under various amount of motion and noise.

3.5 Related Issues

Since optical flow is equivalent to a temporal slice of the ST surface flow projected into the spatial plane, a comparison of the assumptions and robustness of both approaches is of interest. Specifically, we compare our method with gradient-based optical flow algorithms since they are the most similar to ours. Next, we show why our method approximates the correct ST surface flow when an ST surface is generated by object motion that is not parallel to the image plane or when orthographic projection is not assumed. The aperture problem and how our method deals with cases where it exists, is also briefly addressed.

3.5.1 Comparison with Gradient-Based Optical Flow Algorithms

Gradient-based optical flow algorithms typically use Eq. (3.3), the optical flow constraint equation, along with additional assumptions in order to solve for the motion $\frac{\mathrm{d}x}{\mathrm{d}t}$ and $\frac{\mathrm{d}y}{\mathrm{d}t}$. These additional assumptions include [Agga88]:

- optical flow is smooth and neighboring points have similar velocities
- optical flow is constant over some segment of the image
- objects in the scene undergo restricted motion

By examining what, if any, of these three assumption our method makes we can better understand the relationship between our method and gradient-based optical flow approaches.

With respect to the first assumption, our approach does not need to assume neighboring points have similar velocities because it solves for the velocity exactly. We implicitly assume that an ST surface is sufficiently smooth so that a quadratic surface can be fit and the partial derivatives recovered. If the ST surface is sufficiently busy over a small neighborhood, poor partial derivatives may result. But this assumption is also made by gradient-based optical flow approaches since the partial derivatives of the ST volume must be computed there as well. This assumption is not a severe limitation since higher order surfaces can always be used [Besl88].

Since it is not necessary to assume neighboring points have similar velocities, we also do not assume that the optical flow is constant over an image region. Thus, our approach does not make the second assumption given above.

Our model parameterization assumes motion of a contour is a combination of translational and rotational motion parallel to the image plane, i.e., the motion is planar. However, it was shown by Schunck [Schu84] that the optical flow constraint equation is valid only for translation, i.e., our model parameterization is no more restrictive.

3.5.2 Approximating General Motion

The model parameterization assumes that motion in the scene is parallel to the image plane, i.e., there is no rotation in depth. Orthographic projection is also assumed. Therefore, no change in depth occurs. However, if these assumptions are violated, our method does give an approximation to the correct ST surface flow. In this section we show why this is true.

Recall the model parameterization for a contour translating parallel to the image plane is given by

$$\mathbf{X}\,(s,\ t) \quad = \quad (s\ +\ V^s(t),\ h(s)\ +\ V^h(t),\ t) \tag{3.8}$$

where $s\ +\ V^s(t)$ parameterizes the s component of the surface and $h(s)\ +\ V^h(t)$ parameterizes the h component of the surface. If an object rotates in depth or orthographic projection is not used, the resulting ST surface will not be of the form in Eq. (3.8). But if it can be shown that the parameterization in Eq. (3.8) approximates the true ST surface then our method will give an approximation to the true flow. The accuracy of the approximation will depend upon how good the parameterization in Eq. (3.8) approximates the true ST surface.

The ST surface resulting from a contour rotating in depth such that the projection rotates about the h axis is given by

$$\mathbf{X}\,(s,\ t) \quad = \quad (s\ \cos(\theta(t)),\ h(s),\ t) \tag{3.21}$$

A similar argument holds for rotation about the s axis. $\theta(t)$ measures the rotation of the contour as a function of time. To show that Eq. (3.21) is approximated by a parameterization of the form in Eq. (3.8) we need to show that $s\ \cos(\theta(t))$ can be approximated by $s\ +\ V^s(t)$. $\cos(\theta(t))$ is equal to $\sqrt{1\ -\ \sin^2(\theta(t))}$. For small $\theta(t)$ this is approximately equal to $\sqrt{1\ -\ \theta^2(t)}$. Further, $\sqrt{1\ -\ \theta^2(t)}$ is approximately equal to $1\ -\ \theta^2(t)$ for small $\theta(t)$. Substituting this in Eq. (3.21) gives

$$\begin{aligned} \mathbf{X}\,(s,\ t) \quad &= \quad (s\ (1\ \ -\ \ \theta^2(t)),\ h(s),\ t) \\ &= \quad (s\ \ -\ \ s\theta^2(t),\ h(s),\ t) \end{aligned}$$

Viewing $s\theta^2(t)$ as the $V^s(t)$ term, we see that the model parameterization for a translating contour approximates the parameterization for a contour rotating in depth.

This result can be extended by showing that an ST surface generated by a contour moving in depth under perspective projection is also approximated by the model parameterization of a translating contour. The position in an image of a point (x, y, z) in the scene under perspective projection is $(f\frac{x}{z},\ f\frac{y}{z})$ where f is the effective focal length. Applying this to s and $h(s)$ in the model parameterization gives the following parameterization of an ST surface resulting from a contour changing depth in the scene:

$$\mathbf{X}\ (s,\ t) \quad = \quad \left(\frac{f\ s}{z(s,\ t)},\ \frac{f\ h(s)}{z(s,\ t)},\ t\right) \tag{3.22}$$

$z(s,\ t)$ is the depth to a point on the contour which changes with time. Consider a point $p\ =\ \mathbf{X}(s_0,\ t_0)$ on an ST surface. Given a patch of the ST surface around p we will show that this patch can be approximately parameterized by the model parameterization for a translating contour. Approximating the first component of Eq. (3.22), $F(s,\ t)\ =\ \frac{f\ s}{z(s,\ t)}$, with a truncated Taylor series for two variables around the base point $(a_1,\ a_2)$ we find

$$\begin{aligned}
\frac{f\ s}{z(s,\ t)} \quad &= \quad F(a_1,\ a_2) \quad + \\
&\qquad \left[(s\ -\ a_1)\ \frac{\partial}{\partial s}\ F(a_1,\ a_2) \quad + \quad (t\ -\ a_2)\ \frac{\partial}{\partial t}\ F(a_1,\ a_2)\right] \\
&= \quad \frac{f\ a_1}{z(a_1,\ a_2)} \quad + \\
&\qquad \left[(s - a_1)\ \frac{\partial}{\partial s}\ F(a_1,\ a_2)\ -\ (t - a_2)\ \frac{f\ s}{z(a_1,\ a_2)}\ z_t(a_1,\ a_2)\right]
\end{aligned} \tag{3.23}$$

Let $a_1\ =\ s_0$ and a_2 be such that $z(a_1,\ a_2)\ =\ f$. Substituting a_1 and a_2 in Eq. (3.23) gives

$$\begin{aligned}
\frac{f\ s}{z(s,\ t)} \quad &= \quad s_0 \quad + \\
&\qquad \left[(s - s_0)\ \frac{\partial}{\partial s}\ F(s_0,\ a_2)\ -\ (t - a_2)\ \frac{s}{z(s_0,\ a_2)}\ z_t(s_0,\ a_2)\right]
\end{aligned} \tag{3.24}$$

At the point $p\ =\ \mathbf{X}(s_0,\ t_0)$ Eq. (3.24) equals

$$\frac{f\ s_0}{z(s_0,\ t_0)} \quad = \quad s_0 \quad - \quad (t_0\ -\ a_2)\ \frac{s_0}{z(s_0,\ a_2)}\ z_t(s_0,\ a_2) \tag{3.25}$$

Viewing the second term of Eq. (3.25) as the velocity term of $s\ +\ V^s(t)$, we see that at p the first component of Eq. (3.22) is approximated by Eq. (3.25). A similar result can be shown for the second component of Eq. (3.22), $\frac{f\ h(s)}{z(s,\ t)}$. Therefore, the model

parameterization for a translating contour approximates the ST surface generated by a contour moving in depth under perspective projection.

Figures 3.20 through 3.22 show the results of applying our method to a sequence of images rotating in depth (under orthographic projection) and to a sequence where the camera is zooming into the image (under perspective projection). The motion in the sequences was synthetically produced by transforming each image.

There are incorrect results in the lower right of Figure 3.21(b). This resulted becaused that section of the image sequence contains little variation in gray level (see Figure 3.20(a)). With little variation, the gradients were close to 0. Taking the cross product of the two gradient directions to get the direction of motion is noisy when the gradients are close to 0.

The top center of Figure 3.22(b) contains incorrect results which also resulted because of little variation in gray level. Although not visible in Figure 3.20(b), most of the sky region except the center contains clouds. In the cloud areas there was sufficient gray level variation for the correct result to be achieved, but in the non-cloud area there was not enough variation.

3.5.3 The Aperture Problem

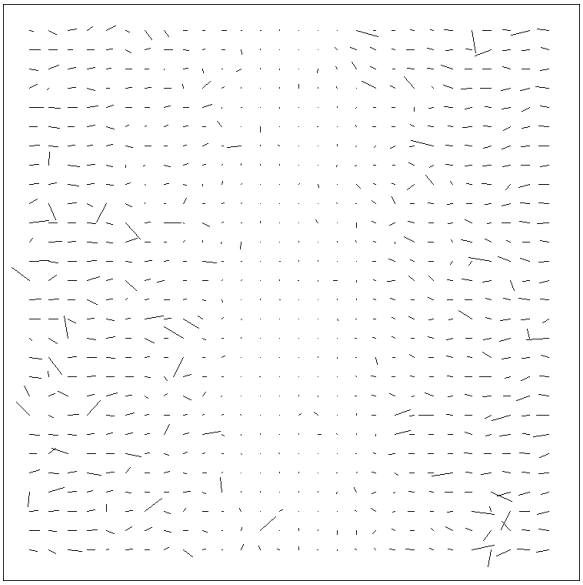
Yamamoto [Yama89] showed that for translational and rotational motion parallel to the image plane, there is an aperture problem when the contour is a straight line and when there is rotation about the center of a circle. Consequently, our method should either be undefined or unable to recover the motion in these cases. In all other cases the true motion can be recovered even when a local method is used.

When using edge images, a straight line will force $E_s = 2\ h_s(s,\ t)h_{ss}(s,\ t)$ to be 0. This results in $\nabla E = 0$. Since the gradient vector is 0, the perpendicular direction is undefined. A "straight line" in a gray level image occurs when the gradient direction of the image at neighboring points is the same. In this case, both functions defined on the ST surface will have all partials of second order equal to 0, i.e., $A_s = A_g = A_t = 0$ and similarly for B . This results in Eq. (3.20), the gradient equation, equaling 0.

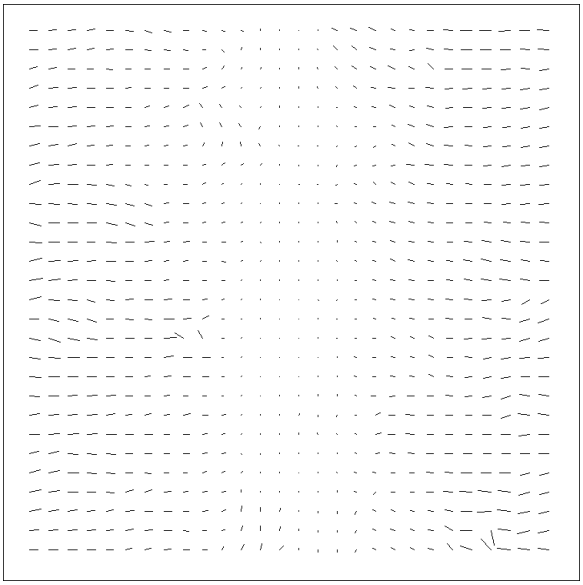
(a)

(b)

Figure 3.20: Two gray level images used in subsequent figures. (a) Halfdome; one frame from a 7 frame sequence of images in which no motion occurred. (b) Krakow; one frame from a 7 frame sequence of images in which no motion occurred.

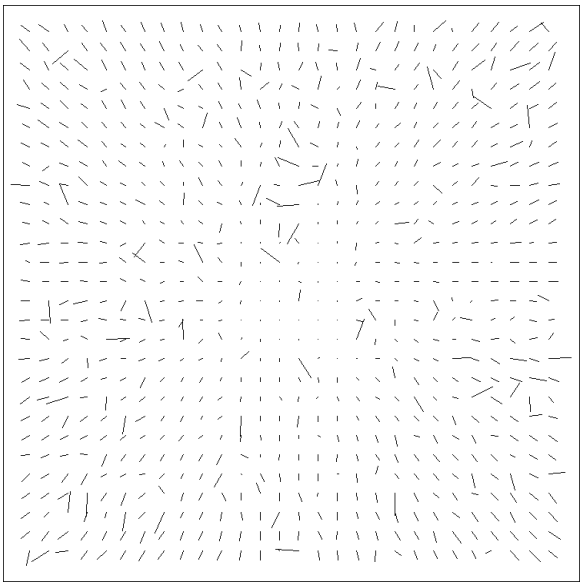


(a)

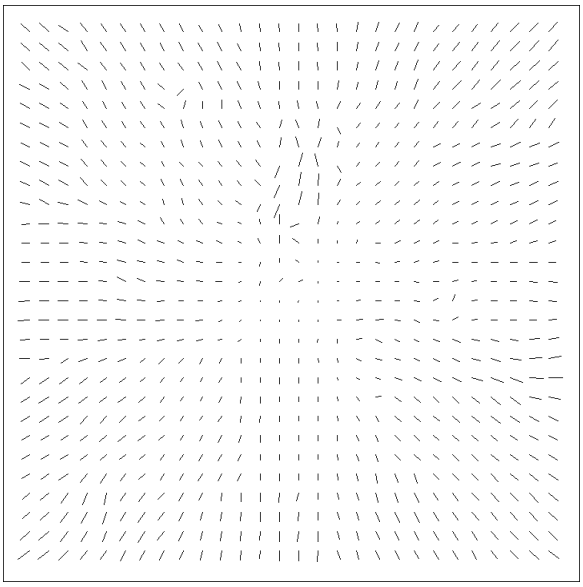


(b)

Figure 3.21: (a) ST surface flow for the middle frame of the Halfdome sequence, where motion was synthetically produced by rotating each frame about a vertical line through the middle of the image at 0.03 radians/frame. This resulted in movement of approximately 0.4 pixels/frame near the left and right edges. (b) The flow in (a) smoothed using a 3×3 kernel.



(a)



(b)

Figure 3.22: (a) ST surface flow for the middle frame of the Krakow sequence, where motion was synthetically produced by zooming into the center of each frame. The movement was approximately 0.6 pixels/frame near the edges. (b) The flow in (a) smoothed using a 3×3 kernel.

When there is circular motion about the center of an arc of a contour, there is no change in either the ST 2-surface or the ST 3-surface, so this will be interpreted as no motion. Similarly, if there is a translational component, only that component is recovered.

These are the only two cases where an aperture problem arises for a rigid contour in motion and our method, correctly, does not find a solution.

3.6 Concluding Remarks

A method was presented for recoving spatiotemporal surface flow, i.e., the instantaneous motion of all points on the generating contour of an ST surface. It was shown that the direction of motion of a point is perpendicular to the gradient direction of a function defined on an ST surface. This function measures the change in arc length on an ST surface. As a result, the motion can be solved for exactly. Because our formulation is well-posed, it is preferable to gradient-based optical flow methods which rely on the ill-posed optical flow constraint equation.

Our method has been implemented using the Monge patch parameterization of an ST surface and test results for five image sequences have been presented. The results show the successful computation of ST surface flow in both edge and gray level image sequences. The sequences using edge images required an intermediate step of segmenting edge points into ST surfaces. Because of this extra computation and possibility of errors, using gray level images is the preferable approach.

We have shown why our method approximates the correct flow in cases where object motion is not parallel to the image plane or when orthographic projection is not assumed. In the future, we plan to investigate ways to extend the ideas developed in this chapter to include motion in depth and contour deformation so the flow in these cases can be computed exactly.

As in the gradient-based optical flow approach, our work assumes that any change in gray level through a sequence of images is due only to motion, i.e., every point in the scene remains the same intensity. We hope to remove, or at least relax, this assumption in future work. Consider an object rotating in depth. As it rotates, its surfaces change

illumination. When this change is gradual, the ST surface will undergo a gradual rise or decline. Gray level variation will still exist, but will be at a higher or lower level depending upon the current illumination. By incorporating this rise and fall of the ST surface into our model parameterization we hope to eliminate the constant brightness assumption.

Chapter 4

Spatiotemporal Flow Curves

In this chapter, spatiotemporal (ST) flow curves are recovered from ST surface flow, and these curves are then grouped to form coherent regions of the ST cube such that each group represents an object or surface in the scene undergoing similar long-range motion. ST flow curves are defined such that the tangent at a point on the curve equals the ST surface flow at that point. Our algorithm forms clusters of similar flow curves and is based the following two intuitive, yet powerful observations. First, a point in an image can only move to at most one point in the next image. Second, a point in an image can come from at most one point in the previous image. When these situations are violated, or it appears that they are violated, occlusion or disocclusion has occurred.

In the first section, we show how the qualitative nature, as opposed to a quantitative nature, of ST flow curves is sufficient for many motion problems. This is done by examining the problem domains and by looking at related work in psychology. In Section 4.2 it is shown how flow curves are recovered from ST surface flow. Section 4.3 describes the types of flow curves that are generated and how they are clustered such that each cluster represents the motion of one coherent object or surface in the scene. Finally, Section 4.4 shows how to detect occlusion and disocclusion from flow curve clusters. Results are presented in Section 4.5. We assume that the ST cube is composed of gray level images. However, our methods are easily modified to also work with edge images. But, as shown in Chapter 3, the additional problem of recovering coherent ST surfaces must be solved when using edge images rather than gray level images.

4.1 Recovering a Qualitative Description

Interpretation of motion in an image, e.g., locating separate objects, can be done *prior* to recovery of 3D scene motion or 3D scene structure. A common paradigm when examining image sequences is to recover 3D structure or 3D motion immediately after optical flow is computed [Agga88]. Recently, some work has even completely skipped the flow recovery step and computed structure and motion directly [Heel90, Taal90, Faug90]. However, a great deal of information exists in the flow field and therefore our approach is to recover a qualitative description of image motion from ST surface flow. Qualitative analysis of image motion is not new. In the past, this usually meant a qualitative study of optical flow or short image sequences [Fran90, Carl88, Verr87, Koen75]. Our work can be viewed as an extension of that problem into the temporal dimension, taking advantage of temporal motion coherence over long image sequences. This allows the recognition of higher-level motions, e.g., cyclic motion [Allm90b], which occur over long image sequences.

By clustering flow curves, a qualitative *grouping* of regions in the ST cube is recovered as opposed to a *segmentation* of the ST cube. Segmentation requires that every pixel in the cube be classified into some set. Our results are more qualitative, classifying pixels only in coherent areas. This approach is desirable for two reasons: first, all gradient-based methods for computing optical flow and ST surface flow are undefined at occlusion boundaries; second, only a qualitative result is needed in many situations.

In practice, optical flow and ST surface flow are not only undefined at occlusion boundaries, but also within a neighborhood around these boundaries. Gradient-based methods for computing flow assume motion and intensity values vary smoothly, an assumption that is violated at occlusion boundaries. To compute the flow, a window around each pixel is used. If any part of the window overlaps an occlusion boundary, the intensity values do not vary smoothly within the window and the computed flow is likely to be incorrect. To segment an ST cube, even these unstable pixels must be classified when, in fact, their values are incorrect.

Since the ST surface flow cannot be computed reliably near occlusion boundaries, our approach recovers a qualitative description of motion. However, in many visual motion tasks, exact localization of motion boundaries is not required. For example, if

one is interested in tracking an object, the exact location of the object boundary is not necessarily as important as, say, where the centroid of the object moves [Alle89]. An automated surveillance system may need to detect only that motion occurred, and track the centroid of the motion [Burt88]. In obstacle avoidance, unless the margin for error is small, the exact boundaries of objects in relative motion are not needed [Nels88]. So rather than attempt to classify pixels based on inaccurate data, exact localization of motion boundaries is sacrificed.

