A VOTING-BASED COMPUTATIONAL FRAMEWORK FOR VISUAL MOTION ANALYSIS AND INTERPRETATION

by

Mircea Nicolescu

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)

August 2003

Dedication

To Monica, with all my love.

Acknowledgments

While working towards my Ph.D. degree I have been very fortunate to be surrounded by a group of wonderful people, whose contribution to this work I would like to acknowledge.

I would first like to express my deepest gratitude to my doctoral advisor, Gérard Medioni, for his extraordinary support and guidance during my graduate years at USC. From his course that introduced me to the fascinating world of computer vision, to the insightful suggestions that have always pointed me to the right direction in my research, to the final advice on refining my dissertation, he has continuously been a source of inspiration, energy and invaluable knowledge. For these, and for many other reasons that helped make this dissertation and the completion of my Ph.D. studies a reality, I will be always indebted to him.

Heartfelt thanks to the other members of my Qualifying and Dissertation Committee: Ram Nevatia, Ulrich Neumann, Norberto Grzywacz, Jeff Rickel, João Hespanha and Laurent Itti, for their thorough and valuable comments that helped shape the final version of my dissertation. In particular, I would like to thank Ram Nevatia for his continuous support, encouragement and advice since my first days as a student at USC. Many thanks to Keith Price, Isaac Cohen, and Andres Huertas for always providing valuable expertise and help with countless issues during these years, ranging from fruitful research discussions to making a stubborn printer actually print.

I would like to thank my labmates in the Computer Vision group for the friendly and stimulating atmosphere that made the past years in the lab a wonderful experience. Special thanks to Elaine Kang, Alexandre François and Philippos Mordohai, for always being there to help, for the great conversations, and for the numerous weekends and nights spent together in the lab before deadlines. I also owe many thanks to Mi-Suen Lee for her invaluable support and advice – before, during and after my internships at Philips Research.

I have saved the last of my acknowledgments for the people whom I can never hope to adequately thank – my family. My love and gratitude to my parents, Veronica and Gabriel, for their support and for the many sacrifices they made so that I can achieve the best in my life. My love and thanks to my grandparents and parents-in-law, Ica and Ticu, Terezia and Marin, for their encouragements and the pride they took in my achievements. And finally, my deepest love to Monica, my wife, for sharing all our joys and efforts throughout these years – our last years as students, and the first in our lives together. I dedicate this dissertation to her, with all my love.

Contents

List of Tables

List of Figures

Abstract

Image motion is a rich source of information for the visual perception system, providing a multitude of cues to identify distinct objects in the scene and infer their 3-D structure and motion. Most approaches rely on parametric models which restrict the types of motion that can be analyzed, and involve iterative methods which depend heavily on initial conditions and are subject to instability. Further difficulties are encountered in image regions where motion is not smooth – typically around motion boundaries.

This dissertation addresses the problem of visual motion analysis and interpretation, by formulating it as an inference of motion layers from a noisy and possibly sparse point set in a 4-D space. The core of the method is based on a *layered 4-D representation* of data and a *voting scheme* for affinity propagation. Within the 4-D space of image positions and velocities, moving regions are conceptually represented as *smooth surface layers*, and are extracted through a voting process that enforces the motion smoothness constraint. By using an additional 2-D voting step that incorporates intensity information (edges) from the original images, accurate boundaries and regions are inferred.

The inherent problem caused by the ambiguity of 2-D to 3-D interpretation is usually handled by adding additional constraints, such as rigidity. However, providing a successful approach that enforces a global constraint has been problematic in the combined presence of noise, multiple independent motions, or non-rigid motion. By decoupling the processes of matching, outlier rejection, segmentation and interpretation, we extract accurate motion layers based on the smoothness of image motion, then locally enforce rigidity for each layer, in order to infer its 3-D structure and motion.

The proposed framework consistently handles both smooth moving regions and motion discontinuities, without using any prior knowledge of the motion model. The method is also computationally robust, being non-iterative, and does not depend on critical thresholds, the only free parameter being the scale of analysis.

The contributions of this work are demonstrated by analyzing a wide variety of difficult cases – opaque and transparent motion, rigid and non-rigid motion, curves and surfaces in motion, from sparse and dense input configurations.

Chapter 1

Introduction

1.1 The Problem

Vision is without doubt our most powerful sense. It allows us to acquire a remarkable amount of information about our surroundings, and to interact intelligently with the environment. The key for such an accomplishment of translating information from the sensors into meaningful knowledge lies in between these two processes. We can do it by continuously building a virtual model of the world, and updating it as the world changes. This is true for any type of perceptual process, but in the vision case the model is by far the most complex, and probably also the most useful for our interaction with the world.

A successful computer vision system must be able to generate such a world model from the same information used by humans in their vision perception process. The main difficulty is that such a problem is underconstrained. The model of our environment should be embedded in a 3-D+t space (three geometric dimensions and a temporal one) augmented with semantic information (such as the separation into distinct objects), while the visual information that we perceive is expressed in a 2-D+t space. The problem is clearly tractable, since humans do it all the time, with remarkable speed and reliability. Yet it is difficult because the human vision process is not a conscious one, and its complex underlying mechanisms are still not well understood.

From a computational point of view, we could consider the process of building a world model from visual information as a refinement of the available data along several levels. We start at the pixel level and we continue to a token structure level, then to an object level, and finally to a scene level.

The first step, from pixels to tokens, is to identify the basic elements that can be used in the next processing stages. They can be simple points, edges, line segments, corners, small blobs with similar intensity etc.

Next, these tokens must be grouped into distinct entities at the object level. These entities should be consistent with what humans perceive as distinct regions in the given image or group of images.

Finally, the whole description obtained so far must be augmented with semantic information concerning the objects and the relationships between each other and with the observer, allowing for understanding at the level of the entire scene.

1.1.1 Motion Analysis and Interpretation

This dissertation addresses a difficult and fundamental problem in computer vision, the analysis and interpretation of visual motion. An example that illustrates the problem is given in Figure 1.1. Given two image frames that contain general motion, the goal is twofold:

- to analyze the image changes in order to establish correspondences between image tokens across frames, as a dense velocity field, and to group tokens into motion regions separated by motion boundaries
- to interpret these changes in order to recover the scene 3-D structure and 3-D motion.

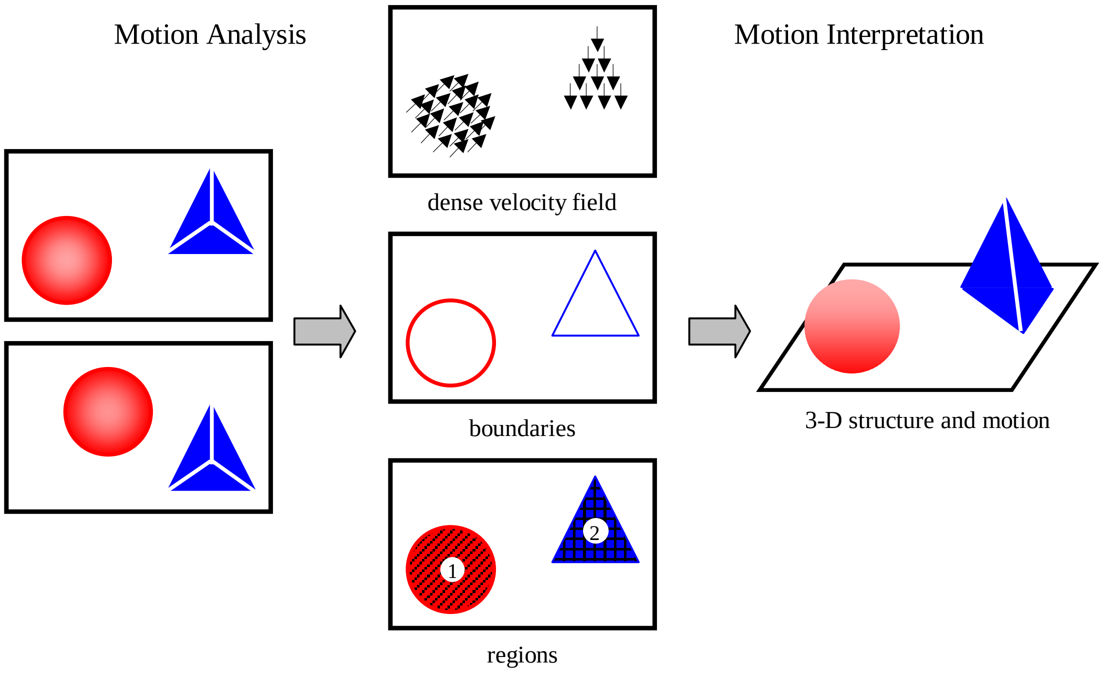


Figure 1.1. Motion analysis and interpretation

As demonstrated by the human visual perception, successful analysis and interpretation are possible from a small number of frames – therefore, this work is focused on the minimal case of inference from two images. Once this basic case is well understood, the processing can be extended to incorporate information from multiple frames, possibly using an incremental inference, where the structure is refined as new frames are added.

Our formulation, that divides the visual process into motion analysis and interpretation, is inspired from various perceptual studies [64][25], which show that establishing correspondences is a low-level process, that takes place prior to interpretation, where matches are established between elementary tokens, based on built-in affinity measures. The tokens involved in matching are not complex structures, but rather primitive image elements, such as points, fragments of edges, or blobs. In this study we only focus on analysis from point tokens.

Based on the changes at image level, the *motion analysis* process is responsible for determining three types of information: pixel velocities, motion boundaries and regions.

The *image velocity* of a pixel is a vector that encodes the motion of the pixel in the image, from one frame to another. Recovering this information is equivalent to establishing a match between a point in the first frame and its corresponding point in the other frame. A *velocity field* is a function defined over a subset of pixels in the image, and whose values are the associated velocity vectors. If this function is defined over the entire image, then the velocity field is called *dense*, otherwise it is a *sparse* velocity field.

In the formulation of the motion analysis problem given above, the ultimate goal with respect to the pixel velocities would be to recover them at each location, thus to determine the dense velocity field. Additional issues that need to be studied in this context are the possible non-existence of correspondences, as some pixels may appear in one image only due to occlusion, and the non-uniqueness in establishing matches, as multiple motions may overlap in the case of transparent motion.

The *motion boundaries* could be defined as the set of curves consisting of pixels where image motion changes abruptly. All pixels that exhibit a homogeneous (i.e. smooth) image motion could be grouped into a *motion region*. Consequently, the motion boundaries represent the sets of curves that separate motion regions in the image.

At a first glance, the three entities described above appear to be placed at different conceptual levels. Indeed, pixel velocity is an easily quantifiable pixel property, and its definition is a clear and formal one. On the other hand, boundaries and regions cannot be formally defined in a mathematical sense, because they are actually more related to the process of visual perception – that is, these entities belong to a perceptual or object level, rather than to the token level. It is not an easy task to formally specify what is meant by “abrupt change in motion” or “smooth motion”. Therefore, the above definitions for motion boundaries and regions should not be considered as based on inherent low-level properties, but on how we perceive them.

At the next conceptual level, the *motion interpretation* process is responsible for building a world model in terms of the 3-D scene structure and 3-D motion of the viewed objects. Such a process, that is performed subsequently to motion analysis and is based on the correspondences and regions recovered from changes at image level, corresponds to a traditional structure from motion problem. In this case, the inference is made by interpreting the changing projections of unrecognized objects in motion. An alternative process, which has been observed in human vision, corresponds to the case of inferring motion from structure, where previously recognized 3-D structure is used in order to derive a motion interpretation. This study is only concerned with the first case – inference of structure from motion – where no past familiarity, or “instant” object recognition is necessary as a prerequisite.

1.1.2 Monocular vs. Motion Cues

A very important issue in the analysis and interpretation of visual motion is the source of information used in the process. In this context, it is necessary to study what types of information are available, how they can be used separately, and how they can be combined into an integrated computational approach. The general process of structure inference from one or more images can be performed based on two sources of information:

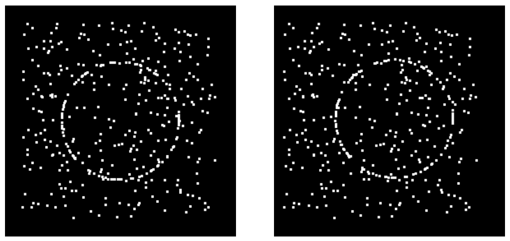
- monocular information
- motion information

Monocular cues are the ones that can be obtained from a single image, such as intensity, color or texture, and they are exclusively employed in several computer vision areas. For example, image segmentation can be attempted based entirely on the intensity or color of the pixels in the image.

Motion cues are used when information is to be extracted from time-varying images. The usefulness of the analysis from motion cues stems from the fact that regions (pixels with similar motion) usually correspond to distinct objects in the scene. Following the same line, monocular image segmentation techniques assume that pixels with similar intensity or color represent distinct objects in the world.

When these assumptions hold, it is worth mentioning that boundaries and regions can be recovered from monocular cues only; that is, from only one image. The human vision system is able to use both monocular and motion cues when available, in addition to patterns of objects or motions that are directly recognized.

Interestingly, in some situations, boundaries, regions and pixel velocities can be also determined from *motion cues only*, when no monocular info is available. Two relevant examples are given in Figure 1.2. If the two frames in each pair are presented in a properly timed succession, a certain motion of image regions is perceived from one frame to the other. However, while in one case the regions can be detected even without motion, only from


(a) Translating circle

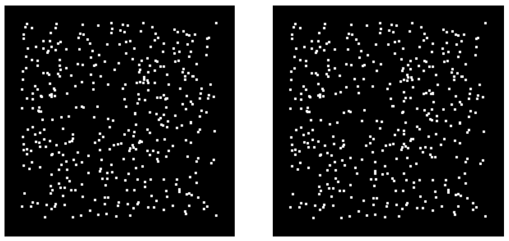

(b) Translating disk

Figure 1.2. Perception from monocular vs. motion cues

monocular cues (in this situation, different densities of points), in the other case no monocular information is available, so the processing relies on motion cues only. Another relevant aspect is the fact that the human vision system not only establishes point correspondences, but also perceives *regions* in motion, although the input consists of sparse points only. This demonstrates that full analysis is possible from motion cues alone, even in a sparse configuration.

It is difficult to ascertain what exact types of information are used in the human perception processes, in what order, what is the granularity of the tokens involved, and how the entire process is carried. Humans are remarkably good in addressing this task, because they are able to bring a vast amount of processing power and knowledge into play. A wide range of higher-level knowledge, such as recognized objects or learned motion patterns, greatly influences the way humans interpret both sequences of images and isolated static scenes. In addition, they are capable of parallelizing the vision processes on many levels. Information from different locations and times in a sequence are frequently processed together, as are pieces of information from different hierarchies of understanding.

The purpose of this study is to determine to what extent this human vision process can be emulated by a computational framework that addresses the problem of motion analysis and interpretation. The proposed framework is able to consistently handle the very difficult case of grouping from motion cues only, and to integrate monocular information when available, such as in the case of real image sequences.

1.2 Computational Perspective

Before presenting the computational approach for motion analysis and interpretation, several issues need to be examined: what are the difficulties, what are the processes involved, and what constraints would be the most suitable for tackling the problem.

1.2.1 What Are the Difficulties?

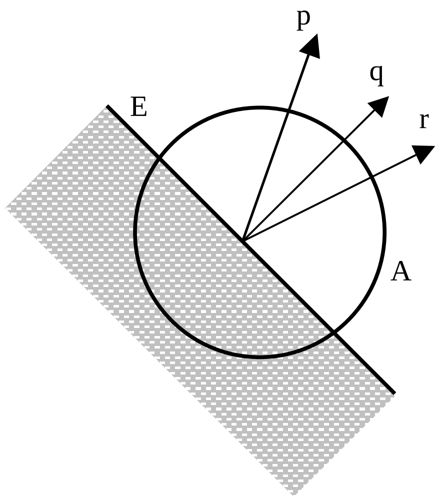
The determination by computational means of the perceived motion of objects in a sequence of images is characterized by a wide range of difficulties. The most important are caused by the aperture problem, by the presence of regions of homogeneous intensity, and by the uncertainty near the motion boundaries.

Aperture problem

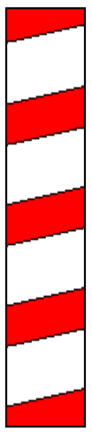
The relationship between image motion and variations of image intensity in time and space is defined by the following equation:

$$\vec{v} \cdot \nabla I + \frac{\partial I}{\partial t} = 0 \qquad (1.1)$$

In this formula $\vec{v}$ is the velocity flow (defined as the "apparent motion of brightness patterns"), and I is the image intensity. This equation is fundamental to intensity-flow calculations and it is called the *optical flow constraint equation*.



(a) Velocity uncertainty



(b) “Barber pole” illusion

Figure 1.3. The aperture problem

A closer look shows that this equation provides a constraint only for the component of the image velocity in the direction parallel to the intensity gradient. As shown in Figure 1.3(a), the motion of the edge *E* is analyzed by a local movement detector that examines a limited area of the image, represented by the aperture *A*. Such a detector can measure only the component of motion in the direction perpendicular to the orientation of the edge, indicated by the vector *q*. The component of motion along the edge is invisible through the aperture, so a local detector cannot distinguish between movement in the directions indicated by *p*, *q*, and *r*. This problem is illustrated by the well-known “barber pole” illusion, shown in Figure 1.3(b), where although the red strips move horizontally, the perceived motion is vertical.

As a consequence of the aperture problem, after computing one motion component based on local measurements, it is necessary to introduce additional constraints to combine these local measurements into a full velocity field.

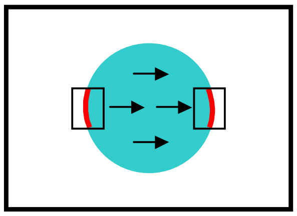


Figure 1.4. Uncertainty near motion boundaries

Regions of homogeneous intensity

Another problem arises when viewing a region lacking texture, with no or very little variation in intensity. In such a case, the motion cannot be locally determined. The explanation is that both the intensity gradient ∇I and the time derivative of the intensity $\frac{\partial I}{\partial t}$ are zero. Therefore, the optical flow $\vec{v}$ in the optical flow equation remains unconstrained, and thus it can take any value.

Uncertainty near motion boundaries

The presence of motion boundaries within an image sequence generates another range of difficulties in motion analysis. Motion boundaries can be seen as the separation between occluding and occluded objects. For the object being currently occluded, there may be pixels that have no correspondence from one frame to the other. Consequently, the apparent motion around boundaries cannot be determined by using any similarity criteria, since the regions being compared must have finite extent. Figure 1.4 gives an illustration of this problem.

In general, this problem can be seen as induced by motion discontinuities at boundaries. From a computational point of view, when additional constraints such as smoothness are used, it is difficult to enforce them while in the same time preserving the discontinuities. To accurately compute the velocity field, the knowledge of the boundaries is required so that the constraints can be relaxed around the discontinuities. But the boundaries cannot be computed without first having determined the pixel velocities. The approach described in this work, which will be described later, addresses this problem.

1.2.2 What Are the Computational Processes?

The problem of motion analysis can be decomposed in two main computational processes: matching and motion capture.

Matching. The matching process is responsible for the computation of pixel velocities from the raw input data. Finding the image velocity for a pixel is equivalent to establishing a match between that point and its correspondent in the next image frame. The output of this process is a (possibly sparse and noisy) velocity field.

Motion capture. Even when the input itself is sparse, as it has been illustrated in Figure 1.2, human vision is able to obtain a dense representation of the entities in motion. The fact that we perceive regions in motion, as opposed to points, is a well known but poorly understood effect called motion capture. The process of motion capture is then responsible for producing as output a dense velocity field, plus boundaries and regions as continuous curves and surfaces.

1.2.3 What Are the Constraints?

The examination of some simple configurations in motion indicates that the visual system incorporates a certain *affinity* measure between tokens, which can be roughly considered as a measure of similarity. This affinity is involved in both processes of matching and motion capture. Indeed, establishing a correspondence between two tokens implies the fact that their mutual affinity (or preference to each other) is greater than the affinity to other tokens. In motion capture, to determine that a token belongs to a certain region is equivalent to establishing that it has a stronger affinity to the tokens in that region than to tokens in other regions.

In order to solve the problem of motion analysis, a successful computational framework must define a way to express the affinities between tokens, while also taking into account and handling the difficulties described in the previous sections. To this purpose, additional constraints need to be introduced.

More specifically, any such constraint must satisfy two requirements:

- define a practical measure of affinities between tokens, so that they can be matched and grouped successfully
- allow the computation of motion (pixel velocities) where local measurements could not provide a complete solution – this being the case of one velocity component missing due to the aperture problem, or the case of unreliable motion due to lack of texture. In this context, the constraints are needed in order to generate a dense output from a sparse and/or noisy input.

Several types of constraints have been usually considered in the literature:

- constant velocity over an area of the image (valid for pure translation)
- constant velocity over small time intervals
- velocity consistent with 2-D rigid motion (valid for rotation and translation of objects in the image plane)

- velocity consistent with 3-D rigid motion
- smooth velocity within image areas that represent distinct objects

Methods that are based on assumptions of constant velocity, or rigid motion, are not sufficient for analyzing the two-dimensional motion that arises from the projection of arbitrary three-dimensional surfaces undergoing general motion in space. Therefore, it is the last constraint – *smoothness* – that is used in this work.

1.3 A 4-D Voting Approach

This dissertation proposes a novel computational framework that addresses the problem of visual motion analysis and interpretation from a perceptual organization perspective, where saliency is used as an affinity measure. This saliency, described in more detail later, will need to encode several perceptual concepts, such as proximity, smoothness and continuity. We claim that tokens, generated by matching corresponding pixels in the two images, form coherent perceptual structures in the 4-D space of image coordinates and pixel velocities, while erroneous matches generate outlier tokens.

The proposed approach to the problem formulated above can be characterized by taking into account two key aspects – data representation and token communication [48]. The next paragraphs describe how these issues are addressed in our work.

1.3.1 Layered 4-D Representation

Finding the velocity field means to assign velocity values at every pixel location in the 2-D image. The process of identifying moving objects in an image means to partition (segment) the

2-D image into regions according to their motion. However, casting the analysis of visual motion as a two-dimensional problem is not the most appropriate solution. The main difficulties appear at motion boundaries, where noisy velocities are abundant. This happens because tokens that are close in the image have a strong mutual influence or affinity, despite the fact that they should belong to different objects.

Accordingly, we believe that the desirable representation for the problem addressed here should be based on a layered description, where regions in motion are represented as smooth and possibly overlapping layers. Next we explain how we embed the problem into such a layered representation.

In any method that seeks to solve the problem of establishing correspondences and recovering the motion boundaries and regions, each token is characterized by four attributes – its image coordinates (x y) and its image velocity with the components (v_x v_y). In general, there may be several candidate velocities for each point (x y), so each tuple (x y v_x v_y) represents a (possibly wrong) candidate match.

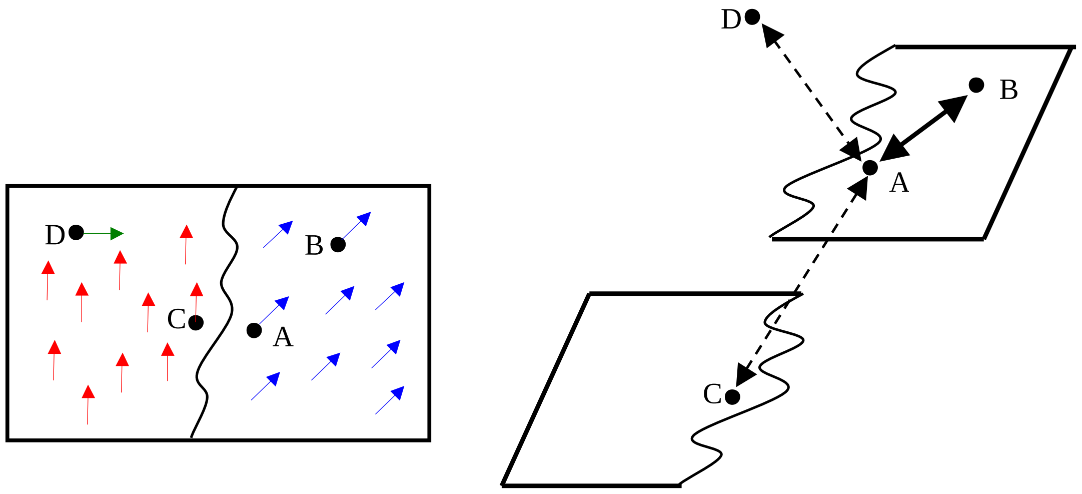
In this context, a natural solution is to encapsulate each token into a (x y v_x v_y) tuple in the 4-D space, so that they are now spatially separated by *both* velocity and image position. This is especially helpful for addressing the problem of uncertainty along motion boundaries, where although tokens are close in the image space, their interaction is now inhibited due to their separation in velocity space.

Within the proposed representation, distinct moving objects appear as *smooth layers* in the 4-D space of image coordinates and velocities. The next section describes how the motion layers are extracted.

1.3.2 Token Communication

As discussed in the previous sections, both matching and motion capture are based on a process of expressing and communicating the affinity between tokens. In the proposed 4-D layered representation, this affinity is based on the token preference for being incorporated into a smooth surface layer. A necessary condition is then to enforce strong support between tokens in the same layer, and weak support across layers, or at isolated tokens.

The example in Figure 1.5 helps illustrate this process. Token *A* exhibits a strong affinity with token *B*, as they belong to the same layer, but receives much less support from token *C*, situated in a different layer, and from token *D*, which is isolated and therefore probably a wrong match. If a 2-D representation were used, tokens *A* and *C* would exhibit a much stronger and undesired interaction due to their proximity in image space.



(a) 2-D representation (b) Layered representation

Figure 1.5. Token interaction in 2-D representation vs. layered representation.

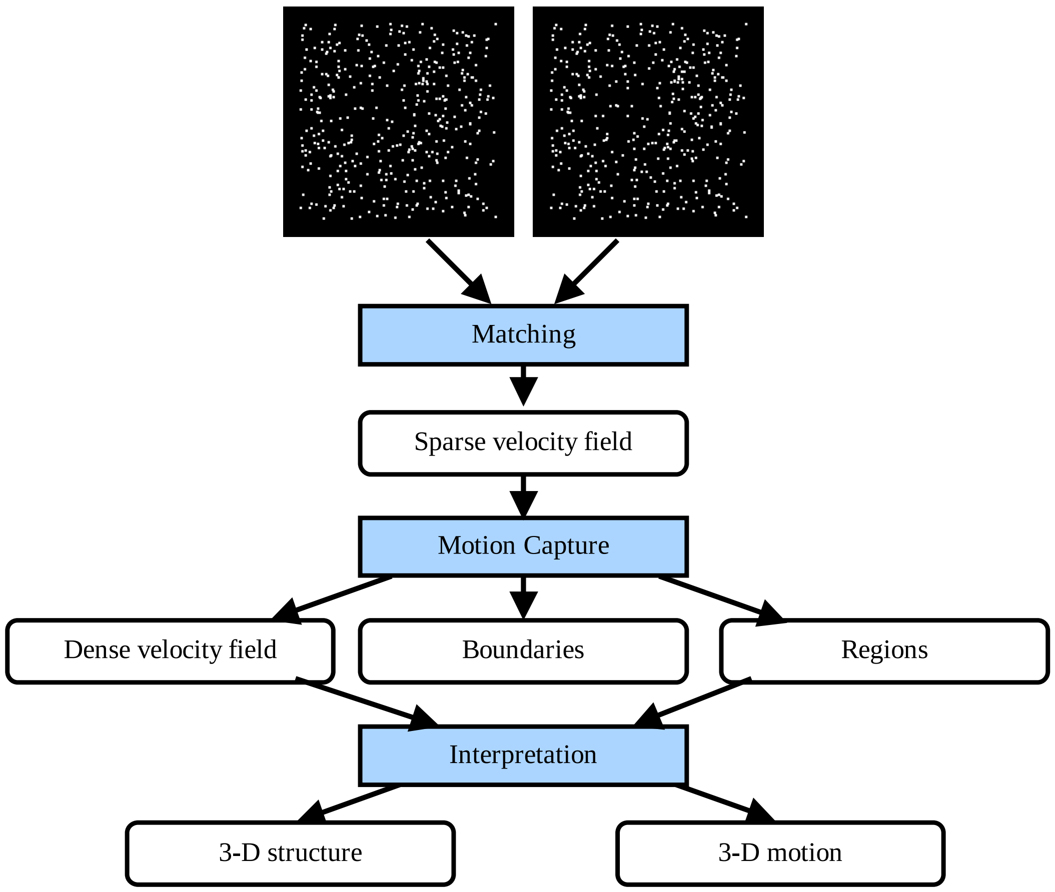


Figure 1.6. Overall view of our approach

By letting the tokens propagate their preferred information, regions that exhibit smooth motion emerge as the most salient smooth surface layers in the 4-D space, while isolated tokens that receive little or no support are identified as outliers. Essentially, the matching problem is expressed as an *outlier rejection* process, while motion capture is performed mainly as a *layer densification* process.

A suitable computational framework that enforces the smoothness constraint while preserving discontinuities is Tensor Voting, here performed in the 4-D space. As the main goal is to extract the motion layers, the affinities between tokens are embedded in the concept of surface saliency exhibited by the data. Communication between tokens is performed through convolution-like tensor voting, where each token casts a vote to its neighbors as a preference for a certain position and orientation. Incorrect matches are then eliminated as they receive little support, and layers are extracted as the most salient surfaces in the 4-D space.

The contribution of this work is demonstrated by addressing the problems of matching, motion capture, and interpretation. The overall view of our approach is illustrated in Figure 1.6. Given two sparse sets of point tokens, 4-D voting is first used to select the correct match for each input point, as the most salient token, thus producing a sparse velocity field. By using the same voting framework, a dense layer representation is determined in the motion capture process, thus inferring dense velocities, motion boundaries and regions. Finally, the 3-D structure and motion of the viewed objects is computed in the interpretation process. We analyze several difficult cases – opaque and transparent motion, rigid and non-rigid motion, curves and surfaces in motion, from sparse or dense inputs, by using motion cues only, or motion augmented by monocular cues.

1.4 Outline

Chapter 2 provides a detailed review of the previous methods used to address the problem of visual motion analysis. Chapter 3 examines the voting framework by first giving an overview of the Tensor Voting formalism, and then discussing how the voting concepts are extended to the 4-D case. Chapters 4 and 5 describe the proposed approach for the matching and motion capture problems respectively, by using the 4-D voting framework. Chapter 6 describes how monocular cues are integrated into the framework, in order to allow for handling real image sequences. Chapter 7 examines the approach for the problem of motion interpretation. Finally, Chapter 8 summarizes our contributions.

Chapter 2

Related Work

2.1 Introduction

The object of this dissertation is to establish a computational framework for the visual motion analysis and interpretation, decomposed here into a matching process that recovers token correspondences as a sparse velocity field, followed by a motion capture process that infers motion boundaries and regions, and an interpretation process that determines the 3-D structure and motion of the viewed objects.

Ullman presents an excellent analysis of the correspondence problem, from both a psychological and a computational perspective [64]. Here we are following his conclusion, that the correspondence formation is a low-level process, which takes place prior to any 3-D interpretation. A certain similarity measure between correspondence tokens, called affinity, is incorporated in the human visual system, and the correspondence between elements is determined from their affinities, via local competition interactions. The entire process is carried in a bottom-up fashion, as correspondences are not established between structured entities, on the basis of their similarity, but is built up from matches between small components of the images. The tokens involved in matching are non-complex elements, such as points, blobs, edge and line fragments. In our approach we only study the case where the input consists of identical point tokens.

A comprehensive description of the motion analysis problem is given in the work of Hildreth [25]. Several additional constraints that need to be incorporated in a computational framework are identified and discussed. According to this analysis, in our research we have employed the motion smoothness constraint, as the most general and probably the most important one that is used by the human visual system.

The problem of visual motion analysis has been intensively studied, and good results have been achieved, although for limited type of scenes, such as those containing a single, smooth and textured surface. Most approaches rely on parametric models that restrict the types of motion that can be analyzed, and involve iterative methods which depend heavily on initial conditions and are subject to instability. Further difficulties are encountered in image regions where motion is not smooth – typically around motion boundaries. This problem has lead to numerous inconsistent methods, with ad-hoc criteria introduced to account for motion discontinuities.

In the area of structure inference from motion, the inherent problem caused by the ambiguity of 2-D to 3-D interpretation is usually handled by adding additional constraints, such as rigidity. However, providing a successful computational approach has still been problematic, especially in the combined presence of noise and multiple independent motions, or even non-rigid motions. In this context it is very difficult to enforce a global constraint, as it is not clear how to handle misfits – which may correspond to outlier noise, non-rigid, or independent motion.

Barron, Fleet, and Beauchemin [5] provide a useful review of the computational methodologies used in motion analysis and interpretation. In the following subsections we discuss the most important research directions that have been investigated in the literature.

2.2 Differential Methods

Differential methods of computing optical flow reduce the problem to that of solving a partial differential equation within spatio-temporal space. The earliest example of this is the work of Horn and Schunck [26]. Such an approach is based on the optical flow constraint equation:

$$\vec{v} \cdot \nabla I + \frac{\partial I}{\partial t} = 0 \tag{2.1}$$

As explained in Chapter 1, this is only one equation providing constraints on the optical flow $\vec{v}$, an unknown with two components. Consequently, only the component of the optical flow normal to the local intensity gradient is constrained, phenomenon known as the aperture problem. The system is therefore underdetermined, requiring that additional constraints be imposed to ensure a unique solution. Horn and Schunck augmented the constraints with a *global* smoothness constraint on the optical flow. Their approach fails to account for discontinuities in the motion field, that are present at motion boundaries.

Using the equation above it is possible to frame the problem in terms of a minimization within a local neighborhood. Lucas and Kanade [37] applied this method by using a small window with Gaussian weighting *w*. The quantity to be minimized is:

$$S = \sum_{x \in R} w^2(x) \left[\vec{v} \cdot \nabla I + \frac{\partial I}{\partial t} \right]^2 \tag{2.2}$$

A combination of this approach with a Bayesian framework is also presented by Simoncelli et al. [55].

Second-order differential techniques, which employ the second-order derivatives of I to constrain the velocity $\vec{v}$ were introduced by Nagel et al. [42][43][44]. The applicable differential equation is derived from a requirement that the local intensity gradient ∇I is conserved in time. In an attempt to properly handle occlusion, Nagel suggested that smoothness only be imposed orthogonal to steep intensity gradients, to prevent smoothing across what could possibly be motion boundaries. The problem is ultimately formulated as a minimization of an energy functional. Unfortunately, the initial constraint does not permit common types of motion such as rotation and expansion.

2.3 Region-Based Methods

In these methods, an attempt is made to determine the most likely displacement *d* of a region *R* between consecutive frames of the image sequence. The parameters in such a calculation are the spatial extent of the region to be matched, the range of possible displacements, and a defined intensity metric between regions. The usual metrics employed are the normalized cross-correlation coefficient (which is to be maximized), or the sum of squared distance measure:

$$D = \sum_{x \in R} w(x)\left[I_2(x) - I_1(x+d)\right]^2 \tag{2.3}$$

Anandan [2] has combined a sum of squared difference metric with a Laplacian pyramid technique [9] to determine optical flow. In this technique, sub-pixel accuracy in this difference metric is achieved with a quadratic approximation. In addition, Anandan employs a smoothness constraint on the optical flow field *v*, which attempts to minimize the sum of the Laplacians of the two components of the flow field, $\nabla^2 v_x + \nabla^2 v_y$. Such a technique suffers from an attempt to smooth the optical flow across motion boundaries, as well as restricting input to data derived from only two neighboring frames.

Singh [56][57] attempts to remedy the latter difficulty by defining a new metric, which is the average of the metrics in forward and reverse time, thereby incorporating information from three neighboring frames. If D is the actual sum of squared difference, the sub-pixel displacement is calculated as a weighted average of all possible displacements, the weighting function being provided by:

$$W(d) = e^{-cD(d)} \tag{2.4}$$

A second step in the processing performs a Gaussian smoothing of the derived optical flow, allowing velocity information to propagate locally. These derived values are then used as input for repeated iterations of a similar calculation. Once again, since the smoothness constraints use no knowledge of motion boundaries, this technique over-smoothes the optical flow at these locations.

2.4 Energy-Based Methods

Energy-based techniques map the optical flow field in spatio-temporal space into Fourier space. A simple application of the shift theorem to the optical flow constraint equation yields:

$$F(k, \omega) = F_0 \delta(\omega + v \cdot k) \tag{2.5}$$

In this equation, F_0 is the Fourier transform of $I(x,0)$, ω is the temporal frequency, and k is the spatial frequency. This shows that the power spectrum of a translating, fixed intensity pattern must exist entirely within a plane passing through the origin of the 2D + t frequency space. This restriction can then form the basis of new methods for calculating optical flow.

Heeger [22][23] exploits this restriction to provide a method that determines optical flow by attempting a least-squares fit to a plane in frequency space. This approach suffers from the defect that any object undergoing rotation or a non-rigid motion such as expansion cannot be adequately represented.

2.5 Markov Random Fields

Several research efforts have investigated the usefulness of Markov Random Fields (MRF) in treating discontinuities in the optical flow [8][24]. Gelgon and Bouthemy [16] perform motion segmentation in three sequential steps. In the first step, the images in a sequence are each partitioned at the pixel level according to an intensity-texture criterion, using a Markov Random Field procedure [30]. This provides a spatial region graph, which acts as an abstraction of the pixel-level representation of the image.

In a second step, the spatial region graph is partitioned according to motion criteria, through the minimization of an energy function. This energy function is a weighted sum of two terms, one being a discrepancy term which favors identical motion labels in the graph when associated motions are similar, the second being a regularization term which favors identical labels for neighboring regions (represented by neighboring nodes in the spatial graph).

In the third step, the optical flow of the regions is determined through a merging of motion data from sets of regions with identical motion labels derived in the previous stage. While this technique gives some good results, it relies heavily on a proper spatial segmentation early in the algorithm, which will not be realistic in many cases. And since the granularity of the spatial region representation is necessarily coarse, little freedom is provided for capturing intricate structure in the optical flow.

2.6 Layered Representations and EM

Significant improvements have been achieved by casting the problem in terms of layered descriptions [11][29][27]. This formalism has many advantages. It is a natural way to accommodate discontinuities present in the motion field. Also, it inhibits information transfer between layers as spatially separated regions, and may resolve local uncertainties.

The work of Ayer and Sawhney [4] attacks this problem through an application of the Expectation-Maximization (EM) algorithm [38] together with a mixture model. The mixture model describes each location in the image in terms of probabilities distributed among a finite set of discrete states. These states can be thought of as corresponding to discrete objects in the image, each providing a smooth flow field. In a given image, an optical flow discontinuity manifests as an abrupt change in the state containing the highest probability density.

An EM algorithm discovers the motion discontinuities in an iterative fashion. During the expectation step, the state probabilities are optimized for the current value for the optical flow. During the maximization step, the parameters governing the optical flow are optimized while holding the state probabilities fixed. An initialization step provides a reasonable set of probability densities.

Since the number of possible states can increase without bound, Ayer and Sawhney incorporate an intermediate Minimum Description Length (MDL) criterion between the expectation and maximization stages, in order to compromise between the simplicity and accuracy of the representation.

While this technique provides a basis for much subsequent study, it still suffers from three major defects. First, the procedure requires an initialization step, which is essentially arbitrary. Second, the algorithm is iterative, and subject to stability concerns. Third, the description of the optical flow is parameterized, and does not permit a general description as would be desirable.

Many other current methods use common motion to group regions, usually performing a parameterized fit to motion data [28][65]. Weiss [66] provides a good overview of the difficulties involved in this estimation process, which range from inadequate representation of motion as rigid, to the over-fitting and instabilities resulting from higher-order parameterizations.

Weiss uses a layered representation in combination with the EM algorithm, where a dense smooth flow field is fit to multiple layers. In this case, the number of layers is computed automatically by initially over-estimating the partitioning. A final step in the algorithm merges layer indices that are judged to be similar.

The expectation step in the algorithm assigns the most likely layer labels to each pixel based upon a Markov Random Field that favors identical labels for neighboring pixels. The maximization step adjusts the optical flow associated with each layer by maximizing the conditional posterior probability of the optical flow based upon a fixed set of pixel layer indices. A functional is minimized which incorporates a Horn-Schunck term as well as term which penalizes a lack of smoothness in the optical flow field.

A novel feature of the method is the representation of the optical flow field as a linear combination of localized flow functions. The Horn-Schunck optical flow criterion is translated into a large-scale linear system combining the localized flow functions with the layer labels. A subsequent substantial reduction in the dimensionality of the linear system renders the method computationally feasible.

While the results obtained by this method are good, it is still an iterative technique, subject to all associated liabilities. In addition, it performs a mathematical fit of the optical flow, even allowing spatially separated regions to influence each other. This precludes any possibility of incorporating higher-level knowledge (e.g. occlusion information) into the calculation. It is possible for unrelated regions to be accidentally merged into a single layer simply because of similar motion profiles, despite the presence of conflicting occlusion evidence, while the merging of spatially diffuse regions is more appropriately the domain of higher-level processing.

2.7 Variational Methods

Due to the ambiguity of the general vision problem, which is inherently ill-posed, attempts have been made to identify and model the physical constraints that make it determined and solvable. The under-constrained nature of the motion analysis problem has led to the development of a class of methods that use variational principles to impose specific physical constraints. These methods are derived from the regularization theory for solving ill-posed problems, which are transformed into a non-linear, scalar functional optimization framework.

However, the discontinuity aspect of the constraint involved is hard to express along with smoothness in such a functional optimization framework. The most limiting factor in the regularization theory is that the solution corresponds to a minimum which globally reduces the error. As such, discontinuities in the data are not preserved.

Recent approaches [17][13] augment the regularization formalism by replacing the quadratic regularization term (usually used to recover a smooth solution) with a particular function of the gradient flow, specifically derived to allow for flow discontinuities in the solution. These techniques attempt to explicitly preserve discontinuities by weakening the smoothing in areas that exhibit strong intensity gradients. The main issues of variational methods are convergence, numerical stability, parameter and initialization dependency. In addition, an incorrect assumption is also made here, that the motion boundaries can always be detected in advance, based on intensity only.

2.8 Basis Set Methods

An example of using basis set methods (in the form of steerable flow fields) is the work of Fleet, Black, and Jepson [14]. At each pixel location, the optical flow is expanded as a linear combination of basis functions. But, in order to accommodate the presence of motion discontinuities, an additional parameter θ is included which incorporates the orientation of a potential motion boundary.

The optical flow equation is used to enable a least squares fit solution to the vectors of unknown coefficients. In order to provide a continuously varying set of motion boundary angular orientations in the basis set, the basis functions are permitted to acquire imaginary components, since coefficients of the form $e^{i\theta}$ are equivalent to rotation operators in the complex plane.

The results of this technique are good, but the use of a gradient descent solution to the resulting system of equations is heavily dependent on initial conditions and parameters governing movement in the coefficient solution space.

2.9 Wavelets

Wu, Kanade, Cohn and Li [67] have applied the wavelet techniques of Cai and Wang [10] to the problem of optical flow determination. Optical flow is described as a linear combination of 2D wavelets. A coarse to fine adjustment is enabled by using different velocity space resolutions in a hierarchical pyramid. This permits capture of a wide range of velocity magnitudes without the instability created by applying coarse to fine adjustment in the intensity space, which can create poor optical flow values at low-resolution where image structure is lost. Instead, full image resolution is used at all levels of the velocity space pyramid.

Based on the optical flow constraint previously described, the sum of squared difference quantity also used by the region-based methods is minimized. While the results of this technique are fairly adequate, motion discontinuities are modeled poorly due to over-smoothing. The presence of iteration in the solution of the system of equations also leaves open the possibility of instability.

2.10 Graph-Based Methods

Shi and Malik [52] have approached the problem of motion segmentation in terms of recursive partitioning of the spatio-temporal space through normalized cuts within a weighted graph [51]. The associated graph encodes similarities between the motion profiles of points in spatio-temporal space, with similar pairs of points supporting graph edges of low weight. The cuts are normalized in such a way as to not favor partitions with small surface area.

Motion profiles encode regional similarities in probabilities of displacements. The results are relatively good, but no prescription is offered for deciding when spatio-temporal space has been adequately partitioned.

2.11 Voting Methods

Little et al. [34] developed a parallel algorithm for computing the optical flow by using a local voting scheme based on similarity of planar patches. However, their methodology cannot prevent motion boundary blurring due to over-smoothing and is restricted to short-range motion only.

The first to propose using Tensor Voting in addressing the motion analysis problem were Gaucher and Medioni [15]. They employ successive steps of voting, first to determine the boundary points as the tokens with maximal motion uncertainty, and then to locally refine velocities near the boundaries by allowing communication only between tokens placed on the same side of the boundary. However, in their approach the voting communication between tokens is essentially a two-dimensional process that does not inhibit neighboring elements with different velocities from influencing each other.

Chapter 3

The Tensor Voting Framework

3.1 Tensor Voting Overview

The use of a voting process for structure inference from sparse data was introduced by Guy and Medioni [18][19] and then formalized into a unified tensor framework [31][32][33] [59][61][40]. The smoothness constraint is used in order to generate descriptions in terms of surfaces, curves, and junctions, from sparse and noisy data in 2-D or 3-D.

The methodology is grounded on two elements: *tensor calculus* for data representation, and non-linear *voting* for data communication. An overall illustration of the method, summarizing its different components, is given in Figure 3.1, which shows the 3-D version.

Each input token can be a point, a point with an associated tangent direction, a point with an associated normal direction, or any combination of the above. Every such token is encoded into a second order symmetric tensor. In a first voting stage, tokens communicate their preferred information in a neighborhood through a predefined tensor field, and cast a tensor vote. The preference information includes proximity, smoothness, and continuity. Each site collects all the votes cast at its location and encodes them into a new tensor. After this refinement process, each token is now a generic second order symmetric tensor, which encodes curve and surface orientation information (given by the tensor orientations), and confidence of this knowledge (given by the tensor size).

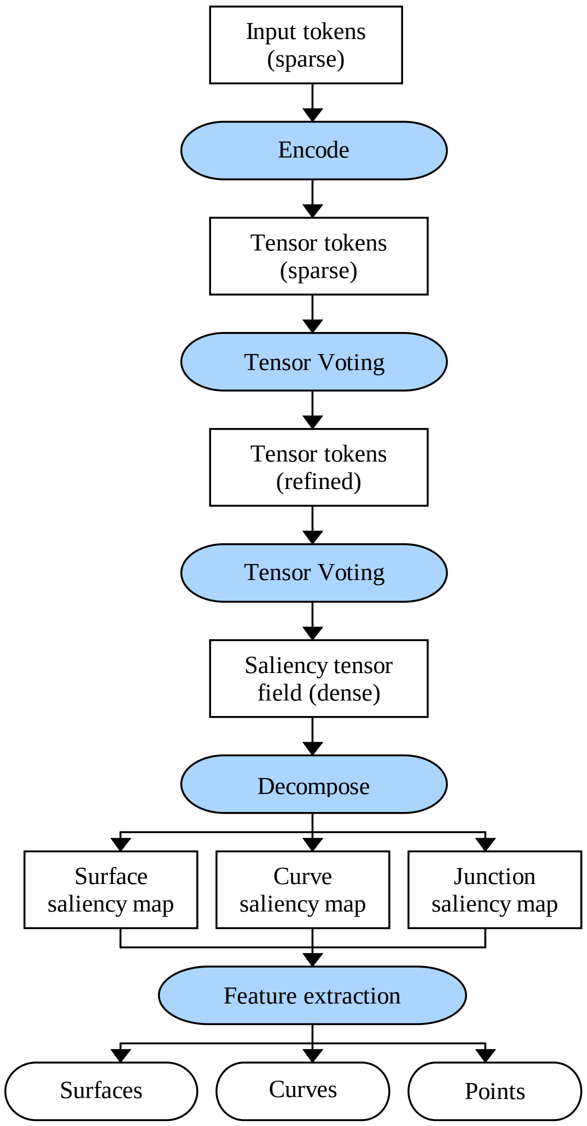


Figure 3.1. Tensor Voting overview

In a second voting stage, these generic tensor tokens propagate their information at each discrete location in their neighborhood, leading to a dense tensor map that encodes feature saliency at every point in the domain. In practice, the domain space is digitized into a uniform array of cells.

The resulting dense tensor map is decomposed by building a saliency map for each feature type. Surface, curve, and junction features are obtained by extracting local extrema of the corresponding saliency values along normal directions. The final output is the aggregate of the outputs for each of the components.

This methodology is non-iterative and robust to considerable amounts of outlier noise. The only free parameter is the scale of analysis, which is indeed an inherent property of visual perception.

3.1.1 Tensor Representation

Points can simply be represented by their coordinates. A local description of a curve is given by the point coordinates, and its associated tangent or normal. A local description of a surface patch is given by the point coordinates, and its associated normal. Here, however, it is not known in advance what type of entity (point, curve, surface) a token may belong to. Furthermore, because features may overlap, a location may actually correspond to multiple feature types at the same time.

To capture the geometric information and its singularities, a second order symmetric tensor is used. It captures both the orientation information and its confidence, or saliency. Such a tensor can be visualized as an ellipse in 2-D, or an ellipsoid in 3-D. Intuitively, the shape of the tensor defines the type of information captured (point, curve, or surface element), and the associated size represents the saliency. For instance, in 2-D, a very salient curve element is represented by a thin ellipse, whose major axis represents the estimated tangent direction, and whose length reflects the saliency of the estimation.

To express a second order symmetric tensor *S*, graphically depicted by an ellipsoid in 3-D, the associated quadratic form is diagonalized, leading to a representation based on the eigenvalues λ_1, λ_2, λ_3 (where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$), and the eigenvectors e_1, e_2, e_3:

$$S = \lambda_1 e_1 e_1^T + \lambda_2 e_2 e_2^T + \lambda_3 e_3 e_3^T \tag{3.1}$$

The eigenvectors represent the principal directions of the ellipsoid and the eigenvalues encode the size and shape of the ellipsoid.

An input token that represents a surface element will be encoded as an elementary *stick tensor*, where e_1 represents the surface normal, while $\lambda_1=1$ and $\lambda_2=\lambda_3=0$. An input token that represents a curve element will be encoded as a *plate tensor*, where e_3 represents the curve tangent, while $\lambda_1=\lambda_2=1$ and $\lambda_3=0$. An input token that represents a point element will be encoded as a *ball tensor*, with no preferred orientation, while $\lambda_1=\lambda_2=\lambda_3=1$. Figure 3.2 shows the elementary tensors that corespond to surface, curve and point tokens, in the 3-D case.

3.1.2 Tensor Decomposition

As a result of the voting procedure, *generic* second-order, symmetric tensors are produced from the elementary tensors described above, therefore the need to handle generic tensors. Any tensor can be expressed as a *linear* combination of these three cases:

$$S = (\lambda_1-\lambda_2)e_1 e_1^T + (\lambda_2-\lambda_3)(e_1 e_1^T+e_2 e_2^T) + \lambda_3(e_1 e_1^T+e_2 e_2^T+e_3 e_3^T) \tag{3.2}$$

In this equation, $e_1 e_1^T$ describes a stick, $(e_1 e_1^T+e_2 e_2^T)$ describes a plate, and $(e_1 e_1^T+e_2 e_2^T+e_3 e_3^T)$ describes a ball.

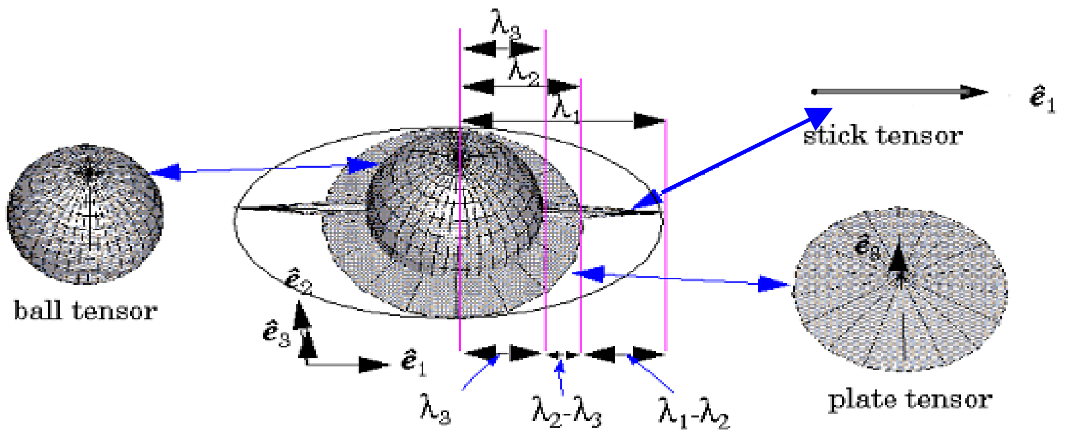


Figure 3.2. Tensor decomposition

At each location, the estimate of each of the three types of information and their associated saliency are captured as follows:

- *point-ness*: no orientation, saliency is λ_3
- *curve-ness*: tangent orientation is e_3, saliency is λ_2 -λ_3
- *surface-ness*: normal orientation is e_1, saliency is λ_1-λ_2

In 2-D, there is no surface-ness, and curve-ness is expressed by e_2 for the tangent orientation, and by λ_1-λ_2 for curve saliency.

3.1.3 Tensor Communication

We now describe the communication and computation scheme, which allows a site to exchange information with its neighbors, and infer new information.

Token refinement and dense extrapolation. The input tokens are first encoded as elementary tensors. In 3-D, a point token is encoded as a 3-D ball. A point associated with tangent direction is encoded as a 3-D plate. A point associated with normal direction is encoded as 3-D stick. These initial tensors communicate with each other in order to:

- derive the most preferred orientation information, or refine the initial orientation if given, for each of the input tokens (*token refinement),* and
- extrapolate the above inferred information at every location in the domain for the purpose of coherent feature extraction (*dense extrapolation*).

In the token refinement case, each token collects all the tensor values cast at its location by all the other tokens. The resulting tensor value is the tensor sum of all the tensor votes cast at the token location.

In the dense extrapolation case, each token is first decomposed into its independent elements, then it broadcasts this information. In this case ball tensors do not vote, as they define isolated features, which do not need to propagate their information. While they may be implemented differently for efficiency, these two operations are equivalent to a *voting* process, and can be regarded as a *tensor convolution* with *voting fields (kernels)*.

Derivation of the voting fields. The size and shape of the voting neighborhood, and the vote strength and orientation are encapsulated in predefined voting fields, one for each feature type – there is a stick voting field, a plate voting field, and a ball voting field in the 3-D case. The fields are generated based on a single parameter – the scale factor σ. Vote orientation corresponds to the best (smoothest) local curve continuation from voter to recipient, while vote strength $VS(\vec{d})$ decays with distance $|\vec{d}|$ between them, and with curvature ρ:

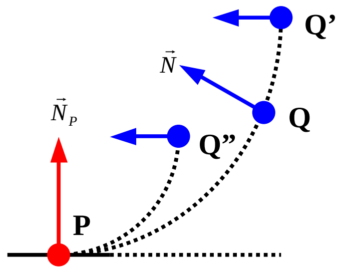
$$VS(\vec{d}) = e^{-\left(\frac{|\vec{d}|^2+\rho^2}{\sigma^2}\right)} \tag{3.3}$$

All voting fields can be derived from the *fundamental 2-D stick field*, by rotation and integration. Figure 3.3(a) shows how votes are generated to build the 2-D stick field. A tensor P where curve information is locally known (illustrated by curve normal $\vec{N}_P$) casts a vote at its neighbor Q. The vote orientation is chosen so that it ensures a smooth curve continuation (through a circular arc) from voter P to recipient Q. To propagate the curve normal $\vec{N}$ thus obtained, the vote $V_{stick}(\vec{d})$ sent from P to Q is encoded as a tensor according to the equation below, where $\vec{d} = \mathrm{Q} - \mathrm{P}$.

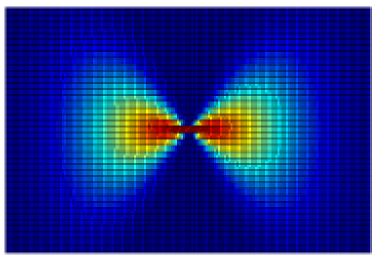
$$V_{stick}(\vec{d}) = VS(\vec{d}) \cdot \vec{N}\vec{N}^T \tag{3.4}$$

Note that vote strength at both Q' and Q" is smaller than at Q – because Q' is farther, and Q" requires a higher curvature than Q. Figure 3.3(b) shows the 2-D stick field, with its color-coded strength. When the voter is a ball tensor, with no information known locally, the vote is generated by rotating a stick vote in the 2-D plane and integrating all contributions, according to the equation below. The corresponding 2-D ball field is shown in Figure 3.3(c).

$$V_{ball}(\vec{d}) = \int_0^{2\pi} R_\theta \, V_{stick}(R_\theta^{-1}\vec{d}) \, R_\theta^T \, d\theta \tag{3.5}$$



(a) Vote generation


(b) 2-D stick field

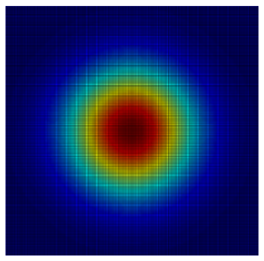

(c) 2-D ball field

Figure 3.3. Voting in 2-D

The 3-D stick field is obtained by first rotating a fundamental 2-D stick field V_{stick} with 90° about the *z*-axis (denote it by V'_{stick}). Then, V'_{stick} is rotated about the *x*-axis, and the contributions are integrated by tensor addition during the rotation. To obtain the 3-D plate field, the 3-D stick field obtained above is rotated about the *z*-axis, integrating the contributions by tensor addition. To obtain the 3-D ball field, the 3-D stick field is rotated about the *y*-axis and *z*-axis, integrating the contributions by tensor addition.

3.1.4 Feature Extraction

At each receiving site, the collected votes are combined through simple tensor addition (sum of matrices $V(\vec{d})$), thus producing generic tensors. During voting, tokens that lie on a salient geometric feature (curve, surface) reinforce each other, and the tensors deform according to the prevailing orientation.

At the end of the voting process, a dense tensor map has been produced, which is then decomposed in three dense maps: the *surface map*, the *curve map*, and the *junction map*. Each voxel of these maps has a 2-tuple $(s, \hat{v})$, where *s* is a scalar indicating strength and $\hat{v}$ is a unit vector indicating direction:

- *Surface map* (*SMap*): $s = \lambda_1 - \lambda_2$, and $\hat{v} = e_1$ indicates the normal direction.
- *Curve map* (*CMap*): $s = \lambda_2 - \lambda_3$, and $\hat{v} = e_3$ indicates the tangent direction.
- *Junction map* (*JMap*): $s = \lambda_3$, and $\hat{v}$ is arbitrary.

These maps are then used as input to an extremal-extraction algorithm similar to a marching process [36], in order to generate features such as junctions, curves, and surfaces. The

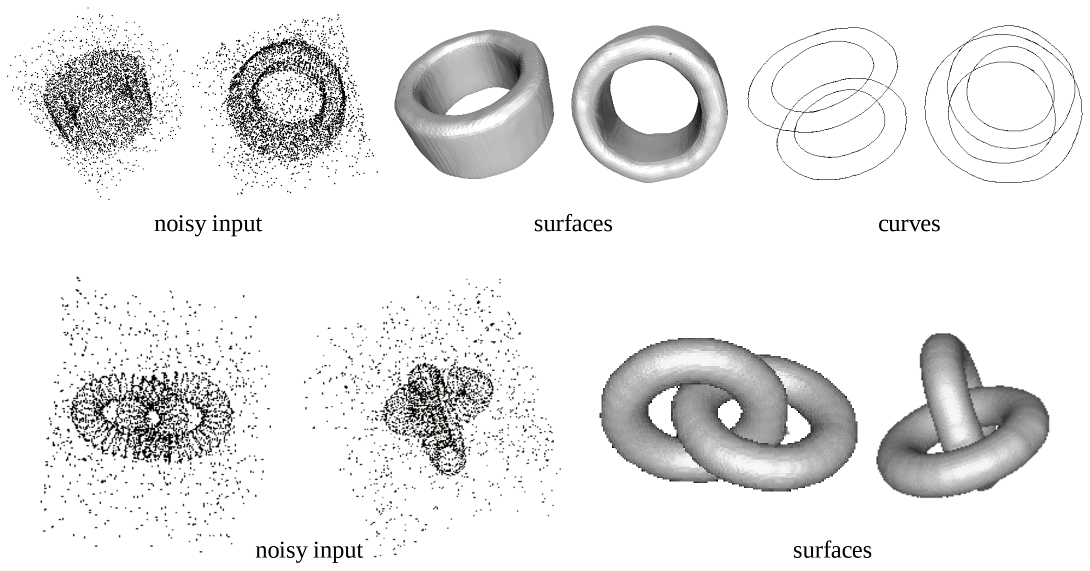


Figure 3.4. Pipe and two linked tori

definition of point extremality, corresponding to junctions, is straightforward: it is a local maximum of the scalar value *s*. A point P is on an *extremal surface* if its strength *s* is locally extremal along the direction of the normal. A point P is on an *extremal curve* if any displacement from P on the plane normal to the tangent will result in a lower *s* value. Detailed implementation can be found in [60][40].

Results of applying the tensor voting methodology in the 3-D case are shown in Figure 3.4.

3.2 Tensor Voting in 4-D

We formulate our methodology for matching and motion capture by using a Tensor Voting framework in a four-dimensional space. In this section we describe how the Tensor Voting formalism is extended to the 4-D case.

The Tensor Voting framework is general enough to be extended to any dimension readily, except for some implementation changes, mainly for efficiency purposes [62]. The main issues that need to be addressed are the *tensor representation* of the features in the desired space, the generation of *voting fields,* and the *data structures* used for vote collection.

Before discussing how the issues above affect the Tensor Voting framework, we need to make a few clarifying comments on the possible geometric varieties that may exist within an arbitrary dimensional space.

In any N-dimensional space there are N types of geometric features (varieties), whose dimensionality ranges from 0 to N-1. The *dimensionality* of each of these features is given by the number of parameters that are needed to describe the feature in a parametric model. For a given feature, the number of such parameters is the same, regardless of the N-dimensional space.

For example, in *any* N-D space (with dimensions x_1, x_2 … x_N) a curve has dimensionality 1, because it is parametrically described through one parameter p, by the set of equations:

$$x_k = x_k(p) \qquad \text{where } k = 1...N \tag{3.6}$$

Similarly, in *any* N-D space a surface has dimensionality 2 because it is parametrically described through two parameters p and q, by the set of equations:

$$x_k = x_k(p,q) \qquad \text{where } k = 1...N \tag{3.7}$$

Moreover, each feature can be considered as defined locally by a number of *tangent vectors*. The number of tangent vectors is the same as the dimensionality of the feature, because each such vector is given by the partial derivatives of the dimension variables x_1, x_2 … x_N with respect to one of the parameters.

For instance, a surface is defined by two tangent vectors:

$$\begin{aligned}\vec{t}_1 &= (\frac{\partial x_1}{\partial p} \quad \frac{\partial x_2}{\partial p} \cdots \frac{\partial x_N}{\partial p}) \\ \vec{t}_2 &= (\frac{\partial x_1}{\partial q} \quad \frac{\partial x_2}{\partial q} \cdots \frac{\partial x_N}{\partial q})\end{aligned} \tag{3.8}$$

Equivalently, if n_t is the number of tangent vectors, a feature can also be locally determined by a number of *normal vectors* n_n = N - n_t. Following the example above, a surface in N-D is also defined by 2 normal vectors. In fact all these tangent and normal vectors together represent an orthonormal basis for the N-dimensional space.

According to the discussion above, in a 4-D space there are four possible features:

- *point* (0-D) - having no tangents and 4 normals
- *curve* (1-D) - having 1 tangent and 3 normals
- *surface* (2-D) - having 2 tangents and 2 normals
- *volume* (3-D) - having 3 tangents and 1 normal

Tensor representation. After this brief geometric interlude, we return to the problem of tensor representation in 4-D. The four possible geometric features mentioned above correspond to the four elementary tensors for the 4-D space. Table 3.1 shows how each of these features is encoded as an *elementary tensor*, by specifying the values of the eigenvalues and eigenvectors in each case.

Table 3.1. Elementary tensors in 4-D

Feature	λ_1 λ_2 λ_3 λ_4	e_1 e_2 e_3 e_4	Tensor
point	1 1 1 1	any orthonormal basis	ball
curve	1 1 1 0	n_1 n_2 n_3 t	C-plate
surface	1 1 0 0	n_1 n_2 t_1 t_2	S-plate
volume	1 0 0 0	n t_1 t_2 t_3	stick

Table 3.2. A generic tensor in 4-D

Feature	Saliency	Normals	Tangents
point	λ_4	none	none
curve	$\lambda_3 - \lambda_4$	e_1 e_2 e_3	e_4
surface	$\lambda_2 - \lambda_3$	e_1 e_2	e_3 e_4
volume	$\lambda_1 - \lambda_2$	e_1	e_2 e_3 e_4

Table 3.2 shows how each of the four geometric features can be extracted from a *generic tensor*, which is produced after voting. It is now clear that by following these rules, the tensor representation can be easily extended into any dimension.

In our computational framework for visual motion analysis we represent the motion layers as surfaces embedded in a 4-D space. Therefore, throughout this study we are mainly interested in extracting salient surfaces from the input data.

Voting fields. The voting fields are a key part of the formalism – they are responsible of the size and shape of the neighborhood where the votes are cast, and also control how the votes depend on distance and orientation. As explained in the previous subsection, there is only one fundamental voting field – the 2-D stick field. All other voting fields – for different features and in higher dimensional spaces – are derived from the 2-D stick.

The 4-D voting fields are obtained as follows. First the 4-D stick field is generated in a similar manner to the 2-D stick field, as it was explained in the previous section and illustrated in Figure 3.3. Then, the other three voting fields are built by integrating all the contributions obtained by rotating a 4-D stick field around appropriate axes. In particular, the 4-D ball field – the only one directly used in this work – is generated according to:

$$V_{ball}(\vec{d}) = \int\int\int_{0}^{2\pi} R_{\theta_{xy}\theta_{xu}\theta_{xv}} \, V_{stick}(R^{-1}_{\theta_{xy}\theta_{xu}\theta_{xv}} \vec{d}) \, R^{T}_{\theta_{xy}\theta_{xu}\theta_{xv}} \; d\theta_{xy} d\theta_{xu} d\theta_{xv} \tag{3.9}$$

where x, y, u, v are the 4-D coordinates axes, θ_{xy}, θ_{xu}, θ_{xv} are rotation angles in the specified planes, and the stick field corresponds to the orientation (1 0 0 0).

Data structures. In the 2-D or 3-D case, the data structure used to store the tensors during vote collection was a simple 2-D grid or a red-black tree. Because we need a data structure that is gracefully scalable to higher dimensions, the solution used in our approach is an *approximate nearest neighbor* (ANN) *k-d tree* [3].

Since we use efficient data structures to store the tensor tokens, the space complexity of the algorithm is linear, or $O(n)$, where n is the input size. The average time complexity of the voting process is $O(\mu n)$ where μ is the average number of tokens in the neighborhood. Therefore, in contrast to other voting techniques, such as the Hough Transform, both time and space complexities of the tensor voting methodology are *independent* of the dimensionality of the desired feature. The running time for an input of size 700 is about 20 seconds on a Pentium III (600 MHz) processor.

Space non-isotropy. A key component of our framework is the 4-D layered representation of data. Within the 4-D space of image positions (x y) and potential pixel velocities (v_x v_y), moving regions are represented as smooth surface layers, and are extracted through a voting process that enforces the motion smoothness constraint.

Although both image positions (x y) and velocities (v_x v_y) are measured in pixels, they represent conceptually different, independent entities. Consequently, the 4-D space used here is not an isotropic one. However, the domain of image velocities is finite, as it is bounded by the image size – for an image with a size of SxS pixels, possible velocities range between $-S$ and S. In practice, since typical image motions are rather small (usually between $-S/10$ and $S/10$), the distance between layers in velocity space is small compared to the image size. Therefore, before voting we scale the velocities (v_x v_y) so that the typical separation between layers is in the same order of magnitude as the image size. For all image sequences analyzed in this work, we have scaled the velocity values with the same factor of 10.

Quantization effects. In the general tensor voting framework, an important issue is the underlying grid that is chosen for extracting dense salient geometric features. The size of voxels in the grid directly influences the feature extraction procedure (similar to a Marching Cubes algorithm), in terms of processing time, memory requirements, and precision of the extracted features.

In our 4-D voting framework for motion analysis, we do not use a marching algorithm to extract dense motion layers. Instead, we generate discrete velocity candidates at each pixel location, collect votes at each candidate, and choose the most salient candidate as the most likely velocity at that pixel. Furthermore, the size of the underlying grid is fixed in the image space (1 pixel), as we are interested in inferring velocity values at each image location. In velocity space, in order to obtain velocity values with subpixel precision, the candidates are generated at every 1/4 pixel.

Chapter 4

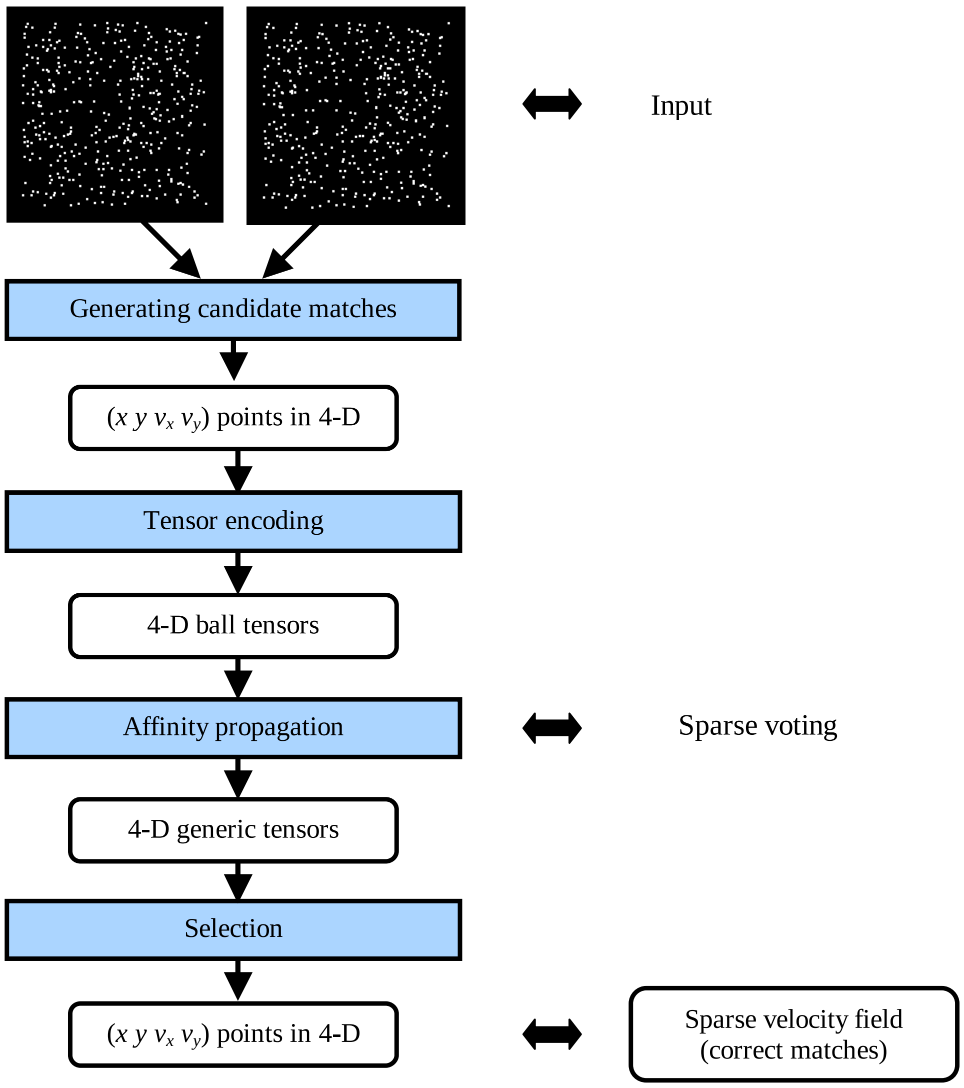
Matching

4.1 Introduction

In this chapter we present our approach for the problem of matching from two sparse sets of identical point tokens, when only motion cues are available [45]. The entire approach is based on the 4-D voting framework that has been presented in the previous chapters. An overview of the processes involved in the various stages of data processing is shown in Figure 4.1.

The input consists of two frames containing identical point tokens, in a sparse configuration. For illustration purposes, we give a step-by-step description of the approach by using a specific example – the random point tokens represent an opaque **translating disk** against a static background. Later we also show how our method performs on several other examples.

In the first stage we generate candidate matches from the image data, as ($x\ y\ v_x\ v_y$) points in the 4-D space. These points are then encoded as 4-D ball tensors, and their mutual affinities are propagated through a step of sparse voting. From the 4-D generic tensors resulted after voting, the ones that have maximal surface saliency are retained, while the others are eliminated as wrong matches. The final result is a set of 4-D points that represent the correct matches, and thus a sparse velocity field.

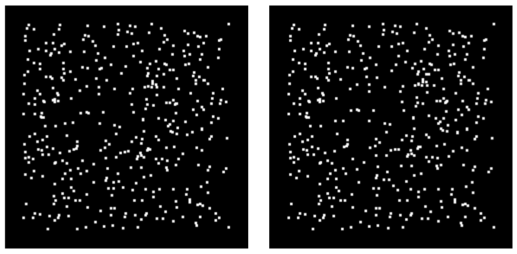


Input
Generating candidate matches
(x y vx vy) points in 4-D
Tensor encoding
4-D ball tensors
Affinity propagation
Sparse voting
4-D generic tensors
Selection
(x y vx vy) points in 4-D
Sparse velocity field (correct matches)

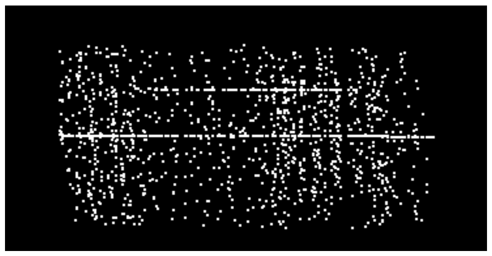
Figure 4.1. The matching process

Since the goal is to extract the correct velocity (match) for each token, while eliminating the wrong matches, the process implemented here can be seen as a classification of the $(x\ y\ v_x\ v_y)$ points into inliers and outliers.

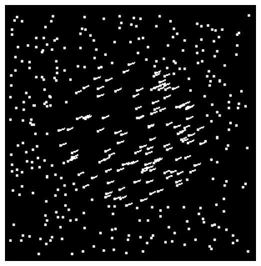
Before presenting in more detail each of the steps involved, we need to make a brief comment on how we display the intermediate results (i.e. those in 4-D). For illustration purposes, the



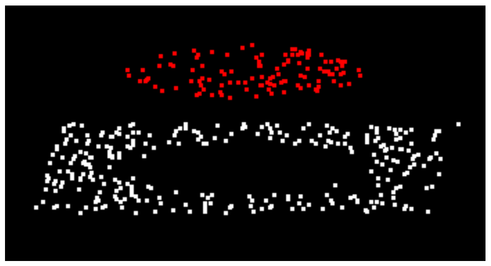
(a) Input



(b) Candidate matches



(c) Sparse velocity field



(d) Recovered v_x velocities

Figure 4.2. Translating disk

last component of each 4-D point has been dropped to allow a three-dimensional display. More specific, the three dimensions shown are the image coordinates *x* and *y* (in the horizontal plane), and the v_x component of the image velocity (the height).

4.2 Generating Candidate Matches

We take as input two sparse sets of identical point tokens, as shown in Figure 4.2(a). Candidate matches are generated as follows: for each token in the first frame, we simply create a potential match with every point in the second frame that is located within a neighborhood (whose size is given by the scale factor) of the first token. The resulted candidates appear as a cloud of (x,y,v_x,v_y) points in the 4-D space. The translation example has 400 input points, and by using the procedure described above we generate an average of 5.3 candidate matches per point, among which at most one is correct.

Note that in the case where monocular information is to be integrated (sequences of real images), this step is replaced by an intensity-based procedure, such as cross-correlation, that produces candidate matches in the form of a sparse and possibly noisy velocity field.

Figure 4.2(b) shows the candidate matches. Note that the correct matches can be already perceived as they are grouped in two parallel layers surrounded by noisy matches.

4.3 Tensor Encoding

Each potential match is then encoded into a 4-D tensor as follows. The tensor position in the 4-D space is given by the point (x y v_x v_y). Next we need to specify the eigenvalues and eigenvectors. Since no information is initially known, each potential match is encoded into a 4-D *ball tensor* – the eigenvalues and eigenvectors are the following:

$$\begin{array}{ll} \lambda_1=1 & e_1 = (\,0\;0\;0\;1\,)^T \\ \lambda_2=1 & e_2 = (\,0\;0\;1\;0\,)^T \\ \lambda_3=1 & e_3 = (\,0\;1\;0\;0\,)^T \\ \lambda_4=1 & e_4 = (\,1\;0\;0\;0\,)^T \end{array} \tag{4.1}$$

The eigenvectors e_1, e_2, e_3 and e_4 given above do not have any special significance – they now simply define an arbitrary orthonormal basis in the 4-D space. The fact that this tensor represents a pure 4-D ball is given by the encoding of the eigenvalues, which show equal preference for all directions. Note that among the 4-D feature saliency values ($\lambda_1 - \lambda_2$ for volumes, $\lambda_2 - \lambda_3$ for surfaces, $\lambda_3 - \lambda_4$ for curves, and λ_4 for points), the only non-zero saliency value is λ_4 which corresponds precisely to the desired situation, that no particular orientation is initially preferred.

4.4 Affinity Propagation

After encoding, each token propagates its preferred information in a certain neighborhood through a step of voting. This is a sparse process, in the sense that votes are accumulated only at input token locations. The size of the neighborhood where each token casts votes is given by the only parameter of our voting scheme, the scale factor σ, which is an inherent characteristic of human vision.

The affinities propagated are encapsulated in the strength and orientation of the votes cast. Since the tensors involved are ball tensors, only the 4-D *ball voting field* is used. The vote strength decays with distance and with the orientation – in the sense that smooth surface continuations are encouraged. The vote orientation corresponds to the best (smoothest) possible local surface continuation from voter to recipient.

During voting there is strong support between tokens that lie on a smooth surface (layer), while communication between layers is inhibited by the spatial separation between tensors in the 4-D space of both image coordinates and velocities. Wrong matches appear as isolated points that receive little or no support.

The output of this process consists of 4-D generic tensors.

4.5 Selection

The next step is to eliminate the incorrect matches. After propagating mutual token affinities in the voting stage, the wrong candidates have received little support compared to the correct ones, which reinforce each other. A measure of this support is encapsulated into the surface saliency.

For each group of tokens that have common (x y) coordinates but different (v_x v_y) velocities we retain the token with the strongest surface saliency (that is, with the maximum value for λ_2-λ_3), while rejecting the others as outliers.

The output represents the good matches in a sparse configuration. Since votes have been cast only at the input token locations, no new points have been inferred. Therefore, the result is a sparse velocity field.

It is worth mentioning that the entire process described in this chapter not only extracts the correct matches, but also *simultaneously* determines the local orientation of the layers at every token location. Indeed, after voting the first two eigenvectors e_1 and e_2 of the tensor give the orientation the two normals to the locally estimated layer at each tensor location. Later in the process of motion capture (in order to extract the layers), these estimated layer orientations will be refined through another voting step that includes only the correct matches recovered here.

For the translating disk example, a comparison with the ground truth shows that the matching was 100% accurate - all 400 matches have been recovered correctly, despite the large amount of approximately 500% noise present. Figure 4.2(c) shows a 3-D view of the recovered matches, where the height represents the v_x component of the velocity, while Figure 4.2(d) shows the recovered sparse velocity field.

4.6 Results

The case illustrated so far may be considered too simple since the only motion involved is translation. Indeed, image regions undergoing translation will be represented in the 4-D space by planar surfaces parallel to the (x y) plane, because all pixels in a region have the same velocity. However, no assumption – such as translational, planar, or rigid motion – has been made. The *only* criterion used is the smoothness of *image* motion. To support this argument, we show next that our approach also performs very well for several other configurations.

Rotating disk (Figure 4.3). The input consists of two sets of 400 point tokens each, representing an opaque rotating disk (about 7°, counter-clockwise) against a static background. The average number of candidate matches per point is 5.6. Comparing the resulting matches with the true motion shows that only 2 matches among 400 are wrong. Our method still works very well in this case, despite the fact that now the motion layer corresponding to the disk is not a horizontal plane, but a tilted surface.

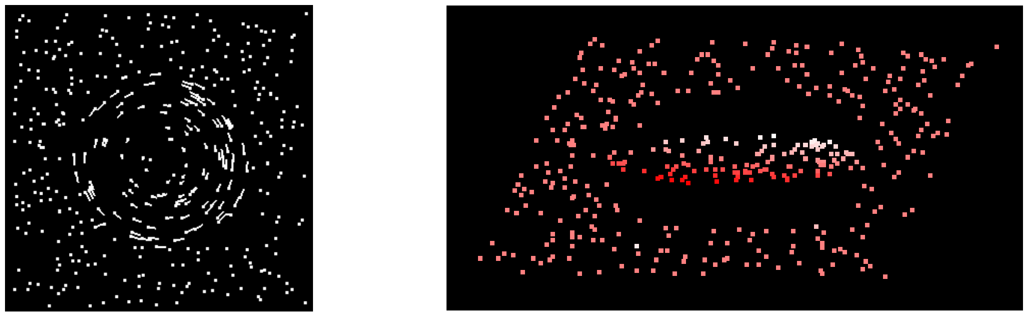


Figure 4.3. Rotating disk

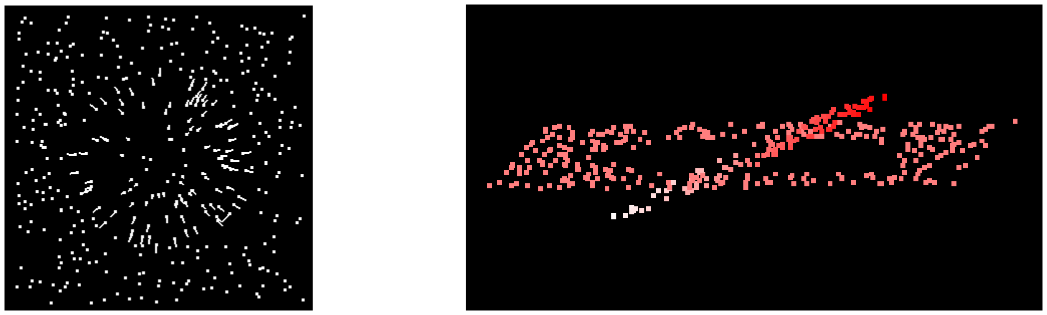


Figure 4.4. Expanding disk

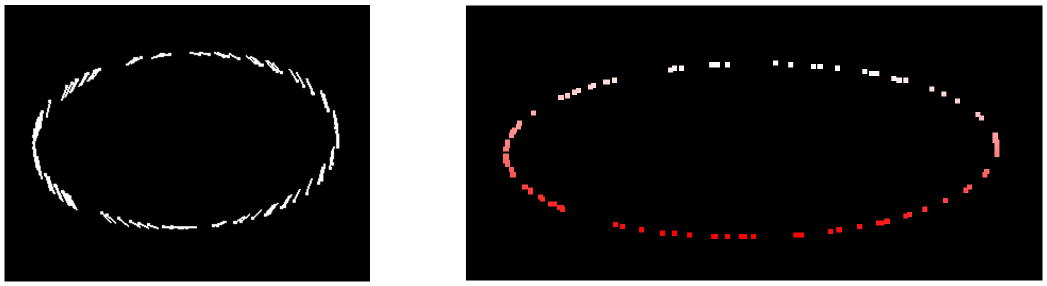


Figure 4.5. Rotating ellipse

Expanding disk (Figure 4.4). The input consists of two sets of 400 point tokens each, representing an opaque disk in expansion against a static background. The average number of candidate matches per point is 6.1. Comparing the resulting matches with the true motion shows that only 1 match among 400 is wrong. This example demonstrates that, without special handling, our framework can easily accommodate both rigid and non-rigid image motions.

Rotating ellipse (Figure 4.5). The input consists of two sets of 100 point tokens each, representing a rotating ellipse. The average number of candidate matches per point is 5.9. Comparing the resulting matches with the true motion shows a 100% accuracy – all the 100 matches have been correctly recovered. Many methods would fail on this example (used in literature to illustrate the aperture effect, and adapted from [25]) – one difficulty is that at the

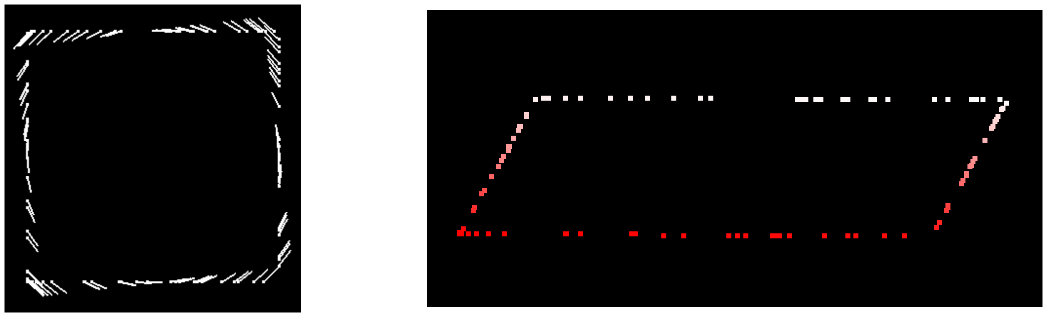


Figure 4.6. Rotating square

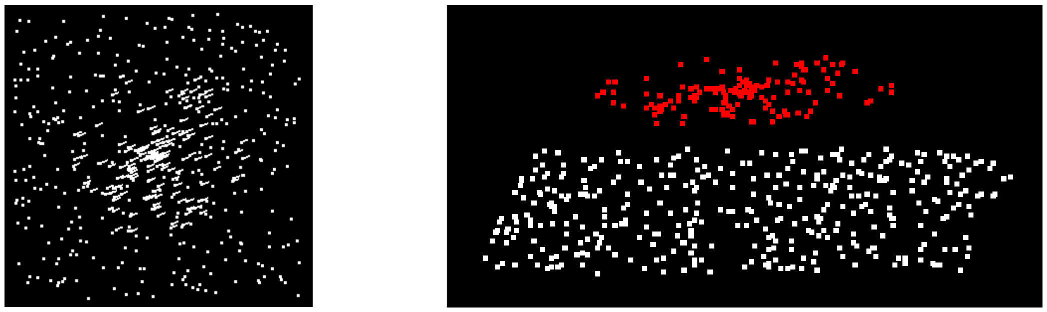


Figure 4.7. Transparent motion

points where the rotated ellipse “intersects” the original one, the velocity could be wrongly estimated as zero.

Rotating square (Figure 4.6). The input consists of two sets of 100 point tokens each, representing a rotating square. The average number of candidate matches per point is 5.7. Comparing the resulting matches with the true motion shows a 100% accuracy – all the 100 matches have been correctly recovered. This example is similar to the rotating ellipse and is used to show that the presence of non-smooth curves does not produce additional difficulty for our methodology.

Transparent motion (Figure 4.7). The input consists of two sets of 500 point tokens each, representing a transparent disk in translation against a static background. The average number of candidate matches per point is 8.9. Comparing the resulting matches with the true motion

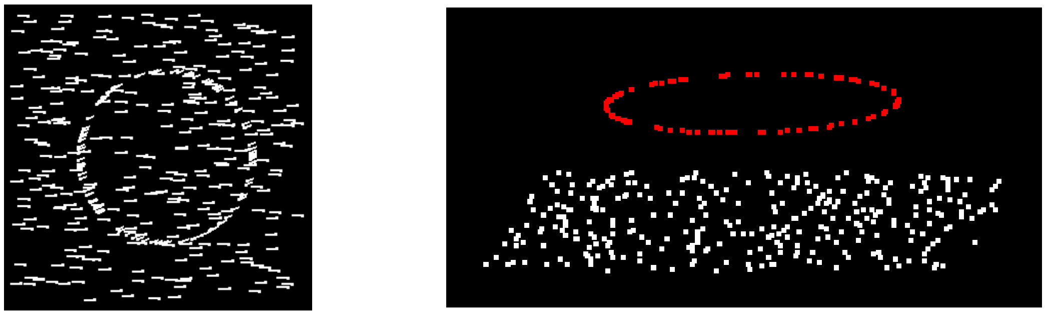


Figure 4.8. Translating circle

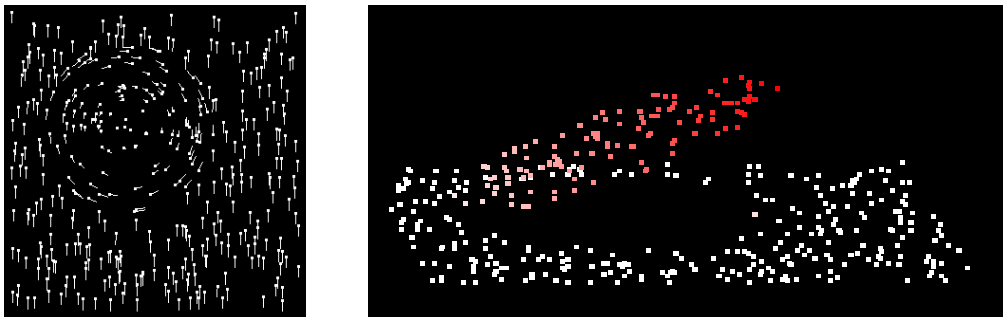


Figure 4.9. Rotating disk – translating background

shows a 100% accuracy – all the 500 matches have been correctly recovered. This example is extremely relevant to illustrate the power of our approach. If the analysis had been performed in a two-dimensional space, most methods would have failed, because the two motion layers are superimposed in 2-D. In our framework, using the 4-D space provides a very natural separation between layers, separation that is consistent with the human perception. In this representation, the process of affinity propagation through voting offers a consistent approach, as the presence of transparent motion does not create any more difficulties than opaque motion.

Translating circle (Figure 4.8). The input consists of two sets of 400 point tokens each, representing a translating circle against a translating background. The average number of candidate matches per point is 6. Comparing the resulting matches with the true motion shows a 100% accuracy – all the 400 matches have been correctly recovered. This example shows that we can successfully handle both curves and surfaces in motion.

Rotating disk – translating background (Figure 4.9). The input consists of two sets of 400 point tokens each, representing an opaque rotating disk (about 7°, counter-clockwise) against a translating background. The average number of candidate matches per point is 5.8. Comparing the resulting matches with the true motion shows that only 2 matches among 400 are wrong. This is a very difficult case even for human vision, due to the fact that at the left extremity of the disk the two motions (of the disk and the background) are almost identical. In that part of the image there are points on different moving objects that are not separated, even in the 4-D space. In spite of this inherent ambiguity, our method is still able to accurately recover correct velocities. The key fact is that the local layer orientations generated through voting are still different from one region to another, and therefore provide a good affinity measure.

Chapter 5

Motion Capture

5.1 Introduction

This chapter describes our approach for the problem of motion capture from two sparse sets of identical point tokens [46]. The process starts with the sparse set of correspondences recovered during the matching step – described in the previous chapter – and is responsible for inferring a scene representation defined by a dense velocity field, motion regions and motion boundaries.

It is interesting to note that in solving the motion capture problem, the proposed framework uses exactly the same token affinity measure that has been used in the matching process. Indeed, in the motion capture process in order to determine that a token belongs to a certain region is equivalent to establishing that it has a stronger affinity to the tokens in that region than to tokens in other regions. It is the same affinity that has been used in matching, in the sense that establishing a correspondence between two tokens implies the fact that their mutual affinity (or preference to each other) is greater than the affinity to other tokens.

Since token affinities here are conceptually the same as in matching, the computational approach that handles them in motion capture is also the same. The motion capture methodology is again based on the 4-D voting framework that has been presented in the

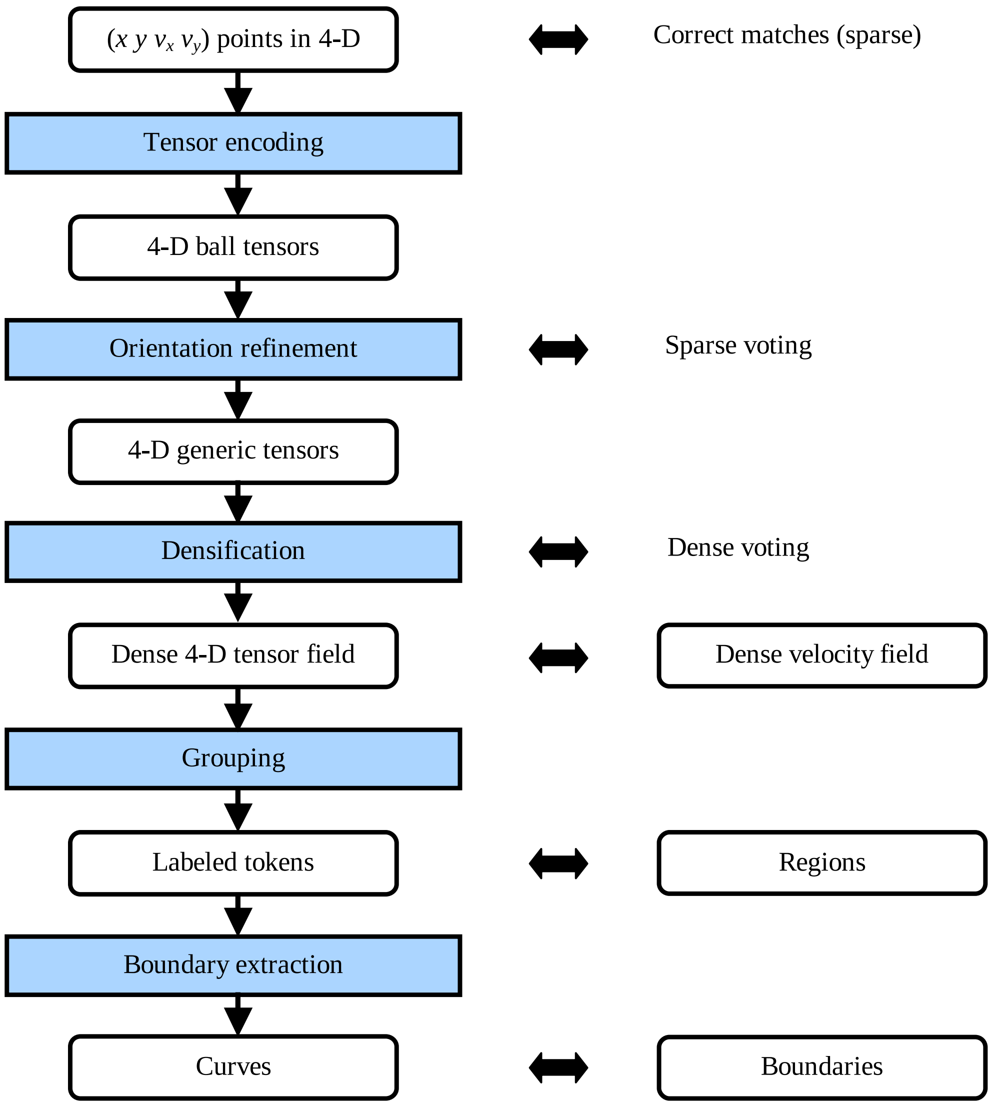
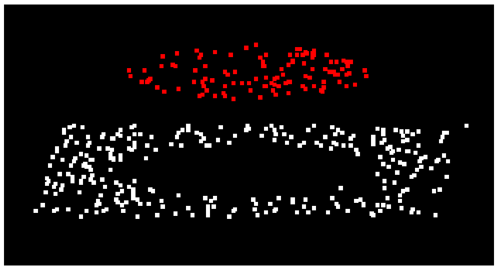
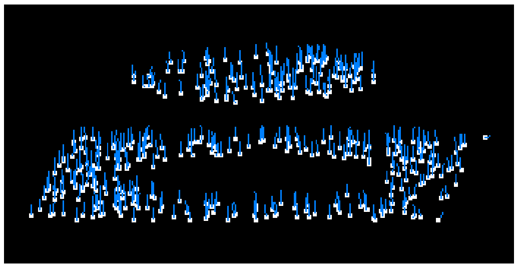


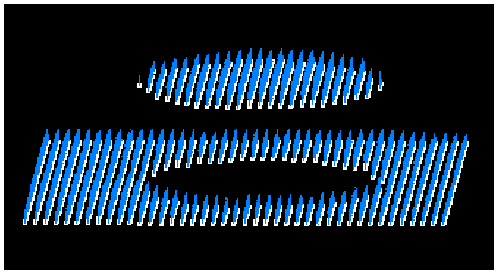
Figure 5.1. The motion capture process

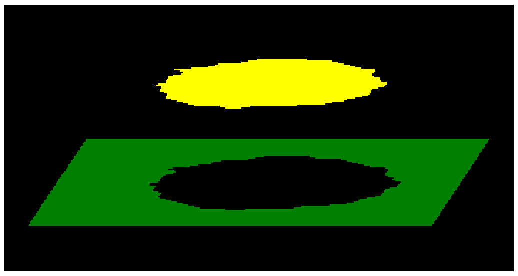
previous chapters. An overview of the processes involved in the various stages of motion capture is shown in Figure 5.1.

The input consists of a sparse velocity field – described by $(x\ y\ v_x\ v_y)$ points in the 4-D space – that has been produced by the matching process. The approach is illustrated with the same


(a) Recovered v_x velocities


(b) Sparse velocity field


(c) Dense velocity field


(d) Regions

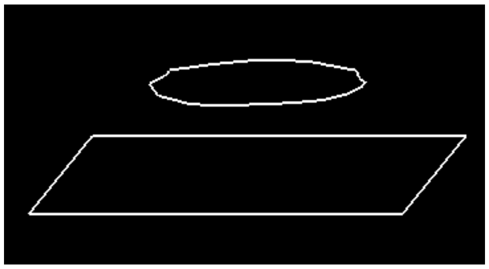

(e) Boundaries

Figure 5.2. Translating disk

specific example – the random point tokens represent an opaque **translating disk** against a static background. Later we also show how the method performs on several other examples.

In the first stage, the 4-D points are encoded as 4-D ball tensors, and a refined description of the layer orientations is obtained through a step of sparse voting. The 4-D tensors resulted after this process are then used in another step of *dense voting*, in order to produce tensors at every image location.

Next, we group the resulted dense set of tokens into layers (regions), based on their velocities and local normal orientations that have been produced at the voting stage. Finally, we extract the motion boundary for each region.

The following sections present each of the processing stages in more detail. Similar to the convention in the previous chapter, for illustration purposes the last component of each 4-D point has been dropped to allow a three-dimensional display. The three dimensions shown are the image coordinates x and y (in the horizontal plane), and the v_x component of the image velocity (the height).

5.2 Tensor Encoding

The input is a set of 4-D points that represent the pixel velocities (matches) recovered in the matching process, as shown in Figure 5.2(a). At this stage we need to obtain an estimation of the layer orientations as accurate as possible. Although local layer orientations have already been determined as a by-product during the matching process (after voting, the eigenvectors e_1 and e_2 represent the normals to layers), they may have been corrupted by the presence of wrong correspondences.

Therefore, we perform an orientation refinement through another sparse voting process, but this time with the correct matches only. To this purpose, every 4-D point is again encoded into a 4-D *ball tensor*.

5.3 Orientation Refinement

As explained above, the goal is to get an accurate estimation of the layer orientation at every token location. The token affinities are propagated again, as in the step of matching, but this time without the corrupting influence of wrong matches.

This process is performed through a step of sparse voting, very similar to the one involved in the matching stage. However, now we are interested in both the saliency values as a measure of affinity, and also in the layer orientations that result after voting. In Chapter 3 we have shown that the voting process simultaneously generates information regarding all geometric varieties (curves, surfaces, volumes, etc.) that exist in the chosen space (in this case 4-D). Because in this process we are looking for the layer orientations – and the layers are surfaces embedded in the 4-D space – the desired orientations (as normals to layers) are found at each token after voting as the first two eigenvectors e_1 and e_2. We remind the reader that in a 4-D space, a surface is characterized by *two* normal vectors.

In figure 5.2(b) we show a 3-D view of the tokens with refined layer orientations. Obviously, only one of the normal vectors is shown at each token.

5.4 Densification

At this stage we have determined the accurate layer orientations at each token location. In order to attain the very goal of the motion capture problem – that is, to recover boundaries and region as *continuous* curves and surfaces, respectively – it is necessary to first infer velocities and layer orientations at every location in the image.

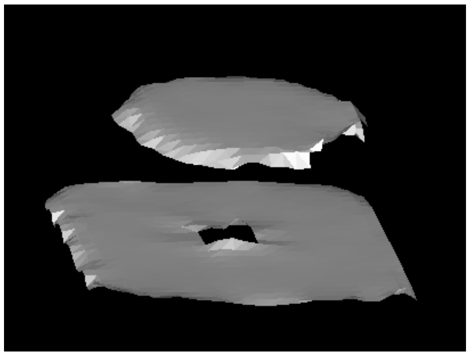


Figure 5.3. Layer over-extension

The previously developed Tensor Voting framework allows for a densification procedure that extracts geometric features such as curves and surfaces from sparse data. However, the way in which this process is conducted is not appropriate for the particular problem addressed in this work. The existing densification algorithm proceeds by choosing the most salient tensor from the sparse set of tokens, and then growing the surface or curve around it, in a manner similar to a marching process. The curve or surface orientation and saliency are estimated at all neighbors of the current token (in a discrete grid) through voting. They are then analyzed with sub-voxel precision in order to determine how the curve or surface crosses the current grid cell (square in 2-D, or cube in 3-D). This procedure is then repeated for the neighboring cell grids that correspond to the direction in which the curve or surface extends. Chapter 3 provides a more detailed description of this process.

The aspect that makes this approach undesirable for the current problem is that it is still not clear when to stop growing. If, for example, a closed surface such as a sphere is to be extracted, the results will be very good. On the other hand, if we need to extract an open surface such as a plane, the resulted surface will be *over-extended*. The reason is that the existing densification process grows the surface until the saliency drops below a certain level,

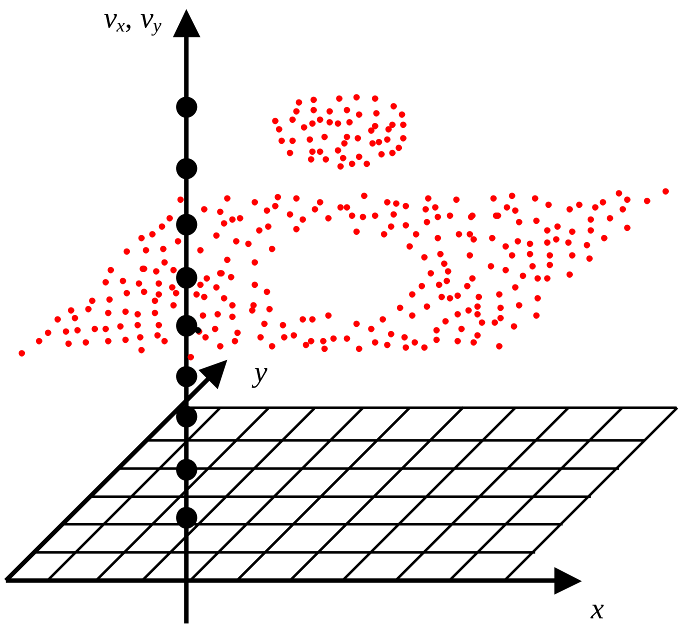


Figure 5.4. Densification

due to the decay with the distance from the supporting tokens. A result of applying this procedure to our translating disk example is shown in Figure 5.3.

Since in addressing the motion capture problem it is crucial to obtain accurate motion boundaries, such an approach is not appropriate. Therefore we devised a different densification scheme that addresses the specific constraints of our problem – to extract dense layers while maintaining the motion boundaries.

The key of our approach lies in the fact that the 4-D space we use is *not isotropic*. We need to obtain a tensor value at every (x y) location in the image, but certainly *not* at every (v_x v_y) location in the velocity space.

Our densification method is illustrated in Figure 5.4. For each pixel (x y) in the image we try to find the best (v_x v_y) location at which to place a newly generated token. The candidates considered are all the discrete points (v_x v_y) between the minimum and maximum velocity values present in the sparse token configuration. To improve the speed, we actually consider only the minimum and maximum velocities from the sparse tokens within a neighborhood of the (x y) point. At each candidate position (x y v_x v_y) we accumulate votes from the sparse tokens, according to the same Tensor Voting framework that has been used so far. After voting, the candidate token whose surface saliency (λ_2-λ_3) is maximal is retained, and its (v_x v_y) coordinates represent the most likely velocity at (x y). By following this procedure at every (x y) image location, we generate a *dense velocity field*.

The densification process described above uses the same token affinities that have been employed in the previous stages, which are encapsulated in the voting framework. The best velocities at each image location are determined by trying different candidates, based on their affinity to the existing, sparse tokens. The token affinity is again measured through the surface saliency, by accumulating support for smooth surface orientations.

The output at this stage is the *dense velocity field*, the first of the three goals that have been mentioned in Chapter 1, when the problem of visual motion analysis has been formulated. Note that in this process, along with velocities (given by the last two coordinates in the 4-D space), we simultaneously infer layer orientations (given by the first two eigenvectors e_1 and e_2 as normals to the layer). Figure 5.2(c) shows a 3-D view of the dense set of tokens that have been generated, including their associated layer orientations.

Although the 4-D representation allows for the presence of overlapping layers, this procedure precludes us from obtaining multiple velocity values at each image location, such as in the case of transparent motion. A potential solution would be to choose not just the most salient velocity candidate, but all the candidates whose saliency values are locally maximal.

However, as it is not very clear how this choice would affect the motion boundaries, such strategy still needs to be further investigated.

5.5 Grouping

After obtaining a dense velocity field, the next step is to group tokens into regions that correspond to distinct moving objects, as perceived by the human vision. A main advantage of our approach is that we have already inferred *both* velocities and layer orientations at each image location.

The smoothness criterion is used again to assign tokens to regions. We start from an arbitrary point in the image, assign a region label to it, and try to recursively propagate this label to all its image neighbors. In order to decide whether the label must be propagated to a neighbor (that is, whether the neighbor and the current point should be placed in the same region), we use the smoothness of both velocity and layer orientation as a grouping criterion. Having both pieces of information available is especially helpful in situations where neighboring pixels have very similar velocities, and yet they must belong to different regions. Most methods that are based only on velocity discontinuities would fail on these cases. We will show such an example later.

The output consists of region-labeled tokens at each image location, which means that we have recovered the motion *regions*, the third of our goals in the problem of motion analysis. After assigning region labels to every token, for illustration purposes we perform a triangularization of each of the regions detected. The resulted surfaces are presented in Figure 5.2(d).

5.6 Boundary Extraction

It may have been noticed that the "upper" surface shown in Figure 5.2(d) does not exactly correspond to a disk. This is a normal outcome, and it is due to the fact that the input set of points was sparse. The irregularity of the surface boundary became apparent in the densification process, and it depends on the density distribution of the input points. Moreover, in terms of the human visual system, this boundary irregularity is more or less perceived according to the scale factor. Indeed, on a small scale (viewing the scene from a small distance) the boundary will appear very "rugged". By increasing the scale factor (viewing from a larger distance), the boundary will be perceived as less and less irregular, until at some point it will appear as totally convex. In fact, at that moment the region boundary is perceived as the *convex hull* of all the sparse input points in the region.

Therefore we have implemented a method to extract the boundary for each region as a "partially convex hull". The process is controlled by only one parameter – the scale factor – which determines the degree of irregularity – in other words, the departure from the actual

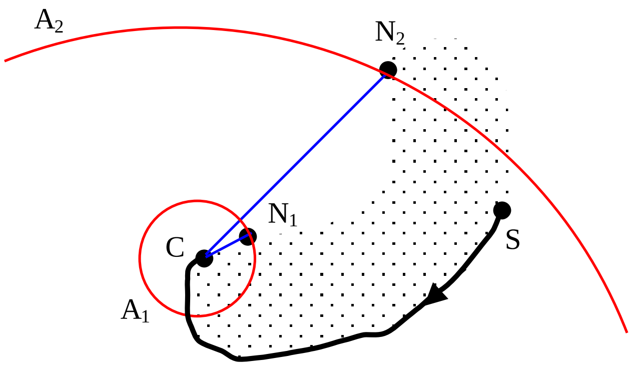
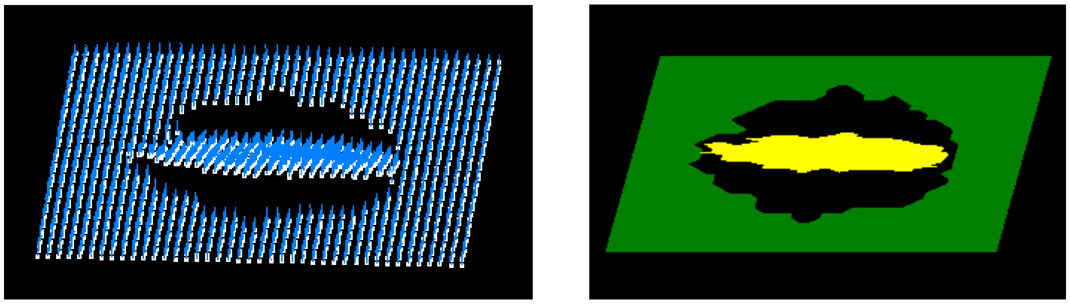


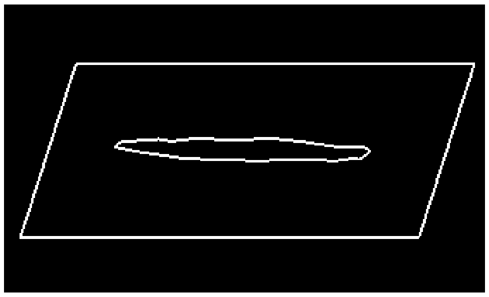
Figure 5.5. Boundary extraction

convex hull. The entire algorithm is performed in the 2-D image space, by using the *x* and *y* coordinates of each token.

The approach is illustrated in Figure 5.5. We start at some arbitrary point *S* on the boundary – for example, the point with the largest *x* coordinate in the region. From there the boundary curve is grown so that at every current point *C*, the curve is *locally convex*. For the current point *C*, the next boundary point N_1 is chosen so that all the points within the neighborhood A_1 are inside (to the right of) the boundary found so far, including the segment CN_1. The size of this neighborhood is given by the scale factor, which determines the perceived level of detail. If the scale factor has a larger value – corresponding to the neighborhood A_2 – then the next point selected from *C* will be N_2, and the resulting boundary will be closer to the true convex hull.

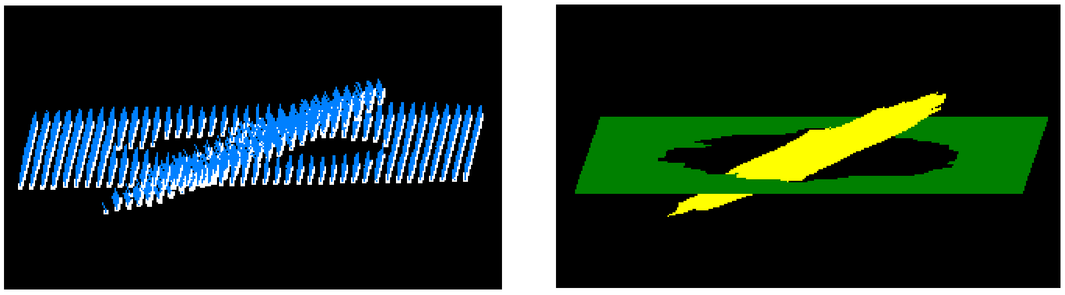


(a) Dense velocity field (b) Regions



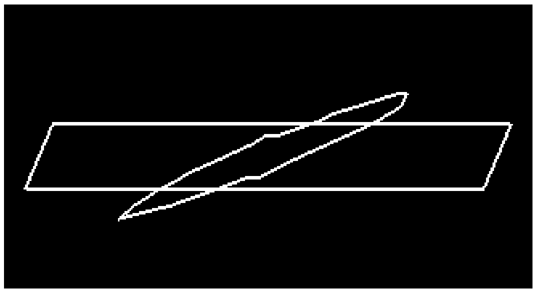
(c) Boundaries

Figure 5.6. Rotating disk



(a) Dense velocity field

(b) Regions



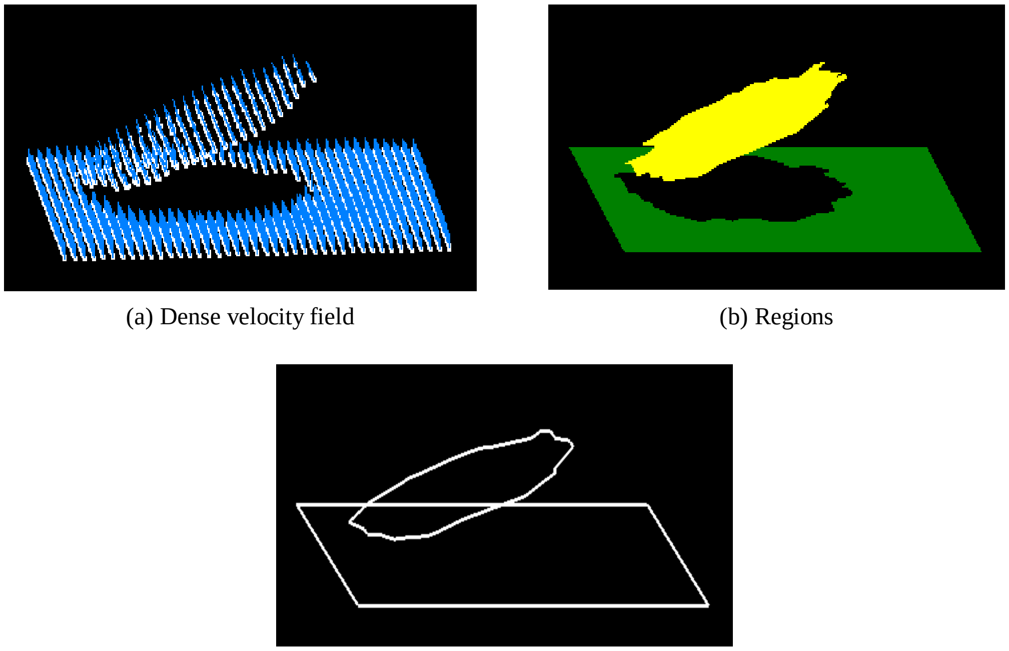
(c) Boundaries

Figure 5.7. Expanding disk

At this point we have determined the *motion boundaries*, the last remaining goal of the three that we enumerated for the problem of motion analysis. The resulted boundary curves for the current example of the translating disk are presented in Figure 5.2(e).

5.7 Results

We conclude this chapter by presenting our results for several other configurations, that illustrate how our approach performs for different classes of sparse inputs, when only motion cues are available. The examples presented below have also been used in Chapter 4 when addressing the matching problem – therefore, a description of their input will not be repeated here.



(a) Dense velocity field

(b) Regions

(c) Boundaries

Figure 5.8. Rotating disk – translating background

For each example we show the dense velocity field as 4-D point tokens, the detected regions as continuous surfaces, and the extracted motion boundaries as continuous curves.

Rotating disk – Figure 5.6.

Expanding disk – Figure 5.7.

Rotating disk – translating background – Figure 5.8.

The last example is the most difficult one and deserves a few comments. As it was mentioned in Chapter 4, this example is problematic even for the human perception, because at the left extremity of the disk the two motions (of the disk and the background) are almost identical. In that part of the image there are points on different moving objects that are not separated even in the 4-D space. In spite of this inherent ambiguity, our method is still able to accurately

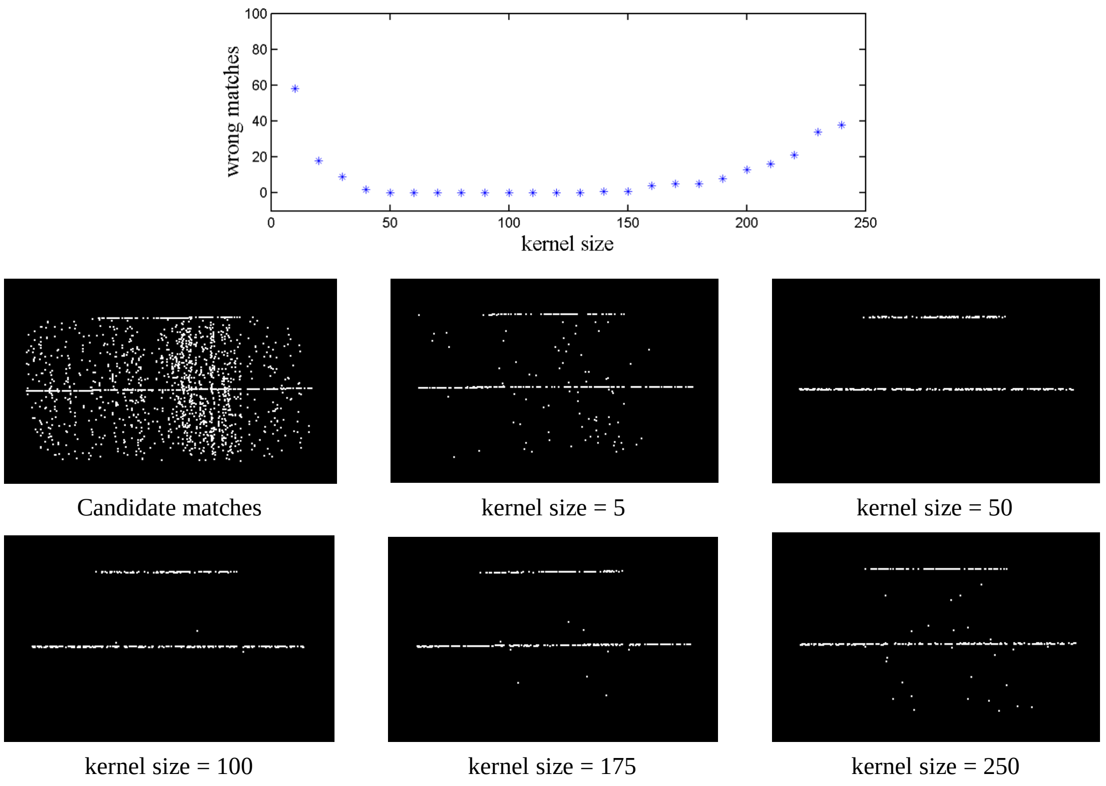
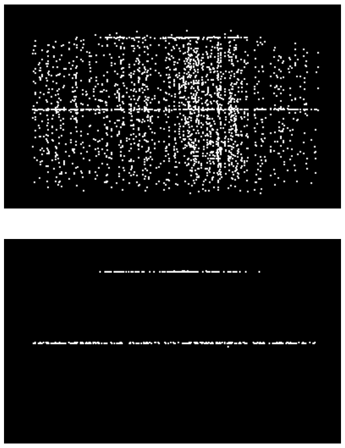


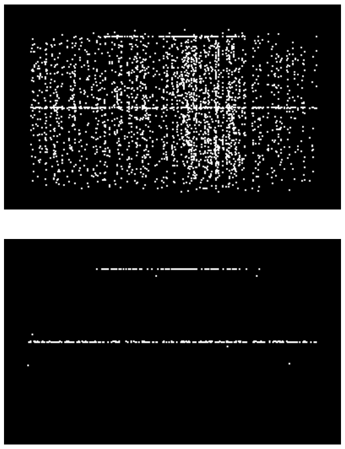
Figure 5.9. Scale factor influence

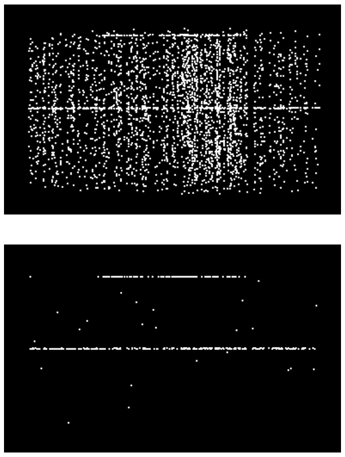
recover the regions and boundaries. The key fact is that in grouping we rely not only on the smoothness of recovered velocities, but also on the smoothness of local layer orientations that are determined simultaneously at every token location.

5.8 Scale Sensitivity

Since the only parameter involved in our voting framework is the scale factor that defines the voting fields (kernels), we analyzed how it influences the quality of the analysis. We ran our algorithm on the translating disk example for a large range of scale values and we found that the method is remarkably robust to varying scale factors. Figure 5.9 shows the number of


noise n = 2


noise n = 3


noise n = 4

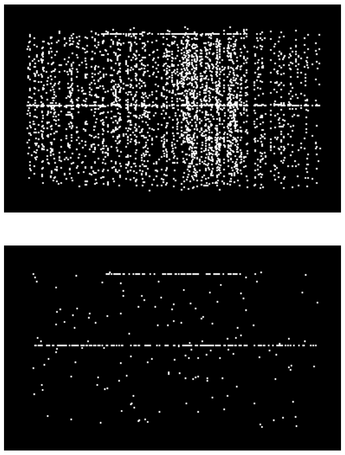

noise n = 5

Figure 5.10. Noise influence

incorrect matches (for an input of 400 points) obtained for different values of the voting field size (in pixels). Comparatively, the image size is 200 by 200.

When the field is too small (kernel size < 30), the input tokens fail to communicate to each other, and the performance starts to degrade abruptly. At the other end (kernel size > 200), the degradation is more graceful, even for voting fields that are larger than the image size. Note the broad range of applicable field sizes, between 50 and 175.

5.9 Noise Sensitivity

To demonstrate the robustness of our method, we have conducted an experiment in which various amounts of noise are added to the data set. The input frames contain 400 point tokens, that correspond to the translating disk sequence. For this experiment, we added $400n$ points randomly selected in the image space, to the data set. At each step, n is incremented by one – see Figure 5.10. It is interesting to note that the performance is very robust to large amounts of noise, and it only starts to fail when $n \geq 5$.

Chapter 6

Integrating Monocular Cues

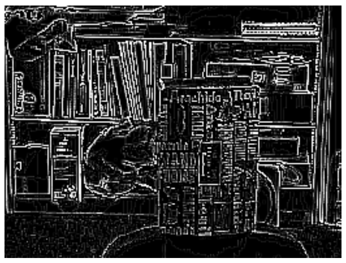
6.1 Introduction

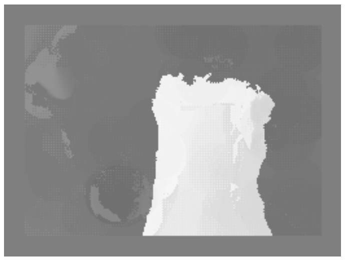
So far we have only presented cases where no monocular information (such as intensity) is available, and the entire analysis has been performed based on motion cues *only*. Human vision is able to handle these cases remarkably well, and their study is fundamental for understanding the motion analysis process. Nevertheless they are very difficult from a computational perspective – most existing methods cannot handle such examples in a consistent and unified manner, or without relying on unrealistic assumptions about the motion.

In the case of real image sequences, in addition to motion cues, there is a wealth of monocular information that can be used in the inference of salient structures. The human vision system is able to decouple the two sources of information, as the analysis is possible from each of them separately. However, when both monocular and motion cues are available, they seem to be used in conjunction, although the integration process is yet unclear.

From a computational point of view, we need to decide how the two types of information should be combined, and in what order. If monocular cues are to be used first, such as in Figure 6.1(b), which shows the intensity edges detected in Figure 6.1(a), it would be very difficult to process such raw information. On the other hand, if motion cues are used, it is much easier to perceive the two salient regions (the box and the background), based on their

(a) One input image


(b) Intensity edges


(c) Velocities

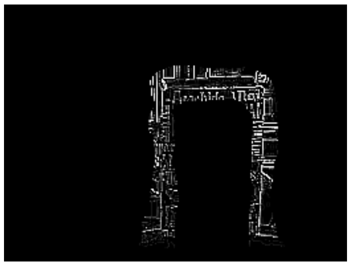

(d) Local boundary refinement

Figure 6.1. Combining motion and monocular cues

different image velocities, as shown by the velocity map in Figure 6.1(c). Although the boundaries are still incorrect, the general position and shape of the regions are detected, and can be *locally* refined by using monocular cues, as intensity edges from original images (Figure 6.1(d)).

This chapter describes the additional difficulties induced by the case of real image sequences, an overview of the proposed method, the extensions to the framework in order to address these difficulties, and the experimental results that have been obtained.

In order to incorporate monocular information into our framework, we need to address a number of problems specific to the case when the input consists of a sequence of intensity images. In particular, the issues to be handled in this context are:

- generation of candidate matches
- rejection of outliers due to image areas lacking texture

- inference of accurate motion boundaries in the presence of occlusion

The pre-processing step where candidate matches are generated is replaced by an intensity-based cross-correlation procedure, where all peaks of correlation are retained as candidate matches. Furthermore, the procedure is repeated for several sizes of the correlation window, in order to capture information at multiple scales.

In areas lacking texture, it is very likely that all matching candidates are incorrect, as there is no reliable intensity information that can be used by the correlation process. In the selection step described in Chapter 4, when only candidates with maximal saliency are retained at each pixel, it is possible that the best candidate is still incorrect. Therefore, an additional step of outlier rejection is employed, where all tokens that received very little support during voting are eliminated.

Producing an accurate motion flow field is very difficult at motion boundaries. The motivation of the motion segmentation problem stems from the fact that motion regions (pixels with similar motion) usually correspond to distinct objects in the scene. Computationally, the problem is addressed by first establishing pixel correspondences between images in order to obtain velocity values at each image location. Based on their velocities, pixels are then grouped into motion regions, separated by motion boundaries, thus producing a segmentation of the image.

However, an inherent difficulty in this process is caused by the presence of the motion boundaries themselves. The very source of information used for segmentation – pixel velocities – are mostly unreliable exactly at the motion boundaries, where the segmentation takes place. The example in Figure 6.2, showing a truck moving from left to right over a static

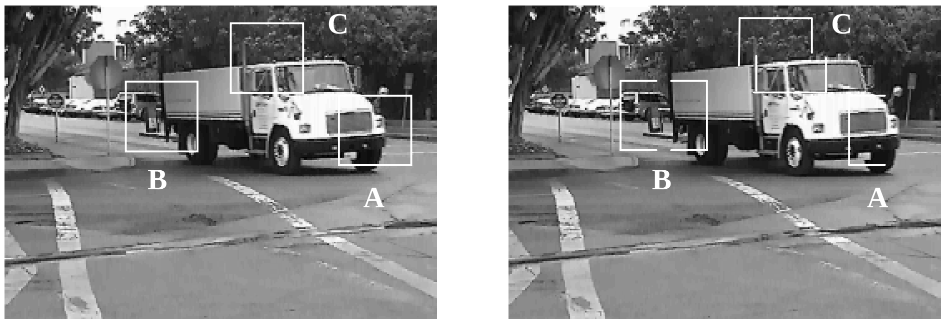


Figure 6.2. Non-similarity at motion boundaries

background, is used to illustrate the problem. From area A that appears in the first image, only half is visible in the second image, the other half being occluded by the moving region. At the opposite side, area B is still visible in the second image, but is now split into two regions, with new, un-occluded pixels in between. Even where no occlusion takes place, such as at the upper boundary, area C is also split in the second image, due to the motion between regions.

Consequently, the apparent motion around boundaries cannot be precisely determined by using any similarity criteria, since the areas being compared must have a finite extent. Moreover, it is not realistic to assume that all the wrong matches can be later removed as noise. Due to the similarity of partial areas, wrong correspondences are often assigned in a consistent manner, resulting in over-extended image regions.

The key observation is that one should not rely on *motion cues* only, in order to perform motion segmentation. Examining the original images reveals a multitude of *monocular cues*, such as intensity edges, that can aid in identifying the true object boundaries. A second glance at Figure 6.2 will confirm it.

In this context, we formulate the problem of motion analysis as a two-component process, that:

- enforces the smoothness of motion, except at its discontinuities
- enforces the smoothness of such discontinuities, aided by monocular cues

Here we present the extensions to our 4-D framework in order to handle real image data, and integrate it with a 2-D voting-based method for accurate inference of motion boundaries [49][47]. The extended approach we developed for the case of monocular cues is based on two voting processes, in different dimensional spaces. First, motion layers as extracted as surfaces in a 4-D space, by using a voting process to enforce the smoothness of motion and determine an estimation of pixel velocities, motion regions and boundaries. This 4-D process is carried out according to the same voting framework that has been presented in the previous chapters. Although noisy correspondences are rejected as outliers after extracting the motion layers in 4-D, there are also wrong matches that are consistent with the correct ones. This mostly occurs at the motion boundaries, where the occluding layer is typically over-extended towards the occluded area.

The remaining stage is to infer the correct motion boundary by adding monocular information from the original images. First we define zones of boundary uncertainty along the margins of layers. Within these zones we create a 2-D saliency map that combines the following information: the position and overall orientation of the layer boundary, and the strength and orientation of the intensity edges from the original images. Then we enforce the smoothness and continuity of the boundary through a 2-D voting process, and extract the true boundaries

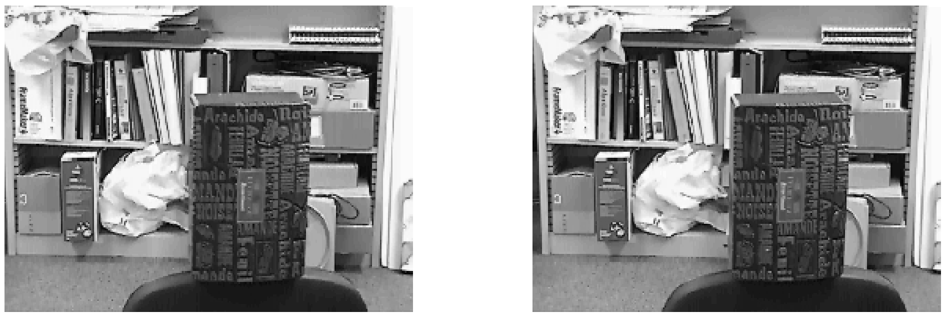


Figure 6.3. Candy box sequence – input images

as the most *salient curves* within these zones. Finally, correct velocities are computed for the pixels near boundaries, as they are reassigned to the appropriate regions.

6.2 Establishing Initial Correspondences

The input consists of two image frames that involve general motion – that is, both the camera and the objects in the scene may be moving. For illustration purposes, we give a description of the proposed approach by using a specific example, the **candy box sequence** – the two images in Figure 6.3 are taken with a handheld moving camera, where the candy box and the background exhibit distinct motions due to their different distances from the camera.

For every pixel in the first image, the goal at this stage is to produce candidate matches in the second image. We use a normalized cross-correlation procedure, where all peaks of correlation are retained as candidates. When a peak is found, its position is also adjusted for sub-pixel precision according to the correlation values of its neighbors. Finally, each candidate match is represented as a (x,y,v_x,v_y) point in the 4-D space of image coordinates and pixel velocities, with respect to the first image.

Since it is desirable to increase the likelihood of including the correct match among the candidates, we repeat this process at multiple scales, by using different correlation window sizes. Small windows have the advantage of capturing fine detail, and are effective close to the motion boundaries, but produce considerable noise in areas lacking texture or having small repetitive patterns. Larger windows generate smoother matches, but their performance degrades in large areas along motion boundaries. We have experimented with a large range of window sizes, and found that best results are obtained by using only two or three different sizes, that should include at least a very small one. Therefore, in all the examples described in this paper we used three correlation windows, with 3x3, 5x5 and 7x7 sizes.

The resulting candidates appear as a cloud of (x,y,v_x,v_y) points in the 4-D space. Figure 6.4 shows the candidate matches. In order to display 4-D data, the last component of each 4-D point has been dropped – the 3 dimensions shown are x and y (in the horizontal plane), and v_x (the height). The motion layers can be already perceived as their tokens are grouped in two layers surrounded by noisy matches.

Extracting statistically salient structures from such noisy data is very difficult for most existing methods. Because our voting framework is robust to considerable amounts of noise, we can afford using the multiple window sizes in order to extract the motion layers.

6.3 Extraction of Motion Layers in 4-D

The process of extracting the motion layers is very similar to that used in the case of sparse input data, when only motion cues are available, and it has been described in detail in the previous chapters. Here we only give a succinct description of the main steps involved,

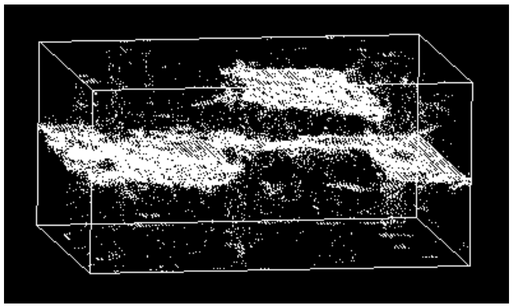


Figure 6.4. Candidate matches

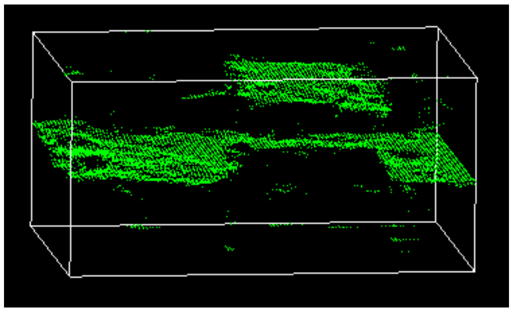


Figure 6.5. Selected velocities

including the extensions that have been made in order to address the problem of motion analysis in the case of intensity images.

Selection. Since no information is initially known, each potential match is encoded into a 4-D *ball tensor*. Then each token casts votes by using the 4-D *ball voting field*. During voting there is strong support between tokens that lie on a smooth surface (layer), while communication between layers is reduced by the spatial separation in the 4-D space of both image coordinates and pixel velocities. For each pixel (x y) we retain the candidate match with the highest surface saliency (λ_2-λ_3), and we eliminate the others as they represent incorrect matches. Figure 6.5 shows a 3-D view of the recovered matches (the height represents v_x).

Orientation refinement. In order to obtain an estimation of the layer orientations as accurate as possible, we perform an orientation refinement through another voting process, but now with the selected matches only. After voting, the normals to layers are found at each token as the first two eigenvectors e_1 and e_2.

Outlier rejection. In the selection step, we retained only the most salient candidate at each pixel. However, there are pixels where all candidates are wrong, such as in areas lacking texture. Therefore now we eliminate all tokens that have received very little support. Typically

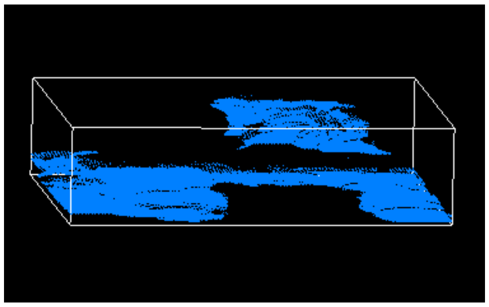


Figure 6.6. Dense layers

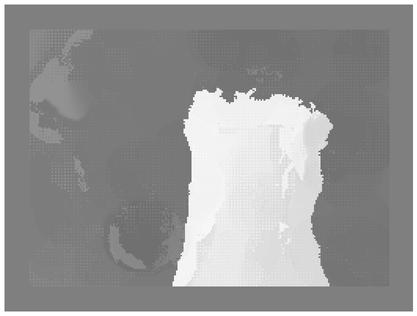


Figure 6.7. Layer velocities

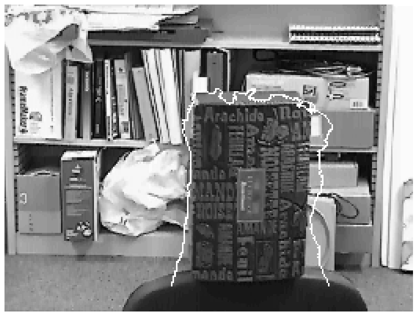


Figure 6.8. Layer boundaries

we reject all tokens with surface saliency less that 10% of the average saliency of the entire set.

Densification. Since the previous step created "holes" (i.e., pixels where no velocity is available), we must infer their velocity from the neighbors by using a smoothness constraint. For each pixel (x y) without an assigned velocity we try to find the best (v_x v_y) location at which to place a newly generated token. The candidates considered are all the discrete points (v_x v_y) between the minimum and maximum velocities in the set, within a neighborhood of the (x y) point. At each candidate position (x y v_x v_y) we accumulate votes, according to the same Tensor Voting framework that we have used so far. After voting, the candidate token with maximal surface saliency (λ_2-λ_3) is retained, and its (v_x v_y) coordinate represents the most likely velocity at (x y). By following this procedure at every (x y) image location we generate a *dense velocity field*. In this process, along with velocities we also simultaneously infer layer orientations. A 3-D view of the dense layers is shown in Figure 6.6.

Grouping. The next step is to group tokens into *regions*, by using again the smoothness constraint. We start from an arbitrary point in the image, assign a region label to it, and try to recursively propagate this label to all its image neighbors. In order to decide whether the label must be propagated, we use the smoothness of both velocity and layer orientation as a grouping criterion. Figure 6.7 illustrates the recovered v_x velocities within layers (dark corresponds to low velocity), and Figure 6.8 shows the layer boundaries superimposed over the first image.

6.4 Boundary Inference in 2-D

At this stage, the extracted motion layers can still be over or under-extended along the motion boundaries. This situation typically occurs in areas subject to occlusion, where the initial correlation procedure may generate wrong matches that are consistent with the correct ones, and therefore could not be rejected as outlier noise.

However, now it is known how many moving objects are present in the scene and where they are. The margins of the layers provide a good estimate for the position and overall orientation of the true motion boundaries. We combine this knowledge with monocular cues (intensity edges) from the original images in order to build a boundary saliency map along the layers margins. Next we enforce the smoothness and continuity of the boundary through a 2-D voting process, and extract the true boundary as the most salient curve within the map.

This procedure is performed in two successive passes – by separately using the horizontal and vertical components of the image gradient. In fact, during the first pass all edges are found, with the exception of the ones “perfectly” horizontal. The second pass is actually used to only detect the remaining edges. Note that the two steps are inter-changeable, and their order is not important.

6.4.1 The Boundary Saliency Map

In the first pass, we start by finding the points that belong to the layer boundaries, identified by changes in region labels along horizontal lines. For each such point (x_c y_c) we define a horizontal *zone of boundary uncertainty*, centered at (x_c y_c). Since the over or under-extension of motion layers is usually within the limits of the correlation window size, we chose the largest size used in correlation as the zone width. The zone height is one pixel.

Next we make use of the monocular cues by computing the image gradient (from the intensity *I* in first image) at each location within the zones of boundary uncertainty:

$$\begin{aligned} G_x(x,y) &= I(x,y) - I(x-1,y) \\ G_y(x,y) &= I(x,y) - I(x,y-1) \end{aligned} \tag{6.1}$$

Since at this pass we are looking for non-horizontal edges, we initialize our saliency map with the horizontal component of the gradient:

$$sal = \left|G_x(x,y)\right| \tag{6.2}$$

This choice is made in order not to be influenced in the analysis by purely horizontal edges, which will be detected during the second pass. Diagonal edges that exhibit a significant horizontal gradient contribute to the saliency map and they are detected in the first pass.

Finally, we incorporate our estimation of the boundary position and orientation, as resulted from motion cues, by introducing a bias towards the current layer boundaries. Within each zone, we define a weight function *W* that is 1 at x_c and decays exponentially by:

$$W = e^{-\frac{(x-x_c)^2}{\sigma_W^2}} \qquad (6.3)$$

where σ_W corresponds to a weight of 0.2 at the zone extremities.

The saliency map is then updated by multiplying each existing value with the corresponding weight.

6.4.2 Detecting the Boundary

At this stage we have a saliency value and an orientation at each location within the zones of uncertainty. However, in order to extract the boundaries we need to examine how neighboring locations agree upon their information, through a voting process.

We proceed by encapsulating all the existing information within a 2-D tensor framework. Since we have boundary orientations, at each location in the uncertainty zones we create a 2-D stick tensor, with the orientation (eigenvectors e_1 and e_2) given by the image gradient, and the size taken from the saliency map:

$$\begin{array}{ll} e_1 = (G_x\ G_y) & \text{(normal to edge)} \\ e_2 = (-G_y\ G_x) & \text{(tangent to edge)} \\ \lambda_1 = sal & \\ \lambda_2 = 0 & \end{array} \qquad (6.4)$$

Next, the tensors communicate through a 2-D voting process, where each tensor is influenced by the saliency and orientation of its neighbors. After voting, the curve saliency values are

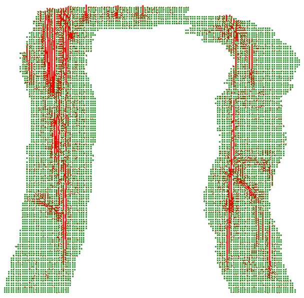
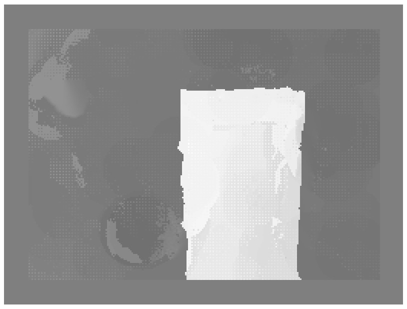


Figure 6.9. Boundary saliency map



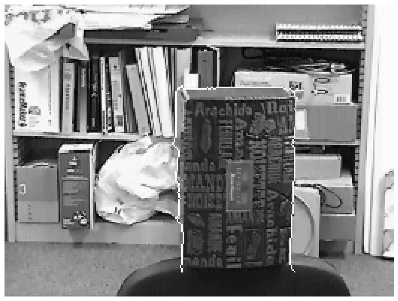


Figure 6.10. Refined velocities

Figure 6.11. Refined boundaries

collected at each tensor as (λ_1-λ_2) and stored back in the saliency map. Figure 6.9 shows the tensors after voting, with the local curve tangent given by the eigenvector e_2. The curve saliency (λ_1-λ_2) is illustrated here as the length of the tangent vector. Note that although strong texture edges are present in the uncertainty zone, after voting their saliency has been diminished by the overall dominance of saliency and orientation of the correct object edges.

The true boundaries are extracted by choosing seeds with maximum curve saliency, and growing the boundary from an uncertainty zone to the next, according to the local curve saliency and orientation.

After marking the detected boundaries, the entire process is repeated in a similar fashion in the second pass, this time using the vertical component of the gradient, in order to detect any horizontal boundaries that have been missed during the first pass.

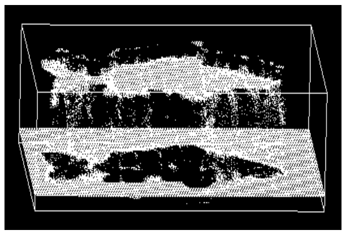
Finally, each zone of boundary uncertainty is revisited in order to reassign pixels to regions, according to the new boundaries. In addition to changing the region label, their velocities are recomputed in a 4-D voting process similar to the one used for densification. However, since region labels are now available, the votes are collected only from points within the same layer.

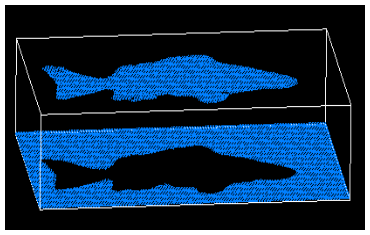
Figure 6.10 shows the refined velocities within layers (dark represents small velocity), and Figure 6.11 shows the refined motion boundaries, that indeed correspond to the actual object.

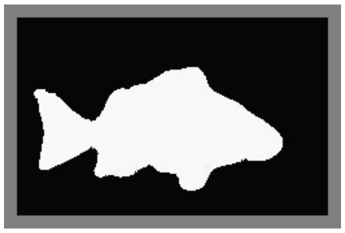
6.5 Results

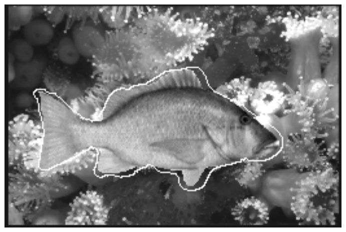
We have also analyzed several other image sequences, and we present here the results obtained. In all experiments we used three correlation windows, with 3x3, 5x5 and 7x7 sizes, and for each window we retained all peaks of correlation. Therefore each pixel in the image had at least 3 candidate matches, among which at most one was correct. For both the 4-D and 2-D voting processes, in all examples we used the same scale factor, corresponding to an image neighborhood with a radius of 16 pixels.

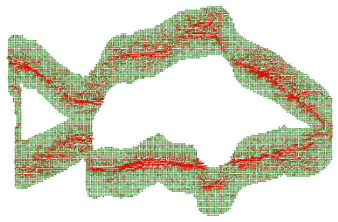
(a) One input image

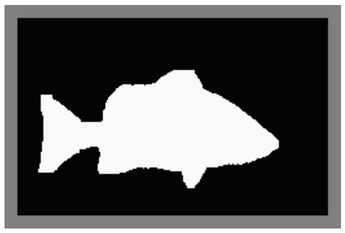

(b) Candidate matches


(c) Dense layers


(d) Layer velocities


(e) Layer boundaries


(f) Boundary saliency map


(g) Refined layers

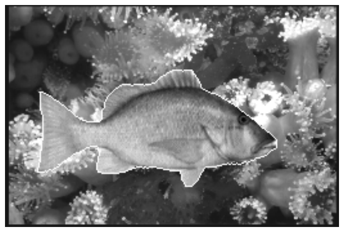
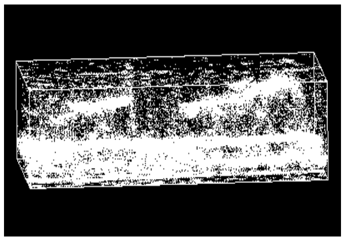
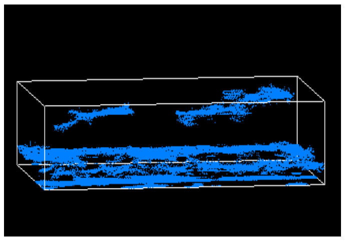

(h) Refined boundaries

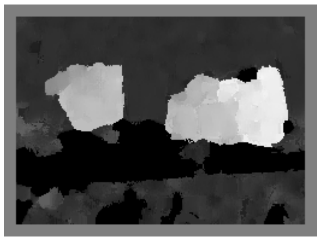
Figure 6.12. Fish sequence

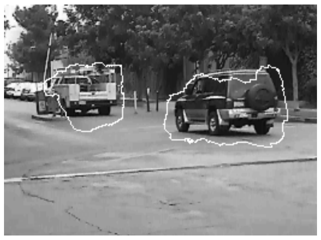
Fish sequence (Figure 6.12). To quantitatively estimate the performance of our approach we created a synthetic sequence from real images. The silhouette of a fish was cropped from its image and inserted at different locations over a background image, in order to generate a motion sequence with ground truth. The average angular error we obtained is 0.42° ± 1.2° for 100% field coverage, which is very low despite the multitude of texture edges from the cluttered background, that were competing with the true object edges. This example is also used to show that we can successfully handle more detailed and non-convex motion boundaries.

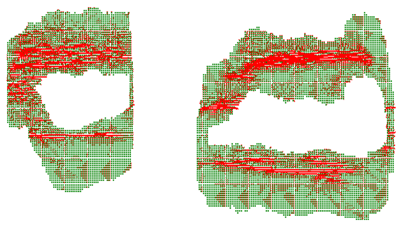
(a) One input image

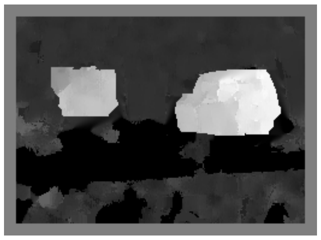

(b) Candidate matches


(c) Dense layers


(d) Layer velocities


(e) Layer boundaries


(f) Boundary saliency map


(g) Refined layers

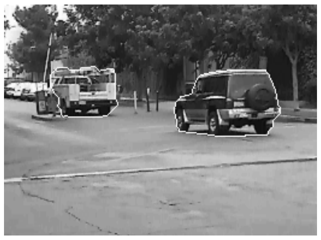
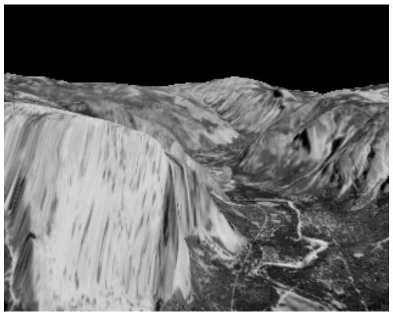
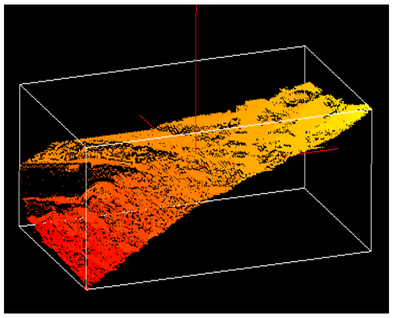
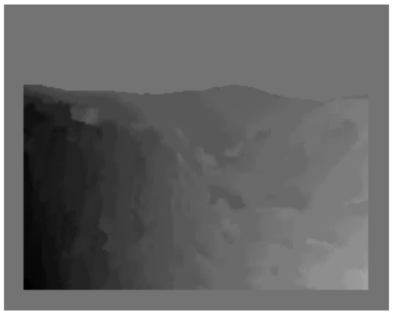

(h) Refined boundaries

Figure 6.13. Barrier sequence

Barrier sequence (Figure 6.13). We analyzed the motion from two frames of a sequence showing two cars moving away from the camera. The analysis is difficult due to the large ground area with very low texture, and because the two moving objects have relatively small sizes in the image. Also note that the *image* motion is not translational – the front of each car has a lower velocity than its back. This is visible in the 3-D view of the motion layers, which appear as tilted surfaces. In fact, our framework does not make any assumption regarding the type of motion – such as translational, planar, or rigid motion – the *only* criterion being used is the smoothness of *image* motion.


(a) One input image


(b) Dense layer


(c) Velocities v_x

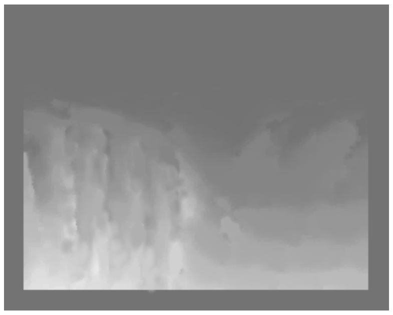

(d) Velocities v_y

Figure 6.14. Yosemite sequence

Yosemite sequence (Figure 6.14). We analyzed the motion from two frames of the Yosemite sequence (without the sky), to further provide a quantitative estimate for the performance of our approach, as compared to other methods. Although this is an artificial fly-through sequence, it uses real images as texture for the valley model. The average angular error obtained by using the described approach is 3.74° ± 4.3° for 100% field coverage. Table 6.1 compares our results with the performance of other methods, as reproduced from [5]. In terms of average angular error, the proposed method significantly outperforms all others, which are also applied on 100% image coverage. Some methods obtain slightly better performance, but only when they restrict the coverage to a lower percent of the image. Also note that most

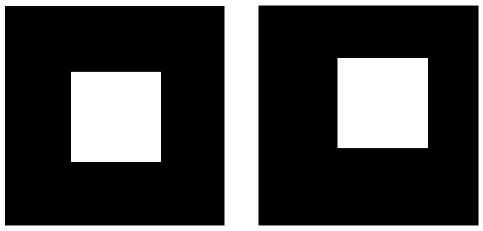
Table 6.1. Yosemite results

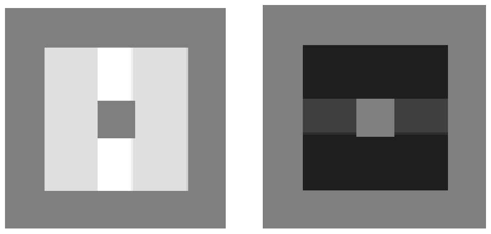
Technique	Average error	Standard deviation	Density
Nicolescu and Medioni	3.74°	4.3°	100%
Anandan	15.54°	13.46°	100%
Uras et al. (unthresholded)	16.45°	21.02°	100%
Horn and Schunck	22.58°	19.73°	100%
Lucas and Kanade ($\lambda_2 \geq 5.0$)	3.55°	7.11°	8.8%
Uras et al. ($det(H) \geq 2.0$)	3.75°	3.44°	6.1%
Fleet and Jepson ($\tau = 2.5$)	4.29°	11.24°	34.1%
Fleet and Jepson ($\tau = 1.25$)	4.95°	12.39°	30.6%
Lucas and Kanade ($\lambda_2 \geq 1.0$)	5.20°	9.45°	35.1%
Uras et al. ($det(H) \geq 1.0$)	5.97°	11.74°	23.4%
Heeger	11.74°	19.0°	44.8%

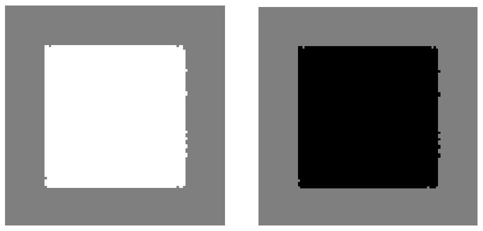
methods listed here require more than two frames for the analysis, while our method uses two frames only.

Uniform square sequence (Figure 6.15). This example, showing a uniform square in translation against a static uniform background, is used to illustrate the main difficulties in motion analysis, and how they are addressed by our method. The experiment is very difficult, as both image regions lack any texture, and the aperture effect is present at each location around the intensity edges.

Figure 6.15(b) shows two intensity maps corresponding to the candidate velocities, separately for the v_x and v_y components. As only the corners provide texture for the correlation process, note the uncertainty around the middle part of the edges, due to the aperture effect. Since there is no possibility to determine the motion of the static uniform background, we assigned zero


(a) Input images


(b) Candidate velocities


(c) Layer velocities

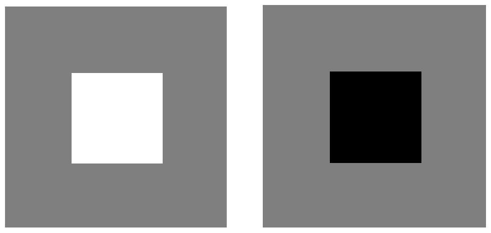

(d) Refined velocities

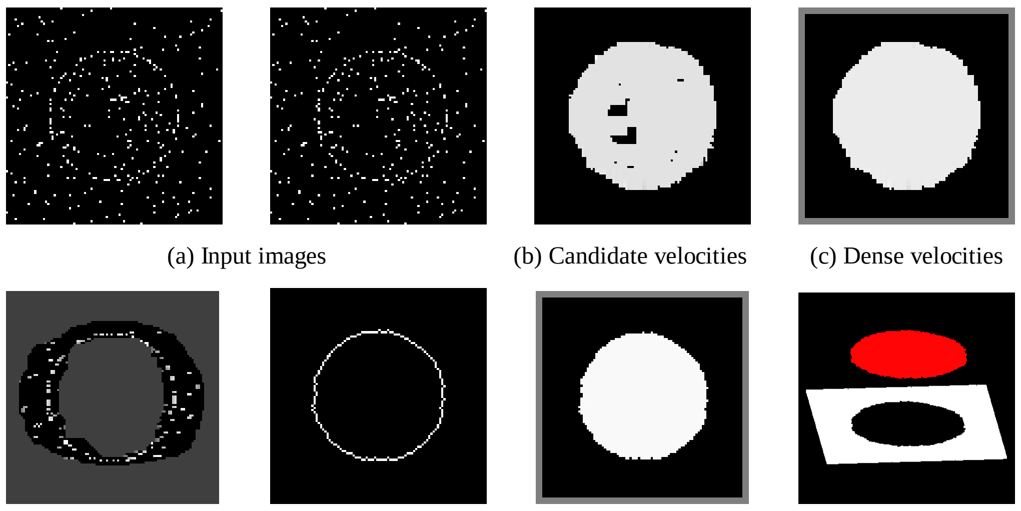
Figure 6.15. Uniform square sequence

motion to all pixels where no texture is found during correlation. This is the reason for the incorrect motion that is reported in the interior of the square.

Figure 6.15(c) shows the layer velocities after the sparse and dense voting. The uncertainty due to the aperture effect has been eliminated, and the interior pixels of the square have been assigned correct velocities during densification.

The remaining problem is that the extracted layers are over-extended (with half the correlation window size), as the only texture used in correlation is represented by the intensity edges. However, the correct layers are inferred by augmenting the motion information with the intensity edges – the refined velocities are shown in Figure 6.15(d).

Sparse disk and boundary sequence (Figure 6.16). In the previous chapters we have shown that our framework is able to perform the analysis based on monocular cues only, when the input consists of sparse data. The extended framework, described in this chapter, handles


(a) Input images (b) Candidate velocities (c) Dense velocities

(d) Boundary gradient (e) Refined boundary (f) Refined velocities (g) Refined layers

Figure 6.16. Sparse disk and boundary sequence

motion augmented with monocular cues, as in real image sequences. Here we show that the extended framework is also able to analyze motion and monocular cues in the case of sparse input data. The example in Figure 6.16 illustrates a sparse input corresponding to a disk in translation against a static background. The monocular cues are represented by the higher density of points on the disk boundary, creating the salient perception of a circle.

As expected, the recovered motion layers (Figure 6.16(c)) are over-extended due to occlusion. However, as the input points from the correct boundary form a salient structure, the true boundary (Figure 6.16(e)) is inferred by the 2-D voting process, and the refined layers indeed correspond to the correct perception of the translating disk, as shown in Figure 6.16(g).

6.6 Handling Transparent Motion

The human visual system can easily distinguish multiple motions that are transparently combined in an image sequence. However, traditional computational models of the motion perception process have largely been confused by scenes with multiple motions. Real world examples of these viewing conditions are common – for example, people looking out of a window often see both the outside world and a reflection of the objects inside the room. As any driver has observed, the human visual system can perform accurate navigation functions even when a large percentage of the image signal is obscured by a corrupting, but coherently organized noise process, caused by rain or snow.

Transparent segregation can be performed on an image sequence that contains moving point tokens, even when a static display of the tokens does not support such segregation. Ullman [64] and Mulligan [41] have shown that human observers could easily segregate two coherently moving sets of point tokens that were unseparable in static presentation. For both image patterns and sparse point tokens stimuli, there is a perception of continuous surfaces corresponding to what was used in the physical or synthetic construction of the stimulus.

Several computational approaches have been developed in order to address the perception of transparent or multi-component motion, from 3 or more image frames. The algorithm of Shizawa and Mase [53][54] computes two velocity vectors for each location in the image, by using an energy integral minimization as a model fitting method, but does not address the problem of perceptual grouping of coherently moving regions of the scene. Other methods [6][28] compute global affine optical flow fields, but use local measurements that are only capable of determining a single velocity estimate at each point.

The technique presented by Irani and Peleg [28] assumes a spatially dominant background, whose parameters can be estimated based on the entire image data, since the outlier contamination from the foreground will be relatively small. The background estimate can be further refined using an iterative robust technique, re-estimating the parameters based only on the points with low residual error. The motion of a foreground object is then estimated based only on the complement of the background support. With multiple objects, these recursive estimate-and-segment approaches will fail when objects exist at the same scale. In this case, the percentage of outliers in the signal, relative to the estimation of either object will exceed the breaking point of the robust estimation method.

Darrell and Simoncelli [12] describe a hypothesize-and-test method, which assumes a prior model of the global motion. Hypotheses are generated by sampling the parameter space, or by fitting initial guesses to samples of data. Other methods [6][58] propose iterative methods that recover multiple motions in the presence of reflections and transparency. However, their techniques are restricted by the use of parametric motion models, such as translational or affine.

A major benefit of our 4-D framework for motion analysis is that it allows for explicit representation of overlapping motion layers, and for affinity propagation within each layer, while inhibiting propagation across layers. Therefore, it can successfully handle images containing reflections and transparency, as the interaction between tokens still takes place within each smooth motion layer, as in the case of opaque motion. The limitation (also shared by most other methods) is that we need to know how many overlapping layers are present in the scene. Such knowledge is needed only for the selection of the most salient velocity candidates, as opposed to the case of opaque motion, where only the most salient one needs to be retained.

In our approach for the analysis of transparent motion we consider the image $I(x,y,t)$ at time t as a combination of two patterns A and B, which have independent motions a and b:

$$I(x,y,t) = A^{ta} + B^{tb} \tag{6.5}$$

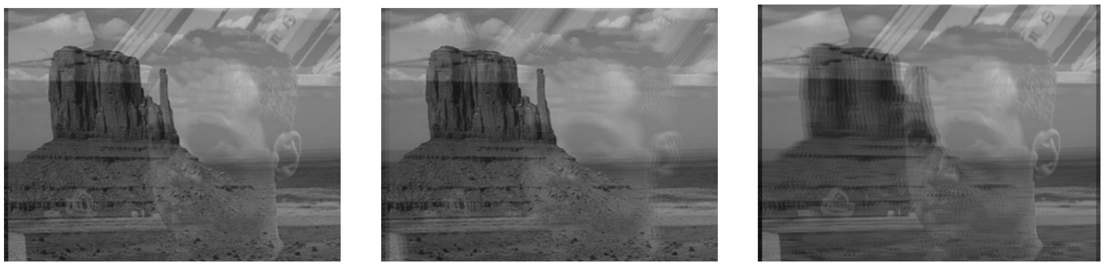
In this equation, A^{ta} denotes pattern A transformed by motion ta. In order to obtain the dominant motion (assume it is a), we run a cross-correlation procedure, followed by a step of voting as described in the Matching section, to eliminate noisy matches. Next we use a "nulling" method [7][58], to estimate the remaining motion b. The pattern component A with velocity a is removed from the sequence by moving each frame with a, then subtracting it from the following frame. The resulting difference images are:

$$\begin{aligned} D_k &= I(x,y,k+1) - I^a(x,y,k) \\ &= (A^{(k+1)a} + B^{(k+1)b}) - (A^{(k+1)a} + B^{kb+a}) \\ &= (B^b - B^a)^{kb} \end{aligned} \tag{6.6}$$

Assuming that we have three frames, the difference images are $D_0 = (B^b - B^a)$ and $D_1 = (B^b - B^a)^b$, which show a pattern $(B^b - B^a)$ moving with a single motion b. We use the same method – cross-correlation followed by voting – to determine motion b from frames D_0 and D_1.

Finally, we put together the two sets of 4-D tokens with velocities a and b, and run a step of dense voting and grouping (as described in the Motion Capture section) on the entire set. This process also fills any holes in the layers, which may have been produced by the noisy matches elimination. Note that the entire procedure recovers the motions without separating the image patterns.

Transparent motion sequence (Figure 6.17). We analyzed the motion from three frames captured with a moving camera, showing a face reflected in a framed picture. One of the input frames is illustrated in Figure 6.17(a). In order to show the accuracy of our results, we compute two "temporal average" images after registering the input frames using the two



(a) An input frame (b) Registered background (c) Registered foreground

Fig. 6.17. Transparent motion sequence

recovered motions, as shown in Figure 6.17(b) and Figure 6.17(c). In each of these, the registered pattern is sharp, while the other one is blurred due to the image motion.

Chapter 7

Three-Dimensional Interpretation

7.1 Introduction

The goal of our approach for visual motion interpretation is the following: given two views of a scene containing general motion, to recover the 3-D structure and 3-D motion of each coherently moving region in the scene. The human vision system is able to make an interpretation according to two alternative processes. One case corresponds to inferring 3-D structure from image motion, by interpreting unrecognized objects in motion. The second case is a process of inferring motion from structure, which uses previously recognized 3-D structure in order to derive a motion interpretation. In this work we focus on the first process, that infers 3-D structure and 3-D motion only from changes in image appearance.

The difficulties at this stage are caused by the inherent ambiguity of the 2-D to 3-D interpretation, which can be handled by adding additional constraints, such as rigidity. From a computational point of view, a problem usually encountered is the combined presence of noise and multiple independent motions, even non-rigid motions. In this context it is very difficult to enforce a global constraint, as it is not clear how to handle misfits, which may correspond to outliers, non-rigid, or independent motion.

The case when the analysis is performed on two image frames and the scene is assumed to be static corresponds to stereo vision. In this context, the assumption of a static scene and a moving camera represents an instance of the 3-D rigidity constraint. In particular, rigidity is used in stereo vision by enforcing the epipolar constraint on the set of point correspondences, in order to obtain the fundamental matrix that describes the rigid configuration of the camera system or, equivalently, the camera motion. As the pixel disparities between the two frames are assumed to have been produced by a single motion only (of the camera), the epipolar constraint is globally enforced on the entire set of correspondences.

In order to estimate the scene geometry and motion from a set of point correspondences, it is often needed to explicitly handle the presence of false matches and independently moving regions. Given a pair of image frames, most methods first obtain the matching points, which are then filtered by an outlier rejection step before they are used to solve for epipolar geometry and for 3-D structure estimation. In the presence of moving objects, image registration becomes a more challenging problem, as the matching and registration phases become interdependent.

The problem of recovering the epipolar geometry and 3-D scene structure has been intensively studied and it is considered well understood. Given two views of a static scene, a set of matching points – typically corresponding to salient image features – are first obtained by methods such as cross-correlation. Assuming that matches are perfect, a simple Eight Point Algorithm [35] can be used for estimating the fundamental matrix, and thus the epipolar geometry of the cameras is determined. A dense set of matches can be then established, aided by the epipolar constraint, and finally the scene structure is recovered through triangulation.

The simplistic approach described above performs reasonably well only in the case when the following conditions are met:

- the set of matches must contain no outlier noise
- the scene must be rigid – i.e., without objects having independent motions

However, the first assumption almost never holds, since image measurements are bound to be imperfect, and matching techniques will never produce accurate correspondences, mainly due to occlusion or lack of texture. In the presence of incorrect matches, linear methods, such as the Eight Point Algorithm, are very likely to fail. The problem can be reliably solved by robust methods, which involve non-linear optimization [1][68], and normalization of data before fundamental matrix estimation [20].

If the second assumption is also violated by the presence of multiple independent motions, even the robust methods may become unstable, as the scene is no longer a static one. Depending on the size and number of the moving regions, these techniques may return a totally incorrect fundamental matrix. Furthermore, even if the dominant epipolar geometry is recovered (for example, the one corresponding to the static background), motion correspondences are discarded as outliers.

The core inadequacy of most existing methods is that they attempt to enforce a *global* constraint – such as the epipolar one – on a data set which may include, in addition to noise, independent subsets that are subject to separate constraints. In this context, it is indeed very difficult to recover structure from motion and segment the scene into independently moving regions, if the two tasks are performed simultaneously.

In order to address these difficulties, we have developed a novel approach that decouples the above operations, allowing for explicit and separate handling of matching, outlier rejection, grouping, and recovery of camera and scene structure. In the first step, we determine an accurate representation in terms of dense velocities (equivalent to point correspondences), segmented motion regions and boundaries, by using only the smoothness of image motion. In the second step we proceed with the extraction of scene and camera 3-D geometry, separately on each rigid component of the scene. Note that our approach follows Ullman's interpretation of visual motion [64], in that the correspondence process is a low-level process, which takes place prior to 3-D interpretation. However, we also perform segmentation before 3-D interpretation, based on smoothness of image motion only.

The main advantage of our approach is that, at the interpretation stage, noisy matches have been already rejected, and matches have been grouped according to the distinctly moving regions present in the scene. Therefore, standard methods can be applied in order to *locally* enforce the rigidity constraint for each segmented data subset.

The computer vision literature offers a multitude of techniques for the estimation of epipolar geometry, scene structure and camera motion. Linear methods, such as the Eight Point Algorithm [35] can be used for accurate estimation of the fundamental matrix, in the absence of noisy matches or moving objects. The algorithm recovers the essential/fundamental matrix from two calibrated/uncalibrated images, by solving a system of linear equations. A minimum of eight points is needed – if more are available, a least mean square minimization is used. To ensure that the resulting matrix satisfies the rank two requirement, its singularity is usually enforced [20].

In order to handle outlier noise, more complex, non-linear iterative optimization methods are proposed [68]. These techniques use objective functions, such as distance between points and corresponding epipolar lines, or gradient-weighted epipolar errors, to guide the optimization process. Despite their increased robustness, iterative optimization methods in general require somewhat careful initialization for early convergence to the correct optimum. One of the most successful algorithms in this class is LMedS [68], which uses the least median of squares and data sub-sampling to discard outliers by solving a non-linear minimization problem.

RANSAC [63] consists of random sampling of a minimum subset with seven pairs of matching points for parameter estimation. The candidate subset that maximizes the number of inliers and minimizes the residual is the solution. Statistical measures are used to derive the minimum number of sample subsets. Although LMedS and RANSAC are considered to be some of the most robust methods, it is worth noting that these techniques still require a majority of the data to be correct, or else some statistical assumption is needed. If false matches and independent motions exist, these methods may fail or become less attractive, since in the latter case, many matching points on the moving objects are discarded as outliers.

In [50], Pritchett and Zisserman propose the use of local planar homographies, generated by Gaussian pyramid techniques. However, the homography assumption does not generally apply to the entire image.

7.2 Overview of the Method

The first step of the proposed method – illustrated in Figure 7.1 – formulates the motion analysis problem as an inference of motion layers from a noisy and possibly sparse point set in a 4-D space. In order to compute a dense set of matches (equivalent to a velocity field) and to segment the image into motion regions, we use the voting-based computational framework described in the previous chapters. By letting the tokens communicate their mutual affinity

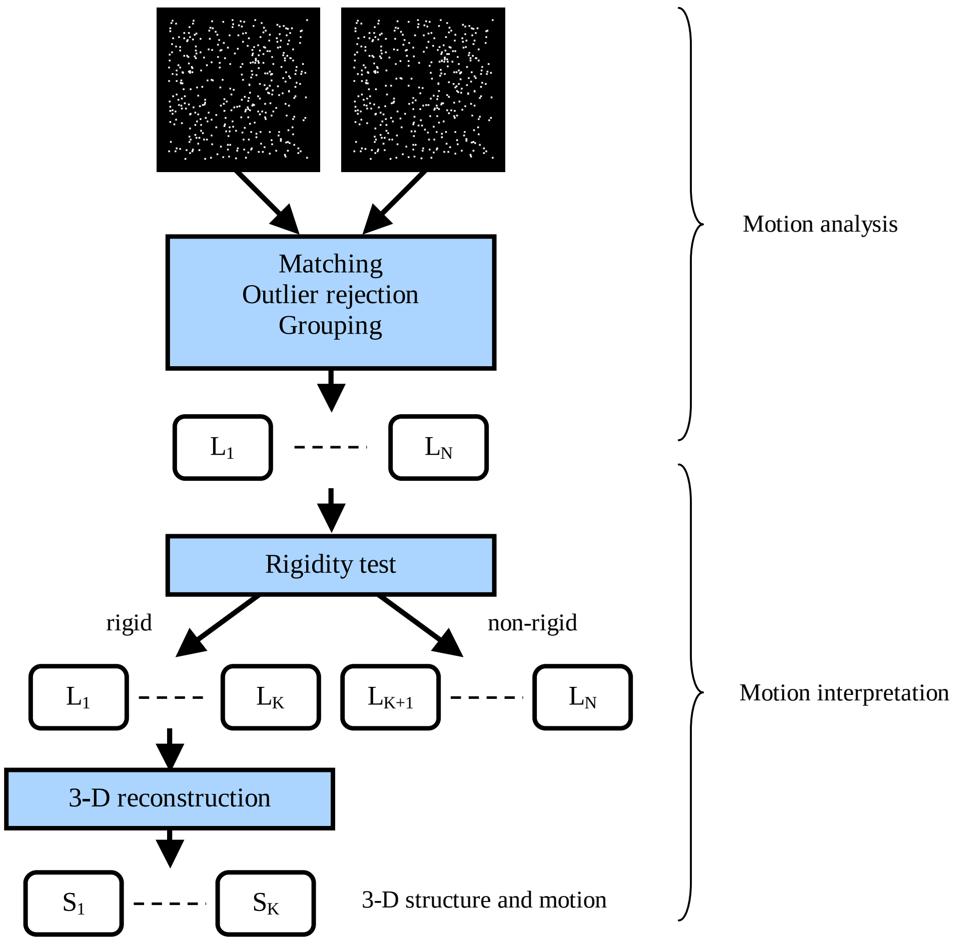


Figure 7.1. Overall view of the approach

through voting, noisy matches are eliminated as they receive little support, and distinct moving regions are extracted as smooth, *salient surface layers*, in the 4-D space.

The second step interprets the image motion by estimating the 3-D scene structure and camera geometry. First a rigidity test is performed on the matches within each region, to identify potential non-rigid (deforming) regions, and also between regions, to merge those that move rigidly together but have separate image motions due to depth discontinuities.

Finally, the epipolar geometry is estimated separately for each rigid component by using standard methods for parameter estimation (such as the normalized Eight Point Algorithm, LMedS or RANSAC), and the scene structure and camera/object motion are recovered by using the dense velocity field.

7.3 The Rigidity Constraint

So far we have not made any assumption regarding the 3-D motion, and the only constraint used has been the smoothness of image motion. The observed image motion could have been produced by the 3-D motion of objects in the scene, or the camera motion, or both. Furthermore, some of the objects may undergo non-rigid motion.

For classification we use an algorithm introduced by McReynolds and Lowe [39], that verifies the potential rigidity of a set of minimum six point correspondences from two views under perspective projection. The rigidity test is performed on a subset of matches within each region, to identify potential non-rigid regions, and also across regions, to merge those that move rigidly together but have distinct image motions due to depth discontinuities.

The following discussion describes each case, as identified by the rigidity test.

Multiple rigid motions. This case occurs when multiple moving regions have been identified by the layer extraction process, and the rigidity test shows that:

- each region moves rigidly
- the regions cannot be merged into a single rigid structure

Consequently, the image sequence cannot correspond to a static scene viewed with a moving camera, as multiple independent motions are present.

Single rigid motion. This is the stereo case, where the scene is static and the camera is moving. However, note that it is still possible to have multiple layers extracted in the stage of grouping from motion, if their image motions do not satisfy the smoothness constraint together, due to depth discontinuities. In this case, the rigidity test will show that the regions actually correspond to a single rigid configuration in motion, and therefore can be labeled as a single object.

Non-rigid motion. This case occurs when objects suffer non-rigid 3-D motion (deformations). The rigidity test will detect that the image motion, while still smooth, is not compatible to any rigid 3-D motion. Therefore, the configuration is recognized as non-rigid, and no reconstruction is attempted.

It is also worth mentioning that the rigidity test is actually able to only guarantee the *non-rigidity* of a given configuration. Indeed, if the rigidity test fails, it means that the image motion is not compatible to a rigid 3-D motion, and therefore the configuration *must* be non-rigid. If the test succeeds, it only asserts that a possible rigid 3-D motion *exists*, that is compatible to the given image motion. However, this computational approach corresponds to the way human vision operates – as shown in [64], human perception solves this inherent ambiguity by always choosing a rigid interpretation when possible.

7.4 Estimating Epipolar Geometry

Given two images of a static scene viewed with a moving camera (or equivalently, a single object in rigid motion), the epipolar geometry is described by the fundamental matrix, which – for any pair of matching points $x \leftrightarrow x'$ in the two frames – satisfies:

$$x'^T F \, x = 0 \tag{7.1}$$

If a sufficient number of point matches is provided, the fundamental matrix can be computed by using a linear method, such as the Eight Point Algorithm [35]. For increased robustness, we choose to use RANSAC [63] to recover the epipolar geometry for each rigid object. RANSAC randomly selects a minimal set of matches (seven point correspondences) to estimate the fundamental matrix, and measures the support for this estimation by using all point matches available. The process is repeated a number of times, and the fundamental matrix that received most support is chosen as the robust fit. The number of samples (sets of matches) that are randomly tried is chosen sufficiently high to ensure with a probability p (usually 95%), that at least one of the random samples of s points is free from outliers. If e is the probability that a selected data point is an outlier, then the number of samples required is:

$$N = \log(1 - p) / \log(1 - (1 - e)^s) \tag{7.2}$$

The RANSAC algorithm is summarized as follows:

1. Repeat for N samples:
    (a) Select a random sample of the minimum number of point matches to make a parameter estimate F.
    (b) Calculate the distance of each match to the epipolar lines defined by F.
    (c) Compute the number of inliers consistent with F.
2. Select the best solution – i.e., the largest consistent data set.

3. Re-estimate *F* using all the data that has been identified as consistent.

Note that the above procedure is applied separately for each rigidly moving region, as determined by the rigidity test. Therefore, inconsistent matches due to independent motion do not appear in the samples selected by the RANSAC algorithm.

7.5 Recovering Camera Motion and Scene Structure

Starting from a set of matches $x \leftrightarrow x'$ in the two images, that correspond to a set of 3-D points X_i, the reconstruction task is to find the camera matrices *P* and *P'*, as well as the 3-D points X_i, so that:

$$x_i = PX_i \qquad x_i' = P'X_i \qquad \text{for all } i \tag{7.3}$$

The camera matrices are determined as follows. Once the fundamental matrix *F* has been computed, the epipoles *e* and *e'* are first determined from:

$$Fe = 0 \qquad F^T e' = 0 \tag{7.4}$$

By using a 3-D coordinate system aligned with the first camera, the camera matrices *P* and *P'* can be then chosen as:

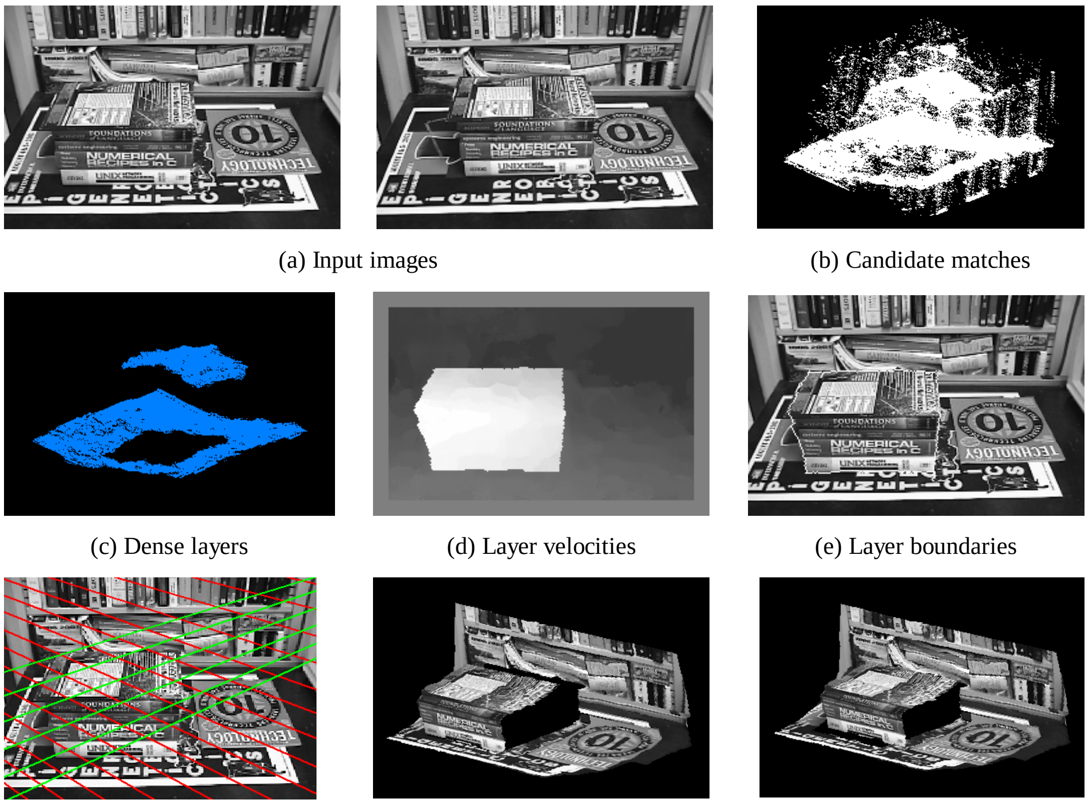
$$P = [I \mid 0] \qquad P' = [\,[e']_\times F \mid e'] \tag{7.5}$$

Note that the concept of camera motion is used here to denote either physical motion of the camera, or of the scene. Indeed, if the camera is static and the scene moves, the configuration is equivalent to a situation where the scene is static, and the camera moves with an inverse motion.

Finally, from the dense set of point matches $x \leftrightarrow x'$ and the recovered camera matrices P and P', the 3-D points X_i can be computed by triangulation. In practice, as back-projecting rays from the measured image points do not intersect precisely, it is necessary to estimate a best solution for the 3-D points. In our approach, we used a solution that minimizes the reprojection error for 3-D points that are mapped to the given image matches [21].

7.6 Results

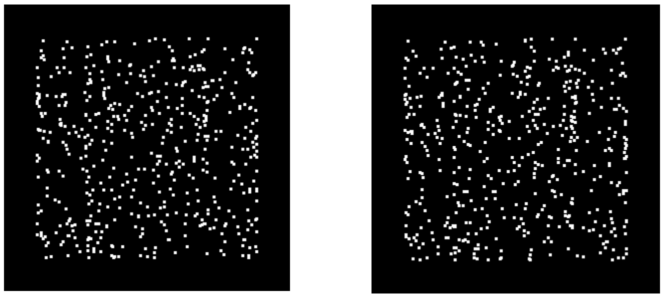
Books sequence (Figure 7.2). The two input images are taken with a handheld moving camera, while the stack of books has been moved between taking the two pictures. This

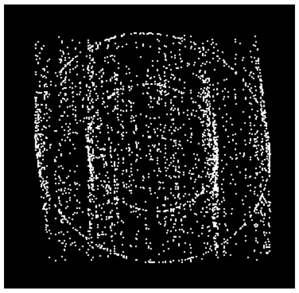

(a) Input images (b) Candidate matches

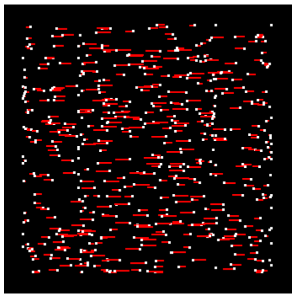
(c) Dense layers (d) Layer velocities (e) Layer boundaries

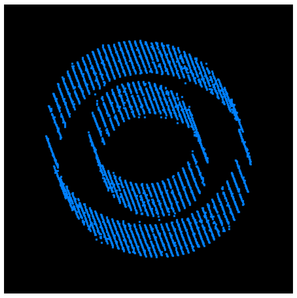
(f) Epipolar lines (g) 3-D structure and motion

Figure 7.2. Books sequence


(a) Input images


(b) Candidate matches


(c) Velocities


(d) Dense layers

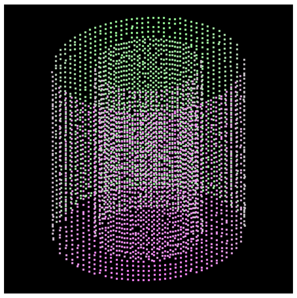
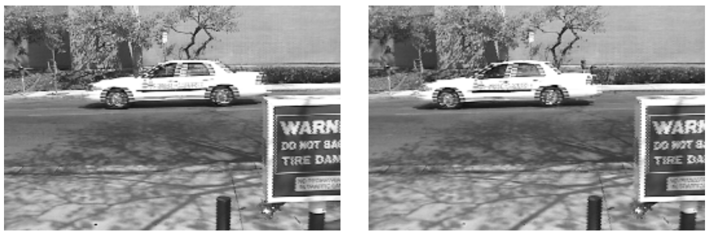
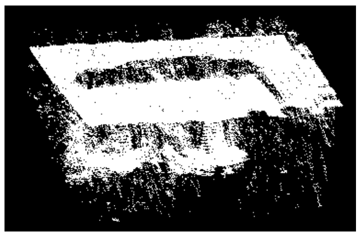

(e) 3-D structure

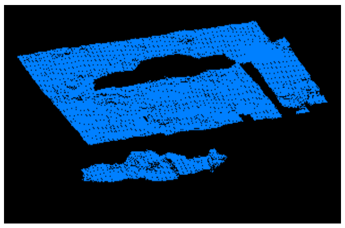
Figure 7.3. Cylinders sequence

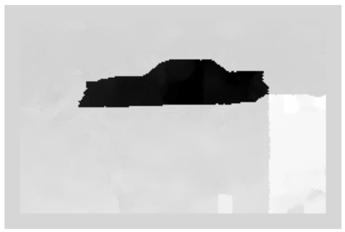
example corresponds to the *multiple rigid motions* case. Two sets of matches have been detected, corresponding to the two distinct regions – the stack of books and the background. The rigidity test shows that, while each region moves rigidly, they cannot be merged into a single rigid structure. The recovered epipolar geometry is illustrated in Figure 7.2(f), while the 3-D scene structure and motion are shown in Figure 7.2(g). Note the displaced stack of books in the 3-D reconstruction shown in the Figure 7.2(g) at left.

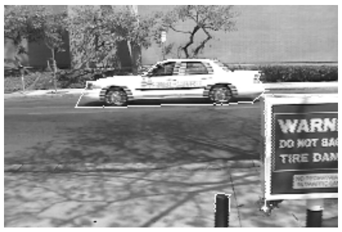
Cylinders sequence (Figure 7.3). This example, also illustrating the *multiple rigid motions* case, is adapted from Ullman [64], and consists of two images of random points in a sparse configuration, taken from the surfaces of two transparent co-axial cylinders, rotating in opposite directions. This extremely difficult example clearly illustrates the power of our approach, which is able to determine accurate point correspondences and scene structure –

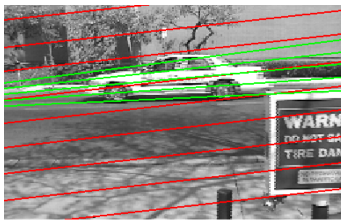

(a) Input images


(b) Candidate matches


(c) Dense layers


(d) Layer velocities


(e) Layer boundaries


(f) Epipolar lines

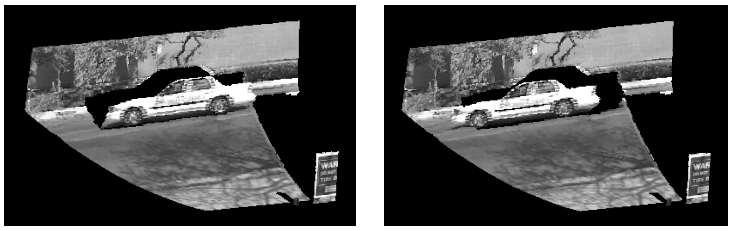

(g) 3-D structure and motion

Figure 7.4. Car sequence

even from a sparse input, based on motion cues only (without any monocular cues), and for transparent motion.

Car sequence (Figure 7.4). In this example, the sign and the background correspond to a rigid configuration and can be merged, while the car exhibits an independent motion.

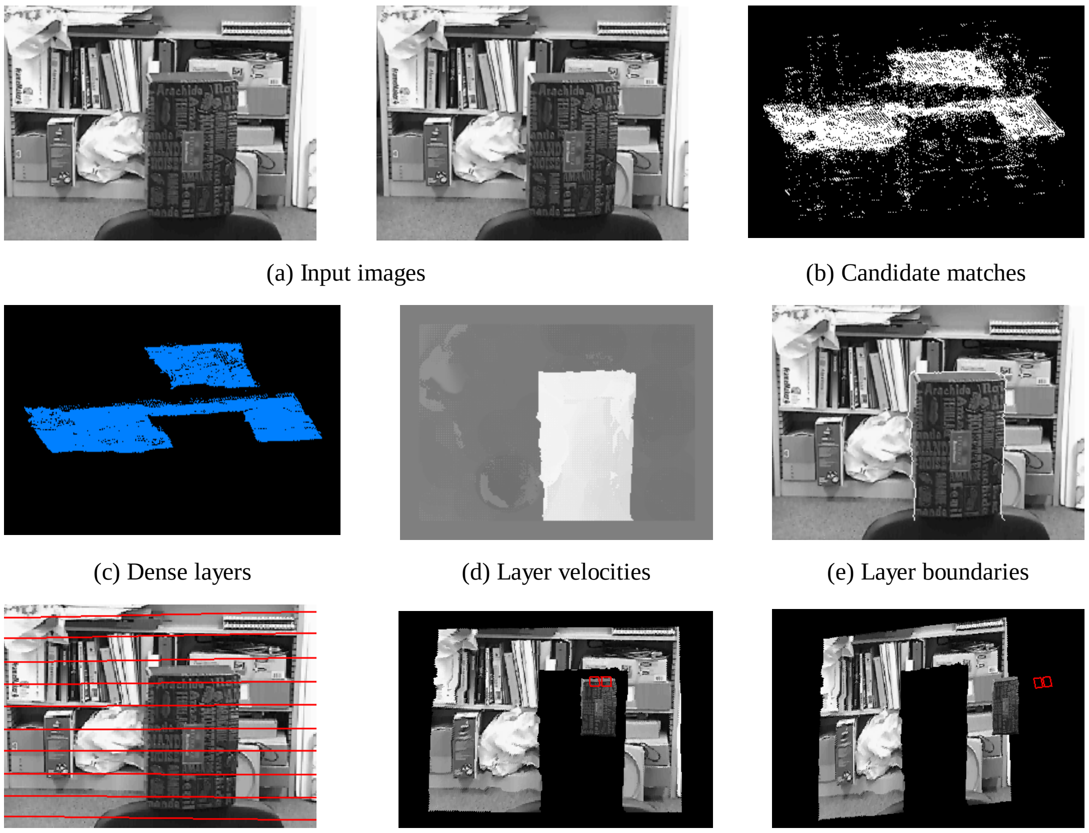
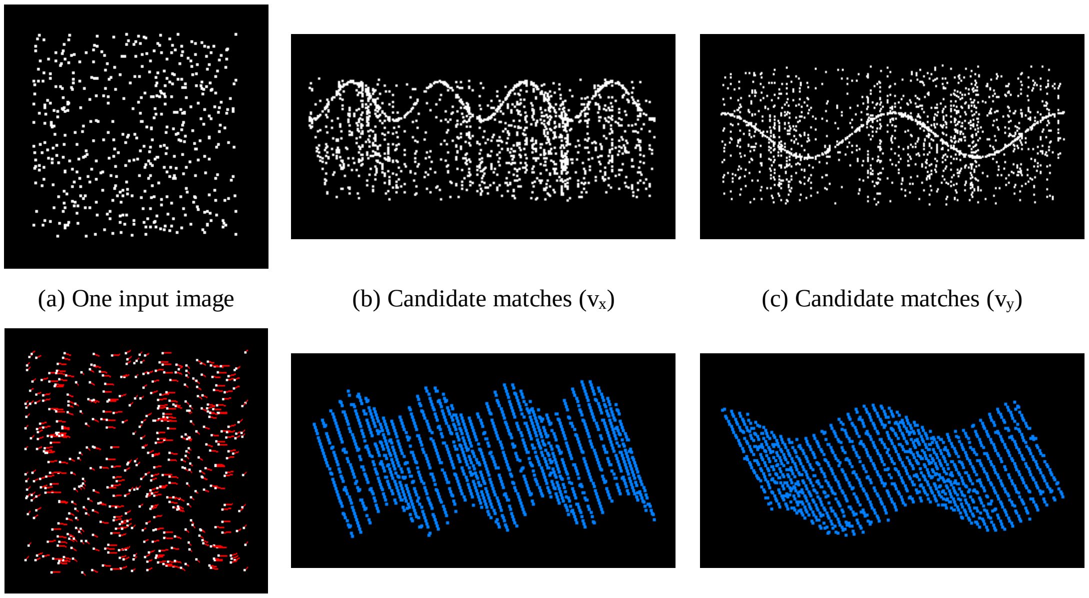

(a) Input images (b) Candidate matches
(c) Dense layers (d) Layer velocities (e) Layer boundaries
(f) Epipolar lines (g) 3-D structure and motion

Figure 7.5. Candy box sequence

Candy box sequence (Figure 7.5). This is the *single rigid motion* case, where the scene is static and the camera is moving. Due to the depth disparity between the box and the background, their image motions do not satisfy the smoothness constraint together, and thus they have been segmented as two separate regions. However, the rigidity test shows that the two regions form a rigid configuration, and therefore are labeled as a single region. The epipolar geometry estimation and scene reconstruction are then performed on the entire set of matches. Along with the 3-D structure, Figure 7.5(g) also shows the two recovered camera positions.

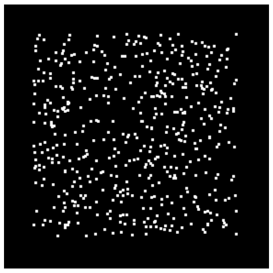


(a) One input image (b) Candidate matches (v_x) (c) Candidate matches (v_y)

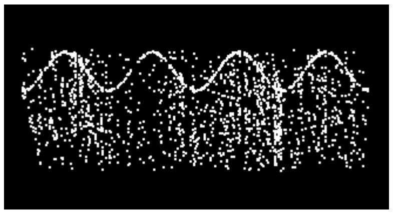
(c) Velocities (d) Dense layers (v_x) (e) Dense layers (v_y)

Figure 7.6. Waving flag sequence

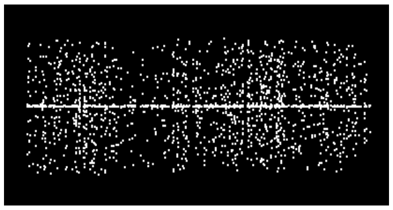
Waving flag sequence (Figure 7.6). This example illustrates the *non-rigid motion* case, and consists of a synthetic sequence where sparse random dots from the surface of a waving flag are displayed in two frames, as observed by a static camera. The configuration is recognized as non-rigid, and therefore no reconstruction is attempted. However, since the *image motion* is smooth, our framework is still able to determine correct correspondences, extract motion layers, segment non-rigid regions, and label them as such.



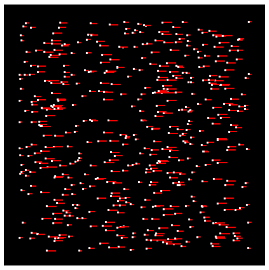
(a) One input image



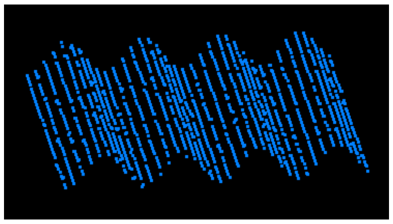
(b) Candidate matches (v_x)



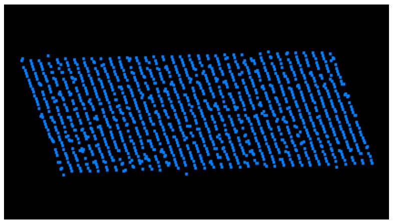
(c) Candidate matches (v_y)



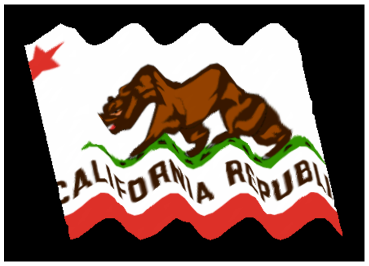
(c) Velocities



(d) Dense layers (v_x)



(e) Dense layers (v_y)



(f) 3-D structure

Figure 7.7. Static flag sequence

Static flag sequence (Figure 7.7). It is interesting to see what happens if the v_y component of the image motion is removed from the previous example. The new image sequence accepts a *rigid* interpretation, which corresponds to a static flag observed by a translating camera. This new configuration is recognized as rigid, and the 3-D reconstruction is shown in Figure 7.7(f) – for illustration purposes, the image of a flag has been mapped on the 3-D model.

Chapter 8

Conclusion

This dissertation presented a novel, voting-based computational framework that addresses the problem of visual motion analysis and interpretation. From two image frames involving general motion, the proposed approach recovers the dense velocity field, motion boundaries and regions, identifies rigid objects, and estimates the 3-D structure and motion for each such object.

We summarize the contributions of this work along four main directions:

- we developed a *4-D layered representation* of data, where the moving regions are conceptually represented as smooth layers in the 4-D space of image coordinates and pixel velocities;
- within this data representation, we employed a *voting scheme for token affinity communication*, where token affinities are expressed by their preference for being incorporated into smooth surfaces, as statistically salient features; communication between tokens is performed by tensor voting, which enforces the motion smoothness while preserving motion discontinuities;
- we consistently integrated both *motion and monocular cues* in our voting framework, in order to fully exploit the information available in real image sequences;

- by using an approach that decouples the processes of matching, outlier rejection and grouping, we segment the scene into rigidly moving regions, and produce a reliable *3-D motion interpretation*.

This formulation offers a consistent and unified approach for visual motion analysis and interpretation, that allows for structure inference without using any a priori knowledge of the motion model, based on the smoothness of motion only, while consistently handling both smooth moving regions and motion discontinuities.

Using a 4-D space for the voting-based motion analysis is essential, since it allows for a spatial separation of the points according to both their velocities and image coordinates. Consequently, the proposed framework allows tokens from the same motion layer to strongly support each other, while inhibiting the influence from other layers or from isolated tokens.

Despite the high dimensionality, the method is computationally robust – it does not involve initialization or iterative search in a parametric space, and therefore does not suffer from local optima or poor convergence problems. The only free parameter is scale, which is an inherent characteristic of human vision, and its setting is not critical.

The contributions of the proposed framework have been demonstrated by successfully analyzing a wide variety of difficult cases – opaque and transparent motion, rigid and non-rigid motion, curves and surfaces in motion, from sparse and dense input configurations.

The current framework is still subject to a number of limitations, that must be addressed in further research efforts. The first issue concerns the improvement of computation time, which prohibits the use of the proposed method for real time applications. A potential way to increase the processing speed is to implement a parallel processing scheme for voting, as the computations at each token are independent.

A second research direction is to incorporate information from multiple views, possibly in an incremental inference, where existing structures are refined as new frames are added. The primary concern here is how the representation must be augmented in order to allow encoding past trajectories, and using their curvature values.

Finally, studying the use of an adaptive scale processing scheme would allow to simultaneously capture details in regions with high density of data, and bridge the gaps where data is very sparse.

Reference List

[1] A. Adam, E. Rivlin, L. Shimshoni, "Ror: Rejection of Outliers by Rotations", *Trans. PAMI*, 23(1), pp. 78-84, 2001.

[2] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion", *IJCV*, 2:283-310, 1989.

[3] S. Arya, D. Mount, N. Netanyahu, R Silverman, A. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions", *Journal of the ACM*, 45(6), pages 891-923, 1998.

[4] S. Ayer, H. Sawhney, "Layered Representation of Motion Video Using Robust Maximum Likelihood Estimation of Mixture Models and MDL Encoding", *ICCV*, pages 777-784, 1995.

[5] J. Barron, D. Fleet, S. Beauchemin, "Performance of Optical Flow Techniques", *IJCV*, 12(1):43-77, February 1994.

[6] J. Bergen, P. Burt, K. Hanna, R. Hingorani, P. Jeanne, S. Peleg, "Dynamic Multiple-Motion Computation", *Artificial Intelligence and Computer Vision*, pp. 147-156, Elsevier, 1991.

[7] J. Bergen, P. Burt, R. Hingorani, S. Peleg, "A Three-Frame Algorithm for Estimating Two-Component Image Motion", *IEEE Trans. PAMI*, 14:9, pp. 886-896, 1992.

[8] Y. Boykov, O. Veksler, R. Zabih, "Markov Random Fields with Efficient Aproximations", *CVPR*, pp. 648-655, 1998.

[9] P. Burt, E. Adelson, "The Laplacian Pyramid as a Compact Image Code", *IEEE Trans. Communications*, 31:532-540, 1983.

[10] W. Cai, J. Wang, "Adaptive Multiresolution Collocation Methods for Initial Boundary Value Problems of Non-Linear PDEs", *SIAM Numer. Anal.*, 33(3):937-970, 1996.

[11] T. Darrell, A. Pentland, "Robust Estimation of a Multilayered Motion Representation", *IEEE Workshop on Visual Motion*, pages 173-178, 1991.

[12] T. Darrell, E. Simoncelli, "Separation of Transparent Motion into Layers Using Velocity-Tuned Mechanisms", *MIT Media Laboratory Vision and Modeling Group – TR-244*, 1993.

[13] R. Deriche, P. Kornprobst, G. Aubert, "Optical Flow Estimation while Preserving its Discontinuities: A Variational Approach", *ACCV*, pp. 290-295, 1995.

[14] D. Fleet, M. Black, A. Jepson, “Motion Feature Detection Using Steerable Flow Fields”, *CVPR*, pages 274-281, Santa Barbara, June 1998.

[15] L. Gaucher, G. Medioni, “Accurate Motion Flow Estimation with Discontinuities”, *Proc. ICCV*, pages 695-702, Corfu, Greece, September 1999.

[16] M. Gelgon, P. Bouthemy, “A Region-Level Graph Labeling Approach to Motion-Based Segmentation”, *CVPR*, pages 514-519, San Juan, Puerto Rico, June 1997.

[17] S. Ghosal, “A Fast Scalable Algorithm for Discontinuous Optical Flow Estimation”, *PAMI*, 18:2, pp. 181-194, 1996.

[18] G. Guy, G. Medioni, “Inferring Global Perceptual Contours from Local Features”, *IJCV*, 20(1/2):113-133, Oct 1996.

[19] G. Guy, G. Medioni, “Inference of Surfaces, 3-D Curves, and Junctions from Sparse, Noisy 3-D Data”, *PAMI*, 19(11):1265-1277, 1997.

[20] R. I. Hartley, “In Defense of the 8-Point Algorithm”, *Trans. PAMI*, 19(6), pp. 580-593, 1997.

[21] R. Hartley, A. Zisserman, “Multiple View Geometry in Computer Vision”, *Cambridge University Press*, 2000.

[22] D. Heeger, “Model for the Extraction of Image Flow”, *J. Opt. Soc. Amer.*, 4:1455-1471, 1987.

[23] D. Heeger, “Optical Flow using Spatiotemporal Filters”, *IJCV*, 1:279-302, 1988.

[24] F. Heitz, P. Bouthemy, “Multimodel Estimation of Discontinuous Optical Flow Using Markov Random Fields”, *PAMI*, 15(12):1217-1232, 1993.

[25] E. Hildreth, “The Measurement of Visual Motion”, *MIT Press*, 1983.

[26] B. Horn, B. Schunck, “Determining Optical Flow”, *Artificial Intelligence*, 17:185-204, 1981.

[27] S. Hsu, P. Anandan, S. Peleg, “Accurate Computation of Optical Flow by Using Layered Motion Representation”, *ICPR*, 1994.

[28] M. Irani, S. Peleg, “Image Sequence Enhancement Using Multiple Motions Analysis”, *CVPR*, pages 216-221, Champaign, Illinois, 1992.

[29] A. Jepson, M. Black, “Mixture Models for Optical Flow Computation”, *CVPR*, pages 760-761, New York, 1993.

[30] C. Kervrann, F. Heitz, “A Markov Random Field Model-Based Approach to Unsupervised Texture Segmentation Using Local and Global Spatial Statistics”, *IEEE Trans. on Image Processing*, 4(6):856-862, 1995.

[31] M.-S. Lee, G. Medioni, "Inferred Descriptions in Terms of Curves, Regions, and Junctions from Sparse, Noisy, Binary Data", *International Workshop on Visual Form*, pages 350-367, Capri, Italy, May 1997.

[32] M.-S. Lee, G. Medioni, "A Unified Framework for Salient Curves, Regions, and Junctions Inference", *Asian Conference on Computer Vision*, pages 315-322, Hong Kong, PRC, Jan. 1998.

[33] M.-S. Lee, G. Medioni, "Inferring Segmented Surface Description from Stereo Data", *CVPR*, pages 346-352, Santa Barbara, CA, June 1998.

[34] J. Little, H. Bulthoff, T. Poggio, "Parallel Optical Flow Using Local Voting", *ICCV*, pp. 454-459, 1988.

[35] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections", *Nature*, 293:133-135, 1981.

[36] W. Lorensen, H. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm", *Computer Graphics*, 21(4), 1987.

[37] B. Lucas, T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proc. DARPA Image Understanding Workshop*, pages 121-130, 1981.

[38] G. McLachlan, K. Basford, "Mixture Models Inference and Applications to Clustering", *Marcel Dekker, Inc.*, New York and Basel, 1988.

[39] D. McReynolds, D. Lowe, "Rigidity Checking of 3D Point Correspondences Under Perspective Projection", *Trans. PAMI*, 18(12), pp. 1174-1185, 1996.

[40] G. Medioni, Mi-Suen Lee, Chi-Keung Tang, "A Computational Framework for Segmentation and Grouping", *Elsevier Science*, 2000.

[41] J. Mulligan, "Motion Transparency is Restricted to Two Planes", *Investigative Ophtalmology and Visual Science Supplement*, 33:1049, 1992.

[42] H.-H. Nagel, "Displacement Vectors Derived from Second-Order Intensity Variations in Image Sequences", *Comput. Graph. Image Process.*, 21:85-117, 1983.

[43] H.-H. Nagel, W. Enkelmann, "An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences", *PAMI*, 8:565-593, 1986.

[44] H.-H. Nagel, "On the Estimation of Optical Flow: Relations between Different Approaches and some New Results", *Artificial Intelligence*, 33:299-324, 1987.

[45] M. Nicolescu, G. Medioni, “Perceptual Grouping from Motion Cues Using Tensor Voting in 4-D”, *Proceedings of the European Conference on Computer Vision*, vol. III, pages 423-437, Copenhagen, Denmark, May 2002.

[46] M. Nicolescu, G. Medioni, “4-D Voting for Matching, Densification and Segmentation into Motion Layers”, *Proceedings of the International Conference on Pattern Recognition*, Quebec City, Canada, August 2002.

[47] M. Nicolescu, G. Medioni, “Motion Segmentation with Accurate Boundaries - A Tensor Voting Approach”, *USC Institute for Robotics and Intelligent Systems Technical Report IRIS-02-418*, December 2002.

[48] M. Nicolescu, G. Medioni, “Layered 4D Representation and Voting for Grouping from Motion”, *IEEE Transactions on Pattern Analysis and Machine Intelligence – Special Issue on Perceptual Organization in Computer Vision*, vol. 25, no. 4, pages 492-501, April 2003.

[49] M. Nicolescu, G. Medioni, “Motion Segmentation with Accurate Boundaries - A Tensor Voting Approach”, to appear in the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, June 2003.

[50] P. Pritchett, A. Zisserman, “Wide Baseline Stereo Matching”, *ICCV*, pp. 754-760, 1998.

[51] J. Shi, J. Malik, “Normalized Cuts and Image Segmentation”, *CVPR*, pages 731-737, San Juan, Puerto Rico, June 1997.

[52] J. Shi, J. Malik, “Motion Segmentation and Tracking Using Normalized Cuts”, *ICCV*, pages 1154-1160, Bombay, India, January 1998.

[53] M. Shizawa, K. Mase, “Simultaneous Multiple Optical Flow Estimation”, *CVPR*, 1990.

[54] M. Shizawa, K. Mase, “A Unified Computational Theory for Motion Transparency and Motion Boundaries Based on Eigenenergy Analysis”, *CVPR*, 1991.

[55] E. Simoncelli, E. Adelson, D. Heeger, “Probability Distributions of Optical Flow”, *CVPR*, pages 310-315, Maui, 1991.

[56] A. Singh, “An Estimation-Theoretic Framework for Image-Flow Computation”, *Proc. ICCV*, pages 168-177, 1990.

[57] A. Singh, “Optical Flow Computation: A Unified Perspective”, *IEEE Computer Society Press*, 1992.

[58] R. Szeliski, S. Avidan, P. Anandan, “Layer Extraction from Multiple Images Containing Reflections and Transparency”, *CVPR*, pp. 246-253, 2000.

[59] C.-K. Tang, G. Medioni, “Integrated Surface, Curve, and Junction Inference from Sparse 3D Data Sets”, *ICCV*, pages 818-824, Bombay, India, Jan. 1998.

[60] C.-K. Tang, G. Medioni, “Extremal Feature Extraction from 3D Vector and Noisy Scalar Fields”, *Visualization '98*, pages 95-102, Research Triangle Park, North Carolina, Oct 1998.

[61] C.-K. Tang, G. Medioni, “Robust Estimation of Curvature Estimation from Noisy 3D Data for Shape Description”, *ICCV*, pages 426-433, Corfu, Greece, Sept 1999.

[62] C.-K. