

CORRESPONDENCE BASED MOTION ANALYSIS

by

Salit Levy Gazit

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Computer Science)

April 1991

Copyright 1991 Salit Levy Gazit

Acknowledgments

I would like to thank my advisor, Dr. Gérard Medioni for introducing me to Computer Vision and for his direction, suggestions, support and encouragement through the duration of my research.

I wish to express my gratitude to Dr. Ramakant Nevatia for the useful discussions I had with him and for agreeing to serve on my dissertation committee.

I am very grateful to Dr. Prasanna Kumar for taking time to serve on my dissertation committee.

I wish to thank everyone at the Institute of Robotics and Intelligent Systems for their help and encouragement. My special thanks go to Dr. Keith Price and to Andres Huertas for their help with problems of software and hardware. I especially appreciate Dr. Price's help in setting up my software at Duke. I also wish to express my gratitude to Dr. Steve Cochran for his help with utility software and our many useful discussions, to Dr. Shou-Ling Peng for help in utility problems and to Bahram Parvin for helpful discussions. I deeply thank Dorothy Steele for her help, especially during my stay at Duke. I deeply thank Hagai Gefen and Jill Shuken for hosting me during my visits to Los Angeles. I thank Hillel and Nava Rom for hosting me during my last visit.

I wish to express my sincere gratitude to the Computer Science department at Duke University for allowing me free use of their computers and facilities, especially the Symbolics Lisp Machine. In particular I wish to thank Pete Boyd for his help in hardware problems and Jeff Tannehill for help in software problems.

I thank my brother, sisters and, in particular, my parents for their support and encouragement through the years.

Finally my deepest gratitude goes to my husband, Hillel Gazit, without whose active support and encouragement I would never have made it.

Contents

Acknowledgments	ii
List Of Figures	vii
List Of Tables	ix
Abstract	x
1 Introduction	1
1.1 Motivation	1
1.2 Our Approach	2
1.2.1 Goals	2
1.2.2 Distinguishing Features	3
1.2.2.1 Dynamically changing features	3
1.2.2.2 Use of shape, length and neighborhood information	3
1.2.2.3 Large disparities and shape deformation	4
1.2.2.4 Hierarchical feature matches	4
1.2.2.5 Multi-frame matches	4
1.2.2.6 Motion segmentation	5
1.3 Organization of this Dissertation	5
2 Previous Work	6
2.1 Motion detection and measurement	7
2.1.1 Intensity Based Approaches	7
2.1.1.1 Difference schemes	8
2.1.1.2 Correlation schemes	8
2.1.1.3 Gradient schemes	8
2.1.2 Feature based Approaches	10
2.1.3 Sequence based approach	13
2.1.3.1 Trajectory based methods	13
2.1.3.2 Spatiotemporal filtering	14
2.1.3.3 Epipolar Plane Image	14
2.2 Motion Segmentation and Estimation	16

2.2.1	Segmentation by Motion	16
2.2.2	Motion Estimation and Depth from Motion	19
2.2.2.1	Estimation from Point Correspondences	20
2.2.2.2	Estimation from correspondence of higher Level feature	22
2.2.2.3	Other (non-correspondence based) estimation methods	23
2.2.2.4	Depth from Motion	24
3	Problem Definition	25
3.1	Introduction	25
3.2	Specific Problems and Goals	26
3.2.1	Large motion	27
3.2.2	Shape deformation due to 3-D motion	27
3.2.3	Occlusion and disocclusion	27
3.2.4	Multi frame matches	27
3.2.5	Motion segmentation	27
3.3	Primitives	28
3.3.1	Some definitions	30
3.4	Computation of the features	30
3.4.1	Adaptive Smoothing	31
3.5	Assumptions and Constraints	34
4	Description of the Matching Algorithm	35
4.1	Introduction	35
4.2	General Description	37
4.2.0.1	The search space	39
4.2.0.2	Selecting the right contour sections	39
4.2.0.3	The similarity function	39
4.2.0.4	Spurious matches	40
4.3	Early Processing	42
4.3.1	Segment Matching	42
4.3.2	Neighborhood Computation	44
4.3.3	Initial sections for matching	44
4.4	Matching super-segments sections based on shape similarity	44
4.4.1	General Description	45
4.4.2	Similarity based Matching Step	49
4.5	Removal of overlapping matches	51
4.5.1	Definitions	53
4.5.2	Computation of approving neighbor matches	54
4.5.3	Remove overlapping matches	56
4.5.4	Matches Retrieval	56
4.5.5	Remove matches with weak neighborhood support	57

4.6	Multi Frame Matches	59
4.6.1	Creating Multi Frame Matches	59
4.6.2	Merging Multi Frame Matches	61
4.7	Hierarchical Matching	64
4.7.1	Example	66
4.8	Critical Evaluation	69
5	The Segmentation Algorithm	70
5.1	Primitives	71
5.2	Pairwise Segmentation	72
5.2.1	General Description and Goals	72
5.2.2	Definition of an Equivalence Relation	72
5.2.3	Analysis	74
5.2.4	Over-segmentation	74
5.2.5	Under-segmentation	74
5.2.6	Opacity Constraint	77
5.3	Multi Frame Segmentation	77
5.3.1	Initial Grouping of Multi Frame Equivalence Sets	77
5.3.2	Splitting multi-frame matches sets	78
5.3.2.1	Definitions	83
5.3.3	Merging multi-frame matches sets	84
5.3.4	Opacity constraint	84
6	Observations and Experimental Results	86
6.1	General Observations	86
6.2	Experimental Results	87
6.2.1	Jeep sequence	87
6.2.2	Advancing car	96
6.2.3	Crossing truck	101
6.2.4	Hallway sequence	106
6.2.5	Rotating car	110
6.2.6	An outdoor scene	113
6.2.7	A robot scene	117
6.2.8	Examples - Conclusion	121
7	Complexity and evaluation	123
7.1	Complexity Analysis	123
7.1.1	Pairwise Matcher	124
7.1.1.1	Segment Matching	124
7.1.1.2	Neighbor Matching	125
7.1.1.3	Similarity Based Matching	125
7.1.1.4	Neighborhood Computation	127
7.1.1.5	Overlap Removal	127

7.1.1.6	Total complexity of the Pairwise Matcher	127
7.1.2	Multi-Frame Matching	128
7.1.3	Hierarchical Matching	129
7.1.4	Segmentation	130
7.1.5	Total complexity for the sequence	130
7.2	Fine to Fine <i>vs</i> Hierarchy	131
7.3	Example	132
7.4	Sensitivity of parameters	133
7.4.1	The matching images	133
7.4.2	Segment threshold	133
7.4.3	Match Length Thresholds	137
7.4.4	Maximal Disparity	138
7.4.5	Initial Section Computation	140
7.5	Shortcomings of the method	142
7.5.1	Heuristic	142
7.5.2	Slow	144
7.5.3	Processing in 2-D	144
7.5.4	Using results for 3-D estimation	145
7.5.5	Segmentation of sparse data	145
7.6	Evaluation - other methods	145
8	Conclusion and Future Research	147
8.1	Summary	147
8.1.1	Conclusion	147
8.2	Suggestions for Further Research	148
Appendix A		
	Computing the area between the sections	159

List Of Figures

3.1	A super-segment	30
3.2	Jeep and train image - features found with different scale params	33
4.1	The complete matching process	36
4.2	The pairwise matcher	38
4.3	Matching line-segments	43
4.4	Why dividing the super-segment to sections is necessary	45
4.5	Initial segment-based section matches	47
4.6	Area between two matching sections: <i>shape similarity score</i> is defined as the area over perimeter squared (to be minimized)	48
4.7	Expanding section matches	48
4.8	Overlapping matches	52
4.9	Neighborhood Support: Adjacent section matches are <i>approving matches</i> except for the highlighted match	55
4.10	Resolving Overlapping matches	58
4.11	Multi-frame Matches	60
4.12	Merging adjacent Multi-frame Matches: (Q_1, Q_2, Q_3) and (R_2, R_3, R_4) are merged into (S_1, S_2, S_3, S_4)	62
4.13	Hierarchical Matching: match features in F_1 and F_2 after smoothing with coarse scale S_h , propagate to next level (scale S_l), then match frames smoothed with fine scale S_l using predictions.	65
4.14	Jeep and train image - features found with different scale params	67
4.15	Jeep and train sequence - hierarchical matches	68
5.1	Pairwise segmentation algorithm	73
5.2	Pairwise Segmentation: image segmented into regions based on similar 2-D motion between neighbor matches.	75
5.3	Segmentation problems	76
5.4	Multi-frame segmentation algorithm	79
5.5	Initial segmentation of the multi frame matches	80
5.6	Incorrect grouping as a result of short multi-frame matches: the rectangle is separated into two regions, because short multi-frame matches were grouped with matches for the ellipse.	81

6.1	Two consecutive frames for the jeep and train scene	88
6.2	Super-segments for the jeep and train scene	89
6.3	Pairwise and multi-frame matches for the jeep and train scene . . .	90
6.4	Pairwise and multi-frame matches for the jeep and train scene . . .	91
6.5	Segmentation results for the jeep and train scene (frame 1)	92
6.6	Segmentation results for the jeep and train scene (frame 6)	93
6.7	Segmentation results for the jeep and train scene (frame 10)	94
6.8	Segmentation for the jeep and train set	95
6.9	Matching results for the advancing car sequence	97
6.10	Multi-frame matches for the advancing car sequence	98
6.11	Segmentation results for the advancing car sequence	99
6.12	Segmentation for the advancing car sequence	100
6.13	Matching results for the crossing truck sequence	102
6.14	Multi-frame matches for the crossing truck sequence	103
6.15	Segmentation results for the crossing truck sequence	104
6.16	Segmentation for the crossing truck sequence	105
6.17	Matching results for the hallway sequence	107
6.18	Multi-frame matches for the hallway sequence	108
6.19	Segmentation results for the hallway sequence	109
6.20	Matching results for the rotating car sequence	111
6.21	Multi-frame matches for the rotating car sequence	112
6.22	Matching results for the outdoor scene	114
6.23	Multi-frame matches for the outdoor scene sequence	115
6.24	Segmentation results for the outdoor scene	116
6.25	Matching results for the robot scene	118
6.26	Multi-frame matches for the robot scene sequence	119
6.27	Segmentation results for the robot scene	120
7.1	Image processed with different segment thresholds for frame 61 of the advancing car sequence	135
7.2	Matching result for frames 61 and 71 of the advancing car sequence using different segment thresholds	136
7.3	Matching result for frames 61 and 71 of the advancing car sequence using different match length thresholds	139
7.4	Matching result for frames 61 and 71 of the advancing car sequence using different maximal disparities	141
7.5	Matching result for frames 61 and 71 of the advancing car sequence using different initial sections lengths	143

List Of Tables

7.1	Information on frames processed by different scales	132
7.2	Timing results for hierarchical <i>vs</i> fine-to-fine single scale matching .	132
7.3	Information on the matched frames	133
7.4	Analysis of matching with different segment thresholds	134
7.5	Analysis of matching with different min match lengths	138
7.6	Runtime with different maximal disparities	140
7.7	Analysis of matching with different initial subsection length	144

Abstract

We propose a Correspondence Based Motion Detection and Analysis Scheme. Our system is capable of handling a large variety of complex scenes, such as scenes with multiple moving objects, 3-D motion, large displacements, occlusion, disocclusion, rotation and shape deformation.

The system is composed of a pairwise matcher, a multi-frame matcher and a segmentation process. Results of the pairwise matcher are combined into multi-frame matches for the whole sequence. The multi-frame matches are then used to segment the image.

Our system uses sections of edge contours of any length or shape. The features are dynamically adapted through the algorithm, to maximize match quality. We show that using unrestricted contour sections has significant advantage over previously used features. We characterize a “good” match by its *similarity*, *length* and *neighborhood approval*, and show how these properties can be utilized by our algorithm to distinguish correct matches from incorrect ones. We show how we can apply our algorithm to a hierarchical set of edgels, using results of a higher level to aid computation in lower levels. We describe our method to combine pairwise matches along the image sequence to obtain *multi-frame matches*.

We also present a motion segmentation method that uses the multi-frame matches as the basis for segmentation. Our algorithm uses the measured 2-D motion of the features and groups together closely localized multi-frame matches that represent a similar 2-D motion through most of their frames.

We show the performance of our algorithm on several real image sequences, indoor as well as outdoor scenes.

Chapter 1

Introduction

1.1 Motivation

Motion analysis is an important research area within the field of Computer Vision. Moving objects and changing scenes surround us. Our own motion cause us to see even stationary scenes changing. The information content associated with a moving object (or objects) can be used not only to estimate its motion parameters but also to assist in segmentation and shape analysis.

The ability to discern motion plays a central role in biological vision systems. Sophisticated mechanisms for extracting and utilizing motion information exist even in simple animals. For example, the frog has an efficient “bug detection” mechanism that responds selectively to small, dark objects moving in its visual field. The ordinary housefly can track moving objects and discover the relative motion between a target and its background, even when the two are identical in texture and therefore indistinguishable in the absence of relative motion [70]. In higher animals, including primates, the analysis of motion is more complicated and some modules may be “wired into” the visual system from the earliest processing stages.

It is not surprising, therefore, that investigators with different backgrounds such as psychophysics, neurophysiology, computer vision and computer graphics, are pursuing research into motion. The interest of psychophysicsts and neurophysiologists in motion centers around the understanding of the biological motion sensing systems. Researchers in computer graphics are interested in the generation of images and, in particular, the generation of moving images. Researchers in computer vision are interested in the analysis, processing and understanding of images.

Analysis of sequences of images requires enormous computing resources. The research in the field has been limited by the available computing power and memory. As technology advances, faster computers, cheaper memories and parallel computers make research in motion more feasible.

A broad set of applications motivates the research in motion. These include medicine, tomography, autonomous navigation, communications and television, video conferencing, meteorology and so on. For example, the application of video conferencing and the desire to transmit sequences of images on channels with limited bandwidth serve as a strong impetus for the research on motion and compression of images. Recognition and tracking of targets is of immense interest to the defense department in every country. Autonomous navigation has many applications in the defense and space industry. In meteorology, satellite imagery provides the opportunity for interpretation and prediction of atmospheric processes through estimation of shape and motion parameters of atmospheric disturbances.

1.2 Our Approach

Most of the research in motion concentrates on detecting and measuring the motion, then evaluating it further for scene analysis. The second part involves estimating the motion parameters, segmenting the image into regions of common motion and scene reconstruction. Each of the above problems is very difficult to solve.

In this section we describe our approach to solving some of these problems.

1.2.1 Goals

1. We wish to develop a stable motion detection scheme, capable of correctly detecting motion for a wide range of real image sequences. The system should be relatively insensitive to poor or noisy images and changes in illumination, should be able to handle occlusion, large motion and shape deformation (due to 3-D motion).
2. We wish to detect sequence motion parameters, rather than just pairwise motion parameters.

3. We wish to use the sequence (multi-frame) motion parameters to segment the image into regions that are likely to correspond to different objects.

1.2.2 Distinguishing Features

Motion detection systems can be divided into three types. Intensity based methods use the intensity data for detection of motion information, sequence based methods use information along the time domain in a sequence of closely sampled frames. Feature based methods match features detected in each frames in order to obtain a sparse motion map. We believe that feature correspondence is the best way to address the above problems. Feature based schemes are, in general, less sensitive to noise, illumination variations. Both intensity and sequence based methods usually require closely spaced frames. Since correspondence is done for a pair of frames, we need to combine the results of matching successive pairs of frames into multi frame matches. The sparseness of the motion information presents special problems in the segmentation.

The distinguishing features of our system are:

1.2.2.1 Dynamically changing features

Selecting the appropriate features is a crucial step in any correspondence scheme. We choose to use edgel contours of varying length and shape. We believe that these features are very natural, as edge contours correspond well to object boundaries. The length and size of the contour sections are dynamically adapted by our algorithm to best fit the corresponding sections. In that respect, our method differs significantly from other correspondence schemes that use static features.

1.2.2.2 Use of shape, length and neighborhood information

Previous correspondence schemes tend to use simple features, such as edgels or line segments. By their very nature, these features are not a very good description of a complex shape and cannot supply enough unique information to resolve ambiguities. On the other hand, a general edgel contour of arbitrary length and shape can give a better and more unique description of the object boundary, and can be matched

more easily if detected correctly. We find that the degree of similarity between the shapes of the two matched contour sections and the length of the sections provide a very reliable indication to distinguish good from bad matches. We are able to adapt the features to maximize match quality.

Some neighborhood information is used by almost every correspondence scheme to verify matches. We find that the number and length of neighboring matches that represent a similar motion is, together with the length and similarity, a very good predictor of the quality of a match. We are able to incorporate these three important factors into a single formula to evaluate the matches.

1.2.2.3 Large disparities and shape deformation

Our algorithm can handle large disparities (we are able to match images which matching features are as much as 150 pixels apart). The key is finding a long and similar match with significant neighborhood support, thus the distance is not a major factor. Our algorithm is also able to handle shape deformation, due to 3-D motion, occlusion, illumination changes. The algorithm can handle arbitrary motion.

1.2.2.4 Hierarchical feature matches

Matching is a very complex and expensive operation. Given a hierarchical set of edgels, we are able to match the images hierarchically by first matching the smallest set of features, propagating the results to the next feature set, then use the predictions to restrict matches in the next level.

1.2.2.5 Multi-frame matches

While edgel or segment matches are fairly easy to combine along a sequence, contour matches are more difficult, as the matches may overlap only partially. We designed an algorithm that splits section matches so as to combine them into multi frame section matches, and also merges adjacent similar matches together to both reduce the fragmentation and improve the quality of the matches. As for the pairwise matches, the length of sections, the similarity between their shapes and the

neighborhood approval are also good indicators for correctness of the multi frame matches.

1.2.2.6 Motion segmentation

Because we only have sparse feature matches, and because we only estimate 2-D motion, segmentation is a difficult task. We group matched sections together based on similarity in the estimated 2-D motion, and their location. Using the multi frame matches rather than only the pairwise matches, however, allows us to perform a more accurate segmentation. We propose a method to tackle the difficult problem where two objects are both close in proximity and have a similar 2-D motion in some frames, and may therefore be grouped together. For stationary images and a side moving camera, our algorithm gives a rough depth segmentation of the image. For a camera that moves in the direction of gaze, the whole image will usually be segmented into one moving object, due to more gradual depth change.

1.3 Organization of this Dissertation

- Chapter 2 contains an survey of related material.
- Chapter 3 describes the problem and the goals of this research.
- Chapter 4 presents our matching algorithm.
- Chapter 5 presents our segmentation algorithm.
- Chapter 6 shows the results of our solution applied to several sequences.
- Chapter 7 presents the complexity of the algorithms and discusses shortcomings of the method.
- Chapter 8 discusses the contributions of this research to the field of motion analysis and suggests further research.
- Appendices:
 - Appendix A presents our approach to compute the shape similarity.

Chapter 2

Previous Work

Processing 2-D images to recover 3-D scenes is a complex task. A single 2-D image can represent an infinite number of 3-D scenes. However, our visual system can give a good interpretation of even a single image, provided it contains some structure. When a sequence of images is available, objects can be identified correctly even if no structure exists (random dot data) based only on the motion (or stereo) information.

Research in motion has concentrated on identifying similarities and differences between time varying image data as a means of detecting structure and reconstructing the 3-D scene. Analysis of images consists of extracting motion information from the image sequence, usually represented as a *displacement field* or *optical flow*. This information is then *segmented*, that is, similar primitives are grouped together based on the motion information obtained in the previous stage. Because the information is 2-D, further interpretation is then done on the segmented image, such as 3-D parameter estimation, depth map, surface reconstruction, object recognition, collision avoidance.

Because of the complexity of the problem, restrictive assumptions are made by the motion detection and estimation algorithms. The most common restrictions are:

- Static camera and moving objects, or
- Moving camera and static scene,
- Limit on the number of moving objects (one or two),
- Limited camera motion (only translation, limit on tilting, etc),

- Limited object motion,
- Known camera motion parameters.

The analysis of time varying imagery may be divided into two parts. The first is the measurement of motion, such as assigning of direction and magnitude of velocity to elements in the image, on the basis of the changing intensity pattern. Several approaches exist for this part, mainly correspondence and gradient based schemes. The second is the use of motion measurements for scene analysis tasks, such as segmentation, motion estimation and 3-D shape estimation.

2.1 Motion detection and measurement

A sequence of time varying frames contains an enormous amount of data, some of which may not be essential for motion analysis. The purpose of this stage is mainly to extract the time varying information and put it in a usable format. Motion in the image can be described in terms of a vector field $V(x, y, t)$ giving the 2-D velocity of a point with image coordinates (x, y) at time t . This computation is the measurement of visual motion. In some cases it is not necessary to measure the velocity field precisely, as when a quick response to a moving object is required.

There are two basic schemes for early processing of motion. These are *intensity based schemes*, which are based directly on the local changes in brightness values. These can be divided into three categories, namely differencing, correlation and gradient schemes; and *feature based schemes* where prominent features of the image such as edges, corners or regions are extracted and matched over time to calculate optic flow or the disparity field of the sequence. A third approach tries to utilize the time information by processing several frames at once and examining the paths of points over time. The Epipolar Plane Image Analysis [11] is a prime example of this approach.

2.1.1 Intensity Based Approaches

Intensity Based approaches employ the local changes in brightness of the images. They are faster and easier to parallelize than the feature based approaches but

require high data volume, are highly sensitive to noise and changes in illumination, and except for some difference schemes, can only handle small motion.

Intensity based schemes are usually divided into three types:

2.1.1.1 Difference schemes

In these schemes one image is subtracted from the other and the result is thresholded. Clusters of non zero points in the thresholded difference image represent moving objects. This requires small amount of computer time, and for this reason these methods are widely used as peripheral, attention attracting, “early warning” schemas and also to separate moving objects from their background [42, 39, 41, 7]. Simplicity and efficiency are the main advantage of difference schemes. The disadvantages lie in the requirement for registered frames, constant illumination and stationary observer. Also moving objects should be far enough apart between frames so they do not overlap, otherwise the shape of the moving object cannot be directly recovered.

2.1.1.2 Correlation schemes

In this method patches of one image are correlated with patches in subsequent images. A peak in the correlation function signifies a match and defines a disparity in the image [43, 4, 12]. The technique suffers from the following limitations: It requires the presence of a detectable texture within each correlation window and so fails in featureless or repetitive texture environment, it tends to be confused by the presence of a surface discontinuity in a correlation window, is sensitive to absolute intensity, contrast and illumination and gets confused in rapidly changing depth fields (e.g. vegetation).

2.1.1.3 Gradient schemes

These methods are all based on the relationship between the intensity gradient at a given point and the temporal intensity changed produced at that point when the image is moving. They allow a point by point determination of disparity based on purely a local criteria and are widely used for the computation of optical flow. If $I(x, y)$ denotes the intensity function of the image, $\frac{dI}{dt}$ is the temporal intensity

change at position (x, y) , G_x, G_y represent the intensity gradient at the image point and u, v are the local velocities in the x, y directions respectively, then we can write:

$$-\frac{dI}{dt} = G_x \cdot u + G_y \cdot v$$

where $\frac{dI}{dt}, G_x, G_y$ are all measurable by the observer. Thus we get one equation with two unknowns u, v . As a result, only one component of the motion can be determined by the above equation. This is often referred to as the *Aperture Problem*. Additional constraints, typically smoothness constraint, are used to overcome this difficulty. To prevent smoothing over motion boundaries several techniques are employed, including clustering or smoothing only along edgel contours (see below).

Additional problems arise due to the fact that spatial and temporal derivatives must be determined discretely depending on local information. Thus this method is highly sensitive to noise and is not able to cope with large disparities, changes in illumination, etc. It also treats the image as moving as a whole. There are some paradoxes with optical flow: for the method to be mathematically correct, the displacement should be small (since we approximate derivatives). But to get more accurate results, we need a large displacement. Also motion measurements are usually incorrect at motion boundaries - precisely where they are most needed.

Horn and Schunk [37] combine the temporal-spatial gradient method with an iterative optimization procedure to estimate the disparity vector of each sample point in the image. This method assumes the rate of change of disparities is limited, and therefore fails at object boundaries. Nagel [58] describes a more general technique based on the gray value variations directly in order to constrain the variation of the displacement field. Clustering methods, trying to cluster points based on their computed velocities using the Hough transform [22, 81, 82], sometimes combining that with clustering by intensity [81]. Peaks in the histogram correspond to dominant velocities and so enable us to handle multiple moving objects.

Hildreth [34] convolves images with a Laplacian of Gaussian mask and computes velocities only along the zero crossings contours. Velocity is estimated by minimizing the sum of two error terms integrated along a contour. The first represents the approximation constraint (the squared difference between measured and

estimated displacement component normal to the contour), the second represents the smoothness requirement (the squared derivative of the displacement field with respect to the arc length along the contour). This approach is interesting in that it only computes the optical flow along zero crossings contours, which represent intensity changes, instead of doing so for the whole image. Since the gradient is zero in homogeneous regions it is reasonable to compute the disparities only along contours. Using zero crossings contours also removes the noise and illumination sensitivity, because we do not deal with the intensity image directly but with edges computed from convolution over a neighborhood. She addresses the aperture problem by using the normal flow component (the flow component in the direction of the intensity gradient). The method suffers from the other above mentioned shortcomings, namely that it treats the whole image as one moving object and that it cannot handle large disparities.

Multi-resolution methods have been proposed mainly to overcome the problem of handling large disparities [13, 5, 4, 59, 31]. The idea is to operate on band-pass image pyramids [13]. The hierarchical disparity computation is performed by a coarse-to-fine algorithm in which the updated disparity field is in each stage projected to the next level. A refinement and relaxation processes are used to detect errors arising from mistakes in previous levels, however this is obviously a problem with all multigrid methods. Anandan [4] suggests a system to compute a dense field of displacement vectors with associated confidence measures. The system contains a spatial frequency decomposer, a matching module, a projection scheme and uses the smoothness constraint. It estimates displacement vectors using coarse to fine without feedback (so errors in higher levels are propagated down).

2.1.2 Feature based Approaches

In these methods, identifiable elements - features - are located and then matched over time. Matching can be established between simple features such as points, line segments, blobs, contour fragments or between complex features such as structured forms or even the images of recognized objects. Primitive features will have many competing possible matches, but have the following advantages: Reduced

preprocessing requirement which is important in motion perception, where computation time is often limited; by using primitive tokens the correspondence process can handle arbitrary objects and complex shape changes; the preprocessing done to obtain the complex forms may introduce errors of its own. Ullman [89] proposes that in humans correspondence is based neither on luminance alone nor relating complex form, but is rather an intermediate level process based on relatively simple properties of the image.

Many approaches have been tried. They differ in their choice of primitives and the criteria used to resolve ambiguities. Many of the matching algorithms work only under limited conditions (only for stereo, when the only motion is in the image plane or when the motion is small). If the chosen features are edgels [52, 38], the correspondence is governed by the distance between the dots. If they are line segments as in [53, 8], relative lengths and orientations can be used to restrict the search space further, thus reducing ambiguities. The metric affinity function can be defined as a function of distance, orientation, length and contrast. Other approaches use vertices [3, 6], local statistics [42, 78], extrema of local grey level curvatures [20], corners [19], regions [69], high curvature points in zero crossings contours [77]. Each method has its own affinity function.

The choice of features is a difficult one. Simple features are easy to detect but hard to match. Edgels, for example, are certainly too local to be used effectively. Methods that use edgels employ some neighborhood criteria to choose between matches, but this is still quite low level. Line segments are better in that they have some continuity information built into them, yet are still local and simple enough that they are easy to extract and the chance of the same line segment to belong to two different objects is very small. However, a match is hard to localize as any point along the segment may match any point along the corresponding segment. Indeed, the line matching algorithms mentioned above (as are most edgel matching algorithms) alleviate this problem only because they are designed for stereo matching. These algorithms do not employ continuity between segments or points of high curvature. They try to “slide” the two line descriptions to get a maximal fit, but they do not employ curvatures for that, though that seems a more natural way. Methods that use corners or curvature extrema require these points to be detected correctly, not an easy problem in itself, especially when curves are smooth.

In matching they need to deal with appearance and disappearance of points (due to occlusion or deformation of the contour due to rotation, expansion or contraction). They do not employ continuity along the contours well.

The correspondence between figures does not depend on affinity alone. Usually a one-to-one requirement exists, and so a feature may not match its highest affinity neighbor, since that has already another match. Some features may not have any match due to disappearance of objects, occlusion and shadows.

Some optimization criterion is required to select the best match for each feature. These criteria range from simple cross correlation [30] to complex and sophisticated graph matching techniques [38]. Relaxation techniques have been used for disparity determination [9, 20].

Sometimes correspondence is solved simultaneously with segmentation or the evaluation of the motion parameters. These methods use a search method in the parameter space, such as the Hough Transform [62].

Region matching has been used to evaluate frames which differ significantly, in which case most matching techniques fail. Price and Reddy [68, 69] as well as Price [67] have shown the use of symbolic representation for the matching. Region matching suffers, however, from sensitivity to illumination changes, occlusion and when the matched objects do not differ much from their background.

Matching is computationally expensive. Methods to reduce this cost include restrictive assumptions, such as a limited disparity range, and coarse-to-fine resolution matching [29, 32].

In chapter 4 we describe our matching method [27]. This algorithm is an extension of the algorithm described in [26]. We use directed edge linked contours as our primitives. The algorithm divides the contours into small pieces and matches them, extending them during the process. Our method differs from most previous methods by its implicit use of *shape* of the contour and *length* of connected match. The features we use are relatively easy to extract. Continuity along the contour is heavily used, and the matching is based on similarity between sections of contours. For these reasons we prefer contours of zero crossings of Laplacian of Gaussian masks which are guaranteed to be closed, are usually long, are not sensitive to small local changes, illumination or noise, and can be used hierarchically (by decreasing the size of the mask). Our algorithm can handle large disparities, as long as the matched

shapes (or rather their zero crossings representations) are similar. By using these arbitrary sections and allowing them to expand, we get matches between contour fragments of any length and shape. The longer the match and the less “straight” the matched contour fragments are, the better the match.

2.1.3 Sequence based approach

The previously described methods consider 2 or 3 frames. Methods using more frames were slow to appear due to the high costs of memory and computation required. These can be divided into *trajectory based methods*, where correspondences are verified by the trajectories they create, *spatiotemporal filtering* that examines closely sampled images in the Fourier domain, and the *Epipolar Plane Image* where a large block of closely sampled frames is examined by creating a set of epipolar plane images along the temporal dimension.

Faugeras [21] analyzes the relation between the motion of 3-D curves to optical flow. He shows that curves that are moving in a “non-elastic” way, such as ropes, the full apparent optical flow (the partial derivative with respect to time when the arclength is assumed constant) can be recovered, while the full real optical flow (the projection of the 3-D velocity field) cannot be recovered. If rigid motion is hypothesized, then in general, the full 3-D structure and motion of the curve can be recovered without explicitly computing the whole flow.

2.1.3.1 Trajectory based methods

Trajectory based recovery has received some attention. This is basically a correspondence scheme that tries to verify correspondence based on the trajectories they create. Jenkin [44] proposed such a scheme to track moving light displays. He assumed small motion, small velocity changes and small direction changes between frames, no occlusion or disappearance of points and known initial 3-D motion parameters. His correspondence is based on smoothness of the trajectory. His work was extended by Sethy and Jain [74]. Like him they assume that points cannot change their velocity or direction of motion abruptly, but they don’t assume the initial motion parameters are given or that the motion is small. They also allow

limited occlusion. Their computation is based on path coherence and smoothness of motion. They formulate the correspondence problem as an optimization problem and propose an iterative algorithm to find trajectories of points in a monocular image sequence. Their approach is interesting but it is not clear how well it can handle many points. Also they select points manually. Their method may not handle noise very well.

2.1.3.2 Spatiotemporal filtering

Another approach to motion measurements is based on spatiotemporal filtering, using a large number of frames sampled closely together in time. A family of motion sensitive filters is used, each of which is selective for motion in a different direction. The basic idea is that some properties of image motion are most evident in the Fourier domain. The power spectrum of a moving one-dimensional signal occupies a line in the spatiotemporal frequency domain. Analogously, the power spectrum of a translating two-dimensional texture occupies a tilted plane in the frequency domain. Adelson and Bergen [1] have pointed out that image motion is characterized by orientation in space-time. Therefore, spatiotemporally oriented filters can be used to detect it. Heeger [33] convolves three-dimensional Gabor energy filters with the input sequence. These filters can easily be tuned to different frequencies and orientation, trading bandwidth for localization. This model uses a family of Gabor energy filters all of which tuned to the same spatial frequency band, but to different spatial orientations and temporal frequencies. However, there is no general principle for selecting the proper scale for an image sequence to generate a family of filters.

The aperture problem is present in the frequency domain as well. For an oriented pattern, such as a two-dimensional sine grating, there is not enough information in the image sequence to disambiguate the true direction of motion. At best, only one of the two velocity components can be extracted.

2.1.3.3 Epipolar Plane Image

This approach was initially proposed by Bolles and Baker [11] and extended by Marimont [51]. In this method, a solid block of data, called the spatio-temporal

volume is constructed by a sequence of very closely spaced frames (corresponding points differ typically by a pixel or less). An *Epipolar Plane Image* or *EPI* is constructed from the sequence of epipolar lines along the temporal dimension. As they assume known camera parameters, linear camera motion, static scene and a fixed viewing angle, their method is actually an extension of Stereo and is used mainly to simplify the matching stage in stereo analysis. The simplest case is lateral motion, in which the camera viewing angle is perpendicular to the direction of motion. In this case each feature point follows a linear trajectory on the EPI. The slope of the line determines the distance from the point to the camera. Occlusion and disocclusion can be detected easily by the merging and unmerging of lines. Their algorithm analyzes each EPI independently to infer the structure of the corresponding epipolar plane, which is in effect a planar slice of the scene, and then to combine results from each planar slice to infer the structure of the entire scene.

Bolles and Baker show that when the camera viewing angle is not perpendicular to the direction of motion, the image point paths are hyperbolic instead of linear. Each hyperbola is from a two-parameter family, and the position of the corresponding point scene can be obtained from these two parameters as from the two parameters of the linear line in the perpendicular case.

Marimont [51] extended their work to deal with general planar motion and to curved objects using projective duality. The principle of duality states that any axiom or theorem about the projective plane remains true in the dual plane. For example, in the dual space, every point maps into a line, two points determine a line and, dually, two lines intersect in a point.

His idea is that a feature point P seen in m frames defines m lines of sight (from the camera to the point). The duals of these lines are points which should all belong to the line dual to P , regardless of the camera path. If the camera path is known, the parameters of each line of sight can be computed, and a point in the dual space constructed. The line corresponding to these dual points can then be computed. Thus we get back a “dual” EPI in which all lines are linear.

For curved objects, the idea is that in the plane, a curved object is merely a curve. As the camera moves around the object, the lines of sight trace out the

tangent envelope of the curve, and the scene point corresponding to the image edge point actually moves along the curve. However, the dual of a differential curve is the dual of its tangent envelope. To recover the shape of the original object, a curve is fitted to the duals of the lines of sight.

Peng [63] applies spatio-temporal reasoning on portions of the sequence to extract temporal information at edge points. Slices along the time axis are cut and processed, allowing for a speedup in processing. Velocity estimates are computed along spatial edge contours. To get a dense velocity map, regions between paths in the temporal slices (called *strips*) are extracted and velocity information is filled in by interpolation. The system requires that shapes of objects do not change much and that objects cannot (dis)appear suddenly. Within a slice, motion is assumed to be translational. The *strip* analysis requires stationary scene and that velocity of slice should change smoothly.

Under these assumptions, the method is able to compute depth and FOE estimation for several real sequences. The performance is slow, but a parallel implementation on the Connection Machine speeds the algorithm considerably.

2.2 Motion Segmentation and Estimation

In the previous section we have presented methods for measurements of motion. The result of this “early processing” stage is a vector field $V(x, y, t)$ giving the velocity of a point (x, y) at time t . In this section we describe methods that use this information for the segmentation of the image into objects, for the computation of 3-D motion parameters and 3-D structure.

2.2.1 Segmentation by Motion

A major step towards inferring the physical properties of objects in the scene, such as 3-D structure and motion parameters, is the segmentation of the image into regions that are likely to correspond to different objects. This early segmentation can be used to guide further processing of the image.

The human visual system can separate a moving object from its surrounding on the basis of motion information alone. It has been shown that the outline of the boundaries of moving objects can be detected even in the absence of intensity edges or texture changes at those boundaries (as in the case of random dot images).

Work on detection of motion boundaries has been categorized into two classes: one is that at an object boundary regions of more distant object will either appear or disappear over time. The other is to observe that two adjacent surfaces undergoing different motions, give rise to motion discontinuity along their boundary. As motion boundaries may be detected either prior to, simultaneously or following the image flow field, different approaches have been used.

To detect motion discontinuities prior to the computation of the flow field, Reichardt *et al.* [70] propose a method where direction selective movement detectors inhibit flicker detectors, when the same movement appears in the center and surround of the motion detectors.

Hildreth [35] uses the normal flow component (the flow component in the direction of the intensity gradient) to detect motion boundaries. She uses the fact that if two adjacent objects undergo different motions v_1 and v_2 , then a normal flow component, with orientation between the directions of v_1 and v_2 , will change sign and/or magnitude across the boundary. She therefore designed a detector that inhibits other detectors of the same type within some neighborhood (similar to Reichardt *et al.*) and excites those having roughly the same orientation but an opposite sign.

Spoerri and Ullman [79] also use the normal flow components as potential displacements. For each image point they use a local histogram in a circular neighborhood (of radius 5 to 8 pixels). As object boundaries give rise to discontinuities in the flow field, the velocities in the circular neighborhood will be clustered around two different points in the histogram. They apply five tests to detect precisely those points whose histograms exhibit this behavior. In addition, the appearance and disappearance of pairs of zero crossings of opposite contrast in the vicinity of a boundary are used to detect occlusion motion boundaries. The method has been

applied to random dot images and natural texture images (results are not shown for real images).

Most methods detect motion discontinuities after the computation of the flow field. Potter [64] uses region growing techniques to group features of similar velocity, assuming the image flow field is due to translation, Clocksin [18] shows that object and depth discontinuities give rise to discontinuities in the magnitude of flow created by an observer translating in a static environment.

Thompson *et al.* [83] show that object boundaries give rise to discontinuities in the image flow field, for the case of unrestricted motion. In principal, these sharp changes could be detected as zero crossings in the Laplacian of the components of the flow field. They derive a zero crossings edge detector to detect these changes. As the smoothness constraint (used by the flow detection algorithm) does not hold in precisely these points, the flow values computed there are usually incorrect.

Adiv [2] partitions a flow field into connected segments, each consistent with a rigid motion of a roughly planar surface, then applies a global, multi pass Hough transform to determine the parameters describing the motion and the plane. The segments are grouped under the assumption that they are created by a single, rigid moving object, by searching for the motion parameters compatible with all segments in the corresponding group.

Nelson [60] describes two complementary methods for the detection of moving objects by a moving observer. The first method, *constraint ray filtering*, uses knowledge about the observer motion. It is based on the fact [84] that in a rigid environment, the projected 3-D velocity at any point in the image is constrained to lie on a 1-D locus in velocity space whose parameters depend only on observer motion. Thus an independently moving object can be detected because its projected velocity is unlikely to fall in that locus. The second method detects rapidly changing motion, under the assumption that unlike the smooth observer motion, the apparent motion of moving objects changes rapidly. Motion detection is done

using a Gradient based method on specialized hardware in real time, where the observer motion is restricted to a few types of known and easily identifiable motion types. The main advantage of the technique is the fact that it can be applied in real time (using specialized hardware). It is a gradient based system, thus susceptible to noise and illumination problems. It requires known smooth observer motion, and rapidly changing object velocity. Objects that are moving smoothly will not be detected, even if their motion defers from the observer's motion.

Frazier and Nevatia [24] detect edge segments of moving objects viewed by a translating camera, A Complex Logarithmic Mapping (CLM) is used to convert the problem from one of detecting a complex motion along both the X and Y axes to one of detecting vertical motion along the angular axis. The system assumes the FOE for the observer motion and the maximal disparity in the sequence are known. The system detects vertical motion of horizontal edges and requires the presence of such edges in the moving objects. This is also a detection scheme, and does not provide structure or grouping of detected moving edges. The motion of the observer and objects is constrained.

Methods that compute image flow fields and motion boundaries in parallel exist too. Wohn and Waxman [92] try to detect motion boundaries by detecting where the approximation of the local flow field by second order polynomials breaks down. Koch *et al.* [47] propose that binary line processes as introduced in the Markov Random Field model [28] can signal boundaries. At locations where such a line process is set, a line or edge is postulated to ensure the smoothness assumption is not imposed across them. They then interpolate a sparse depth map activating the line processes to minimize a non convex energy functional.

2.2.2 Motion Estimation and Depth from Motion

Motion estimation algorithms use output of the results of the early stages to estimate the 3-D motion parameters of the moving object or the observer. Depth from Motion algorithms use the same information to compute depth information for static scenes viewed from a moving camera. The data depends on the methods used at the

previous stages, and may be difference pictures, optical flow, corresponding points, lines, curves, planar or curved surfaces.

The research in computational analysis of time varying images was first induced by the study of cloud motion, so most early papers consider 2-D motion only. Some of them use intensity based techniques, such as cross correlation to measure cloud motion from satellite image data [49]. Wolferts uses a sequence of areal images to analyzes vehicle movement in street traffic, by interactively choosing a vehicle on one frame and then following it using cross correlation [93]. Others use difference methods [41, 43, 40] and gradient methods [22, 14, 37]. Tracking features from frame to frame to estimate the motion is also used [55, 17, 65].

The current research in the field is concentrated in 3-D motion estimation. Tracking 3-D objects in space usually involves determination of their interframe rotation and translation. Formulating the equations relating the 3-D motion of objects in space and their 2-D projections on the image plane is rather straight forward, but solving these non-linear equations is difficult and the solutions, if attained, are very sensitive to noise.

Because the depth information of the scene is lost on the image plane, the translation vector can be determined only upto a scaling factor unless the z -coordinate of at least one point is known. Although this prohibits exact 3-D reconstruction, the observation of a certain number of points in two or more frames is sufficient to yield the position (to scale) and the relative motion to the viewing systems.

Before solving for 3-D motion parameters, equations should be expressed to relate the translation and rotation of an object in space to the 2-D disparities and velocities of its image in the image plane. The majority of the work is done using point or line correspondences as input.

2.2.2.1 Estimation from Point Correspondences

Assuming that point correspondences between five or more points are established in two frames, Nagel develops relations for calculations of translation and rotation matrices [56, 57].

Ullman [89] derives the following theoretical result: Given three distinct orthographic projections of four non-coplanar points in a rigid configuration, the structure and motion compatible with the three views are uniquely determined up to a reflection about the image plane. Roach and Aggarwal [72] applied the rigidity assumption to images of moving blocks.

Tsai and Huang [86] have proposed a method to find the motion of a planar surface patch from 2-D perspective views. They define a set of eight “pure parameters” which can be uniquely determined from two successive image frames by solving a set of linear equations. The actual motion parameters are determined from these eight “pure parameters” by solving a sixth-order polynomial. This method is very sensitive to measurement errors - a 2 percent error in measurement of a feature position on the image plane results in 30 percent error for the calculated rotation matrix and 95 percent error for the calculated translation vector.

Later the same authors investigated the problem of a curved surface patch in motion [87]. They prove that given the image correspondences of eight object points in general positions, the actual 3-D parameters can be determined uniquely without having to solve non-linear equations. This method is also very sensitive to measurement errors - a 1 percent error in the measurement of a feature point may result in 54 percent error in the estimation. A regularization technique is suggested to improve the quality and stability of a solution [94].

Shariat [75, 76] has shown that the solution of the motion problem can be reduced to solving a set of five polynomial equations of orders 2 and 5 with four unknowns, assuming either one point correspondences in five frames, two point correspondences in four frames or four point correspondences in three frames. The system is less susceptible to noise than the previous methods.

Franzen [23] proposes a minimization based solution for the Structure from Motion (SFM) problem when the object is undergoing chronogeneous motion (a class of motion that includes uniform acceleration, constant angular velocity rotation and translation). A second algorithm gives a closed form solution for point or line

segments correspondences, under the assumption that the motion is a uniform acceleration, and a third algorithm recovers refined image plane positions and rotation under the assumption that the motion is pure rotation. The algorithm assumes a stationary camera and a moving object, but the results also apply for stationary scene and moving camera. The first algorithm performs well even in the presence of (known) occlusion and uncertainty in feature position.

2.2.2.2 Estimation from correspondence of higher Level feature

The previous methods have used point correspondences, which in general, are not easy to extract. Therefore some methods using higher level features were proposed. Solutions for the problems of a set of lines in several frames were suggested.

Yen and Huang [95] have proposed an iterative method based on spherical projection and seven line correspondences over three frames for the case of general motion. Liu and Huang [50] show that for general motion six line correspondences over three frames are sufficient. Lee and Rosenfeld [48], using the rigidity constraint, reduce the number of feature points on the object to two and use a straight line analysis. Mitiche *et al.* [54] recover structure and motion using four line correspondences in three views.

Tsai [85] considers two views of a single conic arc and Subbarao and Waxman [80, 90] show that using the rigidity constraint, one planar surface patch in three frames or two planar patches in two frames are sufficient for analyzing the motion of planar surfaces.

Research has also been done as to whether a given motion field could have arisen from two different motions with two corresponding surfaces. It has been shown [80] that two solutions are possible in the planar case and possibly for quadratic patches too. However, Horn [36] shows that the class of surfaces leading to ambiguous motion fields is very restricted and only includes certain hyperboloids of one sheet and some degenerate forms. The reason for this is that for a motion field to be ambiguous, both solutions must have positive depth.

The problems of all correspondence based approaches are mainly due to their sensitivity to errors made by previous processes especially since they rely on error prone processes: Obtaining point or even line or region correspondences is a very difficult problem and small errors are very likely, but the above methods are likely to have large computation errors even when measurement errors are small. Shariat [75] tries to solve this problem by permitting large motion between frames, but then the correspondence problem becomes, of course, harder. Franzen [23] tries to model the amount of uncertainty in the location of the input points.

The methods using line correspondences are expected to be more stable, but they have not been tested on real images.

2.2.2.3 Other (non-correspondence based) estimation methods

Due to the inherent problems of getting reliable and accurate data from correspondence schemes, some methods that try not to use correspondences were developed. Kanatani analyzes the 3-D motion of a planar surface from the motion of its projected 2-D contour image [46]. Rather than use points correspondences, the motion is given by measuring “diameters” of the image contours on the image plane. The estimation technique is rough in that it uses several approximations, but can handle both small and large motion. In [45] another approach using numerical computation of certain line or surface integrals on the image is presented by the same author. This method is more accurate, but can handle well only small motion. Though point correspondences are not needed, correspondence for the image of the planar surface (as a closed contour) should be given.

Tsukune and Aggarwal [88] suggests a method for the case of multiple objects in pure rotation, assuming orthographic projection. The method is based on the parallel vectors found in the optical flow. The work of Adiv [2] was described in subsection 2.2.1. After he computes segments from the optical flow field, he determines their motion using the Hough Transform and then hypothesizes which sets of segments are induced by the same moving object, assuming rigidity. The hypotheses are tested by searching for 3-D motion parameters compatible with all the segments in the set.

The same problems that plague the correspondence based estimation schemes also arise for the optical flow methods. The solution of the equations is not well behaved numerically and is sensitive to noise and, as mentioned before, the computation of optical flow is itself very noise sensitive.

2.2.2.4 Depth from Motion

Knowing the motion of the observer allows us to compute depth similar to the way it is computed for stereo. The depth map can later be used for surface reconstruction, object recognition and other tasks, as the depth map obtained by stereo is used. Observing that the translational component of the observer contains information about the relative depth properties of the environment, Prazdny [66] uses optical flow induced by egomotion to calculate the relative depth map of the environment. Rieger and Lawton [71] use difference vectors of optic flows to approximate the motion of the observer (sensor) and relative depth. They assume a static environment. Bharwani *et al.* [10] assume a known translational camera motion. They use the fact that in translational motion, all points in the image move along the displacement path (defined as the path along the line in the image connecting the point and the Focus Of Expansion (FOE)). Given the axis of translation of the camera, the depth of a point can be determined from the position of the point and the extent of its displacement relative to the FOE. They iteratively improve the accuracy of depth estimates over a sequence of frames.

Chapter 3

Problem Definition

3.1 Introduction

In the survey, we described three different approaches to motion analysis, namely intensity based, feature based and sequence based methods. Each has its own merits and problems.

Intensity-based methods rely on the local changes in the image, and are therefore usually fast and easy to parallelize. However, these schemes are usually very sensitive to noise, contrast and illumination (since they typically use raw data). They require high data volume and, except for the difference schemes, can handle only small motion.

Although difference schemes are able to handle large motion, motion boundaries cannot be clearly recovered, unless the moving objects are so far apart that they do not overlap. Correlation schemes require detectable feature in each correlation window and tend to be confused by the presence of a surface discontinuity in a window. Gradient schemes have been very popular, because of their simplicity and the fact that they represent some physical reality. Unfortunately they are very sensitive to noise, and are not able to handle large motion. The image is treated as moving as a whole. Motion measurements are usually incorrect at motion boundaries. For the method to be mathematically correct it needs small displacements, but for accurate results it needs large displacements.

Sequence-based methods are characterized by using a large volume of closely sampled frames. As a result they suffer from some of the problems plaguing the intensity based schemes.

One approach uses spatiotemporal filtering. Usually there is no general principle to select the proper scale. The Aperture problem is present in the frequency domain as well. It is not clear how well the approach works with real data.

The Epipolar Plane Image (EPI) approach treats the sequence as one spatiotemporal block, and then constructs the EPI from the sequence of epipolar lines along the temporal dimension. Features are detected in each of those planes. Obviously, for the method to be successful, some very restrictive assumptions are made. The frames should be very close - disparity of one pixel or less, the camera motion is restricted (to linear or planar) and should usually be known. Scene should be stationary.

Feature-based methods involve detection of features and then matching. Using detected features has the advantage that usually the method is not sensitive to noise, contrast or illumination. The data volume is low, motion can be less restrictive and disparities larger.

Selecting the “right” features is a difficult task. Low level features are usually easy to detect and reliable, but convey low information context. Therefore they are difficult to uniquely match. High level features convey high information context and therefore are expected to be easy to match. They are difficult to detect correctly, though. Even with good features, matching is both very difficult and computationally expensive. Algorithms are not easy to parallelize and the methods are usually more heuristic. Feature matching results in a sparse disparity map. For accurate segmentation and motion estimation, additional work needs to be done in order to infer the disparity map for all points in the image.

We chose a feature based approach, because of its low susceptibility to noise, contrast and illumination and the low data volume. We believe that a feature based system is the best for handling complex and large motion.

3.2 Specific Problems and Goals

Our goal is to give reliable correspondence and segmentation on a wide range of images and disparities. In particular, we wish to be able to handle the following:

3.2.1 Large motion

Computing the correct correspondence in the presence of large motion is considered such a difficult problem, that most motion detection algorithms do not even try to tackle it. Instead, they only permit disparity of very few pixels, even less than one pixel. On the other hand, ability to detect large motion is useful, because of the significant reduction in the volume of the data, and because it is easier to verify matches representing a large motion, since incorrect motion is more apparent.

3.2.2 Shape deformation due to 3-D motion

Since our goal is to handle a large variety of images, a way to handle 3-D motion is necessary. But as detecting and computing correctly 3-D motion is not possible (as it normally requires the correspondence), we approach the problem by permitting the matched shapes to undergo some deformation. Thus we have some restrictions on the type of motion we allow: The objects cannot move so much that the features they generate are no longer similar in shape. However, as we will show, we were able to match objects with a complex 3-D motion.

3.2.3 Occlusion and disocclusion

Our goal is to be able to handle both occlusion and disocclusion. Thus we need to be able to handle appearance and disappearance of features through the sequence.

3.2.4 Multi frame matches

Correspondence schemes (ours included) typically handle only two frames at a time. Accurate motion detection is not possible using only two frames, thus a method that combines the pairwise matches of successive frames along the sequence is needed.

3.2.5 Motion segmentation

Segmenting the image into regions that are likely to correspond to different objects is a very important step towards the understanding of the scene. Motion Segmentation is a difficult problem. Ideally, we need 3-D motion parameters for every pixel in the

image in order to segment it correctly, but obtaining 3-D motion parameters is a very difficult problem. In addition, feature correspondence computes only a sparse disparity map, further complicating the segmentation.

Our goal is to compute the segmentation of the matched sections, based on the computed 2-D motion and their location.

3.3 Primitives

We believe that feature-based correspondence schemes have strong advantages over area-based schemes, because feature based systems consider much fewer points and are therefore faster than area based systems. By using features such as edgels, curves obtained by spatially linking these edgels, or even some approximation of these curves, the system is less susceptible to errors resulting from noise, change in illumination, etc.

The choice of features is crucial. Low level features, such as edgels are relatively easy to obtain, but matching them correctly is difficult, because of their low information content. Some grouping of edgels into more complex features is necessary to perform matching correctly. Edgels are a very logical basic feature, as they often correspond to object boundaries, and therefore the significant reduction in the amount of information does not necessarily mean a significant reduction in the quality of the information.

The simplest mode of edgel grouping is a linear approximation of a section of an edgel contour. The advantage of segments is that they are relatively local and easy to obtain, yet preserve some important continuity information. The chance of a segment to belong to more than one physical object is usually small, making segments a fairly reliable feature. These reasons lead us to use a simple segment matching algorithm as a first pass in our algorithm. It is a quick and simple method to provide an initial rough selection of possible matches.

Matching line segments has some basic problems. Although some continuity is present, it is usually not enough to resolve ambiguities created by two segments that locally match, but the contours they belong to do not. Most importantly, by their very nature, segments do not preserve the *curvature* information of the contour, although clearly the curvature information gives a more unique information on the

object, and therefore is more useful for matching. Localizing a match along a segment is inherently ambiguous. The *length* of a match, which has been a very reliable predictor for correctness in our algorithm, is not easy to use when matching segments, since extending the match to other segments along the contour contradicts the linearity assumption. Segments are only approximations of the contour, and sometimes a bad approximation (consider a circle, for example). Local changes in the shape of the contour can cause the segment fitting algorithm to break the contour differently.

Another possible feature are *corners*. They allow curvature to be used for the matching. However, corners are difficult to detect, are not a reliable source when the shape of the contour is either a straight line or a smooth curvature (a circle), affected much by occlusion, 3-D motion and noise. Like segments they are local features that lose their basic property (linearity in the case of segments, a single corner in the case of corners) when linked together.

All the above-mentioned features are local, restricted in shape and do not utilize continuity information well. Thus they are difficult to uniquely match.

We believe that sections of edgel contours are the most logical feature for the matching, where a section is any continuous portion of an edgel contour. Clearly linear segments are a special case of contour sections, but when we do not limit ourselves to linear line segments, we can use curvature for the match, as well as use arbitrarily long sections. Obviously a long contour uniquely shaped is easier to identify and to match uniquely.

Handling sections of arbitrary shape and length presents special problems. There is no simple representation, such as a line or a corner, and determining similarity between two curves is a difficult. As the shape and length are not restricted, and because a curve may belong to more than one object (a problem that does not usually exist for the simpler features), we need to be able to determine a maximal match between two portions of curves, when both the locations and length of these portions are not known. A method to break the curves into smaller sections is needed, and a way to extend the smaller ones to get the longer, more reliable match, is needed.

Figure 3.1: A super-segment

3.3.1 Some definitions

A *super-segment* is an object dually represented by a linked edgel contour and by a linear segment approximation of that contour. Every segment points to its begin and end position in the edgel list of the super-segment. The first segment in the list has begin position 0 and the last segment in the list has its end position equal to the length of the contour (in edgels) minus 1. We often use the terms super-segment and edgel contour interchangeably.

A *section* of a super-segment is any continuous portion of its edgel contour.

An example of a super-segment and its relation to edgels, segments and sections is given in figure 3.1.

3.4 Computation of the features

Our algorithm requires the computation of edgel contours. Thus we need to detect edgels and then link them. In principle, the method used to obtain edgels is unimportant, and almost any edge detection algorithm should do.

However, because length and curvature are so important for the matching, we prefer an edge detector that produces well connected boundaries, does not break corners or results in a lot of short lines, or a lot of noisy edgels.

In our previous work [27] we used contours of zero crossings of (large) Laplacian of Gaussian masks [16]. We got very good matches using these features, but

the results were hard to use because Gaussian smoothing tends to shift the edgels from their correct positions and may create spurious edgels. The number of edgels increases dramatically for smaller scales, resulting in a more difficult and expensive matching. Also zero crossings of LoG masks may not correspond precisely to object boundaries.

To overcome these difficulties, we now use adaptive filters [73] at different scales to smooth the images. We then detect edgels in the smoothed images using Canny's edge detector [15], link them, using an extension of our zero crossings linking algorithm [25] and extract segments and super-segments at each scale.

The advantage of this approach is that smoothing with adaptive filters preserves the causal properties of Gaussian filters without shifting the edgels or creating spurious ones.

The following is a more detailed description of the adaptive smoothing algorithm.

3.4.1 Adaptive Smoothing

The extraction of features such as intensity discontinuities from an image is an essential task in early vision. Those discontinuities correspond to the physical boundaries of the objects present in the scene, but, because of the complexity of the physical world and of the imaging apparatus, and of multiple sources of noise, the image to be processed is complex, and the detection of such discontinuities is non trivial. Furthermore, a crucial idea based on physiological observations is that an image can be interpreted at a few different scales depending on how much detail is taken into account. Hence, many researchers in the vision community recently focused on multiple scale features extraction.

Out of the major approaches proposed in the literature to tackle this challenging problem, Witkin [91] introduced the concept of scale-space: the idea is to embed the original image in a family of derived images $I(x, y, \sigma)$ obtained by convolving the original image with a Gaussian kernel using different values of σ . Larger values of σ , the scale-space parameter, correspond to images viewed at coarser resolutions. The major drawback of Gaussian smoothing is that features extracted at larger values of σ are not localized correctly since edges start interacting. It is then necessary to track coarser features across the different scales downwards to the finest scale in

order to find their correct location, which poses a difficult correspondence problem in 2-D.

The purpose of Adaptive Smoothing recently introduced by Saint-Marc and Medioni [73] is to smooth an intensity image (for instance) while preserving intensity discontinuities. This is achieved by repeatedly convolving the image with a very small averaging filter modulated by a measure of intensity discontinuity at each point. A relatively small number of iterations is needed to obtain a smooth image and a parameter k equivalent to σ in Gaussian smoothing fixes the amplitude of the discontinuities to be preserved. Since the latter are preserved, they are directly localized, hence no tracking is needed as opposed to Gaussian smoothing. The adaptive smoothing of an image $I(x, y, n)$ is performed as follows:

$$I(x, y, n + 1) = \frac{1}{R} \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I(x + i, y + j, n) c(x + i, y + j, n)$$

with $R = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} c(x + i, y + j, n)$

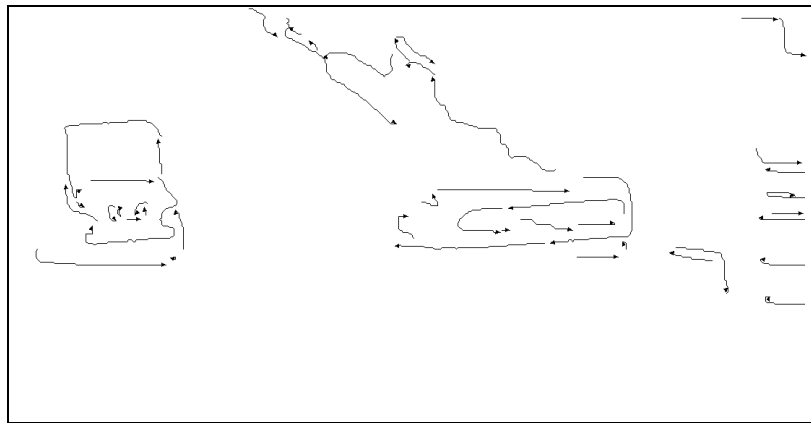
where $c(x, y) = f(d(x, y)) = e^{-\frac{d(x, y)^2}{2k^2}}$ with $d(x, y) = \sqrt{G_x^2 + G_y^2}$

$d(x, y)$ represents the magnitude of the gradient $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})^T = (G_x, G_y)^T$, computed in a 3×3 window.

Figure 3.2 shows the results of edge detection after adaptive smoothing with 2 different values of the parameter k , 16 and 8, for one frame of the jeep and train sequence. The number of iterations n was fixed to 10. The results show that the features detected at different scales are very well localized and do not move. Figure 4.14 shows the result of applying adaptive smoothing and edge detection for two frames of the sequence, and figure 4.15 shows how the properties of adaptive smoothing can be utilized for matching the frames.



(a) Jeep sequence - Frame 4



(b) Super-segments of (a) - scale 16



(c) Super-segments of (a) - scale 8

Figure 3.2: Jeep and train image - features found with different scale params

3.5 Assumptions and Constraints

The following constraints are assumed. However we were able to handle some deviation from them (for example, objects seen through a car window, occlusion, rotation and pure rotation and 3-D motion).

- **Solidity Assumption**

Objects are assumed to be solid with all parts having same motion.

- **Opacity Assumption**

Objects cannot be transparent.

- **Shape preservation**

The shape of an object should not change abruptly between adjacent frames, so that similarity in shape between corresponding features can be detected. In particular, rotation should be gradual (especially in the xy plane).

Chapter 4

Description of the Matching Algorithm

4.1 Introduction

In this chapter we describe our matching algorithm. We begin by a general description followed by a more detailed description of each step and the problem it solves.

Given an image motion sequence, our goal is to get a maximal multi-frame contour-section correspondence. Our solution is to first compute pairwise correspondences between every adjacent pair of frames along the sequence (for n frames, there are $n - 1$ such pairs), then combine the pairwise matches into multi-frame matches. Using a hierarchical set of features allows us to first match features at the higher level, then propagate the results to lower levels, using higher level matches as a guide in matching at the lower levels.

Figure 4.1 contains a diagram of our correspondence algorithm. The motion sequence is composed of frames F_1, \dots, F_n and each frame is smoothed by scales s_1, \dots, s_k , with $s_i > s_{i+1}$. Matching is performed from higher level down, and results at each level are propagated to the next lower one.

Sections 4.2 through 4.5 describe our pairwise correspondence algorithm. Section 4.2 gives a general description of the problem, issues and solutions. Section 4.3 describes the initial segment matching algorithm, section 4.4 describes the first stage of the contour section matching algorithm, in which a large set of possible contour section matches is selected based on shape similarity, and section 4.5 describes the refinement stage, in which spurious matches are discarded and a non-overlapping subset of matches is chosen based on their similarity and neighborhood support.

Figure 4.1: The complete matching process

Section 4.6 describes our multi-frame matching algorithm, whereby the pairwise matches are combined into multi-frame matches. Section 4.7 describes our hierarchical correspondence scheme.

4.2 General Description

Given two sets of super-segments detected in the two matched images, both smoothed with the same scale parameter, a set P of predicted matches computed by the hierarchical matching algorithm (see section 4.7) and the maximal disparity d , our intent is to find for each edgel contour section in one image, the matching section in the other image. Our matching criteria is to maximize *shape similarity* between contour sections and *length* of matching sections. This is done by computing the area between the (translated) matching sections and dividing it by the length of the sections, squared. The match that yields the lowest score wins. Squaring the length is necessary to ensure that short sections that are locally similar would not get preference over long sections for which the area is not as small. This is important since we expect shape deformation (due to the 3-D motion of the objects), and because in general the longer the matching sections are, the better chance there is for the match to be correct.

The above creates a very large number of possible section matches. The next step is to discard incorrect or extraneous matches. We combine the previously computed shape similarity measure with length and neighborhood approval and use the combined measure as our criterion to distinguish between correct and incorrect matches. Initially each match is obtained independently, based only on its location and similarity score. Thus a wrong decision at an early stage does not influence our selection of other matches. All matches have to be computed before we evaluate them further. The good matches are expected to have neighbor matches with similar motion. This is obvious considering that the contour sections represent parts of a moving rigid body. Figure 4.2 contains a diagram of our pairwise matching algorithm.

The following problems need to be addressed:

Figure 4.2: The pairwise matcher

4.2.0.1 The search space

In order to decrease the run-time of the algorithm and reduce ambiguity, we would like to decrease the number of possible matches in an early stage, yet minimize the effect of wrong decisions made in the early stages of the algorithm.

We restrict the search space by using the given maximal disparity, predicted matches (from higher level) and the results of matching the line segments. The assumption is that where no line segments matched (according to orientation and strength), the section contours cannot match either. This is done by step 4.3 of the algorithm.

In addition, we use matches from higher levels to predict location of matches of lower levels. Predictions made by matching previous frames in the sequence could be used also, but this part is not implemented.

4.2.0.2 Selecting the right contour sections

Edgels may disappear or appear due to either physical causes such as 3-D motion, occlusion, shadows or errors in the processing steps (edge detection, thresholding, linking). As a result, contours that actually belong to different objects may merge, or an object contour may break into several smaller disconnected contours or even partly disappear. Even given the approximate matches of the previous stage, it is still difficult to define where the sections should be “broken” into smaller sections. Our solution is to break the sections arbitrarily, so that the resulting smaller sections are short enough to not cover more than one match, and long enough to provide meaningful information. Then, using a “bottom up” approach, expand the match along the contour. This is done by step 4.4.

4.2.0.3 The similarity function

Intuitively, for a given section in the left image, we look for the “most similar” section in the right image. There are no algorithms for matching 2-D arbitrary curves (allowing for expansion, contraction, deformation or a slight rotation), when only part of the contours may match. Our solution is to use the area between the sections (after translating one of them to the beginning of the other) as the main

criterion, and divide it by the length squared, in order to prefer longer sections. Note that the area between two sections can be computed in time proportional to the lengths of the sections.

4.2.0.4 Spurious matches

We define spurious matches as either incorrect matches or possibly correct but overlapping matches. Our goal is to have the maximal number of non-overlapping correct matches. This is the main problem of the algorithm. The previous stage may create *many* matches, and as it discards none, we are left with a set of matches, the great majority of which are spurious. The number increases dramatically as the disparity increases. For example, over 95% of the matches created for the jeep and train sequence (with disparity 150) were spurious.

We say that correct matches usually have either of the two following properties:

- Long *or*
- Have supportive neighbors.

A correct match is expected to be long, since corresponding contour sections are similar in shape, and we try to maximize the length of the match when computing it. A correct match is expected to have supportive neighbor matches (matches of neighbor sections that represent similar motion), because the contour sections represent part of a rigid physical body. Other contour sections representing other parts of the body are close and have similar motion. Since every match is computed independently, we can assume that in the absence of repetitive objects or straight lines, the chance of many incorrect neighboring matches to represent the same motion is very small. This holds true even when there are some straight lines or repetitive structures, as can be seen in the examples, because the correct interpretation of a repetitive scene usually yields the best neighborhood support.

Thus, when comparing two overlapping matches, we choose the one that has a better combination of shape similarity, length and neighborhood support. Short matches with little or no support are discarded.

Notation

Let F_1 and F_2 be the matched frames.

A super-segment is an object dually described by its edgel contour and the segment approximation of that contour.

For frame F_k ($k = 1, 2$) let S_k be the set of super-segments $s_{i,k}$ of F_k .

The *position* of an edgel (or point) in a super-segment contour is its location in the ordered list of points that describes the contour. The position of the first point is always 0 and the position of the last point is the length (in points) of the contour minus one.

A *section* of a super-segment (or segment) s_i is a connected portion of its edgel list. It is characterized by its super-segment id and the begin and end positions in its edgel contour, as (s_i, b_i, h_i) .

Define the set of segments

$$A_k = \{a_{i,k} \mid \text{There exists super-segment } s_{j,k} \text{ that contains segment } a_{i,k}\}.$$

For every segment $a_{i,k} \in A_k$, let $\theta_{i,k}$ and $\ell_{i,k}$ represent its orientation and length (measured in pixels) respectively.

Since a segment is a linear approximation of a contour section, and is therefore a portion of some super-segment, we can say that every point along the segment has a position associated with it, that is the position of this point in the super-segment contour.

Let the *maximal disparity* d be defined as the maximal distance allowed between two corresponding features (measured in pixels). The maximal disparity is of course different for every sequence.

Let the *neighborhood disparity* d_n be defined as the maximal distance allowed between two neighboring features (measured in pixels). We have chosen to define the neighborhood disparity to be some constant fraction of the image dimension, as we assume that there is a relation between the image size and the object size.

4.3 Early Processing

4.3.1 Segment Matching

The following algorithm computes for each segment $a_{i,1} \in A_1$ a subset of segments $a_{j,2} \in A_2$ that can match $a_{i,1}$ (and vice versa):

For each segment $a_{i,1} \in A_1$ define a *disparity window* $w_d(a_{i,1})$ in which corresponding segments from A_2 must lie, and define similar windows for segments in A_2 . We use a rectangular window parallel to the segment with width $2d + 1$ and height $2d + \ell_{i,1}$.

We say that $a_{j,2}$ *intersects* $w_d(a_{i,1})$ within positions $b_{j,2}$ and $e_{j,2}$ when the portion of $a_{j,2}$ corresponding to these positions lies within $w_d(a_{i,1})$, and no larger portion does. Similarly, $a_{i,1}$ *intersects* $w_d(a_{j,2})$ within positions $b_{i,1}$ and $e_{i,1}$ when the portion of $a_{i,1}$ corresponding to these positions lies within $w_d(a_{j,2})$ and no larger portion does.

Clearly, if the whole segment $a_{i,k}$ falls in the disparity window, $b_{i,k}$ and $e_{i,k}$ are the begin and end positions of this segment.

We say that the pair $[(a_{i,1}, b_{i,1}, e_{i,1}), (a_{j,2}, b_{j,2}, e_{j,2})]$ is a *potential match*.

This potential match is a *segment match* if $a_{i,1}$ and $a_{j,2}$ have a similar orientation (as defined by equation 4.1), and the middle point of the shorter segment intersects the window of the other segment.

$$|\theta_{i,1} - \theta_{j,2}| \leq \theta + \frac{\pi}{2} \cdot \ell \cdot \left(\frac{1}{\ell_{i,1}} + \frac{1}{\ell_{j,2}} \right) \quad (4.1)$$

θ and ℓ are constants.

In our experiments we used $\theta = \frac{\pi}{6}$ and $\ell = 1$ (See [61]).

Figure 4.3 contains an example of the segment matches (the large parallelogram) and neighbors (the small parallelogram).

Note that the maximal disparity d may be very large. Where a prediction is available, we therefore use it instead. If a section containing (most of) segment $a_{i,1}$ is predicted to have a certain 2-D motion, there is no reason to search much further than around the predicted location. Thus, instead of creating the large w_d window for this segment, we create two *small* (disparity of 5 pixels or less) windows - w_0 around the location of $a_{i,1}$ and w_p around the location of the predicted match.

Figure 4.3: Matching line-segments

Only segments that intersect these windows are selected. If no significant match is found in these windows (if the prediction was incorrect for example), only then do we search the large w_d window.

The resulting segment matches (rather sections-of-segments matches) have each the form $[(a_{i,1}, b_{i,1}, e_{i,1}), (a_{j,2}, b_{j,2}, e_{j,2})]$.

4.3.2 Neighborhood Computation

For every segment $a_{i,k} \in A_k$ compute the set of neighbor sections-of-segments in a very similar way to the computation of the *potential* matches in the previous section. Define the window $w_{dn}(a_{i,k})$ as above.

We say that the pair $[(a_{i,k}, b_{i,k}, e_{i,k}), (a_{j,k}, b_{j,k}, e_{j,k})]$ are *neighbors* by the same definition as above. Note that $a_{i,k}$ and $a_{j,k}$ belong to the same image.

4.3.3 Initial sections for matching

Since the position of every edgel along the super-segment contour is unique, we can replace in each of the above pairs, the segment id by the containing super-segment id.

Thus the segment matches have the form: $[(s_{i,1}, b_{i,1}, e_{i,1}), (s_{j,2}, b_{j,2}, e_{j,2})]$ where $s_{i,k}$ is the super-segment containing segment $a_{i,k}$.

The neighbor pairs become $[(s_{i,k}, b_{i,k}, e_{i,k}), (s_{j,k}, b_{j,k}, e_{j,k})]$.

4.4 Matching super-segments sections based on shape similarity

In this section we describe our method of computing an initial set of section matches. The results of the segment matches and predictions from higher levels are used to restrict the search space. Each match is given a score representing its quality (based on similarity and length).

Figure 4.4: Why dividing the super-segment to sections is necessary

4.4.1 General Description

When we try to match edgel contours we are faced with the fact that edgels may disappear or appear due to either physical causes such as occlusion or errors in the processing steps (edge detection, thresholding, linking). As a result, contours that actually belong to different objects may merge, or an object contour may break into several smaller disconnected contours or even partly disappear.

Figure 4.4 illustrates this problem.

A possible solution would be to try to determine where a contour starts to represent a different object and cut it there, using by some method of corner detection. We do not believe that this is a good method. Apart from the fact that corner detection is in itself a difficult problem and no good simple solution for it exists, dividing a contour in the corners makes matching more difficult, because in trying to minimize the area between two contours, the curvature is more useful than linear piecewise segments.

Our solution is to divide the contours arbitrarily into small enough sections that will most probably not be larger than the true matching sections, yet long enough to convey meaningful information. Results of the segment matching can be used to determine maximal matching sections (by simply combining adjacent segment matches together). These maximal sections are then divided into smaller sections, each of which is then independently matched by comparing it to a possible matching contour section and searching along that section for the best match. Results of the

segment matching step are used to restrict the size of the matching contour section: if a section p_1 of a super-segment $s_1 \in F_1$ is compared to a super-segment $s_2 \in F_2$, the matching section p_2 has to have most of the segments overlapping it match the segments overlapping p_1 .

An illustration of this process is given in figure 4.5.

Choosing the “right” size for these smaller sections is an issue here - too short a section will not yield a meaningful match; too long a section may include more than one object. A constant section size penalizes short matches and a constant number of sections per contour penalizes long ones. Our solution is as follows: For every maximal section, any predicted match overlapping it is a section. The remainder of the maximal section (that may be quite fragmented after cutting out the predictions) is partitioned into sections of length approximating $\frac{2 \cdot p}{\log(p)}$, where p is the length of the shorter of the two maximal sections matched.

For each section p_1 in s_1 , find the section of s_2 within which p_1 may match (use matches of p_1 segments to determine the range). Then slide p_1 along this section and look for the most similar section p_2 .

The *similarity* measure is defined as follows: Assume p_2 is translated to the location of the first point in p_1 (or within a pixel, due to inaccuracies in the edge detection process, mentioned earlier). Then compute the *area* A between the two sections. p_1 is *most similar* to p_2 if the ratio $\frac{A+1}{(|p_1|+|p_2|)^2}$ is minimal for all possible p_2 's.

Intuitively, the above measure finds the sections that are most similar in shape, as illustrates in figure 4.6. The match cannot contain the contour portion where the sections are no longer similar, because the area between them is large, compared to the lengths of the sections.

If p_1 is a section taken from a predicted match, this search is much simpler. Only matches that represent a 2-D translation that is similar to the predicted match can be selected.

Once a possible match is detected, “extend” it by adding adjacent edgels as long as the similarity (previously defined) decreases. To reduce time complexity, this step can be done using a binary search type operation.

Figure 4.7 illustrates the process.

Figure 4.5: Initial segment-based section matches

Figure 4.7: Expanding section matches

Note that extending the matches can create many competing matches: if a super-segment was divided into several sections, each of which was then matched correctly and then extended, we may get several nearly identical (and overlapping) matches. However, this is actually viewed as an advantage. Having several competing matches for the same area *increases* the possibility that there be at least one correct match among them. That is true because every match is computed independently. Many correct but overlapping matches are easy to detect, because they have overlapping sections and similar translations. They can also be used to reinforce neighboring correct matches, since the number and total length of the correct matches is thus larger.

4.4.2 Similarity based Matching Step

1. Initial Section Computation

Let M be the set of initial section matches. Initially M is the set of segment matches obtained by previous step. Join adjacent sections together:

While there exist a pair of matches

$$m_1 = [(s_1, b_{1,1}, e_{1,1}), (s_2, b_{2,1}, e_{2,1})] \text{ and}$$

$$m_2 = [(s_1, b_{1,2}, e_{1,2}), (s_2, b_{2,2}, e_{2,2})] \text{ such that}$$

$$\text{for } i = 1, 2, \min(|b_{i,1} - e_{i,2}|, |b_{i,2} - e_{i,1}|) < c;$$

remove both m_1 and m_2 from M and replace them by the match

$$m_3 = [(s_1, \min(b_{1,1}, b_{1,2}), \max(e_{1,1}, e_{1,2})), \\ (s_2, \min(b_{2,1}, b_{2,2}), \max(e_{2,1}, e_{2,2}))].$$

2. Determining the matching sections

Let $m = [(s_1, b_1, e_1), (s_2, b_2, e_2)] \in M$ be an initial match. Divide (s_1, b_1, e_1) into sub-sections $(s_{1,i}, b_{1,i}, e_{1,i})$ as follows:

Every predicted match (obtained from previous level) that overlaps this section is a sub-section. The remaining portions are divided into equal length sub-sections, as explained in the previous paragraph.

Figure 4.5 illustrates the idea.

3. Section Matching

For every sub-section $(s_{1,i}, b_{1,i}, e_{1,i})$ slide it along the matching section (s_2, b_2, e_2) and search for the *most similar* sub-section $(s_{2,j}, b_{2,j}, e_{2,j})$, under the following restrictions:

- Use segment matches to restrict search space (only sections covered matching segments are possible candidates).
- The 2-D translation of the match should be less than the maximal disparity d .
- If the the checked sub-section has a predicted match (and therefore a predicted 2-D translation), the 2-D translation of the match should be close to the predicted translation.

4. Match Extension

Extend the match computed in the previous step, by adding adjacent points to both sections, as long as the similarity (computed as explained by previous paragraph) does not decrease.

Notes

Matching each piece is done independently, so non unique matches are allowed, since we hope that at least one “catches” its correct location. Dividing the initial large section into smaller pieces is necessary since the sections often do not fully match, but portions of them do (due to motion of objects, changes in illumination, occlusion or errors of the edge detector). Extending the matches is necessary, as long matches are much more reliable than shorter ones, so good matches are better distinguishable from bad ones.

We choose a bottom up approach, in which we break the initial matching sections into small enough pieces and try to match each such section, then try to extend the match as long as the shape of the curve is similar enough. (Another option is to determine where is the best place to “break” a super-segment, but this is complicated, since it requires finding corners, junctions and other high level features, and may fail when we have occlusion and motion.)

4.5 Removal of overlapping matches

The previous steps compute every match independently of other matches. No comparative evaluation of matches is made and none is discarded. The result is a very large set of matches, the great majority of which are spurious.

We divide the spurious matches into two groups:

- *Correct* matches that are contained by better (read, longer) correct matches,
or
- Incorrect matches.

While the first case is easy to identify and solve, the second problem is difficult to solve. In this section we propose a solution that is based on both similarity, length and neighborhood approval.

Figure 4.8 contains an example to illustrate the problem. The left contour is covered by 3 overlapping matches. For a one to one solution, no one edgel can match more than one other edgel. Thus the overlaps need to be resolved.

Our solution is based on the fact that *correct* matches are usually either

- Long or
- have a high neighborhood approval.

A correct match is expected to be long, because every match is extended to cover the maximal similar section. Even if the initial section used for the search was short, if the match that was computed was correct and there was room for extending it in both contours, this should have been done by the previous step. The above would not work, of course, if the contour shape was deformed so much that no similarity was found, and indeed in that case the algorithm fails. But this can only happen if the image of the object changes very significantly between frames, violating the assumption on which our algorithm is based.

A correct match may be short if the contour was fragmented, since we make no attempt to link disconnected contour segments. However, that obviously results in many neighboring matches that represent a similar 2-D motion, and thus a high neighborhood approval. This is also expected to happen for a long correct match.

Figure 4.8: Overlapping matches

The reason for that is our rigidity assumption. If objects are fairly rigid, images of nearby portions of the same object have similar 2-D motion, even if the motion of the object is not 3-D.

Our solution is thus to compute for every match the approval score, depending on the number of neighbor matches that have similar motion. Resolving overlaps is done by choosing the match that combines a high similarity, is long and has a high approval score.

We assume that every segment keeps a list of all the matches that overlap it. In addition, every segment keeps a list of its neighbor pairs. Thus computing neighbor matches for any given match is simply a union of the matches covering the segment portions that overlap the match and the matches covering the segment portions that are neighbors of the segment portions that are covered by the match. A more precise definition is given in the next section.

4.5.1 Definitions

Let $s_{i,k}$ and $s_{j,k}$ be the id's of the i^{th} and the j^{th} super-segment in the k^{th} frame of the sequence (note that both super-segments belong to the same frame F_k). Let $b_{i,k} \leq e_{i,k}$ be positions in the edgel contour of $s_{i,k}$ and $b_{j,k} \leq e_{j,k}$ be positions in the edgel contour of $s_{j,k}$.

Definition 1 *We say that the pair of sections $[(s_{i,k}, b_{i,k}, e_{i,k}), (s_{j,k}, b_{j,k}, e_{j,k})]$ are a neighbor pair if the distance between their closest points is less than d_n .*

(Note that this is an extension of the definition given in section 4.3.2).

Definition 2 *We say that two matches*

$$m_1 = [(s_{1,1}, b_{1,1}, e_{1,1}), (s_{1,2}, b_{1,2}, e_{1,2})] \text{ and}$$

$$m_2 = [(s_{2,1}, b_{2,1}, e_{2,1}), (s_{2,2}, b_{2,2}, e_{2,2})]$$

are neighbor matches if there exists a neighbor pair

$$[(s_{1,1}, b'_{1,1}, e'_{1,1}), (s_{2,1}, b'_{2,1}, e'_{2,1})] \text{ such that}$$

the sections $(s_{1,1}, b_{1,1}, e_{1,1})$ and $(s_{1,1}, b'_{1,1}, e'_{1,1})$ overlap or

the sections $(s_{2,1}, b_{2,1}, e_{2,1})$ and $(s_{2,1}, b'_{2,1}, e'_{2,1})$ overlap. In other word, m_1 and m_2

are neighbor matches if their sections are (even partially) neighbors.

Note that a match is a neighbor of itself.

Definition 3 *The approving matches of a match are neighbor matches that represent similar a 2-D translation. Note that a match is always an approving match of itself.*

Definition 4 *The non-approving matches of a match are neighbor matches that represent a non similar 2-D translation (this is actually the complement of the approving matches within the set of all the neighbor matches).*

Definition 5 *The approving length of a match is defined to be a function of the total length of its approving matches and their number. Thus we allow few strong approving matches or many short approving matches.*

Definition 6 *The non-approving length of a match is similarly defined to be a function of the total length of its non approving matches and their number. Thus we allow few strong non approving matches or many short non approving matches.*

Definition 7 *The score s of a match is a combination of its shape similarity score (previously defined), length difference between the two sections and the approving length of the match (which represents the neighborhood support). The lower the score the better.*

Our formula gives lower score to matches which sections vary significantly in size, are short or have a low neighborhood support:

$$s = s_m + c \cdot \frac{|l_1 - l_2| + 1}{a_m \cdot l_1^2}$$

where s_m is the similarity score, l_1 and l_2 are the lengths of the left and right sections respectively, and a_m is the approving length of the match. c is a constant.

Figure 4.9 illustrates how approving matches relate to each other. Figure 4.3 shows how neighbors are initially computed.

4.5.2 Computation of approving neighbor matches

We defined *approving neighbor matches* of a match m to be neighbor matches that represent a similar 2-D translation to m .

Figure 4.9: Neighborhood Support: Adjacent section matches are *approving matches* except for the highlighted match

Assume that every segment a_i (of super-segment s_i) in the image has two sets, one set contains all *neighbor pairs* $[(s_i, b_i, e_i), (s_j, b_j, e_j)]$, where the first section of the pair is a sub-section of the section covered by a_i . The second set is the set of all the *matches* of s_i $[(s_i, b_i, e_i), (s_j, b_j, e_j)]$, that overlap the section covered by a_i .

Thus, computing the *neighbor matches* of a match $m = [(s_{i,1}, b_{i,1}, e_{i,1}), (s_{j,2}, b_{j,2}, e_{j,2})]$ can be done by simply collecting all matches that either overlap m or belong to *neighbors* of segments of m .

The following algorithm computes the neighbor matches of m :

```

for every match  $m = [(s_{i,1}, b_{i,1}, e_{i,1}), (s_{j,2}, b_{j,2}, e_{j,2})]$  do
  for every segment  $a_{i,1}$  overlapped by  $m$  do
    Collect the section represented by the segment and all its neighbor pairs
    that overlap the match.
  od
  for every segment  $a_{j,2}$  overlapped by  $m$  do
    Collect the section represented by the segment and all its neighbor pairs
    that overlap the match.

```

od

Find all matches that overlap the sections found in the previous stage.

od

A neighbor match is an *approving neighbor match* if it has similar 2-D translation. We permit a larger error for very long matches or matches that represent a large translation. The reason is that long matches have a higher probability of being correct, and that matches that represent a large 2-D translation will represent very large 3-D motion in the scene. Obviously the images of such objects are expected to deform more than images of objects that hardly move. Thus a larger error should be permitted.

4.5.3 Remove overlapping matches

Two possible overlaps exist, either *complete* overlap, where one match (or rather, either of its sections) contains the other, or *partial* where the matches share a common section, but neither is contained by the other. For every pair of overlapping matches, compare their scores using definition 7 above. Choose the match that has the lower score and the remainder of the other match (if any). Try to join both matches into one match if possible, that is, when both have same super-segments and joining the sections yields a better match.

This step typically removes 95% of the matches. However, due to the large number of incorrect matches, we expect some good matches to be removed and some bad matches to be kept. Therefore further evaluation is needed. We evaluate the matches based on the approving and non-approving lengths only, since we believe that at this stage the greater majority of the matches are correct and therefore the approving and non-approving lengths actually reflect the correctness of the match. A correct match is expected to have a large approving length and a small non-approving length.

4.5.4 Matches Retrieval

In this step we retrieve possibly correct discarded matches by computing approving length for every match in the set of matches resulting from step 4.4 using only

matches that survived the overlap removal step. Only matches that have a non-zero approving length are kept for further evaluation. Obviously some matches will overlap and many duplicates are expected, strengthening the scores of correct matches. We then apply the previous step (compute approving length and removal of overlaps) to this set.

This step is necessary, as mentioned in the previous section, because of the large reduction in the number of matches. Because so many matches were spurious, some incorrect matches may have enough support to survive when a large percentage of the matches are incorrect, but are expected to be removed when only the surviving matches are considered for computing the approving length. Since the incorrect matches may have overlapped correct matches that were therefore removed, retrieving all approving matches from the original set should undo this effect. We compute approving length for the matches using only the “preferred” ones that survived the last step, because they are more credible.

4.5.5 Remove matches with weak neighborhood support

The previous step is expected to remove incorrect matches when correct matches overlapping them exist, by iteratively removing weak matches overlapping stronger ones. This technique is very powerful, but does not solve the problem of incorrect matches that are not overlapped by better matches, and therefore remain. However, using the same reasoning as before, we say that weak matches that have a very low approval length are probably incorrect.

A simple relaxation technique is used to remove weak matches. For every match compute the approving and non-approving lengths. An approving match is a neighbor match that has a similar translation.

We allow long matches and matches with a large translation a larger error. The reason for that is that very long matches are expected to be correct, and we therefore want to use them more liberally. A very long match can also be used to approve other matches with no other support. Matches that represent a large motion in 2-D represent a large 3-D motion as well. We expect a large 3-D motion to produce larger contour deformation in the image, and therefore allow a wider 2-D translation range for the object they represent.

Figure 4.10: Resolving Overlapping matches

A match will be discarded if the approving length is much smaller than the non-approving length (at least 5 times smaller, or more if the match is long).

We iterate until no more matches are discarded.

Figure 4.10 shows how the better matches are selected.

4.6 Multi Frame Matches

Matching only two images is not useful for motion detection and estimation. Two frames are not enough to estimate 3-D parameters, except for limited types of motion. In addition the use of several frames is useful when verifying a match and for motion segmentation. Tracking motion through time cannot be done with only two frames.

In this section we describe our algorithm for combining pairwise matches along a sequence into multi frame matches, that are section matches extending over more than two frames. Initially the multi frame matches are formed by selecting the common part of two pairwise (and later multi frame) matches. This results in large fragmentation, which we remedy by merging multi frame matches together wherever possible (as long as no super-segment contradiction exists). The merge operation is fairly difficult, since the merged multi frame matches may not start or end in the same frame, and thus matches need to be extended in the missing frames.

4.6.1 Creating Multi Frame Matches

Assume the image sequence contains n frames, F_1, \dots, F_n , and that pairwise matches are available for all successive pairs of frames.

Our method for combining pairwise section matches into multiple section matches is fairly simple.

Let m_1 and m_2 be two pairwise matches defined as follows:

$m_1(p_i, p_{i+1})$ be a match between sections p_i (in frame i) and section p_{i+1} (in frame $i + 1$).

$m_2(q_{i+1}, q_{i+2})$ be a match between sections q_{i+1} (in frame $i + 1$) and section q_{i+2} (in frame $i + 2$).

Note that p_{i+1}, q_{i+1} are sections in the same frame $i + 1$.

Then p_{i+1} and q_{i+1} either have no points in common, they fully overlap (one of them is a sub-section of the other) or they partly overlap.

If the sections overlap either fully or partly, then we can probably *combine* m_1 and m_2 into a 3-frame match m_3 .

This problem is very similar to the overlapping of matches discussed previously.

Figure 4.11: Multi-frame Matches

In this case our solution is as follows:

1. Compute the overlapping sub-section of p_{i+1} and q_{i+1} , say r_{i+1} .
2. Compute r_i , the sub-section of p_i (in frame i) that best matches r_{i+1} .
3. Compute r_{i+2} , the sub-section of q_{i+2} (in frame i) that best matches r_{i+1} .
4. Create a 3-frame match $[r_i, r_{i+1}, r_{i+2}]$.

This process is iteratively applied to obtain multi-frame matches of $k + 1$ consecutive frames $[r_i, r_{i+1}, \dots, r_k]$, where $k > 1$ and $1 \leq i < k \leq n$. The only restriction being that a multi-frame match should contain at least three frames, but can begin or end anywhere along the sequence. This allows us to handle disappearing or newly appearing points.

Figure 4.11 contains an example of a multi-frame match.

4.6.2 Merging Multi Frame Matches

The simple algorithm described in the previous section results in of fragmentation.

We try to merge adjacent multi frame matches if the following holds:

$$\text{Let } \overline{m_1} = [p_i, p_{i+1}, \dots, p_r] \text{ and } \overline{m_2} = [q_j, q_{j+1}, \dots, q_s]$$

be two multi frame matches, where p_m is a section in frame m of the form

$$(s_{p_m}, b_{p_m}, e_{p_m}) \text{ (} q_n \text{ is defined in a similar way).}$$

The multi-frame matches $\overline{m_1}$ and $\overline{m_2}$ can be *merged* if the following conditions hold:

1. Let $l = \max(i, j)$ and $h = \min(r, s)$, then $l \leq h$ (there is at least one common frame).
2. There is no super-segment contradiction in the common frames, that is,
 $s_{p_m} = s_{q_m}$ for every $l \leq m \leq h$.

We denote this common super-segment as simply s_m .

3. For every common frame $l \leq m < h$, the sections
 $r_m = (s_m, \min(b_{p_m}, b_{q_m}), \max(e_{p_m}, e_{q_m}))$ and
 $r_{m+1} = (s_{m+1}, \min(b_{p_{m+1}}, b_{q_{m+1}}), \max(e_{p_{m+1}}, e_{q_{m+1}}))$ can form a valid pairwise match. We permit some error correction to be made here, however, since the original matches may not have been correctly or accurately localized. If one of the matches was incorrect, the correction may result in different sections than expected).
4. Assume (without loss of generality) that $i < l$ (thus $\overline{m_1}$ starts at a lower frame). Then there should be a section r_{l-1} containing p_{l-1} that can match r_l (the lowest merged section), a section r_{l-2} that can match r_{l-1} and so fourth.
5. Assume (without loss of generality) that $j > h$ (thus $\overline{m_2}$ ends at a higher frame). Then there should be a section r_{h+1} containing q_{h+1} that can match r_h (the highest merged section), a section r_{h+2} that can match r_{h+1} and so fourth.
6. The merged multi frame match should not “telescope” in either direction (that is, the lengths of the sections along the merged multi frame match should not decrease significantly over time).

Figure 4.12: Merging adjacent Multi-frame Matches: (Q_1, Q_2, Q_3) and (R_2, R_3, R_4) are merged into (S_1, S_2, S_3, S_4)

If the multi frame matches can be merged, they are replaced by the merged multi match and the merging process continues.

An example is given in figure 4.12.

The fragmented multi-frame matches $Q = [Q_1, Q_2, Q_3]$ and $R = [R_2, R_3, R_4]$ can be merged into a the multi-frame match $S = [S_1, S_2, S_3, S_4]$.

Note that the merge algorithm not only reduces fragmentation, it also adds new information by extending matched sections that are covered by only one multi frame match. Merging multi frame matches this way allows us also to remove some incorrect multi frame matches, because the merged multi frame matches contain

longer sections for each frame, and are thus more likely to be positioned more accurately.

Identifying pairs of multi frames matches that can be merged is a non trivial task. A trivial algorithm is simply to compare every match to every other match, with number of comparisons equal to the number of multi frame matches squared. Every such cycle needs to be repeated as long as even one pair was merged, because the result may allow us to merge other multi frame matches with it.

One should notice, however, that two multi frame matches can only be merged if they have to have common frames, and same super segments for all the common frames. Since the number of matches for each super segment is quite small, the number of multi frame matches in which this particular super segment can participate is also very small. Thus, for each multi match, there is only a small number of other multi frame matches that can be merged with it, and all of them should have common frames and super segments with it.

To identify pairs of multi frame matches that can be merged, we create a *bucket array* B of multi frame matches. $B[i, j]$ contains all multi frame matches that have a section of the j^{th} super-segment of frame i . By the definition above it is obvious that if two multi frame matches can be merged, both have to belong to the same bucket. Since the number of multi frame matches per bucket is small (as we explained before), we can simply compare all its multi frame matches to each other. Merging a pair of multi frame matches requires updating all the buckets they can belong to. The original two multi frame matches are deleted and replaced by the new multi frame match.

Clearly, after processing all the buckets, only buckets that had a merged multi frame match added need to be checked in subsequent iterations. The process is repeated for these buckets, until there are no more changes.

4.7 Hierarchical Matching

Applying Adaptive Smoothing with different scales to the image, results in a hierarchical set of edges. The finer the scale, the more edges are obtained. However, the contours are not hierarchical, as linking is done independently for each scale. Matching features at a coarser scale is expected to be easier and faster, since they are sparser. Using the results as *predictions* for matching in finer level is very desirable, but requires identifying matching sections for the same image, processed with different scales.

Our idea is to *match* the same image processed with different scales, and then combine the results to obtain predictions.

Figure 4.13 illustrates the idea. It describes a hierarchical match between two frames, F_1 and F_2 . First the match is done at the coarser scale S_h . Then the results are propagated as predicted matches to the finer scale S_l and the predictions are used for the matching in the finer scale.

Assume that the images are $image_1$ and $image_2$, the coarser scale is h and the finer one is l .

Then we have 4 sets of super-segments: $S_{1,h}$, $S_{2,h}$, $S_{1,l}$ and $S_{2,l}$, where $S_{i,s}$ is the set of super-segments detected in $image_i$ after smoothing it at scale s .

Note that the edgels of $S_{i,h}$ are actually a subset of the edgels of $S_{i,l}$, but the super-segments themselves may not have this property, since edgels may be linked differently for different scales (at junctions, for example).

The hierarchical matching algorithm is as follows:

1. Match $S_{1,h}$ and $S_{2,h}$ to obtain a match $M(1, 2, h) : S_{1,h} \Rightarrow S_{2,h}$ for the coarser scale.
2. Match $S_{1,h}$ and $S_{1,l}$ to obtain a match $M(1, h, l) : S_{1,l} \Rightarrow S_{1,h}$ between the two scales for the first image.
3. Match $S_{2,h}$ and $S_{2,l}$ to obtain a match $M(2, h, l) = S_{2,h} \Rightarrow S_{2,l}$ between the two scales for the second image.
4. Combine the results to obtain *predicted matches* for the images smoothed by l , by applying the multiple-matches algorithm.

Figure 4.13: Hierarchical Matching: match features in F_1 and F_2 after smoothing with coarse scale S_h , propagate to next level (scale S_l), then match frames smoothed with fine scale S_l using predictions.

Result: a match $S_{1,l} \Rightarrow S_{1,h} \Rightarrow S_{2,h} \Rightarrow S_{2,l}$.

Remove the two middle matches to get the predicted matches:

$P(1, 2, l) : S_{1,l} \Rightarrow S_{2,l}$.

5. Use $P(1, 2, l)$ to match $S_{1,l}$ and $S_{2,l}$ to obtain $M(1, 2, l) : S_{1,l} \Rightarrow S_{2,l}$ for the finer scale.

Of course, steps 2 to 5 can be repeated for smaller scales.

In the first step we match a relatively sparse edgel map, since the image is smoothed with a very coarse scale filter. We therefore do not expect many competing matches, and the matching is relatively easy. The second and third steps are almost trivial, since the edgels shift by at most a pixel. We use our matching algorithm with a disparity of 2. The fourth step is a mere application of our multiple matches algorithm, described in [27] and also later in section 4.6. Although this method was originally designed for a motion sequence, it is easily applicable here. In the fifth step we use the predicted matches as guide for the matching algorithm in several steps. We describe this algorithm in sections 4.2 through 4.5.

4.7.1 Example

Figures 4.14 and 4.15 show an example for the application of the hierarchical match. Subfigures 4.14(a) and (b) contain the two consecutive frames from a sequence of moving toy jeep and train. subfigures 4.14(c) and (d) contain the contours detected after smoothing with a mask of 16, subfigures 4.14(e) and (f) contain the contours detected after smoothing with a mask of 8 and subfigures 4.14(g) and (h) contain the contours detected after smoothing with a mask of 4.

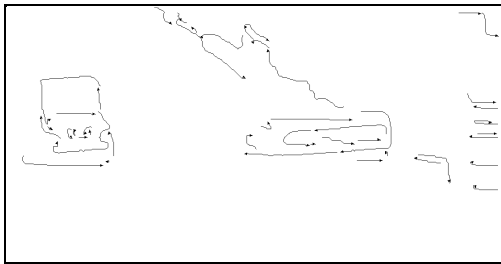
Subfigure 4.15(a) contains the matches computed for a scale parameter of 16, subfigure 4.15(b) contains the predicted matches computed from mask 16 for a scale parameter of mask 8, subfigure 4.15(c) contains the matches computed for mask 8 using the predicted matches of previous step, subfigure 4.15(d) contains the predicted matches computed from a scale parameter of 8 to a scale parameter of 4 and subfigure 4.15(e) contains the matches computed for a scale parameter of 4 using the predicted matches of the previous step. These are the final matches for this pair of images.



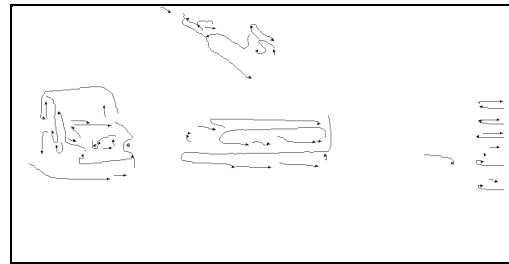
(a) Jeep sequence - Frame 4



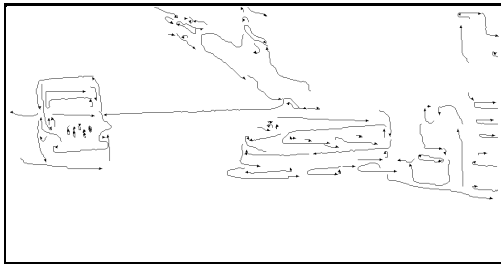
(b) Jeep sequence - Frame 5



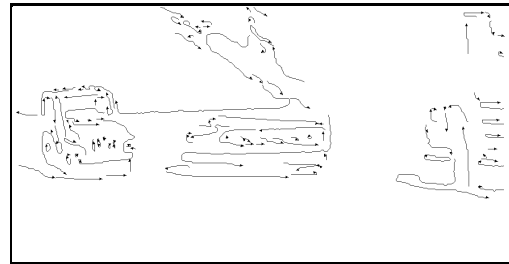
(c) Super-segments of (a) - scale 16



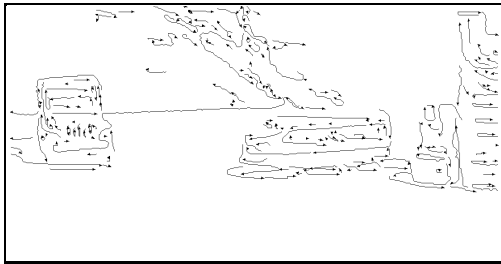
(d) Super-segments of (b) - scale 16



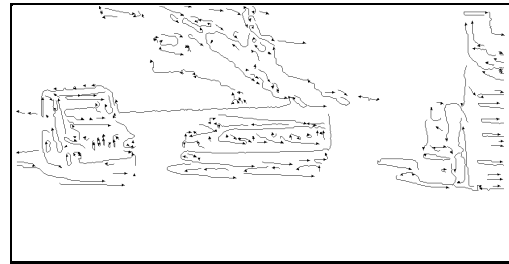
(e) Super-segments of (a) - scale 8



(f) Super-segments of (b) - scale 8



(g) Super-segments of (a) - scale 4



(h) Super-segments of (b) - scale 4

Figure 4.14: Jeep and train image - features found with different scale params

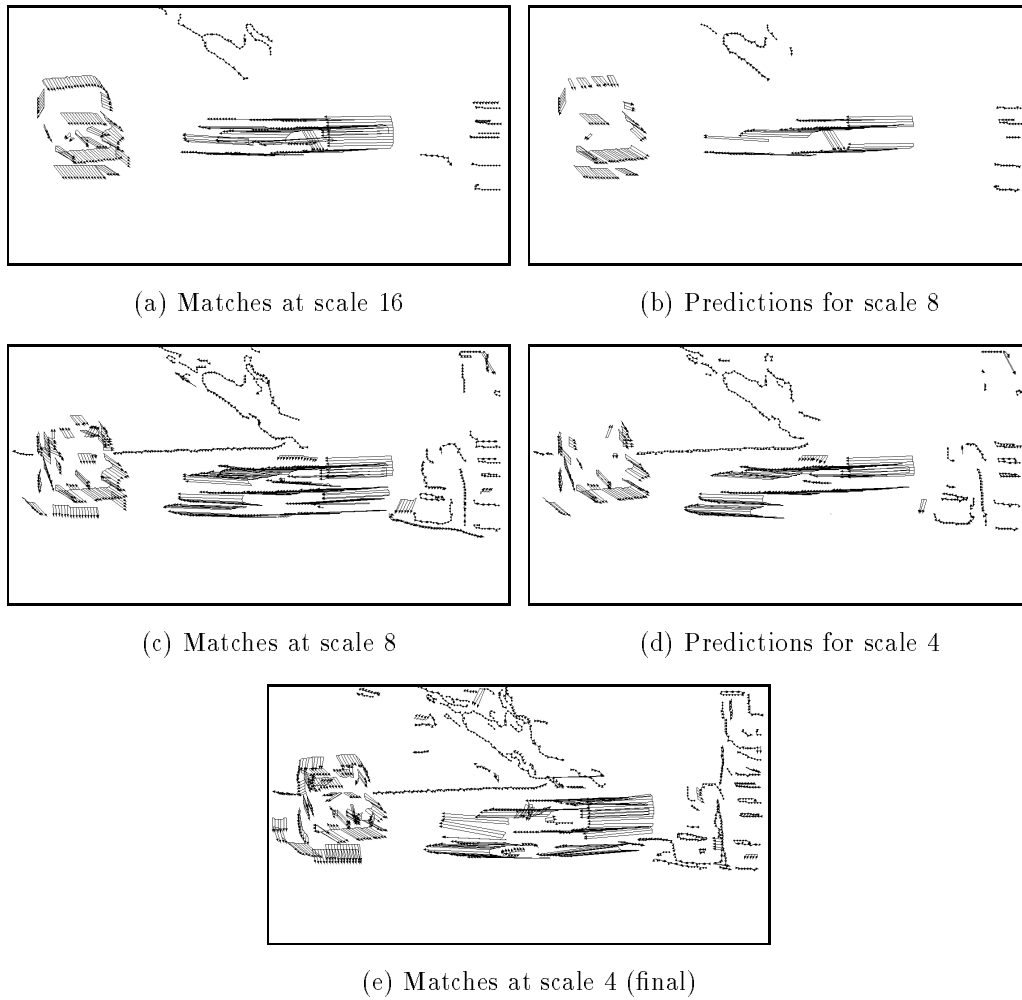


Figure 4.15: Jeep and train sequence - hierarchical matches

4.8 Critical Evaluation

We have shown an algorithm to compute correspondences between 2 frames with very few constraints. We suggested the use of edgel contours and sections of contours. Correspondence was based on shape similarity between matching sections and on translation similarity between matches. Performance was demonstrated by applying our method to a number of real images. Using a multi-scale approach enables us to use better primitives.

The advantages of the matching method were discussed in the previous sections:

We use continuity and contour sections of arbitrary shape and size in matching, that are easily expandable along the contour, thus permitting us to adapt the features throughout the algorithm in order to obtain the best fit. The *length* and *similarity* of matching sections together with *neighborhood approval length* were proved extremely reliable in predicting the correctness of a match.

Combining pairwise matches into multi-frame matches is a critical step for further evaluation of the motion information.

The advantages of adaptive filters are in their better localization of edgels, edgels that represent real events, and their hierarchical nature. The advantages of using multi-scale hierarchical matches are: reduced complexity and the ability to match well even with smaller masks.

Applying this method to real images produces very good results. We have been able to match many different scenes, indoor as well as outdoor, with both objects and camera moving, using large disparities. Our algorithm handles well even complex 3-D motion.

Chapter 5

The Segmentation Algorithm

Segmenting an image into regions that are likely to correspond to different objects is an important step towards inferring the physical properties of objects in the scene.

Most Motion Segmentation algorithms rely on a dense flow field and are therefore inappropriate for us, due to the sparseness of the data. Our solution is similar to [64], in that we group together adjacent features based on their perceived motion, but our algorithm uses the multi frame matches for the segmentation, instead of the pairwise matches only. We only rely on the 2-D translation computed by our matching algorithm, despite the fact that it represents only a rather coarse estimate of the actual motion. It is clear that given the 3-D motion of every point in the image, segmenting it should be easy. Unfortunately, inferring the 3-D motion from the 2-D computation is a very difficult task. There are many algorithms for motion estimation, but most of them require matching points and a very high level of accuracy in their location. As the primitive for the matching in our algorithm is a *section* of an edgel contour, further processing would have to be done in order to obtain point matches from the section matches, and the continuity information would be lost. Note also that edge detection shifts the edges by as much as two pixels, whereas most 3-D motion estimation algorithms expect sub-pixel accuracy. Thus finding the 3-D motion parameters using our matches is not easy. Our goal is to find an algorithm that utilizes the continuity and neighborhood information inherent in the section matches. Obviously, accurate motion parameters are not necessary for segmentation.

In this chapter we describe our segmentation algorithm. The algorithm uses both the pairwise section matches and the multi-frame section matches as primitives. Pairwise matches are grouped into equivalence sets of matches based on common motion and neighborhood. Multi frame matches are initially grouped together if no contradictions between their pairwise equivalence sets exists. Then further processing is done to refine the solution: Some sets of multi-frame matches should have been joined together; others should be split. We propose solutions to these problems.

5.1 Primitives

We repeat some definitions for convenience.

Given a set of frames F_0, \dots, F_n , let M_i be the set of section matches between F_i and F_{i+1} . Let \overline{M} be the set of multi-frame matches for the sequence. In a multi-frame match, the individual sections for every frame that participates are subsections of pairwise matches for these frames (but this is not always the case, because merging multi-frame matches together may result in sections that are not contained in any pairwise match).

Formally, a *section* of a super segment s_i in frame F_i that begins at position b_{s_i} and ends at position e_{s_i} in its edge list, is denoted (s_i, b_{s_i}, e_{s_i}) .

Two sections (in the same image) are *neighbors* if the shortest distance between them is less than some fixed neighborhood distance d_n .

A *pairwise section match* is a pair of two sections $[(s_i, b_{s_i}, e_{s_i}), (s_{i+1}, b_{s_{i+1}}, e_{s_{i+1}})]$, and the 2-D translation (r_{s_i}, c_{s_i}) represented by the match.

A *multi-frame match* is a tuple of pairwise matching sections. Let \overline{m}_j be a multi-frame match, f the first frame of the multi-frame match and $u_i = (s_i, b_{s_i}, e_{s_i})$ the section of \overline{m}_j corresponding to frame F_i . Then

$$\overline{m}_j = [u_f \ u_{f+1} \ \dots \ u_k] = [(s_f, b_{s_f}, e_{s_f}), (s_{f+1}, b_{s_{f+1}}, e_{s_{f+1}}), \dots, (s_k, b_{s_k}, e_{s_k})]$$

and their respective 2-D translations

$$[(r_{u_f}, c_{u_f})(r_{u_{f+1}}, c_{u_{f+1}}), \dots, (r_{u_k}, c_{u_k})].$$

Usually (but not always), for any successive pair of sections u_i and u_{i+1} in a multi-frame match, there exists a *pairwise* match that includes it, since the multi-frame matches were obtained by combining the pairwise matches along the sequence.

The *size* of a multi-frame match is defined as the sum of lengths of all sections belonging to the match, multiplied by the square root of the number of frames of the match. (In other words, a combination of lengths of sections and length of sequence).

The *total size* of a set of multi-frame matches is the sum of the sizes of the multi-frame matches (as previously defined), multiplied by the square root of their number. (In other words, a combination of matches size and number of matches).

5.2 Pairwise Segmentation

5.2.1 General Description and Goals

The Pairwise Segmentation algorithm tries to segment the image using two successive frames only. Our idea is to eventually combine the pairwise results into a segmentation for the whole image. Using the pairwise matches as the first step has the following benefits:

- More neighborhood context. Multi frame matches are a subset of the set of all pairwise matches (as explained before). Thus they convey less neighborhood information. By using the pairwise matches we can utilize as much context information as is available.
- Independent segmentation for every pair of frames means that a mistake in one frame is easy to correct when processing the multi-frame match, since two sets of multi-frame matches that actually belong to the same object would likely share some pairwise equivalence sets.

5.2.2 Definition of an Equivalence Relation

Given the frames F_i and F_{i+1} and the set of matches M_i , define a relation Q over the set of matches as follows:

```

procedure  $\overline{Q}$ 
 $j \leftarrow 0$ 
Repeat
  Select a match  $m \in M_i$ ;
   $M_i \leftarrow M_i - \{m\}; S_j \leftarrow \{m\}$ ;
  Repeat
    For every  $m_1 \in M_i$ 
      If there exists  $m_2 \in S_j$  such that  $Q(m_1, m_2)$  is true
        then  $M_i \leftarrow M_i - \{m_1\}; S_j \leftarrow S_j \cup \{m_1\}$ ;
  until no changes
   $j \leftarrow j + 1$ ;
until  $M_i = \emptyset$ 

```

Figure 5.1: Pairwise segmentation algorithm

For any pair of matches $m_1, m_2 \in M_i$, where m_j represents the match between sections $u_{i,j}$ and $u_{i+1,j}$ for $j = 1, 2$;

$Q(m_1, m_2)$ is true *iff* either $u_{i,1}$ is a neighbor of $u_{i,2}$ or $u_{i+1,1}$ is a neighbor of $u_{i+1,2}$, and the two matches have similar 2-D translations. In other words, m_1 and m_2 are *approving matches*.

Define a relation \overline{Q} as follows:

$\overline{Q}(m_1, m_2)$ is true *iff* there exists a path in Q between m_1 and m_2 , that is, there exist matches m_3, m_4, \dots, m_k such that $Q(m_1, m_3), Q(m_3, m_4), \dots, Q(m_k, m_2)$ are all true.

It is trivial to show that Q is not an equivalence relation, since transitivity is not preserved, but that \overline{Q} , on the other hand, is an equivalence relation. Therefore \overline{Q} implies a *unique partition* of the matches into equivalence sets.

Our algorithm consists of computing these equivalence sets and partitioning the matches accordingly. It is formally described in figure 5.2.2.

The sets S_0, \dots, S_j are the equivalence sets of \overline{Q} .

An example is given in figure 5.2. Sets A, \dots, E represent the segmentation of the frame based of a pairwise segmentation. Adjacent pairwise matches are repeatedly joined together by their 2-D motion, implying a partition of the image plane. Adjacent sections with different 2-D motions (A and D for example) are not

joined. Far apart sections with similar motion (B and D for example) are also not joined.

5.2.3 Analysis

Ideally, these equivalence sets describe regions of matches where each region represents motion different from that of adjacent regions. Thus moving objects should be separated. For a static image we expect to get depth segmentation.

We define an *object* to be any continuous part of the scene that shares a common motion. Thus, two adjacent *physical* objects having the same motion will be considered one object. A physical object with several parts moving differently (such as a human body), is divided into different objects based on the motion of its different parts.

The following problems need to be addressed by our algorithm:

5.2.4 Over-segmentation

Over-segmentation happens when an object is separated into several equivalence sets.

One possible reason is if no neighborhood information exists to link the different sets. This is possible for a large object that is either fairly featureless or is partly occluded (such as the background).

Another possible reason is if the matching algorithm computed different 2-D motions for different parts of the object. This is an obvious outcome if the object is moving along the z direction or is rotating. It could also be the result of an incorrect match that represents an incorrect motion.

An example is given in figure 5.3.

5.2.5 Under-segmentation

An equivalence set contains (portions of) more than one object.

This can be the result of matching errors or when two (or more) separate objects have similar 2-D motion in some frames and are also close in proximity, even though they have distinct 3-D motion. This is very likely to happen when the motion of an

Figure 5.2: Pairwise Segmentation: image segmented into regions based on similar 2-D motion between neighbor matches.

(b) Under-segmentation: two objects are merged into one region because of neighborhood support

Figure 5.3: Segmentation problems

object differs little from that of the background or if the object is very far from the observer.

An example is given in figure 5.3.

5.2.6 Opacity Constraint

Two or more equivalence sets may cover the same physical area in the image, as we make no use as yet of our assumptions of solidity and opacity.

We prefer not to try and tackle these difficult issues in the pairwise segmentation level, as motion information that is based on two frames only is often not sufficient for resolving object boundaries.

5.3 Multi Frame Segmentation

Our multi-frame segmentation uses the pairwise segmentation as a guide, by separating the multi-frame matches based on the results of the pairwise segmentation. This initial segmentation may have the same problems described by the previous algorithm. However, an incorrect grouping is easier to detect when there are more than two frames. That is because two separate sets that actually belong to the same object are very likely to have at least one pairwise equivalence in common. Sets that contain matches for two objects will have low connectivity between matches of the two objects and therefore can also be separated.

5.3.1 Initial Grouping of Multi Frame Equivalence Sets

Let $S_{i,j}$ be the equivalence set j for frame F_i . That is, $S_{i,j}$ contains pairwise matches between frames F_i and F_{i+1} , grouped together by the relation \overline{Q} .

For every multi-frame match

$$m = [(s_f, b_{s_f}, e_{s_f}), (s_{f+1}, b_{s_{f+1}}, e_{s_{f+1}}), \dots, (s_k, b_{s_k}, e_{s_k})]$$

create a tuple $t = (S_{f,j_f}, S_{f+1,j_{f+1}}, \dots, S_{k,j_k})$ where S_{i,j_i} is the index of the pairwise equivalence set that includes the pairwise match that most overlaps the sections (s_i, b_{s_i}, e_{s_i}) and $(s_{i+1}, b_{s_{i+1}}, e_{s_{i+1}})$.

Note that S_{i,j_i} may be undefined, if the match was the result of a merge between several multi-frame matches. We consider it as a “don’t care” set.

Define a relation $Q_{\overline{M}}$ over the multi-frame matches as follows:

For every pair of multi-frame matches \overline{m}_1 and \overline{m}_2 ; $Q_{\overline{M}}(\overline{m}_1, \overline{m}_2)$ is true *iff*

- \overline{m}_1 and \overline{m}_2 have at least two common frames, that is $\max(f_1, f_2) < \min(k_1, k_2)$, and
- Both multi-frame matches have the *same* equivalence sets $P_{i,j}$ for the common frames.

Note that $Q_{\overline{M}}$ is not an equivalence relation, because transitivity is not preserved.

Our partition of the multi-frame matches is based on ensuring that every multi-frame match \overline{m}_1 in a set will have at least one other multi-frame match \overline{m}_2 in this set such that $Q_{\overline{M}}(\overline{m}_1, \overline{m}_2)$ is true, and that no contradictions due to the non transitive nature of $Q_{\overline{M}}$ occur. Note that this partition may not be unique, as a different order of matches may yield somewhat different sets.

Our algorithm to partition the multi-frame matches is described in figure 5.3.1

Let \overline{M} be the set of multi-frame matches.

An example for the initial partitioning is shown in figure 5.5.

The resulting partition of the multi-frame sets is a collection of sets of multi-frame matches with non-contradicting pairwise equivalence sets. Unfortunately, the two problems mentioned in the previous sections still exist, and in the next section we show our solution to them.

5.3.2 Splitting multi-frame matches sets

In this subsection we describe our method for detecting sets with several objects and how to separate them.

The problem of several different objects (or portions of objects) joined into one set is more difficult to resolve. We base our solution on the fact that if two objects were (partially) joined together, thus have similar 2-D motion and close proximity in some part of the sequence, but have different 3-D motions, then there have to be another part of the sequence where they cannot be joined, unless their 3-D motion

```

procedure  $Q_{\overline{M}}$ 
While  $\overline{M} \neq \emptyset$  do
  Select a multi-frame match  $\overline{m}$  and assign it to a new multi-frame set  $\overline{S}$ .
  Remove  $\overline{m}$  from  $\overline{M}$ .
  Repeat
    For every  $\overline{m} \in \overline{M}$  do
      If there exists  $\overline{m}_1 \in \overline{S}$  such that  $Q_{\overline{M}}(\overline{m}, \overline{m}_1)$  is true and
        For every  $\overline{m}_2 \in \overline{S}$  either
           $Q_{\overline{M}}(\overline{m}, \overline{m}_2)$  is true or
           $\overline{m}$  and  $\overline{m}_2$  share no common frames;
        then add  $\overline{m}$  to  $\overline{S}$  and remove it from  $\overline{M}$ .
    od
  until no new matches are added to  $\overline{S}$ 
od

```

Figure 5.4: Multi-frame segmentation algorithm

differs so little as to render them indistinguishable. Therefore, these erroneous multi-frame sets would typically have many short multi-frame matches for the first or last frames, a few long or short multi-frame matches for the middle frames, but fairly small connectivity between the short multi-frame matches of the first frames and those of the last frames. (Note that “short” and “long” in this context refer to the number of frames participating in the match.) Therefore separating the set into several subsets based on begin and end frame of each match, has a high likelihood of having most of the matches of one object in one pool and most of the matches of the other object in another pool.

Figure 5.6 illustrates the problem. Multi-frame section matches for the rectangle are divided between two segmented regions, because short multi-frame matches, representing motion through only 3 or 4 frames, were joined with multi-frame matches of the ellipse.

An example for this problem occurs in the sequence of the advancing car. In this sequence the observer is moving towards the car, and therefore the background is also deemed “moving.” In the first several frames, the motion of the car is practically indistinguishable from that of the background. The algorithm therefore

Figure 5.5: Initial segmentation of the multi frame matches