Related work in psychology supports the view that localization of occlusion boundaries is not required in order to recover a useful description of motion. Johansson [Joha73] attached lights to the joints of a person and then filmed the person undergoing motion, such as walking, in the dark. Only the motion of the lights was visible in these *moving light displays* (MLD). When people observed these MLD's they almost immediately recognize the motion. Clearly, localization of boundaries is not required for recognition of motion since no boundaries exist in an MLD, only the path of the lights through time and space exists. Note that these paths are precisely ST flow curves. This suggests that ST flow curves represent salient properties of the image sequence and are useful for motion description and recognition.

Related work in computer vision falls into two categories, that dealing with segmenting an ST cube and that dealing with segmenting an optical flow field. The latter can be viewed as a 2D version of our problem. However, while the problem appears similar, approaches must differ since temporal coherence is the dominate constraint in our method and optical flow segmentation deals with flow at a single instant in time. In addition to working with an instant in time, most optical flow segmentation approaches attempt to localize discontinuities in the flow [Litt87, Thom85, Reic83] even though the image flow is in error in these areas [Schu86]. As stated earlier, our approach does not deal with these areas, but rather uses areas where the flow is well defined.

Jain segmented an ST cube by examining the signs of three principle curvatures of an ST 3-surface [Jain88]. These signs were determined by the type of the motion that generated the surface, translational for example. As such, only the type of motion was used and not the direction of motion, as we do when using ST surface flow. Later, Liou and Jain presented a volume-growing algorithm based on gray level [Liou90]. While both

of these approaches address the problem of ST cube segmentation, they used temporal and spatial gray-level coherence as opposed to temporal and spatial motion coherence.

Peng and Medioni presented an algorithm that analyzed oriented slices of an ST cube [Peng89]. An edge operator was applied to each slice and the resulting edge slices were then examined to recover the normal component of motion. λ and Y junctions, indicators of occlusion and disocclusion respectively, were located in each slice. The problem with their approach is that edge detectors model an edge as a sudden intensity change, a model not satisfied near λ and Y junctions. Since our method need not localize boundaries of occlusion and disocclusion, this is not a problem. Our method will identify occlusion and disocclusion but there is no reliance on detecting boundaries such as λ or Y junctions.

Flinchbaugh and Chandrasekaran segmented a sequence of images; but rather than stress the importance of using the motion of objects over long sequences, the instantaneous motion was used for clustering [Flin81]. Further, the common and relative motion of points was recovered prior to clustering the points. Alternatively, our approach proposes that clustering of points can occur prior to recovering higher level motion descriptions such as common and relative motion.

4.2 Recovering Flow Curves

In this section we show, given a sequence of images forming an ST cube, how the motion of points through the cube are recovered. The first step is the computation of the *spatiotemporal surface flow*, i.e., the instantaneous motion of each point in the ST cube. The spatiotemporal surface flow is computed as described in Chapter 3 and gives a vector for each pixel in an ST cube indicating the motion of that pixel. Given the ST surface flow over many frames, flow curves through the ST cube are then recovered. Loosely, flow curves are started in the first frame and flow through the cube to the most recent frame. In theory, a flow curve could be started at every pixel in the first frame. For practical reasons, flow curves are started at uniformly-spaced intervals in the first frame, e.g., one curve for each non-overlapping 16×16 block of pixels. Nothing other than computation time prevents a more dense placement of flow curves.

An ST flow curve can be represented as a parameterized space curve. A parameterized ST 3D curve, α(t), is a map $\alpha : \mathbf{I} \rightarrow \Re^3$ of an open interval $\mathbf{I}$ = (a, b) of the real line $\Re$ into $\Re^3$. α defines a correspondence which maps each $t \in \mathbf{I}$ into a point $\alpha(t) = (x(t), y(t), t) \in \Re^3$.

Using prime to denote the partial derivative with respect to t, the vector

$$(x'(t), y'(t), 1) = \alpha'(t) \in \Re^3$$

is called the tangent or velocity vector of the curve α at t.

Assume for the moment that the ST surface flow is smooth. Given a smooth ST surface flow, an ST flow curve α is defined such that the velocity vector of α at each point equals the ST surface flow at that point. (See Figure 1.1.) Let the ST surface flow be defined by

$$\mathbf{F} : \Re^3 \rightarrow \Re^3 \qquad \mathbf{F}(\mathbf{x}) = (\Delta x, \Delta y, 1)$$

Requiring the velocity of α to equal $\mathbf{F}$ is equivalent to

$$\alpha^{'}(t) = \mathbf{F}(\alpha(t)) \tag{4.1}$$

A given flow curve has the initial condition $\alpha_i(0) = (x_0, y_0, 0)$, where (x_0, y_0) are the coordinates of the pixel in the first frame. Using coordinates (x, y, t) for the ST cube, the preceding equation can be rewritten as the simultaneous equations

$$\begin{aligned} x^{'}(t) &= F_1(x(t), y(t), t) \\ y^{'}(t) &= F_2(x(t), y(t), t) \\ t^{'}(t) &= F_3(x(t), y(t), t) \quad = \quad 1 \end{aligned}$$

with initial conditions

$$(x(0), y(0), t) = (x_0, y_0, 0)$$

where $\mathbf{F} = (F_1, F_2, F_3)$.

Many methods can be used to solve this system of equations given the starting points in the first frame and the ST surface flow, $\mathbf{F}$, defined at every pixel in the ST cube. However, since $\mathbf{F}$ is only defined at coordinate points and must be interpolated at intermediate pixels, the relatively simple Runge-Kutta method [Pres88, Gear71] is appropriate. More sophisticated methods where the increment in t varies depending upon the complexity of

F are not used since there is no reason not to use the smallest increment in t available, namely 1.

Two issues remain regarding the recovery of flow curves: when do they terminate and when are they created? That is, how is the interval associated with a flow curve determined? Clearly, flow curves must be created in the first frame and terminated in the most recent frame. But this is not sufficient. For example, Figure 1.1(c) shows flow curves starting where the background becomes disoccluded. These issues will be addressed in Section 4.4.

The results of the Runge-Kutta method give a sequence of points that define each flow curve. In order to compute shape-description properties of flow curves such as curvature, a quadratic curve is fit to each set of points. A separate curve segment, centered at each point, is fit for every point making up a flow curve. This is done using a 1D version of the quadratic surface fitting procedure described by Besl and Jain [Besl86]. Once the quadratic curve segment is fit, the partial derivatives at a point can be recovered and the curvature computed using the following equation:

$$\kappa = \frac{\sqrt{A^2 + B^2 + C^2}}{\left((x')^2 + (y')^2 + (t')^2\right)^{3/2}}$$

where

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix} \qquad B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix} \qquad C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}$$

The velocity vector at a point of a flow curve, $(x'(t), y'(t), 1)$, will also be a useful curve descriptor since it gives the instantaneous direction and speed of motion through the ST cube. Also, the difference between two velocity vectors of two flow curves at time t measures the instantaneous difference in speed and direction of the two flow curves.

In summary, flow curves are recovered from the ST surface flow using the Runge-Kutta method. For each curve the Runge-Kutta method gives a sequence of points that define the curve. After fitting a quadratic curve segment to each point of each curve, the curvature and velocity vector at each point can be computed as shape descriptors of the curve.

4.3 Clustering Flow Curves

Once flow curves are recovered from an ST cube they must be grouped, or clustered, so that all the flow curves that make up a cluster are "similar." Ideally, similarity should be defined so that two flow curves are similar if they both are generated by the motion of a single rigid object. Torsion, curvature and velocity are three flow curve descriptors that can be used to measure similarity. How and why these measures will correctly group curves as similar will be discussed below. How to cluster flow curves using these measures will then be described.

4.3.1 Describing Flow Curves

Any space curve, and hence any flow curve, is completely defined, up to rigid rotation and translation, by its curvature and torsion. Intuitively, the curvature at a point on a curve measures how fast the curve pulls away from the velocity or tangent vector at the point. The normal vector at a point is defined as the instantaneous change of the velocity vector at that point. The plane formed by the velocity vector and the normal vector is called the osculating plane. The torsion at a point measures how fast the curve pulls away from the osculating plane. So the curvature is a measure of curving in the plane and the torsion is a measure of twisting out of the plane. Given a starting point, an orientation, and the curvature and torsion values at every point on a curve, that curve is uniquely defined [DoCa76]. Therefore, the torsion and curvature are good measures of the shape of curves [Allm90a, Mokh88, Hoff88]. By classifying the different types of motion that can occur in the scene, and the resulting projected motion, one can gain an understanding of the types of flow curves that can be produced. Then it can be shown how torsion, curvature, and velocity can be used so that two curves generated by the same object will be considered similar.

Objects in a scene can move in only two ways — they can translate and rotate. These motions can be further classified as parallel to the image plane or in depth, resulting in five separate categories of motion: translation parallel to the image plane, translation in depth, rotation parallel to the image plane, rotation in depth, and no motion. The

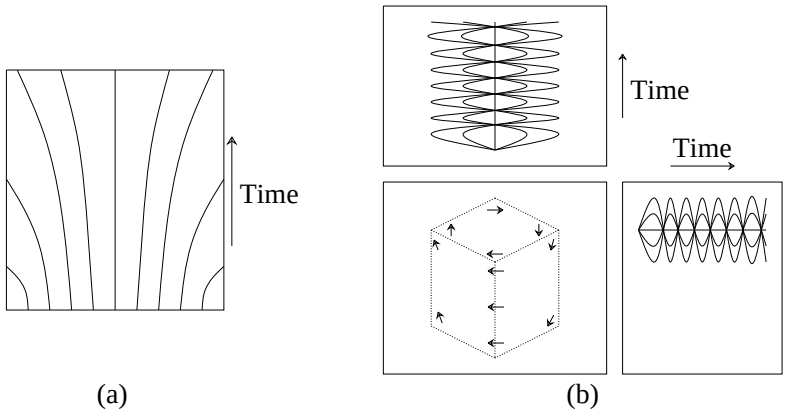


Figure 4.1: (a) Flow curves from a slice of the ST cube where an object is moving toward the camera. The focus of expansion is in the center of the slice. (b) The lower-left view shows the ST surface flow of a rotating cube. Time is into the page. The top and right views show the flow curves for the top surface of the box.

resulting flow curves for each type of motion can be studied by examining how the curvature, torsion and velocity change in each case.

When an object is not moving or is translating at constant speed parallel to the image plane, the flow curves generated are straight lines as shown by the right object in Figure 1.1. Curvature in this case is zero and torsion is undefined. The velocity vector indicates the direction and speed of motion.

If orthographic projection is used, there is no apparent motion when an object translates in depth. In the case of perspective projection, flow curves will curve away from the focus of expansion (foe). See Figure 4.1(a). Because the flow curves are planar, torsion is always equal to zero and curvature varies along the curve. Curvature is larger the farther the flow curves are from the foe. Since the flow curves form a symmetric pattern around the foe, the velocity vector varies between flow curves.

An object rotating parallel to the image plane results in corkscrew-like flow curves. The curvature increases from the center of rotation where it equals zero to the curvature of the flow curve farthest from the center of rotation. Torsion is undefined at the center of rotation and decreases with distance from the center. The velocity vectors are cyclic along the curve and increase with distance from the center of rotation.

Motion	Spatial			Temporal		
	Curvature	Torsion	Velocity	Curvature	Torsion	Velocity
No Motion	Zero	Undefined	Zero	Zero	Undefined	Zero
Translate Parallel To Image Plane	Zero	Undefined	Constant	Zero	Undefined	Constant
Translate In Depth	Increases	Zero	Increases	Increases	Zero	Increases
Rotate Parallel To Image Plane	Increases	Decreases	Increases	Constant	Constant	Cyclic
Rotate In Depth	Increases	Decreases	Increases	Cyclic	Cyclic	Cyclic

Table 4.1: Curvature, torsion and velocity changes between curves (spatial) and along curves (temporal). "Increases" and "Decreases" is with respect to distance from the center of rotation or focus of expansion.

To illustrate the types of motion that can be produced when an object rotates in depth, Figure 4.1(b) shows flow curves resulting from a cube rotating in depth. The flow curves resulting from the top face have an oval, corkscrew shape. Curvature, torsion and velocity behave the same as for rotation parallel to the image plane except the values along the curve are cyclic. The flow curves generated by the side faces of the cube are also oval shaped but since these faces turn away from the viewer, the flow curves merge with the background after each face becomes occluded.

The torsion values produced by these four types of motion are of limited usefulness for distinguishing between these cases. This does not mean that torsion is a poor measure for distinguishing between space curves in general, only that it is poor for the types of space curves produced in an ST cube. Here, since curvature varies as torsion does, or inversely to it, torsion will not help in distinguishing flow curves. Hence, using only curvature and velocity, all flow curves of one type can be distinguished from the other types.

Table 4.1 summarizes the above analysis by showing how curvature, torsion and velocity change along a flow curve (temporally) and between flow curves (spatially) of the same type. For example, consider the flow curves associated with an object rotating parallel to the image plane. The curvature and torsion along a flow curve is constant and

the velocity is cyclic.

Using Table 4.1 it can be shown that curvature is sufficient to distinguish between the four motion types. Given two different types of curves from Table 4.1, it must be shown that the curvature values along the curves differ. This will show that curvature is sufficient to distinguish the different types of flow curves.

Consider the Temporal, Curvature column of Table 4.1. Ignoring for the moment the No Motion row, curvature along a curve varies differently for the four types of flow curves. So given two different types of flow curves, the difference between these curves must be large since the curvature values along the curves are different most everywhere. One exception is a flow curve with zero curvature, generated by an object translating parallel to the image plane, and a flow curve at the center of rotation or foe. Since these two curves are similarly shaped, no measure can distinguish them based on shape alone. See Section 4.5 for how position can be used in this case.

The only types of flow curves that curvature cannot distinguish are the ones resulting from translation parallel to the image plane and no motion because in both cases the curvature is zero. Therefore velocity is also required in general to classify flow curves uniquely according to the type of motion that generated them.

Torsion could be used to differentiate between most of the motion types, but nothing is gained by using torsion as well as curvature. Therefore, because torsion adds nothing and is a third derivative operation making it very susceptible to noise, only curvature and velocity are used. Other work has also used only curvature for similar reasons [Allm90b].

From these five primitive cases of 3D motion, we have shown how curvature and velocity are sufficient for describing and classifying flow curves. When objects undergo a combination of these types of motion, accelerate, or there is camera motion, the flow curves become more complicated of course. In these cases, velocity becomes less useful for clustering and therefore curvature must be used as the primary description of a curve.

4.3.2 Flow Curve Difference Measures

Given a set of flow curves, we would like to be able to cluster those curves so that all the curves in each cluster were generated by a single rigid object. Typically, a clustering algorithm is given a set of items and a way of measuring the "difference" between objects. For a given number of clusters there is an optimal clustering of the items that minimizes the total difference. However, finding this optimal clustering requires exponential time. Therefore, most clustering algorithms attempt to find a clustering that is close to the optimal but requires considerably less time to compute.

All clustering algorithms suffer from the fact that the algorithm does not know how many clusters exist *a priori*. In many applications it is reasonable to expect that this information is supplied. However, in our case this would amount to specifying how many objects are moving in the scene. Clearly this is an undesirable assumption. Therefore we will use heuristic techniques for automatically determining how many clusters exist [Hart75, Rome84].

A clustering algorithm needs a function that measures the difference between two items. It has already been shown why curvature and velocity are sufficient to distinguish between ST flow curves, but it has not been shown how, using these measures, to compute the difference between a pair of curves. We will first show how this is done in the continuous case. After converting this result to the discrete case, a very intuitive difference measure is obtained.

The curvature and velocity of a curve can be thought of as functions, κ and α':

$$\kappa(\alpha(t)) \rightarrow \Re \qquad\qquad \alpha'(t) \rightarrow \Re^3$$

κ and α' can be considered as points in Hilbert space. Hilbert space is an infinite dimensional vector space such that the vectors have finite length. For example, the vector $(1, \frac{1}{2}, \frac{1}{3}, ...)$ has finite length $\sqrt{(1)^2 + (\frac{1}{2})^2 + (\frac{1}{3})^2 + \cdots}$ so it is a vector in Hilbert space.

Consider the function $\kappa(\alpha(t))$ on the interval $t_i \leq t \leq t_j$. $\kappa(\alpha(t))$ can be considered as a vector with a continuum of components along the interval. Intuitively, the squared length of such a vector is the summation of all the squared components, which becomes

the integral of $\kappa(\alpha(t))$. I.e.,

$$\|\kappa(\alpha(t))\|^2 = \int_{t_i}^{t_j} [\kappa(\alpha(t))]^2 dt$$

Since the length of κ and α' (or the distance to the point in Hilbert space) can be computed, the distance between two functions in Hilbert space can also be found. The direction from one function to another is the component by component difference of the two functions. The squared distance then is just the squared length of this difference vector. Since the length of each function is finite, the length of the difference vector, or the distance between the functions, will also be finite.

The discussion above assumes that the functions κ and α' have finite length. For every point in some bounded interval, κ and α' each have a finite value. The values can be arbitrarily large, but they are still finite. Therefore over the bounded interval the length of the function is bounded. Again, the length of each function can be arbitrarily large, but it will be finite.

In the discrete case the integral is replaced with a summation and we arrive at the following distance or difference measure between functions:

$$\begin{aligned}
\|\kappa(\alpha_1) - \kappa(\alpha_2)\|^2 &= \sum_{t=t_i}^{t=t_j} [\kappa(\alpha_1(t)) - \kappa(\alpha_2(t))]^2 \\
\|\alpha_1' - \alpha_2'\|^2 &= \sum_{t=t_i}^{t=t_j} [\alpha_1'(t)) - \alpha_2'(t)]^2 \\
&= \sum_{t=t_i}^{t=t_j} \left[\sqrt{(x_1'(t) - x_2'(t))^2 + (y_1'(t) - y_2'(t))^2 + (t-t)^2)} \right]^2 \\
&= \sum_{t=t_i}^{t=t_j} \left[(x_1'(t) - x_2'(t))^2 + (y_1'(t) - y_2'(t))^2 \right]
\end{aligned}$$

Note this summation is over a fixed segment of the two curves. This segment can be the entire curve from time 0 to the current time or say, the most current T steps. In any case, for each time, the difference is computed, squared and summed. A weighted average of these two difference measures is used to compute the difference between flow curves. See Section 4.5 for how these weights are set.

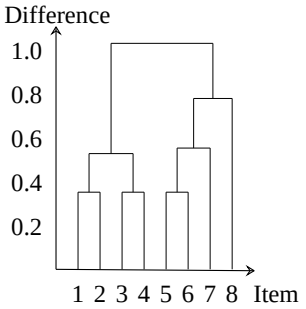
4.3.3 Clustering Algorithms

Given a difference measure, many possible clustering algorithms could be used. As stated earlier, most clustering algorithms require the number of clusters be given *a priori*. In order to determine the number of clusters, we chose to use a hierarchical clustering method. This algorithm uses the first n frames to initially determine the number of clusters at time n. Next, K-Means clustering clustering is used to update the clusters at each subsequent time step. K-Means works in an iterative manner, updating the clustering as each flow curve is extended into the current frame.

The hierarchical clustering method used for initially determining the number of clusters begins by forming singleton clusters with each flow curve calculated over the interval [0,...,n]. The two closest clusters are then combined. At each step, the curvature and velocity values of the merged clusters are averaged and become the curvature and velocity values of the new cluster. The iterative combination of pairs of clusters with the smallest inter-cluster difference continues until only one cluster remains. This clustering process forms a binary tree, with the singleton clusters at the leaves and the final cluster at the root. The difference between two adjacent nodes in the tree is defined by the difference between the two associated clusters. See Figure 4.2.

Clearly, in most cases, clustering should stop before only one cluster remains. One heuristic, suggested by Romesburg [Rome84], determines the number of clusters such that the decision is least sensitive to error in the difference measure. Figure 4.2(b) shows the number of clusters obtained for different widths of the ranges of difference between the clusters in Figure 4.2(a). Romesburg suggests that the decision to terminate merging pairs of clusters is least sensitive to error when the width of the range is the largest. For Figure 4.2 this results in deciding that two clusters exist in the data.

We use a heuristic similar to Romesburg to identify the number of clusters and hence number of moving objects. Ideally, there should be a "clear" point, indicated by a large difference, where, by combining clusters, flow curves from different objects are merged. When the minimum difference between clusters is greater than a threshold times the minimum difference in the previous iteration, hierarchical clustering terminates. The number of clusters at this point is the number used for further processing.



(a)

# Clusters	Range of difference	Width of range
8	$0.0 < diff < 0.35$	0.35
2	$0.78 < diff < 1.04$	0.26
3	$0.56 < diff < 0.78$	0.22
5	$0.35 < diff < 0.54$	0.19
4	$0.54 < diff < 0.56$	0.02
1	$1.04 < diff < \infty$	-

(b)

Figure 4.2: (a) The leaves show the initial singleton clusters. The vertical axis indicates the difference between the two clusters being merged. For example, the difference between items 1 and 2 is 0.35. The difference between items 3 and 4 is also 0.35. The difference between these two merged clusters is 0.54. (b) Number of clusters obtained for different widths of the ranges of difference for the tree shown in (a).

Romesburg's heuristic is desirable since it does not require a threshold. But consider the situation where there are three moving objects and a static background. The number of clusters in this case should be four. Our method will terminate when there are four clusters remaining since the difference between all pairs of remaining clusters is high. But there is no reason to believe that the minimum difference with four clusters remaining will be greater than with three clusters remaining, or two clusters, etc., as required using Romesburg's heuristic.

Once the initial number of clusters has been determined and the flow curves have been clustered, a different, more incremental-type clustering is performed. For each new frame, the curvature and velocity of each flow curve's next point is computed. Then, for each flow curve, using a fixed width interval ending at the current time, the difference of the flow curve and its cluster's mean is computed. If this difference is greater than the difference of the flow curve and another cluster's mean, the flow curve is moved into the other cluster. This is called the K-Means clustering method [Hart75, Rome84]. Note that the number of clusters does not change, but it allows flow curves to move between existing clusters. This is important in areas of occlusion and disocclusion.

4.4 Occlusion and Disocclusion Detection

Detection of areas of occlusion and disocclusion is straightforward once the initial clusters have been computed. Occlusion is characterized by a cluster associated with an occluded surface merging into a cluster associated with an occluding surface. Occlusion areas are areas where this merging occurs. Disocclusion areas exist where flow curves split from one cluster to another. In practice, clusters do not actually split since once a flow curve is following the ST surface flow associated with an object, it cannot flow out of this area into the ST surface flow area associated with the object being disoccluded. Consequently, disocclusion results in areas where no flow curves exists. So rather than using splitting clusters, disocclusion is detected by noting areas where flow curves do not exist.

The two intuitive observations that form the basis of our algorithm were defined earlier as: a point in an image can only move to at most one point in the next image, and a point in an image can come from at most one point in the previous image. Figure 1.1(b) shows the ST surface flow for a scene with two objects. In regions away from occlusion boundaries, these observations are not violated. However, at occlusion boundaries two points flow to one point in the next frame.

Figure 1.1 shows the ideal case where an occlusion discontinuity occurs. In practice this discontinuity appears as a strip of random flow with the width of the strip dependent upon the neighborhood size used to compute the ST surface flow. This occurs because gradient-based methods to compute ST surface flow are undefined at occlusion and disocclusion boundaries. A method that tries to find these occlusion and disocclusion boundaries by looking locally for this discontinuity will have to actually look for a strip of random-like flow.

If we ignore these unstable boundary areas by simply extending the flow curves, they will eventually leave the unstable strip and enter a more stable area, an area away from an occlusion or disocclusion boundary. Figure 4.8 shows the flow curves for an image sequence where one object is moving in front of another object. The flow curves generated by the occluded surface quickly move though the area where the flow is undefined and merge with the flow curves generated by the occluding surface.

Our two observations concerning the uniqueness of point motion can be viewed on two

different levels. Viewed locally, the the observations do not hold true when two points flow to one point in the next frame or a single point flows to more than one point in the next frame. Viewed globally, the the observations are violated when flow curves from one cluster merge with another cluster or flow curves split from a cluster. By using the observations at the global level, we avoid having to rely on areas where the ST surface flow is undefined.

Given the neighborhood size used to compute the ST surface flow, the width of the random-like strip can be estimated. The temporal neighborhood used to compute the difference between flow curves can be chosen large enough so that any noise in the curve due to the occlusion region will be minimal. Therefore it is expected that a flow curve will stay in the cluster representing the occluded object and then switch into the cluster representing the occluding object.

Once a flow curve has merged into another cluster and has remained there for some period of time, it could be deleted. Other than computation time, there is no reason for this step, however.

The situation is similar at disocclusion boundaries. Flow curves must be created in regions where the density of flow curves drops below some threshold. Note there is no problem if too many curves exist. Near boundaries of disocclusion it is expected that the density of flow curves will decrease because all curves will follow the disoccluding object and no flow curves will follow the disoccluded object.

4.5 Results

In this section results of the algorithm on five image sequences are presented. Also, while not necessary from a theoretical standpoint, an additional flow curve descriptor is presented that helps to correctly cluster flow curves. We also show why it is desirable to use the position of flow curves when computing the difference between them.

Figures 4.3 to 4.7 show the flow curves and resulting clusters for five image sequences. The set of clusters are the result of the initial hierarchical clustering algorithm. The results of the subsequent cluster updating procedure using K-Means is shown in Figures

4.8 and 4.9. Figure 4.10 shows the occlusion boundaries resulting from the sequences shown in Figures 4.6 and 4.7. All image sequences were synthetically generated by transforming a planar region of a phone book page over a larger image of a table top. Each image sequence was smoothed using a 3D Gaussian-weighted kernel with standard deviation of 2 pixels. ST surface flow was computed at every fourth pixel in every frame. See Chapter 3 for details on computing ST surface flow. The resulting flow for each sequence was convolved with a $3\times3\times3$ median filter and smoothed using a 3D Gaussian-weighted kernel with standard deviation 0.66. Flow curves started in the first frame are spaced 16 pixels apart resulting in 196 flow curves.

Figure 4.3 shows hourglass shaped flow curves resulting from a rectangular region rotating about a vertical line through the middle of the region. Flow curves near the center line of rotation are almost straight and therefore clustered with similarly shaped flow curves associated with the background.

Figure 4.4 shows corkscrew shaped flow curves resulting from a circular region rotating parallel to the image plane. The curvature of flow curves near the center is close to zero and increases for curves farther away from the center. Since the curvature values are similar, the curves were clustered together. The velocity vectors are in all directions so they do little to help the curves form a cluster. However, the *speed* of the vectors are similar. Speed is defined as the length of the spatial component of the velocity vector. Because it does not measure direction, it is unaffected by velocity vectors that are oriented in different directions. Using speed when measuring the difference between curves and clusters amounts to using the heuristic that flow curves associated with a single object move with similar speed in the ST cube. Note that speed could not be used instead of velocity since two objects translating with the same speed but in different directions would then be clustered as one object.

In general, using both speed and curvature may still not be enough to cause the flow curves near the center of rotation to be clustered with flow curves farther from the center. This is because the curves near the center of rotation are shaped like the curves associated with a static background; therefore, no matter what difference measure is used there may be problems computing the correct clusters if only motion is used. To alleviate this problem, the position of flow curves can also be used. By including position

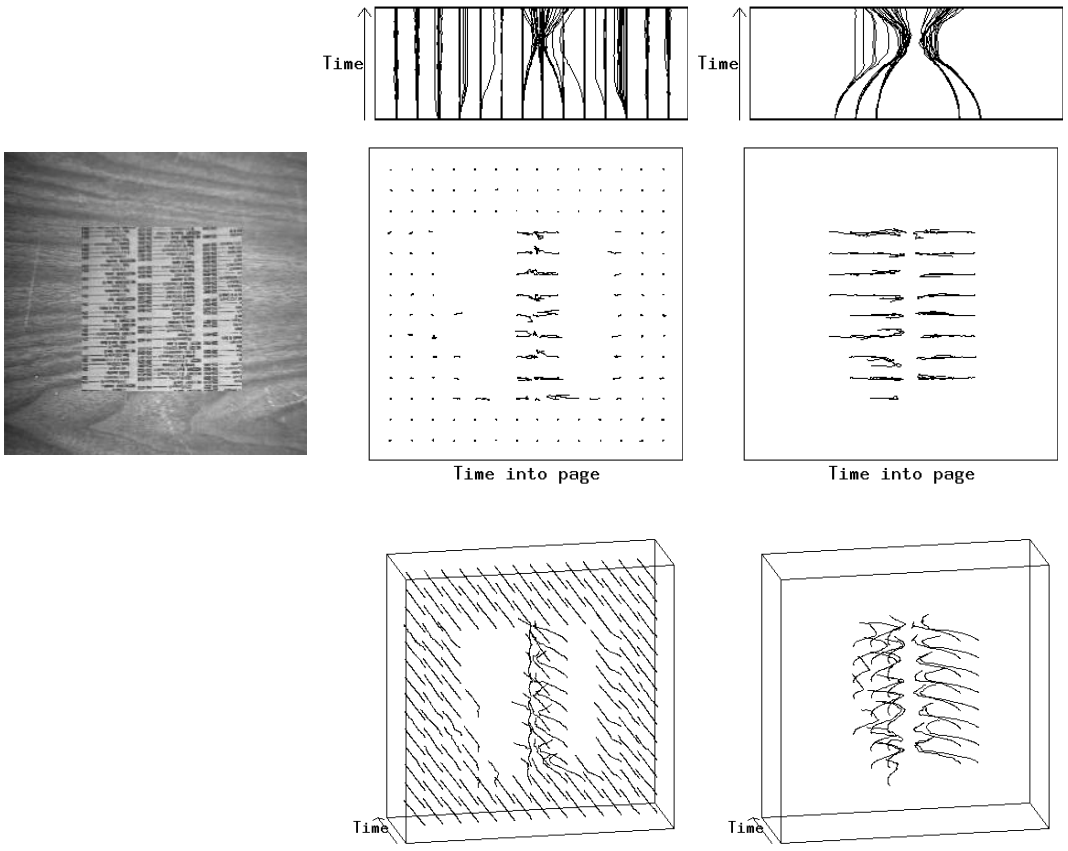


Figure 4.3: ST flow curves from a 115 frame sequence of a phone book page rotating against a static background about a vertical line through the middle of the page at 0.02 radians/frame. One frame of the sequence is shown on the left. The center and right columns show the top, front and oblique views of the two clusters. Note that some flow curves far from the center of rotation failed to follow the rotating page. Most of these curves then followed the background and became part of the background cluster as shown by the top view of the background cluster. The flow curves near the vertical line of rotation are almost straight and therefore clustered with the background.

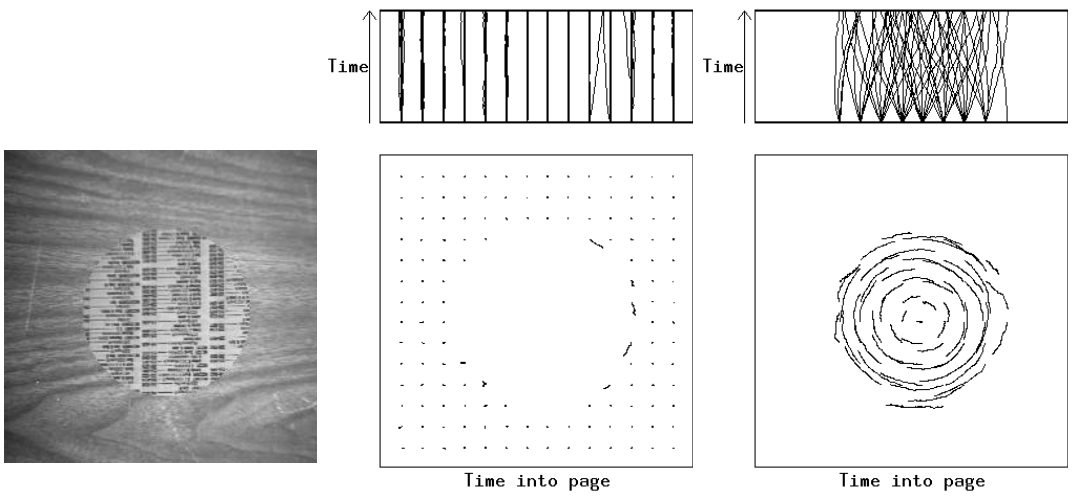


Figure 4.4: ST flow curves from a 115 frame sequence of a phone book page rotating about its center at 0.008 radians/frame against a static backgound. One frame of the sequence is shown on the left. The shape of the flow curves resulting from the background (center) and the phone book page (right) are straight and corkscrew shaped, respectively.

information, flow curves near the center will cluster with those curves nearby it rather than with areas farther away. To compute the spatial distance between two clusters, the minimum spatial distance between all pairs of curves is used.

Using distance when measuring the difference between curves and clusters amounts to using the heuristic that flow curves associated with an object are near each other. Since the nearest neighbor measure is used to compute the distance between clusters, this heuristic does not imply that a cluster cannot extend across large areas. Also, while the nearest neighbor measure may force flow curves near the center of rotation of a disk rotating parallel to the image plane (Figure 4.4) to be clustered correctly, it does not necessarily force flow curves near the center of rotation of a region rotating in depth (Figure 4.3 and 4.5) to be clustered correctly. The power of this heuristic depends upon the smallest distance between the center of rotation and the background. If this distance is small, or zero as in Figure 4.3, the heuristic has no effect. If this distance is large, as in Figure 4.4, the heuristic has the desired effect.

As shown in Figure 4.6 and 4.7 the results of the hierarchical clustering are not always perfect. Extra clusters result in these sequences because of the non-homogeneity of the

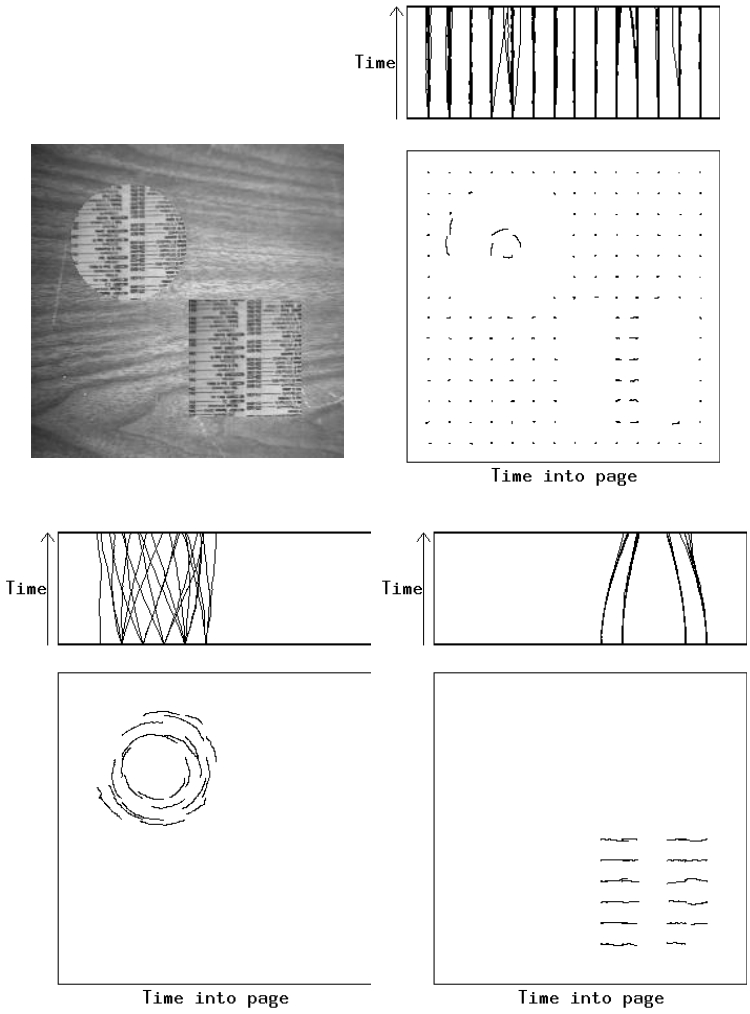


Figure 4.5: ST flow curves from a 115 frame sequence of a phone book page rotating about a vertical line through the middle of the page at 0.02 radians/frame and a phone book page rotating about its center at 0.01 radians/frames. One frame of the sequence is shown at the top. The front view of the resulting clusters is shown below. The flow curves near the centers of rotation are almost straight and therefore clustered with the background.

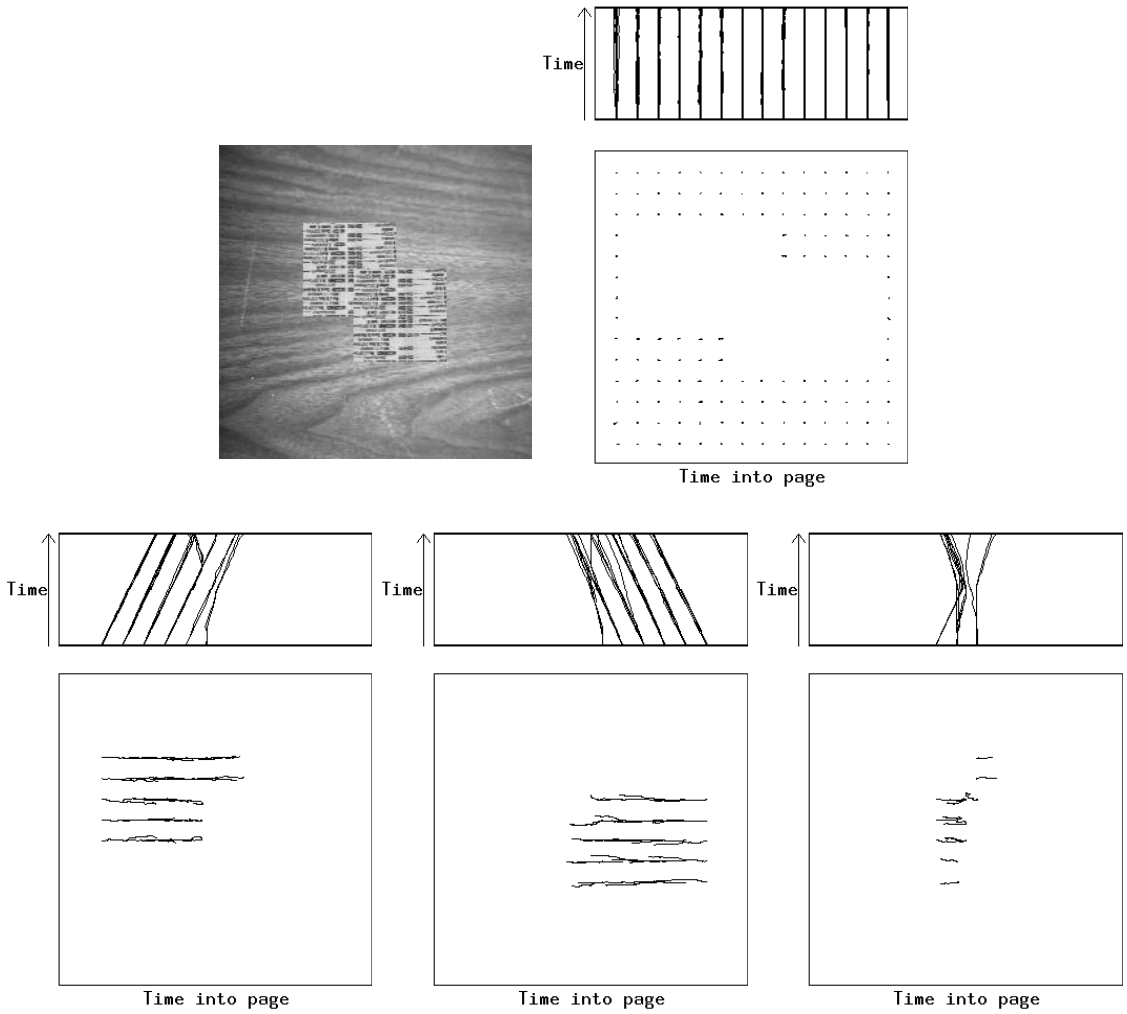


Figure 4.6: ST flow curves from a 115 frame sequence of two phone book pages, one translating left at 0.4 pixels/frame and partially occluding the other, which is translating right at 0.4 pixels/frame. One frame of the sequence is shown on the left. The front and top views of the resulting four clusters are shown. A fourth cluster results in the area where the two pages overlap because the flow curves in this area are shaped like flow curves generated by the left page for a while then shaped like the flow curves generated by the right object. This results in the flow curves shaped like no other flow curves so a fourth cluster remains. As the flow curves were extended into subsequent frames, the flow curves in the extra cluster became shaped more like the flow curves following the occluding object. K-Means then moved the flow curves into the cluster associated with the occluding object. This is shown in Figure 4.8.

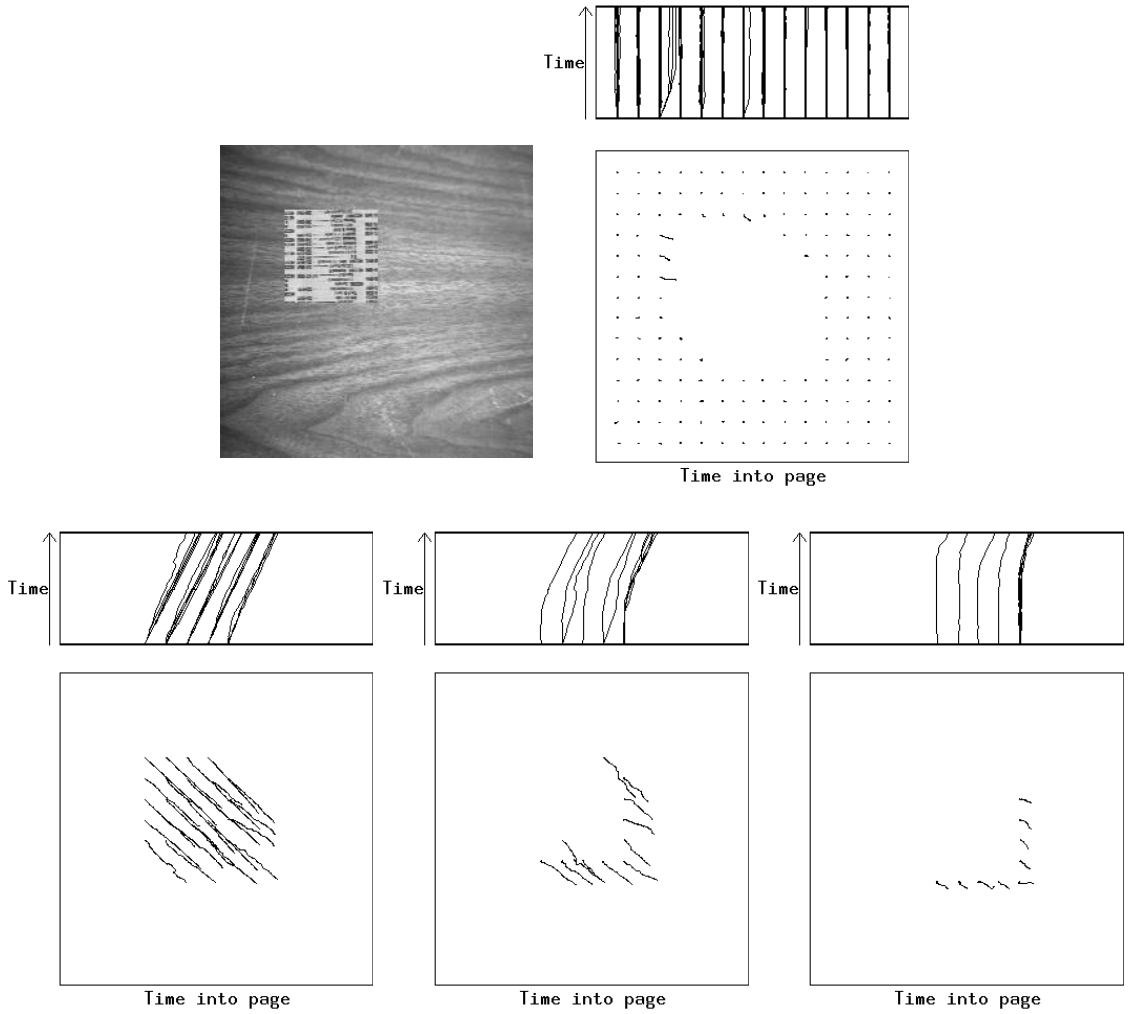


Figure 4.7: ST flow curves from a 115 frame sequence of a phone book page translating down and right at 0.4 pixels/frame. One frame of the sequence is shown on the top left. The front and top views of the four clusters resulting from the hierarchical clustering are shown. Two unexpected clusters resulted in areas where occlusion takes place through the initial portion of the sequence.

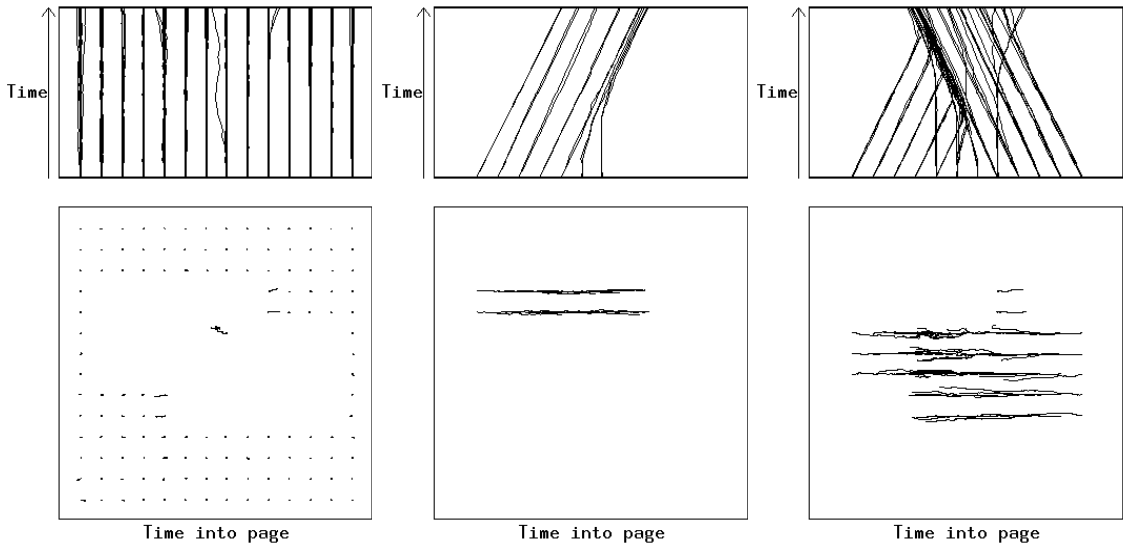


Figure 4.8: ST flow curves from Figure 4.6 extended to 170 frames. The top and front views of the three clusters resulting from K-Means are shown. The number of flow curves in the fourth cluster was low enough that it was deleted. All of the flow curves that were generated by the lower part of the object translating right have merged into the cluster associated with the object translating left. Also, flow curves generated from the background have merged into the clusters for the translating objects. There are two flow curves in the right cluster that should be in the center cluster. These errors are corrected by K-Means after the flow curves are extended a few more frames.

flow curves though the beginning of the sequence. Flow curves that are near occlusion boundaries are shaped like flow curves generated by one object for a while then shaped like flow curves generated by another object. Also, the times that the curves switch from following a partially occluded object to following an occluding object vary, resulting in differently shaped flow curves.

In Figure 4.6 a fourth cluster results in the area where the two pages overlap because the flow curves in this area are shaped like flow curves generated by the left page for a while then shaped like the flow curves generated by the right object. This results in the flow curves shaped like no other flow curves so a fourth cluster remains.

In Figure 4.7 two unexpected clusters result in the area near the occlusion boundary. Flow curves in the area where the background is just becoming occluded form one cluster and flow curves in the area that has been occluded longer form another cluster. This

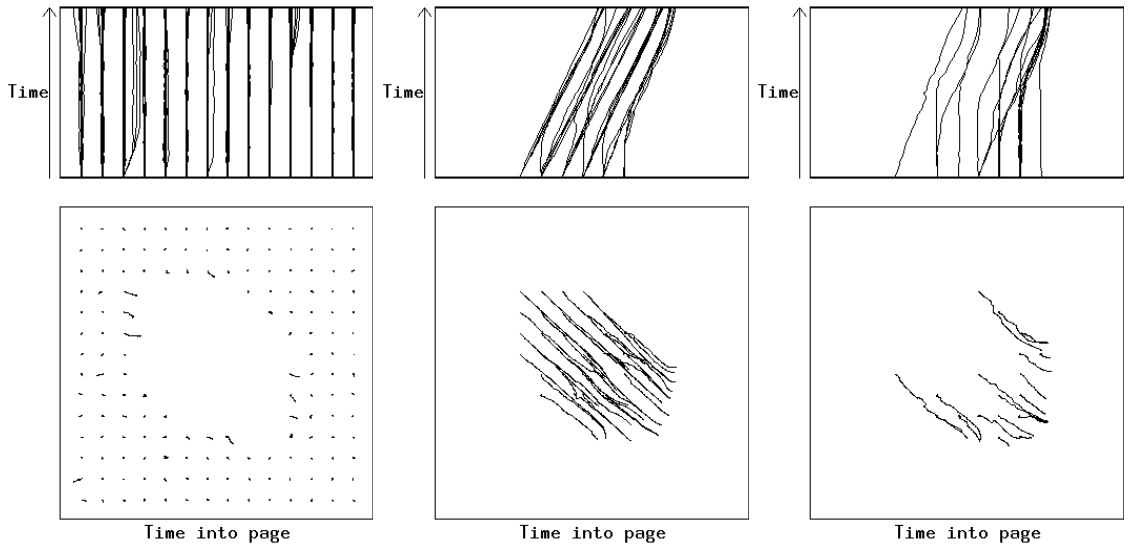


Figure 4.9: ST flow curves from Figure 4.7 extended to 170 frames. The top and front views of the remaining three clusters resulting from K-Means are shown. One of the unexpected clusters was deleted when the number of flow curves in the cluster dropped below threshold. An extra cluster of flow curves near the occlusion boundary remains. Flow curves generated from the background near the leading edge of the page have merged into this additional cluster. As the sequence is extended, the flow curves in the extra cluster will move into the cluster associated with the translating object.

results in four groups of differently shaped flow curves: straight with no spatial component (background), straight with a spatial component (page translating), bent in the middle (became occluded part way through the sequence) and curved at the end (just became occluded).

Some of the errors in the sequences shown in Figures 4.6 and 4.7 are corrected by K-Mean clustering. As discussed below, when the number of flow curves in a cluster drops below threshold, the cluster is deleted. The extra cluster in Figure 4.6 and one of the extra clusters in Figure 4.7 are deleted in this manner. However, one extra cluster remains for the single translating page sequence (see Figure 4.9). In order to conclude that this cluster should be deleted, reasoning about the merging of flow curves between clusters has to be performed. Typically, flow curves move from the background cluster into the extra cluster and then move into the cluster associated with the translated page. This "detour" into the extra cluster is indicative of a cluster resulting from an occlusion

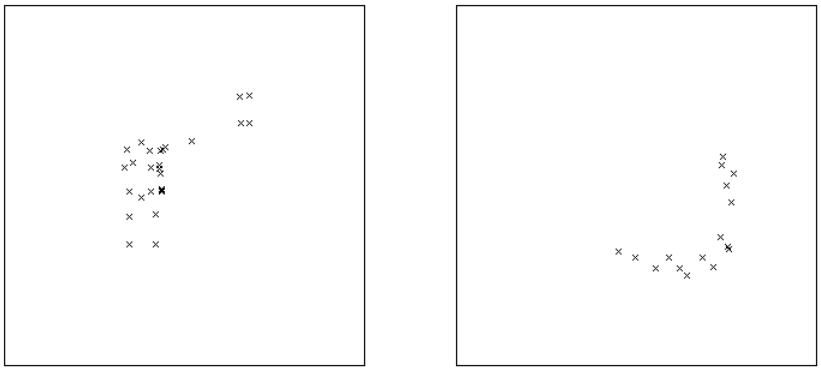


Figure 4.10: Occlusion boundaries from Figure 4.8 where two regions translate (left), and from Figure 4.9 where the region translates down and right (right). The position of each "x" indicates the position where a flow curve changed clusters. The occlusion boundaries for the left image also show the background flow curves merging into the phone book pages and the flow curves associated with the left page merging into the right page. The occlusion boundaries for the right image show the background flow curves merging into the phone book page.

area and should be deleted.

In the current algorithm, when computing the difference between curves and clusters, all components used to compute the difference are weighted equally. With the exception of the weight for the position of flow curves, similar results were obtained for other combinations of weights. When the weighting for position was low, flow curves near the center of rotation were clustered with the background. The fixed interval used to compute the difference between curves and clusters, was 33 frames. However, similar results were obtained when 23 frames were used. Based on empirical results, when the minimum distance between clusters is greater than 3.5 times the minimum difference in the previous iteration, hierarchical clustering was terminated and the remaining clusters were then used by K-Means. When the costs between clusters is very small it is common to that the minimum difference is greater than 3.5 times the minimum difference in the previous iteration. In order to prevent premature termination of hierarchical cluster, the minimum difference between clusters must be greater than a threshold in order for hierarchical clustering to terminate.

It is possible for an entire object to become occluded by another object. In this case

the cluster associated with the occluded object should be deleted when the number of flow curves in the cluster becomes small. Based on the initial spacing of flow curves, this threshold was chosen to be 10. This threshold was also used for deleting clusters of erratic flow curves resulting from occlusion. For example, Figure 4.6 shows a cluster which resulted because flow curves followed one object for a while then followed another after the object became occluded. During K-Means the flow curves merged into the cluster for the occluding object and the cluster for the occluded object was deleted because its size fell below the threshold.

4.6 Concluding Remarks

We have presented a method that takes advantage of temporal coherence over very long image sequences. Flow curves are recovered from the ST surface flow. Flow curves are then clustered such that each cluster represents the motion of a single surface or object through the ST cube. By analyzing the merging and splitting of clusters, occlusion and disocclusion boundaries are identified.

With coherent areas of the ST cube identified, we have a good starting point for higher-level understanding of the motion in an image sequence. For example, each area can be reduced to a single representative flow curve. This results in a significant reduction in the amount of data used in further processing. This is an important property in spatiotemporal image analysis since the amount of data is very large. With these representative flow curves, motion recognition can be performed without prior knowledge of the objects undergoing motion. Johansson's results with MLD's suggests that this is possible and that using flow curves is a promising approach [Joha73].

The next chapter will present an intermediate-level motion problem, cyclic motion detection, and show how ST flow curves can be used to detect cyclic motion without prior recognition of the objects undergoing the motion. Chapter 6 will discuss possible future approaches to computing high-level motion descriptions using ST flow curves.

Chapter 5

Cyclic Motion Detection

In Chapters 3 and 4 low-level and intermediate-level representations of motion were defined. In this chapter this representation is used to detect cyclic motion. There are three reasons for examining this problem. First, the problem of cyclic motion detection is an interesting problem worthy of study. Section 5.1 will explain how this problem, while recognized by the psychology community, has been largely ignored by the computer vision community. Just as the human visual system can recognize objects and their motion using only motion information (as in Johansson's MLDs), the human visual system (HVS) does not require prior recognition of objects in order to detect cyclic motion. The second reason for studying the cyclic motion problem is to test the computational plausibility of motion-based recognition. Specifically, how can cyclic motion be recognized without prior object recognition? Finally, while examining the properties of cyclic motion, it becomes clear that the representation laid out in Chapters 3 and 4 is very appropriate for this problem. So this chapter is also a verification of the representational power of spatiotemporal (ST) flow curves.

This chapter is organized as follows. In the next two sections the cyclic motion problem and its properties are described. In Section 5.3 cyclic motion is formally defined as repeating curvature along a path of movement. ST surfaces are introduced in Section 5.4 as the appropriate low-level motion description from which cyclic motion is recovered. Section 5.4 continues by showing that cyclic motion in the scene that is preserved under projection is retained by the ST surfaces. It is then shown how to recover the cyclic

shape of ST surfaces using ST curves and curvature scale-space.

While all the methods presented in this chapter can be extended for use with a sequence of gray-level images, we restrict our discussion to edge image sequences. Since the goals of this chapter are to show the plausibility of detecting higher-level motion prior to object recognition and to show the usefulness of ST flow curves, we will not examine how to recover ST flow curves using ST surface flow from an edge image sequence. Rather, we will detect ST flow curves more directly by tracking points. However, nothing excludes the use the more general approach using ST surface flow.

5.1 Problem Description

Many natural objects undergo cyclic motion. Examples include a human walking, a person riding a bike, a running dog, a swinging pendulum, and a bouncing ball. The fact that people perceive these motions as repeating demonstrates that the HVS is capable of detecting cyclic motion. The study of humans' ability to perceive a stimulus as cyclic[1] dates back to the late 19th century [Bolt94] and has continued as an active area of research [Frai78]. While the ability of the HVS to detect cyclic stimuli has been recognized [Frai78], there is little research in the area. Most of the research concentrates on the human perceptual system's ability to perceive auditory rhythm. In this chapter we will examine the detection of cyclic stimuli in vision.

Clearly, the perception of rhythm can be accompanied by a recognition of the rhythm. For example, when Gabrielsson [Gabr73] presented subjects with complete musical patterns and measured their perception of rhythm, the subjects could also recognize the pattern if they had heard it before. The analog of this in the visual domain was shown by Johansson [Joha73] using moving light displays (MLD). While the recognition of the rhythms was not central to those studies, one can extrapolate from the results and conclude that the perception of rhythm is accompanied by recognition. This chapter addresses the detection of cyclic motion within the context of motion recognition.

There has been considerable work examining the HVS's ability to recognize movement

[1]The psychology community refers to a repeating stimulus as "rhythmic" rather than "cyclic".

that contains cyclic motion, but the detection of the cycles was not the focus of the studies [Joha73]. Johansson's MLDs contained cyclic motion in the form of human walking, running, cycling and dancing. When subjects were presented with an MLD of walking, they always recognized the motion after the first one or two steps.

Unfortunately, Johansson's work examined subjects' ability to recognize the high-level motion of an MLD, not any particular cyclic motion contained in the MLD. In this chapter we address the problem of recovering a cyclic motion description from an input sequence using computational methods. This problem, while recognized by the psychology community, has been largely ignored by the computer vision community. By addressing the computational issues we can better understand the HVS's ability to perform this task.

5.2 Properties of the Cyclic Motion Detection Problem

By examining the HVS's ability to recover and describe cyclic motion, we can learn pertinent properties of the cyclic motion detection problem. Further, these properties give a better understanding of the problem and suggest a form of a solution. These properties include: 1) recognition of objects is not necessary for the recovery of cyclic motion; 2) many frames are necessary before cyclic motion becomes evident; 3) cyclic motion does not depend on absolute position; and 4) cyclic motion can occur at multiple scales.

In order to substantiate the hypothesis that motion recognition can occur before object recognition, we will show in this chapter that detecting cyclic motion in an image sequence can be performed without prior recognition of the object undergoing the cyclic motion. Other results also indicate that the perception of intermediate descriptions is accomplished prior to recognition of the stimulus. For example, Barrow and Tenenbaum [Barr81] used photomicrographs of pollen grains to argue that the HVS can recover surface shape and orientation without recognition. When Gabrielsson [Gabr73] presented subjects with songs, they could detect rhythm within the music even without having

heard the song previously. These psychophysical results suggest that intermediate-level descriptions can be recovered prior to object recognition. Further, it is reasonable to conclude that if the HVS is presented with a "movie" of photomicrographs of pollen grains moving in a cyclic fashion, that the HVS could detect the cyclic motion even though it has no familiarity with the objects.

Since cyclic motion of non-recognizable objects, e.g., pollen grains, can be recovered, static models of objects are not a necessary component of a solution. Rather than a model-based solution, a motion-based solution is required. It is not the object itself that is the appropriate level of motion description, but rather the motion of lower-level primitives such as surfaces.

Cyclic motion is an important problem because it is an example of an intermediate-level motion description. It is a lower-level description than walking, for example, but it requires more than simply making changes between a few frames explicit, as in low-level motion analysis. Since cyclic motion is an intermediate-level description, one would expect that it should be computed from a low-level motion description. The fact that even the simplest of cyclic motions, e.g., a swinging pendulum, requires many frames before its cyclic behavior is recoverable, suggests that the low-level motion representation must describe long-range motion sequences.

Recently, there has been a trend toward using more than a few frames to analyze an image sequence [Jain88]. Cyclic motion description requires long sequences of frames since many frames are required before a motion repeats. These considerations suggest a low-level representation that coherently represents many frames, i.e., an ST cube. The concise, coherent nature of this representation has also lead others to use it to examine long sequences of frames [Bake88b]. Because of the coherent nature of the cube, ST surface flow and ST flow curves can be computed. Since ST flow curves represent the motion over long sequences, it is an ideal representation from which to detect cyclic motion.

Just as cyclic motion detection does not depend on the object undergoing the cyclic motion, as we argued earlier, it also does not depend on the absolute position where the motion takes place. It is not the position that is essential to characterizing cyclic motion, for example, but rather how the object moves that is important. For example,

the perception of cyclic walking motion is not dependent on the absolute position of the person because the person is never in the same place at any two separate times. So the definition and detection of cyclic motion must be invariant to position.

Finally, cyclic motion can occur at multiple scales. For example, consider a bird flying in a cyclic pattern. The bird displays the cyclic motion of its flight pattern and also the finer cyclic motion of its flapping wings. Thus, a complete cyclic motion description should describe all levels of cyclic motion.

In summary, we have argued that cyclic motion detection and description

- does not depend on prior recognition of objects
- does not depend on absolute position information
- must make use of long-range temporal sequences
- must be sensitive to multiple scales

5.3 A Formal Definition of the Cycle Detection Problem

Before we can define cyclic motion we must have a representation of motion. We need a representation that makes the essence of cyclic motion explicit. One possibility is to represent the motion of an object by its position over time. However, we are more concerned with *what* kind of motion occurs rather than *where* it occurs. Thus, it is not position that is essential to characterizing cyclic motion, but rather how the object moves that is important. For example, the motion produced by a person walking parallel to the image plane is considered cyclic. But clearly this perception of cyclic motion is not dependent upon the position of the walking person because they are never in the same place at any two separate times. We need a definition of cyclic motion that is invariant to the position of the object. Curvature and torsion along the path of movement of an object represents movement and is invariant to position. In fact, the curvature and torsion along the path uniquely defines the path up to a rigid transformation [DoCa76]. However, for our purposes, curvature alone will be sufficient to describe how an object

moves. We choose this over torsion because it is possible that some object's motion is restricted to a plane, e.g., a swinging arm. The torsion along this path of motion would always be 0 since torsion along a planar curve is always 0. Curvature, on the other hand, will not always be 0.

In the remainder of this section we define the cycle detection problem for a single point and then extend it to rigid objects. Having defined cyclic motion, we will apply the definition to ST surfaces and curves. We will prove that if an object undergoes 3D cyclic motion which is preserved under projection, then the resulting ST surfaces must retain this cyclic information. We will show how to place ST curves on ST surfaces and how to use these curves to recover the cyclic information.

5.3.1 The Cycle Detection Problem for a Point

Consider the situation of a single point moving in three dimensions. Let $\alpha(t) = (x, y, z)$ parameterize the path of a point. Since α is a space curve, properties associated with space curves, e.g., curvature, can be applied to the curve defined by α. We make no assumptions about the smoothness of α.

Let $\kappa(t)$ define the curvature at $\alpha(t)$. Since we made no assumptions about the smoothness of α it is possible that for some t, $\kappa(t)$ is undefined. In this case we will say that $\kappa(t) = \infty$. And for t_1 and t_2, if $\kappa(t_1) = \infty$ and $\kappa(t_2) = \infty$ then $\kappa(t_1) - \kappa(t_2) = 0$. We say that α undergoes cyclic motion with period of length $t_2 - t_1$ if:

$$\max\{|\kappa(t) - \kappa(t + (t_2 - t_1))| \quad : t \in [t_1, t_2]\} < \epsilon$$

With this definition, only two periods of cyclic motion are needed before it can be classified as cyclic.

This definition refers to motion in the 3D scene. When we make use of this definition in later sections, we will refer to 3D cyclic motion that is preserved under projection, i.e., cyclic motion along the axes perpendicular to the line of sight. Nature does not usually conspire against us so we assume that if there exists cyclic motion in the image sequence, then there existed cyclic motion in the scene.

There are a few points worth making about the types of movement that would be considered cyclic under this definition. Time intervals during which the curvature in the

intervals is similar are defined as cyclic. Consider a point moving in a straight line. The curvature is always zero. So given any two intervals, the curvature will always be equal during these intervals. In this degenerate case, a point displays cyclic motion with an arbitrary period. Similarly, a point moving in a circle with constant speed displays cyclic motion with an arbitrary period since the curvature is constant. This definition fails to define a cycle as one revolution of the circle.

Goddard [Godd89] used change in angular velocity as a primitive to detect movements such as walking. A point moving in a circle with constant speed does not change its angular velocity. So Goddard's primitive would also fail to define the period of cyclic motion as one revolution. Similarly, the Trajectory Primal Sketch of Gould and Shah [Goul89] does not directly represent position information so it would also fail to define the period as one revolution. A definition of cyclic motion that is sensitive to such cases would only obscure the type of motion of the point. As stated earlier, it is the type of motion that we wish to detect.

5.3.2 The Cycle Detection Problem for a Rigid Object

Given our definition of cyclic motion for a point, we now extend the definition to rigid objects. This extension is motivated by the following theorem:

Theorem 1 *If a point of a rigid body undergoes cyclic motion, then with probability 1 all other points on the rigid body undergo cyclic motion with the same period.*

Proof Sketch Any instantaneous motion of a rigid body in three dimensions can be uniquely described by a *twist* [Coxe61]. A twist is described by an axis l, an angular velocity ω, and a translational velocity v along l. Since a twist describes the 3D motion of any rigid object, it can be applied to an entire rigid object or a point on the rigid object. In other words, a point on a rigid object can be thought of as a rigid object. Therefore, the point's motion can be defined by a twist. Since a twist is unique, the twist describing the motion of the point must be the same as the twist describing the motion of the entire rigid object that contains the point. Further, the twist must be the same twist describing the motion of any single point on the object. The twists along the path

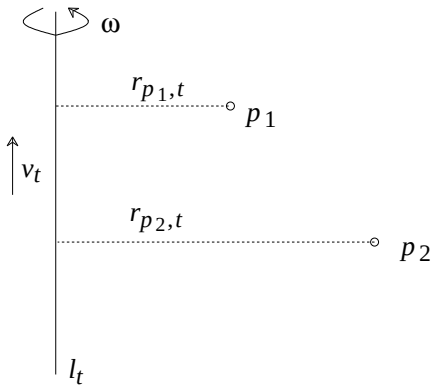


Figure 5.1: A *twist* at time t.

of a point determine the curvature of the path of the point. Let p_1 be the point feature of the solid object that displays cyclic motion. Let $twist_t$, defined by l_t, ω_t and v_t describe the motion for p_1 time t. The twists along the path of p_1 determine the curvature of the path of p_1. Let $\kappa(p_1, t)$ define the curvature of p_1 at time t. In the simplified case where v remains zero, $\kappa(p_1, t)$ equals $r_{p_1,t}$ where $r_{p_1,t}$ is the perpendicular distance from p_1 to l_t (Figure 5.1). In the case where v is not 0, the curvature along the path of movement of a point is a function of l, ω, v and their first and second partial derivatives with respect to arc length along the path of movement.

We need to show that given the curvature of p_1 at t, the curvature of another point p_2 at t is dependent upon the curvature of p_1 at t. That is,

$$\kappa(p_1, t) = \delta\kappa(p_1, t)$$

for some δ. From Figure 5.1 it is clear that in the simplified case where $v_t = 0$, $\delta = r_{p_2,t}/r_{p_1,t}$. In the case where $v_t \neq 0$, δ is a function of $r_{p_2,t}/r_{p_1,t}$, v and their partial derivatives. This shows that given the curvature of p_1 at t we can find the curvature of any other point at time t.

p_1 is the point undergoing cyclic motion. So the sequence of curvature values for p_1 is repeating. The sequence of curvature values for p_2 is determined by the sequence of curvature values for p_1. So if p_1 is cyclic, p_2 is also cyclic. This assumes that $r_{p_1,t}$ is never 0 so δ is always defined. In the continuous case, the probability that p_1 lies exactly on l_t is 0. Therefore, with probability 1, if a point on a rigid object undergoes cyclic motion,

then all other points on the rigid object undergo cyclic motion with the same period.

□

This theorem assumes the objects are moving in 3-space. One would like to know the effect of the projection into a 2D image. In this initial study we assume objects do not rotate in depth. So if a point projects into the image in one frame, it will always project into the image. If a point undergoes cyclic motion, its orthographic projection will also undergo cyclic motion, as will all other points of the rigid object that project into the image.

If any point on a rigid object undergoes cyclic motion, all other points on the rigid object undergo cyclic motion with the same period. So we say a rigid object undergoes cyclic motion if any point feature of the object undergoes cyclic motion. And a point undergoes cyclic motion as defined in Section 5.3.1.

5.3.3 The Cycle Detection Problem for an Articulated Object

An articulated object is composed of rigid objects connected by joints. A rigid part of an articulated object undergoes cyclic motion as defined above. We need a definition of cyclic motion for a set of rigid objects connected by joints. A set of rigid parts of an articulated object undergoes cyclic motion if each rigid part undergoes cyclic motion and there exists a dependency between the periods of cyclic motion. A dependency exists between solid parts if the ratio of their periods remains constant. For example, the ratio of the period of a swinging forearm and the period of the upper arm remains constant for a walking person.

This definition is motivated by the fact that most articulated objects that the human visual system would classify as displaying cyclic motion have this dependency property. The definition prevents jointed, rigid objects with no consistent pattern to the periods of their parts from being classified as displaying cyclic motion.

5.4 Finding Cyclic Motion of Articulated Objects

We argued earlier for the need of a low-level motion representation that can describe long-range motion sequences. ST volumes are 3D structures that are built by "stacking" a dense sequence of image frames. The ST volume is a viewer-centered, 3D (x-y-time) description. If we sample densely enough in time and an edge operator is applied, the ST volume will contain surfaces and volumes created by the surfaces. These surfaces and volumes represent object motion swept out through time.

Figure 5.2 shows two image sequences and the resulting ST surfaces. The first set of data consisted of 66 frames of a cube translating parallel to the image plane. Each frame is a 128 by 128 binary image created by a Difference-of-Gaussian edge operator. The second test data set consisted of 33 frames of a single object with one joint and its two major parts undergoing a "flapping" motion. The two wings "flap" with different periods.

The objects in the scene undergo cyclic motion and the resulting ST surfaces have an obvious cyclic shape. It is this cyclic shape that we wish to detect since cyclic motion in the scene generates cyclic ST surfaces. In order to detect this cyclic shape we require a description of the ST surfaces from which the cyclic shape can be detected. ST curves, defined in the next subsection, will be detected on ST surfaces such that if an ST surface is cyclic the ST curves will be cyclic. For example, Figure 5.3 shows ST curves, which are clearly cyclic, generated as a result of following the curvature extrema of the ST surfaces in Figure 5.2. So cyclic motion in the scene resulted in cyclic ST surfaces which resulted in cyclic ST curves. Finally we can detect the cycles in the ST curves by detecting repeating curvature values along the curves.

The first step is to show that cyclic motion in the scene generates cyclic ST surfaces.

Theorem 2 *If a solid object in a scene undergoes cyclic motion such that the cyclic motion is preserved under projection, the ST surfaces that correspond to intervals of cyclic motion will be piecewise isomorphic.*

Proof Sketch Consider an ST surface such that there exist ridges on the surface and every point on a ridge has the same value for the time coordinate. We call this ridge a

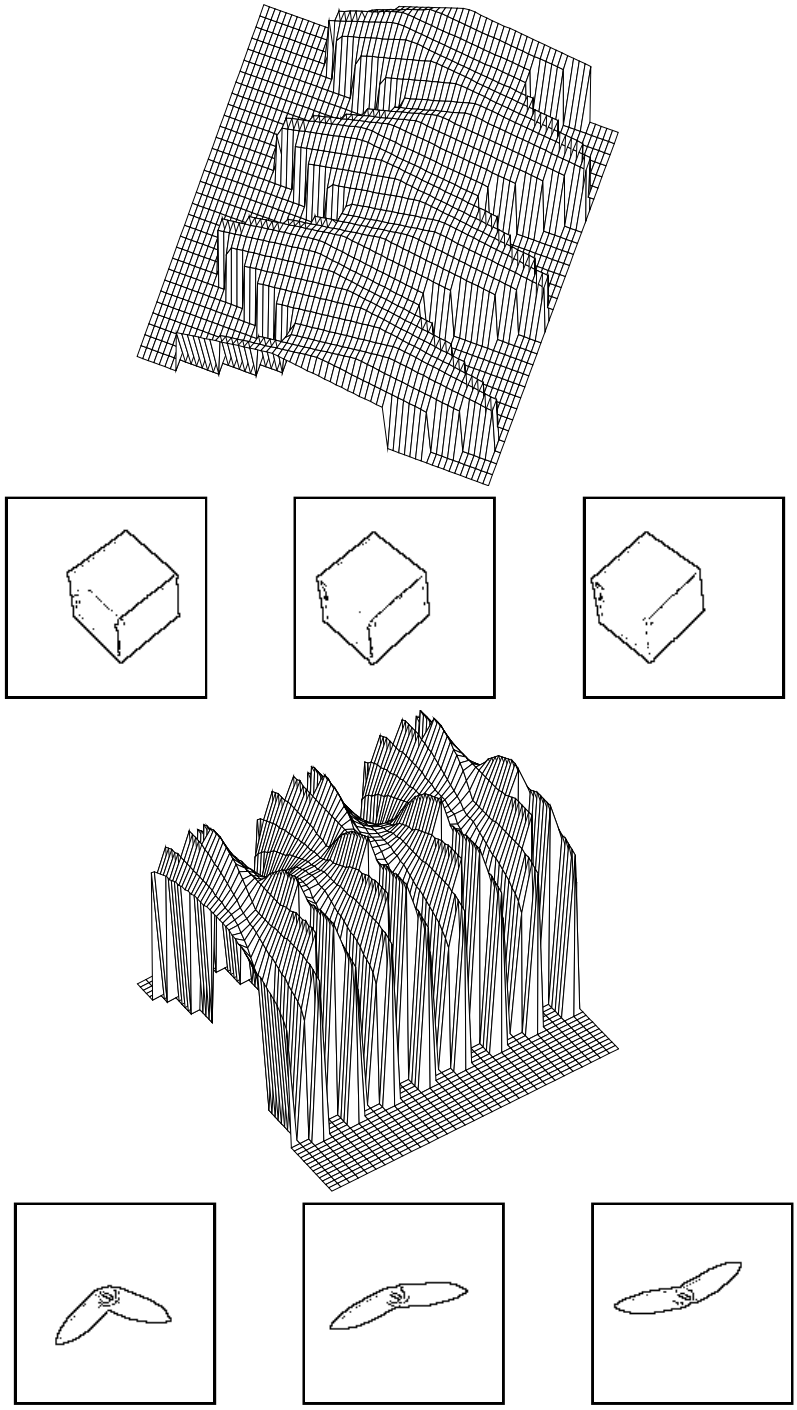
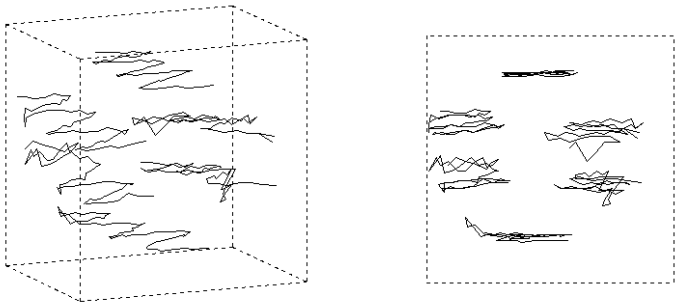
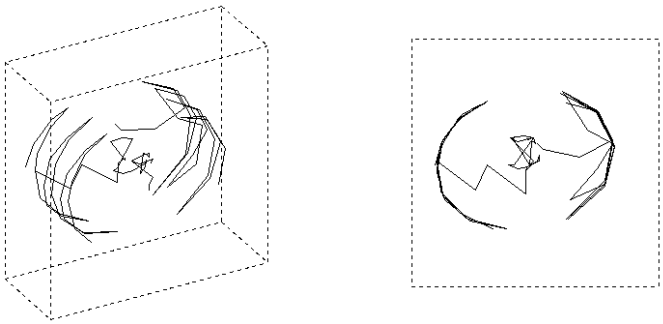


Figure 5.2: Three frames of the translating cube and flapping wings image sequence. The top half of the resulting ST surfaces are shown above each sequence.



(a)



(b)

Figure 5.3: ST curves for the translating cube image sequence (a) and for the flapping wings image sequence (b). Time is into the page for all views.

temporal ridge since it results from a non-smooth change of velocity of an object at some time. Define a *patch* of an ST surface as the surface between two temporal ridges. Two surfaces are piecewise isomorphic if the first patch of surface 1 is isomorphic to the first patch of surface 2, the second patches are isomorphic, etc. A curve, C, in a sequence of frames generates an ST surface. The type of movement, translation, rotation or a combination of both, determines the surface generated by C. Given a parameterization of the ST surface in terms of C and the movement of C, an isomorphic mapping between the corresponding patches of the ST surfaces can be defined. See Appendix A for more details.

□

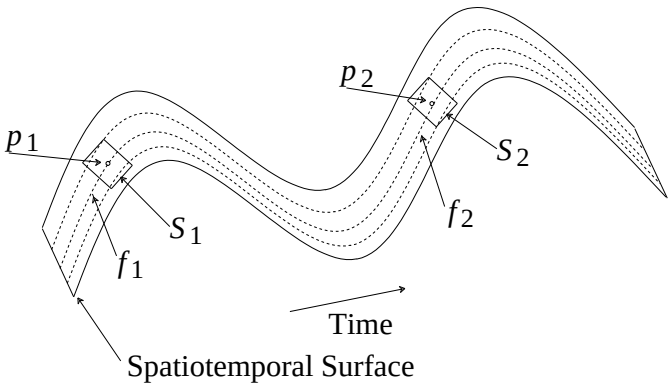


Figure 5.4: f_1 and f_2 are consistent since the tangent vectors of f_1 and f_2 at p_1 and p_2 are congruent.

5.4.1 Spatiotemporal Curves for Cyclic Motion Detection

In this section we show how to detect ST curves on ST surfaces so that a set of ST curves describes the surface. Figure 5.4 shows a sinusoidal surface with dashed ST curves that give a qualitative description of the surface. ST curves will be detected on ST surfaces such that whenever an ST surface is cyclic the ST curves are also cyclic.

An ST curve is any curve on a ST surface such that the time component of the curve is strictly increasing. Consider p_1 and p_2, two points on a ST surface (Figure 5.4). Let S_1 and S_2 be two congruent surface patches that contain p_1 and p_2, respectively. That is, S_1 and S_2 are neighborhoods of p_1 and p_2. Since S_1 and S_2 are congruent, there exists some translation, T, and/or rotation, R, which when applied to S_1 makes S_1 coincide with S_2. Given an ST curve f_1 containing p_1 and an ST curve f_2 containing p_2, f_1 and f_2 are consistent at p_1 and p_2 if the tangent vectors of f_1 and f_2 at p_1 and p_2 coincide after translation T and rotation R. f_1 and f_2 are consistent over an interval if they are consistent at every point within the interval. In other words, two ST curves are consistent if they go in the same relative direction for any two congruent neighborhoods of points on the ST curves. Note that f_1 and f_2 can be different intervals of the same ST curve.

Theorem 3 *If an ST curve on a ST surface is consistent and the surface represents cyclic behavior, i.e., if the surface is piecewise isomorphic, then the curvature of the ST curve will be cyclic.*

Proof If we can show that the curvature at every point of ST curve f_1 within surface patch S_1 equals the curvature at the corresponding point of ST curve f_2 within congruent surface patch S_2, f_1 must preserve any cyclic behavior present in the surface. So we must show that given a translation T and rotation R that makes S_1 and S_2 coincide, applying T and R to f_1 results in f_1 having the same curvature as f_2 at every point.

Since the ST curve is consistent, corresponding tangent vectors along the curve coincide after a translation and rotation. Since the tangent vectors along f_1 and f_2 coincide, we are left to show that rotation and translation of all the tangent vectors by T and R does not affect the curvature of the space curve. This is equivalent to showing that translation and rotation of a space curve does not affect its curvature [DoCa76].

Since the curvatures of f_1 and f_2 are always equal for corresponding points, the ST curves will be cyclic whenever the ST surface is cyclic.

□

From Theorem 3, our only requirements for detecting ST curves are that their temporal component is strictly increasing and they are consistent. If we take a temporal slice of an ST surface at time 0, we have the edge map of the image of a scene at that time. These edge points can be connected into contours and the curvature extrema detected. Doing this for each time slice and connecting corresponding curvature extrema over time defines a set of consistent ST curves.

In our implementation, curvature extrema, positive and negative, whose absolute value was greater than a threshold were detected in each frame and correspondence was established. The linking of the extrema into ST curves for the two sequences in Figure 5.2 are shown in Figure 5.3.

5.4.2 Finding Repeating Patterns in Spatiotemporal Flow Curves

Given a set of consistent ST curves extracted from ST surfaces, we can now infer cyclic motion from cyclic structure in the ST curves. All ST curves which lie on an ST surface generated by the projection of a rigid object, or rigid part of an articulated object,

will have the same period of cyclic motion. ST curves that track other point features, such as surface markings or vertices, will also have the same period of cyclic motion. We propose using curvature scale-space to detect repeating patterns in ST curves.

Mokhtarian [Mokh88] showed how to construct curvature scale-space images for a space curve [2]. It is constructed much like the traditional scale-space image of a 1D signal. A space curve is one dimensional with curvature values defined at every point along the curve. However, since it is not possible to give curvature a sign for non-planar curves, level-crossings rather than zero-crossings are used to construct the scale-space. The curve is smoothed by smoothing coordinate values of each point on the curve with coordinate values of neighboring points on the curve. As the curve is smoothed, the curvature level-crossings are recomputed. The horizontal axis of the curvature scale-space image is arc length and the vertical axis is the amount of smoothing.

Curvature scale-space has many properties that make it desirable, regardless of the application Mokhtarian [Mokh88]. Two properties, however, are particularly relevant to the problem of cyclic motion detection. Cyclic motion can occur at many scales. Since curvature scale-space represents curvature over many scales, it is a natural representation to use. Fine cyclic motion, e.g., the flapping wings of a bird, will be observable at fine scales, whereas coarse cyclic motion, e.g., a cyclic flight path of a bird, will appear at coarse scales.

Second, curvature scale-space is position invariant. As argued earlier, when looking for repeating patterns we do not want to concern ourselves with spatial information. Curvature scale-space is invariant to position since curvature is invariant to position.

For portions of the curve that are congruent, we expect the corresponding portions of the scale-space to be similar. Since scale-space represents large features as well as small ones, we can use a coarse-to-fine search procedure, first looking for repeating patterns at coarse scales where the matching is computationally less demanding, and then moving to finer scales.

Figure 5.5 shows curvature scale-space images for several ST curves. Features were defined as local maxima along with the two contours that extend to the left and the right

[2]Code for the calculation of curvature scale-space was provided by Farzin Mokhtarian.

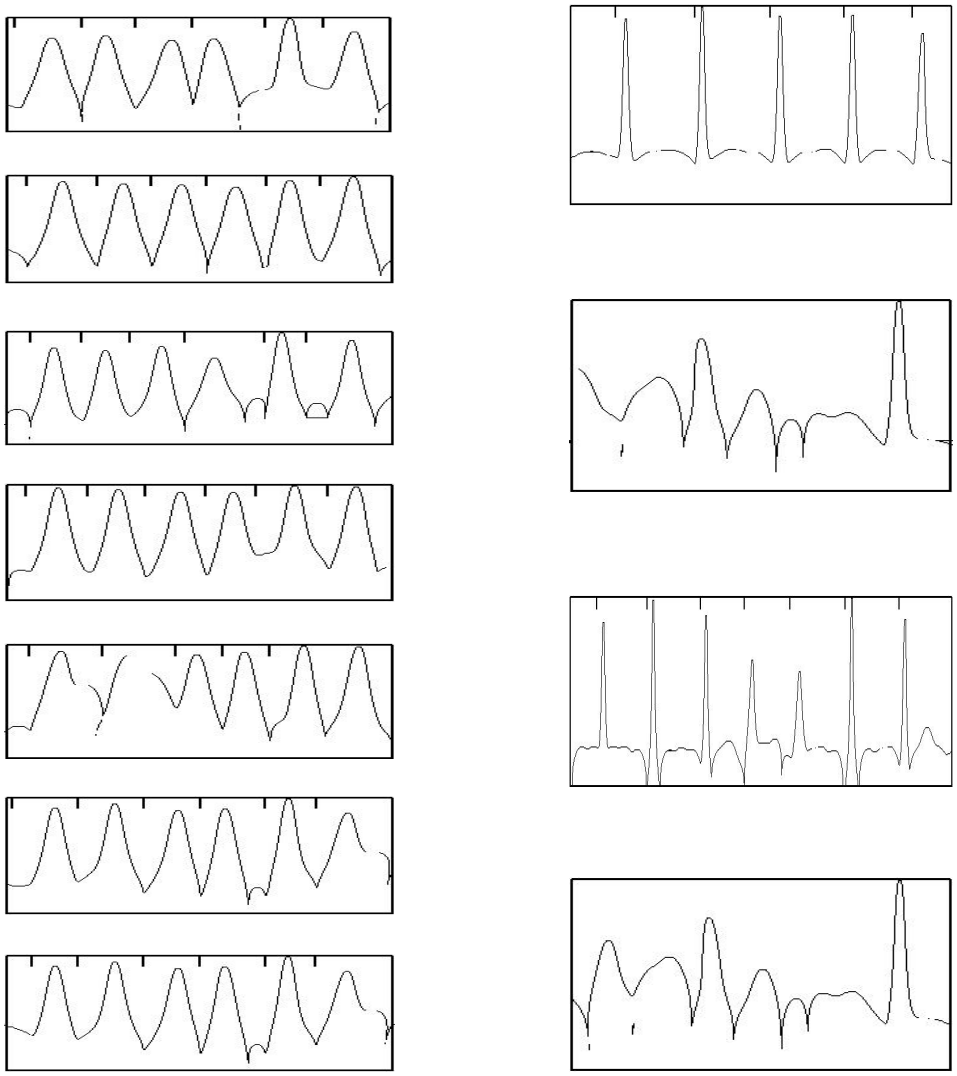


Figure 5.5: Curvature scale space images for a set of ST curves from the translating cube sequence (left) and the wings sequence (right). The horizontal axes are arc length and the vertical axes are the amount of smoothing. Cycles are marked with short tick marks across the top of each image. The cost associated with the second and fourth image on the right was so high that no cycles were found.

of each local maximum. Note that the contours do not extend to the finest scale as in traditional 1D scale-space images but the features still resemble arches.

To detect repeating patterns in the features, we used a uniform cost algorithm that is similar to Mokhtarian's [Mokh86]. The algorithm begins by creating a node for every possible pair of features in scale-space. A match cost is computed for each node, measuring how *different* the two features are. Expansion of the lowest cost node continues until a solution is reached, i.e. a repeating pattern has been verified over the entire domain. The final result of the algorithm is the repeating pattern in curvature scale-space with the lowest cost. If the cost of this pattern is above threshold then it is concluded that there is no repeating pattern. The results of this algorithm are shown in Figure 5.5. The curvature scale-space images on the left were computed from the ST curves for the translating cube sequence (Figure 5.3a) and the curvature scale-space images on the right were computed from the ST curves for the flapping wings image sequence (Figure 5.3b). Cycles are marked with short tick marks across the top of each image. The second and fourth curvature scale-space images on the right of Figure 5.5 were generated from the ST curves following the curvature extrema formed by the joining of the two wings. Since these extrema repeatedly disappeared and reappeared they were not accurately tracked and no cyclic pattern was found in the ST curves.

All consistent ST curves that lie on an ST surface generated by the projection of a rigid part will be similar. In particular, Theorems 1 and 3 show that the ST curves on this ST surface will all be cyclic with the same period. Using this property ST curves can be clustered based on the similarity of their cyclic behavior. ST curves that are not similar are hypothesized to be tracking points on separate rigid objects. In the translating cube sequence the periods of the cycles of all seven curves were sufficiently similar that it was concluded that only one rigid object exists in the scene. In the flapping wings image sequence it was concluded from the periods of the cycles that there are two separate rigid parts.

5.5 Concluding Remarks

The goals of this chapter were to show that an intermediate-level motion description, i.e., cyclic motion, could be recovered prior to object recognition and to show that ST flow curves are a powerful representation from which to compute this description. The successful detection of cyclic motion shows the possibility of the first goal and verifies the representational power of ST flow curves.

In order to detect cyclic motion, a procedure based on tracking curvature extrema in spatiotemporal images was defined. A curvature scale-space representation was then used to efficiently detect repeating patterns. Since torsion is also a space curve shape descriptor, torsion scale space could be use in addition to curvature scale space to detect repeating patterns. One desirable property of torsion scale space is that its features are well understood. See Appendix B.

When the repeating cyclic motion is as simple as the examples shown in this chapter, the curvature scale-space consists of spike-like features. Rather than using a uniform cost search, we could compute the Fourier transform of the scale-space image and recover the frequency of the peaks. While this will work for simple repeating curvature values, it is not clear how well it will work for more complicated repeating motions.

These results depended on the assumption that objects do not rotate in depth. The assumption of no rotation in depth was necessary because point tracking was used to detect ST curves. This assumption can be dropped if we compute ST flow curves by using ST surface flow and solving Eq. 4.1 (as described in Chapter 4). This alone will not guarantee that ST curves will be consistent. If an object is rotating in depth then flow curves will "flow" into the background as an object turns away from the viewer. However, this self-occlusion will be detected by the cluster analysis described in Chapter 4. Using this occlusion information a more sophisticated cyclic detection algorithm can be developed.

Future work on cyclic motion detection should experiment with more complex images sequences, including varying scales of cyclic motion, rotation in depth, and occlusion and disocclusion.

Chapter 6

Summary

In this thesis we proffered that high-level motions, e.g., walking, can be recognized prior to recognizing the object, e.g. a person or animal. In order to achieve this goal, new low-level and intermediate-level motion representations were defined for describing long-range motion.

Spatiotemporal (ST) surface flow is a low-level representation that indicates the instantaneous motion of points in an ST cube. Rather than use the partial derivatives of the surface as done in most gradient-based optical flow methods, the gradient of a function on the ST surface was used. It was observed that arc length of a contour does not change if that contour is moved in the direction of motion on the surface. Motivated by this observation, a function measuring arc length change was defined. The direction of motion of a contour undergoing motion parallel to the image plane was shown to be perpendicular to the gradient of this function. More generally, this gradient also approximates the direction of motion when object motion in the scene is not parallel to the image plane or when perspective projection is used.

Using ST surface flow, ST flow curves were then recovered and used to group coherent regions of the ST cube such that each group represents an object or surface in the scene undergoing similar long-range motion. ST flow curves were defined such that the tangent at a point on the curve equals the ST surface flow at that point. Our grouping algorithm forms clusters of similar flow curves and is based the following two intuitive, yet powerful

constraints called the temporal uniqueness constraints. First, a point in an image can only move to at most one point in the next image. Second, a point in an image can come from at most one point in the previous image. When these constraints are violated, or it appears that they are violated, occlusion or disocclusion has occurred.

To show the possibilities for using this framework for high-level motion-based recognition, the cyclic motion problem was addressed. Cyclic motion is formally defined as repeating curvature values along a path of motion. A procedure was presented for cyclic motion detection using ST surfaces and ST curves. The projected movement of an object generates ST surfaces. ST curves were detected on the ST surfaces, providing an accurate, compact, qualitative description of the ST surfaces. Curvature scale-space of the ST curves was then used to detect intervals of repeating curvature values.

6.1 Future Directions

The future directions of this work are of two types: experimental and theoretical. Experimentally, some of the results of this thesis could to be more thoroughly tested for their usefulness in real-world situations. The theoretical motivation and foundation of the work is well understood, however, the results should be examined when the theory is not completely met, i.e., when varying levels of noise present. The robustness of ST surface flow was examined in Chapter 3, but the robustness of computing clusters of ST flow curves can be examined. In future work, when the relative and common motion is computed (see below) it may be desirable to use a small number of representative flow curves for each cluster in order to reduce computation. In this case, it is important to understand the robustness of the computation. Chapter 4 discussed how to terminate hierarchical clustering in order to determine the initial number of clusters. Intuitive motivation for how to terminate this process was presented, but experiments could be run to test the robustness of the intuition.

Theoretically, the intermediate-level and high-level motion descriptions need to be analyzed further. This thesis has developed a rich but concise intermediate-level motion description but it is unlikely that this description will be sufficient for high-level motion understanding. In an effort to better understand what needs to be computed at

the intermediate level, we will examine the human visual system (HVS) to glean what intermediate-level motion features should be recovered.

Given an intermediate level description of motion as computed here, the high level, or coordinated sequences of events needs to be recovered. How this should be performed is an interesting and useful problem to address. One approach is currently being examined by Goddard [Godd89, Godd88a].

The following subsection lays out an approach for computing additional intermediate-level motion descriptions based on the HVS.

6.1.1 Computing Relative and Common Motion

Given a segmentation of an ST cube, we would like to use that segmentation to recognize high-level motions. A first step toward this goal could be to compute a set of generic, non-model-based motion features. By looking at the HVS we can glean what these features might be. In this section a common perceptual phenomenon of the HVS system will be examined.

When the human visual system is presented with a rolling wheel it perceives the wheel as translating across the field of view and rotating about the center of the wheel. Even though the path of a point on the rim of the wheel is a cycloid[1] the HVS does not perceive this cycloid. Rather the HVS perceives the *common motion* of translation and the *relative motion* of rotation about the center of the wheel. The *absolute motion* of the cycloid is generally not perceived.

Even though the absolute cycloidal motion is not perceived, the following equation holds [Cutt82]:

$$\text{common motion} \quad + \quad \text{relative motion} \quad = \quad \text{absolute motion} \tag{6.1}$$

We can show how this equation holds true for the case of the rolling wheel. Let $C(t)$, $R(t)$ and $A(t)$ parameterize the path of the common, relative and absolute motions,

[1] Interior points form prolate cycloid paths.

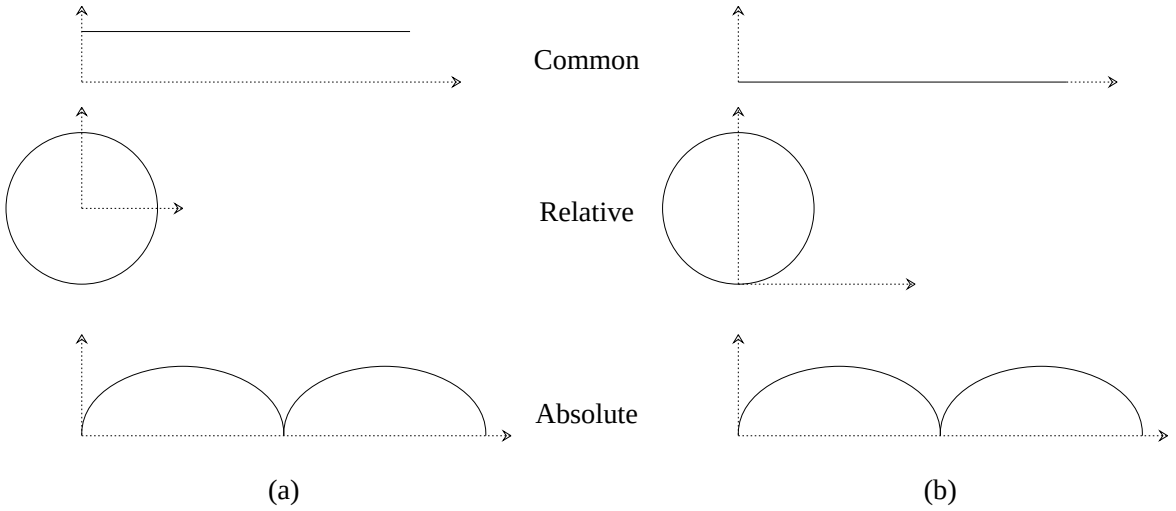


Figure 6.1: Two possible perceptions of common, relative and absolute motions for a point on the rim of a rolling wheel.

respectively[2]. For a rolling wheel we have:

$$\begin{aligned} C(t) &= (t, 1) \\ R(t) &= (-\sin(t), -\cos(t)) \end{aligned}$$

Adding $C(t)$ and $R(t)$ we have:

$$\begin{aligned} A(t) &= C(t) + R(t) \\ &= (t - \sin(t), 1 - \cos(t)) \end{aligned}$$

which parameterizes a cycloid. Figure 6.1(a) shows the three curves. Note that this is only one solution to Eq. (6.1). Another solution is obtained by vertically translating $C(t)$ and $R(t)$, as shown in Figure 6.1b. Another, more obscure, solution is shown in Figure 6.2. The solution shown is generally not perceived by the HVS. In Figure 6.2, the relative and common motion still combine to equal the absolute motion, but the curves are not as "simple" as in Figure 6.1.

There is no unique solution for the relative and common motion because there is only one equation, Eq. (6.1), and two unknows, A and C. Our goal is to find a solution that

[2]Recall that a parameterized curve in $\Re^2$ is a map α: I $\rightarrow \Re^2$ of an open interval I = (a, b) of the real line $\Re$ into $\Re^2$.

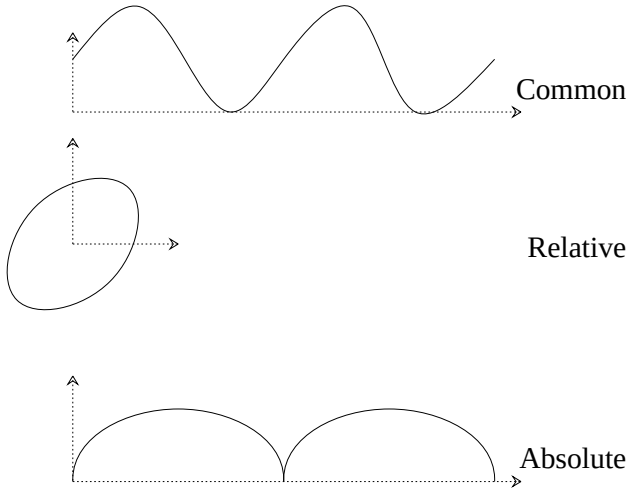


Figure 6.2: An unlikely perception of common, relative and absolute motions for a point on the rim of a rolling wheel.

has the simplest relative and common motion. This simplicity will be defined formally such that by minimizing this quantity, we arrive at a solution to Eq. 6.1 that is similar to the solution shown in Figure 6.1 rather than the solution shown in Figure 6.2.

In general, the absolute motion is the motion once a (arbitrary) viewer-relative frame of reference is specified. The common motion is the motion of the whole configuration relative to the viewer. Finally, the relative motion is the motion relative to another element. These loose definitions of relative and common motion are based on how the HVS perceives motion. There are no "correct" solutions to Eq. (6.1), only solutions that are similar to the HVS's perception. Therefore, we will examine how the HVS perceives these motions and use that as an additional constraint to solve Eq. (6.1).

Psychologists have long recognized this perceptual phenomenon [Joha50, Hoch57, Wall65], but have failed to developed a complete model of how the HVS recovers relative and common motion [Joha73, Cutt82]. A common component of those models is to minimize the complexity of the relative and/or common motion [Hoch53, Hoch60, Cutt82]. Minimizing the motion can be viewed as making it as simple as possible. Unfortunately, how and what is being minimized is usually very poorly defined. In the remainder of this section, a possible definition of what to minimize will be formally defined and the recovery of common and relative motion will be posed as a non-linear minimization problem.

Let $A(t)$, $C(t)$ and $R(t)$ parameterize the absolute, common and relative motions as a function of time, respectively. We know that $A(t_i) = C(t_i) + R(t_i)$. Consider segments of $A(t)$, $C(t)$ and $R(t)$ where $t \in [t_i, t_j]$. In the discrete case

$$\begin{aligned} A(t_i) &= C(t_i) + R(t_i) \\ A(t_{i+1}) &= C(t_{i+1}) + R(t_{i+1}) \\ &\vdots \\ A(t_j) &= C(t_j) + R(t_j) \end{aligned}$$

These are the constraints that must be satisfied while minimizing C and R, i.e., making C and R as simple as possible.

We will minimize the following function, f:

$$f(C, R) = \sum_{t=t_i}^{t_j} [\kappa_t^2(C(t)) + \kappa_t^2(R(t))]$$

where κ defines the curvature along a path and κ_t defines the derivative of curvature with respect to t, i.e., the change of curvature along the path.

For planar curves, κ uniquely determines the "shape" of the curves. Points on a curve where the curve changes shape rapidly will have large values of curvature whereas points on the curve that do not change shape, i.e., where the curve is straight, will have curvature equal to 0. For a curve to be simple, it should not change its shape often, i.e., its curvature should not change often. Therefore, $f(C, R)$ will be minimized when the partial derivatives of curvature of C and R are minimized.

The constraints listed above and f are all that is needed to find the common and relative motions. A, the absolute motion, is recovered from the image sequence, then a minimization procedure is used to solve the system.

Unfortunately, a minimization procedure for non-linear problems may never find a solution. There are two approaches one can take to this problem: first, a routine that linearly approximates f, the function to be minimized, can be used, or second, f can be changed so that it is linear. In the first case, the first and second derivatives of f are computed and used by the minimization procedure. In our case, this is possible since the

derivatives of f are computable. However, the optimal solution will not be guaranteed. Therefore the second alternative may be desirable.

The problem was stated above as a non-linear minimization problem because it is an intuitive way to understand the problem. Since it may be unreasonable to expect a procedure to solve the problem as stated above, we need to simplify the problem, i.e., find a new f that is linear. We need this new function, f', to be linear yet capture the essence of f. f' needs to behave as f does. For example, if f increases then f' must increases. f' can be viewed as a linear heuristic of f.

f is composed of the derivative of the curvature function of two curves. To find a suitable f', we need a function that is similar to curvature, but simpler. Curvature is a second derivative operation, measuring the change of the tangent vector at a point along a curve. Therefore the derivative of curvature measures the second-order change of the tangent vector. Therefore, a possible f' would be a function of the third derivative of the flow curve. Approximations to the first derivative can be found by a linear function that computes the spatial difference around a point. Similarly, higher-order derivatives can be found by computing the difference of lower-order derivatives around a point. By composing f' of these differences, we may be able to find a linear function that behaves as a hueristic for f.

Using one of these two approaches, we should be able to compute the relative and common motions such that the computed results correspond with the HVS's perception. We can then combine the results from the other intermediate-level motion representations presented in this thesis.

6.2 Contributions

The goal of this thesis was to examine a new computational framework for processing long image sequences. This paradigm is an alternative approach to intermediate-level and high-level motion analysis. Traditionally, motion was used to recover scene structure and the structure was then used to index into a database of object models. In this thesis, we proposed a new motion description hierarchy and claimed that coordinated

sequences of events such as walking and throwing can be recovered using this hierarchy *before* identifying the objects (e.g., legs and arms) performing the motion.

This claim has been made by only a few other researchers [Godd88a, Goul89], and this thesis is one of the first thorough investigations of the claim. Also, the specific approach based on recovering ST surface flow and ST flow curves, is novel and has great potential as demonstrated by the ease with which ST flow curves were used to detect cyclic motion and by the equivalency of ST flow curves to the paths of lights in Johansson's MLDs. Since the human visual system can so readily recognize MLDs, the paths of the lights, and hence ST flow curves, must represent the fundamental salient features of image motion.

In addition to defining a new computational framework, this thesis has defined low-level and intermediate-level motion descriptions that are applicable to other areas of computer vision. ST surface flow can be used to recover scene structure in much the same way that optical flow is used to recover scene structure. But the temporal coherence of ST surface flow should improve structure-from-motion results. ST flow curves can also be used for object tracking. However, since ST flow curves are computed directly from ST surface flow, correspondence-based tracking of points, and its associated difficulties, is not required.

It is important to note that in none of the image sequences were the objects or their motion known *a priori*. Nevertheless, we are now able to compute answers to all of the following questions:

- What is the instantaneous motion of points in an image?
- Where are the moving objects in an image sequence?
- What is the background?
- Which objects become occluded or disoccluded?
- What is the path of moving objects?
- Are objects undergoing cyclic motions?

Given our ability to compute answers to these questions, we are now able to begin work on how to recover the the high-level, abstract motion that objects are undergoing. By

computing this abstract motion prior to identifying the objects undergoing the motion, we investigate a novel and very promising approach to object and motion recognition.

Appendix A

Proof Sketch of Theorem 2

Theorem 2 If a solid object in the scene displays cyclic motion such that the cyclic motion is preserved under projection, then the spatiotemporal surfaces that correspond to intervals of cyclic motion will be piecewise isomorphic.

Proof Sketch Consider a spatiotemporal surface such that there exist ridges on the surface and every point on a ridge has the same value for the time coordinate. We call this ridge a *temporal ridge* since it results from a nonsmooth change of velocity of an object at some time. Define a *patch* of an ST surface as the surface between two temporal ridges. Two surfaces are piecewise isomorphic if the first patch of surface 1 is isomorphic to the first patch of surface 2, the second patches are isomorphic, etc. Let C be a closed curve in the image that is the result of the projection into the x-y plane of a solid object undergoing cyclic motion. Let

$$x = f(u), \qquad y = g(u), \qquad \text{a} < u < \text{b}, \qquad f(u) > 0, \qquad g(u) > 0$$

be a parameterization for C. As the object in the scene moves the curve in the image also moves. We will now parameterize the surface, S, generated by C moving in the image. The resulting surface will exist in x-y-t space. We will break the types of movement of C into three cases.

Case I: C Translates

Let v_x and v_y be the translational components of the curve in the x and y directions, respectively. Let d_x and d_y be the displacement of C in the x and y

directions, respectively. We obtain the following map

$$\mathbf{H}(u,t) = (f(u) + tv_x + d_x, g(u) + tv_y + d_y, t)$$

from the open set U =$\{(u,\ t) \in \Re^2;\ \mathrm{a} < u < \mathrm{b};\ \mathrm{c} < t < \mathrm{d};\ \mathrm{c},\ \mathrm{d} \geq 0\}$ into S.

Case II: C Rotates

Let $(i,\ j)$ be the point about which C rotates. Let ω be the angular velocity of C. Let θ be the angle that a point on C has rotated through. We obtain the following map

$$\mathbf{H}(u,t) = (|i - f(u)|\ \cos(t\omega),\ |j - g(u)|\ \sin(t\omega), t)$$

from the open set U =$(u,\ t) \in \Re^2;\ \mathrm{a} < u < \mathrm{b};\ \mathrm{c} < t < \mathrm{d};\ \mathrm{c},\ \mathrm{d} \geq 0$ into S. If necessary, displacement terms can be added to the x and y components.

Case III: C Translates and Rotates

We simply combine the components of the parameterization for Cases II and III.

It is possible that over some time interval, C will undergo a combination of these movements, changing parameters with time. For example, C may translate at one speed for a time, continue translating at a different speed, then rotate about some point for a time, and finally rotate about a different point for a time. But we can simply break this sequence of movements into separate parameterizations for each translation, rotation, non-movement and combination of translation and rotation. One parameterization starts up where the previous one leaves off to give a continuous surface. Each of these parameterizations define a piece of the surface. It is these pieces that will be isomorphic. In the case where C accelerates, another parameterization can be considered or the ST surface generated by C while accelerating can be treated as a series of translation and rotation changes.

Two surfaces are piecewise isomorphic if there exists a bijective function mapping one patch to another patch. We need to show that given two surface patches, one patch generated by C during one cycle of motion, the other patch generated by C during a corresponding cycle of motion, the two patches are isomorphic.

We assumed that the cyclic motion of a rigid object is preserved under projection. This means that for any point on the object, it displays cyclic motion under projection.

Let S_1 and S_2 be two surfaces patches that were generated by C at corresponding times in the cyclic motion of C. Let $\mathbf{H}_{S_1}$ and $\mathbf{H}_{S_2}$ be the parameterizations of S_1 and S_2, respectively. Let patch S_i start at t_i. Since there are three different types of parameterizations we need to show that for any of these parameterizations, corresponding patches are isomorphic.

Case I: Translation

Given the parameterization for two translating patches:

$$\mathbf{H}_{S_1}(u,t) = (f(u) + tv_x + d_{x_1}, g(u) + tv_y + d_{y_1}, t)$$

$$\mathbf{H}_{S_2}(u,t) = (f(u) + tv_x + d_{x_2}, g(u) + tv_y + d_{y_2}, t)$$

we need to show that there exists a bijective mapping from S_1 to S_2. The function:

$$h([x,y,t]) = [x,y,t] + [d_{x_2} - d_{x_1}, d_{y_2} - d_{y_1}, t_2 - t_1]$$

where t_i is the start time of patch S_i, is a bijection that maps points from S_1 to S_2. So S_1 and S_2 are isomorphic.

Case II: Rotation

The definition of cyclic motion uses the curvature of the path of motion rather than the position of the path. Suppose an object is undergoing cyclic motion such that this cyclic motion is preserved under projection. All this says is that the curvature is repeating. Let t_1 and t_2 be two corresponding times in different periods. Since these times correspond we know that the curvature at corresponding points is the same. Given that the projection is rotating in the image plane, we need to show that there is only one possible center of rotation for C that could give rise to the curvature values of the points of C over time.

The curvature of a point is equal to the inverse of the distance from that point to the the center of rotation. Given that the curvature of point p_i on C is k_i, its distance from the center of rotation is $1/\mathrm{k}_i$. We need to show that there is only one center of rotation point such that the distance from that center of rotation point to p_i is $1/\mathrm{k}_i$ for all p_i on C. We will show this by assuming that two centers of rotation exist and show that C must then be a straight line. Let (x_1, y_1) and

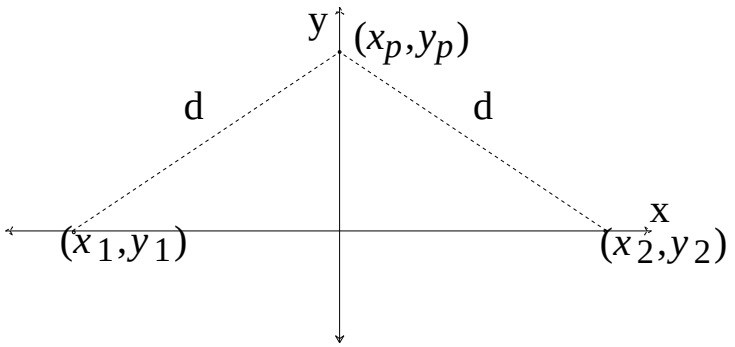


Figure A.1: The distance from (x_1, y_1) to (x_p, y_p) equals the distance from (x_2, y_2) to (x_p, y_p).

(x_2, y_2) be the two centers of rotation. Position the coordinate system so that y_1 and y_2 are both zero and the y-axis is a perpendicular bisector of the line segment joining (x_1, y_1) and (x_2, y_2). See Figure A.1. Let (x_p, y_p) be a point distance d away from (x_1, y_1) and (x_2, y_2) We need to show that (x_p, y_p) lies on the y-axis. In other words, we have to show that $x_p = 0$. The distances from (x_1, y_1) to (x_p, y_p) and from (x_2, y_2) to (x_p, y_p) are:

$$\mathrm{d} = \sqrt{(x_p - x_1)^2 + (y_p - y_1)^2} \qquad \mathrm{d} = \sqrt{(x_p - x_2)^2 + (y_p - y_2)^2}$$

respectively. Setting these two equal and solving for x_p we find that x_p equals 0. So we have shown that if there are two possible center of rotation points then C must be a straight line. Since C is assumed to be a closed curve, i.e. not a straight line, there can only be one center of rotation.

Let (i, j) be the center of rotation point. There are two directions that C can rotate around (i, j). But the curvature of the paths of the points of C for rotating one direction will be the negation of the curvature values if C rotates in the other direction. So there is actually only one direction that will give the correct curvature values for the points of C. We have now shown that C must rotate in a unique direction around a unique point. Given the parameterization for two rotating patches S_1 and S_2:

$$
\begin{aligned}
\mathbf{H}_{S_1} &= (|i - f(u)|\ \cos(t_1\omega) + |j - g(u)|\ \sin(t_1\omega), \\
&\quad -|i - f(u)|\ \sin(t_1\omega) + |j - g(u)|\ \cos(t_1\omega), \\
&\quad t) \\
\mathbf{H}_{S_2} &= (|i - f(u)|\ \cos(t_2\omega) + |j - g(u)|\ \sin(t_2\omega), \\
&\quad -|i - f(u)|\ \sin(t_2\omega) + |j - g(u)|\ \cos(t_2\omega), \\
&\quad t)
\end{aligned}
$$

we need to show that there exists a bijective function from S_1 to S_2. The function:

$$
h([x, y, t]) = [x, y, t] \begin{bmatrix} \cos((t_1 - t_2)\omega) & -\sin((t_1 - t_2)\omega) & 0 \\ \sin((t_1 - t_2)\omega) & -\cos((t_1 - t_2)\omega) & 0 \\ 0 & 0 & t_1 - t_2 \end{bmatrix}
$$

where t_i is the start time of patch S_i, is a bijection that maps points from S_1 to S_2. So S_1 and S_2 are isomorphic.

Case III: Translation and Rotation

We combine the translation and rotation of Case II and Case III to show that S_1 is isomorphic to S_2. Since C is translating, the center of rotation of C is translating also. But it remains unique for all t.

This shows that S_1 is isomorphic to S_2 in all cases. So the ST surfaces generated by the cyclic motion of C are piecewise isomorphic.

□

Appendix B

The Structure of Torsion Scale-Space

Theorem 4 *Let Γ be a 3D space curve in class C_1 and let Γ_σ be the evolved version of Γ. If all curves Γ_σ are in C_1, then all extrema occurring at regular points on contours in the torsion scale space image of Γ are maxima*[1].

Proof: Torsion is defined as follows:

$$t(s) = \frac{-\alpha'(u) \times \alpha''(u)\alpha'''(u)}{|\mathrm{k}(u)|^2}$$

This is equivalent to:

$$-\begin{vmatrix} \alpha'''_{\mathrm{x}} & \alpha'''_{\mathrm{y}} & \alpha'''_{\mathrm{z}} \\ \alpha'_{\mathrm{x}} & \alpha'_{\mathrm{y}} & \alpha'_{\mathrm{z}} \\ \alpha''_{\mathrm{x}} & \alpha''_{\mathrm{y}} & \alpha''_{\mathrm{z}} \end{vmatrix}$$

This is equivalent to:

$$\frac{\begin{vmatrix} X_{uuu} & Y_{uuu} & Z_{uuu} \\ X_u & Y_u & Z_u \\ X_{uu} & Y_{uu} & Z_{uu} \end{vmatrix}}{|\mathrm{k}(u)|^2}$$

[1]This proof is based on a similar proof showing the same property for curvature scale space of planar curves [Mack88].

On any contour in torsion scale space

$$\mathrm{k}(u,\sigma) \;=\; 0$$

And the fact that all Γ_σ are in C_1 this is equivalent to:

$$X_{uuu}[Y_u Z_{uu} - Y_{uu} Z_u] \;-\; Y_{uuu}[X_u Z_{uu} - X_{uu} Z_u] \;-\; Z_{uuu}[X_u Y_{uu} - X_{uu} Y_u] \;=\; 0$$

To exploit the properties of the heat equation, it is convenient to change variables and let

$$t = \frac{1}{2\sigma^2}$$

So we define

$$\begin{aligned} x(u,t) &= X(u,\sigma) \\ y(u,t) &= Y(u,\sigma) \\ z(u,t) &= Z(u,\sigma) \end{aligned}$$

$$\begin{aligned} \alpha(u,t) \quad = \quad & x(u,t)_{uuu}[y_u(u,t)z_{uu}(u,t) - y_{uu}(u,t)z_u(u,t)] \;- \\ & y(u,t)_{uuu}[x_u(u,t)z_{uu}(u,t) - x_{uu}(u,t)z_u(u,t)] \;- \\ & z(u,t)_{uuu}[x_u y(u,t)_{uu}(u,t) - x_{uu} y(u,t)_u(u,t)] \end{aligned}$$

The functions $x(u,t)$, $y(u,t)$ and $z(u,t)$ are obtained by convolving

$$\frac{1}{\sqrt{4\pi t}}\; e^{-(1/4t)u^2}$$

with the original curve coordinates $x(u)$, $y(u)$ and $z(u)$ respectively, and so they satisfy the heat equation:

$$\begin{aligned} x_{uu}(u,t) &= x_t(u,t) \\ y_{uu}(u,t) &= y_t(u,t) \\ z_{uu}(u,t) &= z_t(u,t) \end{aligned}$$

The implicit function theorem is satisfied on contours $\alpha(u,t) = 0$ because of the assumption that all Γ_σ are smooth. Therefore we can write:

$$t \;=\; t(u) \qquad\qquad \frac{dt}{du} \;=\; -\frac{\alpha_u}{\alpha_t}$$

The theorem will be proven if we can show that for all points such that $t_u(u) = 0$ we have $t_{uu}(u) < 0$. Observe that

$$t_u(u) \;=\; 0 \quad iff \quad \alpha_u(u,t) \;=\; 0$$

At an extremum where the above holds, we have

$$t_{uu}(u) \;=\; \frac{d}{du}\left(\frac{-\alpha_u}{\alpha_t}\right) \;=\; \frac{\partial}{\partial u}\left(\frac{-\alpha_u}{\alpha_t}\right) + \frac{\partial}{\partial t}\left(\frac{-\alpha_u}{\alpha_t}\right)\frac{dt}{du} \;=\; \frac{-\alpha_{uu}}{\alpha_t}$$

So we must show that if $\alpha(u,t) = \alpha_u(u,t) = 0$ then

$$\frac{\alpha_{uu}}{\alpha_t} \;>\; 0$$

We shall show that these conditions require

$$\frac{\alpha_{uu}}{\alpha_t} \;=\; 1$$

which proves the theorem.

We will expand only the first part of our equation for α:

$$\begin{aligned}\alpha_u &= x_{uuuu}[y_u z_t - y_t z_u] \;+\; x_{uuu}[y_{uu}y_{uu} + y_u z_{uuu} - y_{uu}z_{uu} - y_{uuu}z_u] \\ &= x_{tt}[y_u z_t - y_t z_u] \;+\; x_{tu}[y_u z_{tu} - y_{tu} z_u]\end{aligned}$$

Then

$$\begin{aligned}\alpha_{uu} &= x_{ttu}[y_u z_t - y_t z_u] \;+\; x_{tt}[y_u z_{tu} + y_{uu} z_t - y_t z_{uu} - y_{tu} z_u] \;+ \\ &\quad x_{tuu}[y_u z_{tu} - y_{tu} z_u] \;+\; x_{tu}[y_u z_{tuu} + y_{uu} z_{tu} - y_{tu} z_{uu} - y_{tuu} z_u] \\ &= x_{ttu}[y_u z_t - y_t z_u] \;+\; x_{tu}[y_u z_{tt} + y_t y_{tu} - y_{tu} z_t - y_{tt} z_u]\end{aligned}$$

Now we calculate α_t. Again we only expand the first part of α.

$$\alpha_t \;=\; x_{ttu}[y_u z_t - y_t z_u] \;+\; x_{tu}[y_{tu} z_t + y_u z_{tt} - y_{tt} z_u - y_t z_{tu}]$$

At this point α_{uu} and α_t are almost equal. We need to show:

$$x_{tu}[y_t z_{tu} - y_{tu} z_t] \;=\; 0$$

This by itself is not true. But if we include the analogous terms from the other parts of α which we dropped earlier then they will sum to zero.

All the appropriate terms are:

$$x_{tu}[y_t z_{tu} - y_{tu} z_t] \;-\; y_{tu}[x_t z_{tu} - x_{tu} z_t] \;-\; z_{tu}[x_t y_{tu} - x_{tu} y_t]$$

which equals zero.

Bibliography

[Adel85] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Optical Society of America*, 2:284 – 299, February, 1985.

[Agga88] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images - A review. *Proc. IEEE*, 76:917–935, 1988.

[Akit84] K. Akita. Image sequence analysis of real world human motion. *Pattern Recognition*, 17:73–83, 1984.

[Alle89] P. K. Allen. Real-time motion tracking using spatio-temporal filters. In *Proc. Image Understanding Workshop*, pages 695–701, 1989.

[Allm90a] M. Allmen and C. R. Dyer. Computing spatiotemporal surface flow. In *Proc. 3rd Int. Conf. Computer Vision*, pages 47–50, 1990.

[Allm90b] M. Allmen and C. R. Dyer. Cyclic motion detection using spatiotemporal surfaces and curves. In *Proc. 10th Int. Conf. Pattern. Recognition*, pages 365–370, 1990.

[Allm91] M. Allmen and C. R. Dyer. Long-range spatiotemporal motion understanding using spatiotemporal flow curves. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 303–309, 1991.

[Aloi88] J. Aloimonos. Visual shape computation. *Proc. IEEE*, 76:899–916, 1988.

[Ande85] C. H. Anderson, P. J. Burt, and G. S. VanDerWal. Change detection and tracking using pyramid transform techniques. *Proc. SPIE Intelligent Robots and Computer Vision*, 579:72 – 78, 1985.

[Badl79] N. I. Badler and S. W. Smoliar. Digital representations of human movement. *Computing Surveys*, 11:19–38, March, 1979.

[Bake88a] H. H. Baker. Building surfaces of evolution: the weaving wall. *Image Understanding Workshop*, pages 1031–1040, 1988.

[Bake88b] H. H. Baker and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Proc. Computer Vision and Pattern Recognition Conf.*, pages 2–9, 1988.

[Bake89] H. H. Baker and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *Int. J. of Computer Vision*, 3:33–49, 1989.

[Barl65] H. B. Barlow and W. R. Levick. The mechanism of directionally selective units in rabbit's retina. *J. Physiol. London*, 178:477–504, 1965.

[Barr81] H. G. Barrow and J. M. Tenenbaum. Computational vision. *Proc. IEEE*, 69:572–595, May 1981.

[Bert84] B. I. Bertenthal, D. R. Proffitt, and J. E. Cutting. Infant sensitivity to figural coherence in biomechanical motions. *Journal of Experiment Child Psychology*, 37:213–230, 1984.

[Bert85] B. Bertenthal, D. Proffitt, N. Spetner, and A. Thomas. The development of infant sensitivity to biomechanical motions. *Child Development*, 56:531–543, 1985.

[Besl86] P. J. Besl and R. C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Comp. Vision, Graphics, and Image Proc.*, 33:33–80, 1986.

[Besl88] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10:167–192, 1988.

[Boll87] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *Int. J. Computer Vision*, 1:7–55, 1987.

[Bolt94] T. L. Bolton. Rhythm. *American J. of Psychology*, 6:145–238, 1894.

[Bosc67] C. Bosche. Computer-generated random-dot images. In M. Krampen and Pl Seitz, editors, *Design and Planning: 2. Computers in design and communication*, pages 87–91. Hastings House, 1967.

[Brad85] M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. *Proc. 2nd Int. Symp. on Robotics Research*, pages 5–16, 1985.

[Brus83] A. R. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21:3–20, 1983.

[Burt88] P. J. Burt. Smart sensing within a pyramid vision machine. *Proc. IEEE*, 76:1006–1015, 1988.

[Buxt83] B. F. Buxton and H. Buxton. Monocular depth perception from optical flow by space-time signal processing. *Proc. Roy. Soc. London B*, 218:27–47, 1983.

[Buxt84] B. F. Buxton and H. Buxton. Computation of optical flow from the motion of edge features in image sequences. *Image and Vision Computing*, 2:59–75, 1984.

[Cann86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Carl88] S. Carlsson. Information in the geometric structure of retinal flow field. In *Proc. 2nd Int. Conf. Computer Vision*, pages 629–633, 1988.

[Coxe61] H. S. M. Coxeter. *Introduction to Geometry.* Wiley, New York, 1961.

[Cutt81a] J. Cutting. Coding theory adapted to gate perception. *Journal of Experimental Psychology: Human Perception and Performance*, 7:71–87, 1981.

[Cutt81b] J. Cutting and D. Proffitt. Gait perception as an example of how we may perceive events. In R. Walk and H. L. Pick, editors, *Intersensory perception and sensory integration*, pages 249–273. Plenum, New York, 1981.

[Cutt82] J. E Cutting and D. R. Proffitt. The minimum principle and the perception of absolute, common, and relative motions. *Cognitive Psychology*, 14:211–246, 1982.

[DoCa76] M. DoCarmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, 1976.

[Enge86] S. A. Engel and J. M. Rubin. Detecting visual motion boundaries. *Proc. IEEE Workshop on Motion*, pages 107–111, 1986.

[Faug90] O. Faugeras. On the motion of 3D curves and its relationship to optical flow. In *Proc. 1st European Conf. Computer Vision*, pages 107–117. Springer-Verlag, 1990.

[Flee84a] D. J. Fleet. The early processing of spatio-temporal visual information. Technical report, University of Toronto, 1984.

[Flee84b] D. J. Fleet, A. D. Jepson, and P. E. Hallet. A spatio-temporal model for early visual processing. Technical Report RCBV-TR-84-1, University of Toronto, 1984.

[Flin81] B. E. Flinchbaugh and B. Chandrasekaran. A theory of spatio-temporal aggregation for vision. *Artificial Intelligence*, 17:387–407, 1981.

[Frai78] P. Fraisse. Time and rhythm perception. In E. C. Carterette and M. P. Friedman, editors, *Handbook of Perception, Volume VIII*, pages 203–254. Academic Press, New York, 1978.

[Fran90] E. Francois and P. Bouthemy. The derivation of qualitative information in motion analysis. In *Proc. 1st European Conf. Computer Vision*, pages 226–230. Springer-Verlag, 1990.

[Fu68] K. S. Fu. *Sequential Methods in Pattern Recognition and Machine Learning.* Academic Press, New York, 1968.

[Fuku72] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, 1972.

[Gabr73] A. Gabrielsson. Adjective ratings and dimension analysis of auditory rhythm patterns. *Scandinavian J. of Psychology*, 14:244–260, 1973.

[Gear71] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, 1971.

[Godd88a] N. H. Goddard. Recognizing animal motion. *Proceedings of Image Understanding Workshop*, pages 938–944, 1988.

[Godd88b] N. H. Goddard. Representing and recognizing event sequences. *Proc. CMU Summer Connectionist Workshop*, 1988.

[Godd89] N. H. Goddard. The interpretation of visual motion: Recognizing moving light displays. *Proc. IEEE Workshop on Motion*, pages 212–220, March 1989.

[Gold88] D. B. Goldgof, H. Lee, and T. S. Huang. Motion analysis of nonrigid surfaces. *Proc. Computer Vision and Pattern Recognition Conf.*, pages 375–380, 1988.

[Goul89] K. Gould and M. Shah. The trajectory primal sketch: A multi-scale scheme for representing motion characteristics. *Proc. Computer Vision and Pattern Recognition Conf.*, pages 79–85, 1989.

[Grzy89] N. M. Grzywacz, J. A. Smith, and A. L. Yuille. A common theoretical framework for visual motion's spatial and temporal coherence. In *Proc. Workshop Visual Motion*, pages 148–155, 1989.

[Hans78] A. R. Hanson and E. M. Riseman. Visions: A computer system for interpreting scenes. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 303–333. Academic Press, New York, 1978.

[Hara83] R. M. Haralik, L. T. Watson, and T. J. Laffey. The topographic primal sketch. *Int. J. Robotics Research*, 2:50–72, 1983.

[Hart75] J. A. Hartigan. *Clustering Algorithms.* Wiley, 1975.

[Heeg87] D. J. Heeger. Optical flow from spatiotemporal filters. *Proc. First Int. Conf. on Computer Vision*, pages 181–190, 1987.

[Heel90] J. Heel. Direct estimation of structure and motion from multiple frames. A.I. Memo 1190, MIT, 1990.

[Henn69] V. Henn and O. J. Grüsser. The summation of excitation in the receptive fields of movement-sensitive neurons of the frog's retina. *Vision Research*, 9:57–69, 1969.

[Hoch53] J. Hochberg and E. McAlister. A quantitative approach to figural goodness. *Journal of Experimental Psychology*, 46:361–364, 1953.

[Hoch57] J. Hochberg. Effects of the gestalt revolution: The Cornell symposium on perception. *Psychological Review*, 64:73–84, 1957.

[Hoch60] J. Hochberg and V. Brooks. The psychophysics of form: Reversible perspective drawing of spatial objects. *American Journal of Psychology*, 73:337–354, 1960.

[Hoff88] D. D. Hoffman and W. Richards. Representing smooth plane curves for recognition: Implications for figure-ground reversal. In W. Richards, editor, *Natural Computation*, pages 76–82. MIT Press, 1988.

[Hogg83] D. Hogg. Model-based vision: A program to see a walking person. *Image and Computer Vision Computing*, 1:5–20, 1983.

[Horn81] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Jain79a] R. Jain, W. N. Martin, and J. K. Aggarwal. Segmentation through the detection of changes due to motion. *Computer Vision, Graphics and Image Processing*, 11:13–34, 1979.

[Jain79b] R. Jain and H. H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:206–214, 1979.

[Jain81] R. Jain. Dynamic scene analysis using pixel-based processes. *IEEE Computer*, 14:12–19, 1981.

[Jain83] R. Jain. Extraction of motion information from peripheral processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:489–503, 1983.

[Jain88] R. Jain. Dynamic vision. In *Proc. 9th Int. Conf. Pattern Recognition*, pages 226–235, 1988.

[Jaya83] S. N. Jayaramamurthy and R. Jain. An approach to the segmentation of textured dynamic scenes. *Computer Vision, Graphics and Image Processing*, 21:239–261, 1983.

[Joha50] G. Johansson. *Configurations in event perception.* Almqvist & Wiksell, 1950.

[Joha73] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14:201–211, 1973.

[Joha76] G. Johansson. Spatio-temporal differentiation and integration in visual motion perception. *Psychological Research*, 38:379–393, 1976.

[Jule71] B. Julesz. *Foundations of Cyclopean Perception.* The University of Chicago Press, Chicago, 1971.

[Kana81] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[Koen75] J. J. Koenderink and J. J. Van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22:773–791, 1975.

[Lee88a] C. H. Lee. Structure and motion from two perspective views via planar patch. *Proc. ICCV*, pages 158 – 164, 1988.

[Lee88b] J. S. Lee and C. Lin. A novel approach to real-time motion detection. *Proc. Computer Vision and Pattern Recognition*, pages 730 – 735, 1988.

[Lett59] J. Y. Lettvin, R. R. Maturana, W. S. McCulloch, and W. H. Pitts. What the frog's eye tells the frog's brain. *Proc. Inst. Radio Engrs.*, 47:1940–1951, 1959.

[Lett61] J. Y. Lettvin, R. R. Maturana, , and W. H. Pitts W. S. McCulloch. Two remarks on the visual system of the frog. In W. A Rosenblith, editor, *Sensory Communication*. MIT Press, 1961.

[Leun87a] M. K. Leung and Y. H. Yang. Human body motion segmentation in a complex scene. *Pattern Recognition*, 20:55–64, 1987.

[Leun87b] M. K. Leung and Y. H. Yang. A region based approach for human body motion analysis. *Pattern Recognition*, 20:321–339, 1987.

[Limb75] J. O. Limb and J. A. Murphy. Estimating velocity of motion images in television signals. *Computer Graphics and Image Processing*, 4:311–327, 1975.

[Liou89] S. P. Liou and R. C. Jain. Motion detection in spatio-temporal space. *Computer Vision, Graphics, and Image Processing*, 45:227–250, 1989.

[Liou90] S. P. Liou and R. C. Jain. A parallel technique for three-dimensional image segmentation. In *Proc. 10th Int. Conf. Pattern Recognition*, pages 201–203, 1990.

[Litt87] J. J. Little, G. Blelloch, and T. Cass. Parallel algorithms for computer vision on the connection machine. In *Proc. 1st Int. Conf. Computer Vision*, pages 587–591, 1987.

[MacA83] L. MacArther and R. Baron. Toward an ecological theory of social perception. *Psychological Review*, 90, No. 3:215–238, 1983.

[Mack88] A. K. Mackworth and F. Mokhtarian. The renormalized curvature scale space and the evolution properties of planar curves. *Proc. Computer Vision and Pattern Recognition*, pages 318–326, 1988.

[Marr78] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London B*, 200:269–294, 1978.

[Marr80] D. Marr and L. Vaina. Representation and recognition of the movements of shapes. *AI Memo 597*, October, 1980.

[Marr81] D. Marr and S. Ullman. Directional selectivity and its use in early visual processing. *Proc. Roy. Soc London B*, 211:151–180, 1981.

[Marr82] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[Medi84] G. Medioni and R. Nevatia. Description of 3-d surfaces using curvature properties. *Proc. Image Understanding Workshop*, pages 291–299, 1984.

[Mokh86] F. Mokhtarian and A. Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:34–43, January 1986.

[Mokh88] F. Mokhtarian. Multi-scale description of space curves and three-dimensional objects. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 298–303, 1988.

[Nels88] R. C. Nelson and J. Y. Aloimonos. Using flow field divergence for obstacle avoidance: Towards qualitative vision. In *Proc. 2nd Int. Conf. Computer Vision*, pages 188–196, 1988.

[O'Ro80] J. O'Rourke and N. I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:522–536, November, 1980.

[Peng89] S. L. Peng and G. Medioni. Interpretation of image sequences by spatio-temporal analysis. In *Proc. Workshop Visual Motion*, pages 344–351, 1989.

[Pres88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[Rash80] R. F. Rashid. Towards a system for the interpretation of moving light displays. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2, No. 6:574–581, November, 1980.

[Reic83] W. Reichardt, T. Poggio, and K. Hausen. Figure-ground discrimination by relative movements in the visual system of the fly. Part ii: Towards the neural circuitry. *Biological Cybernetics*, 46:1–30, 1983.

[Rome84] H. C. Romesburg. *Cluster Analysis for Researchers*. Lifetime Learning, 1984.

[Rubi85] J. Rubin and W.A. Richards. Boundaries of visual motion. Technical Report AI Memo 835, MIT, 1985.

[Rune81] S. Runeson and G. Frykholm. Visual perception of lifted weight. *Journal of Experimental Psychology: Human Perception and Performance*, 7, No. 4:733–740, 1981.

[Schu84] B. G. Schunck. The motion constraint equation for optical flow. *Proc. 7th Int. Conf. on Pattern Recognition*, pages 20–22, 1984.

[Schu86] B. G. Schunck. The image flow constraint equation. *Computer Vision, Graphics, and Image Processing*, 35:20–46, 1986.

[Spet87] M. Spetsakis and J. Aloimonos. Closed form solution to the structure from motion problem from line correspondences. *Proc. Sixth National Conference on Artificial Intelligence*, pages 738–743, July 1987.

[Stev79] K. A. Stevens. Representing and analyzing surface orientation. In P. H. Winston and R. H. Brown, editors, *Artificial Intelligence: An MIT Perspective*, volume 2. MIT Press, 1979.

[Taal90] M. A. Taalebinezhaad. Direct recovery of motion and shape in the general case by fixation. In *Proc. Int. Conf. Computer Vision*, pages 451–455, 1990.

[Terz88] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[Thom85] W. B. Thompson, K. M. Mutch, and V. A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE Trans. Patt. Analysis Machine Intell.*, 7:374–383, 1985.

[Tou74] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Reading, Mass., 1974.

[Ullm79] S. Ullman. *The Interpretation of Visual Motion*. MIT Press, Cambridge, Mass., 1979.

[VanS85] J. P. H. VanSanten and G. Sperling. Elaborated reichardt detectors. *J. Optical Society of America*, 2:300 – 321, February, 1985.

[Verr87] A. Verri and T. Poggio. Against quantitative optical flow. In *Proc. 1st Int. Conf. Computer Vision*, pages 171–180, 1987.

[Wall65] H. Wallach. Visual perception of motion. In G. Kepes, editor, *The nature and art of motion*. Braziller, 1965. (Revised in H. Wallach, *On perception*. Quadrangle, 1976).

[Wats85] A. B. Watson and A. J. Ahumada. Model of human visual-motion sensing. *J. Optical Society of America*, 2:322 – 342, February 1985.

[Webb82] J. A. Webb and J. K. Aggarwal. Structure from motion of rigid and jointed objects. *Artificial Intelligence*, 19:107–130, 1982.

[Yama89] M. Yamamoto. A general aperture problem for direct estimation of 3-d motion parameters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:528–536, 1989.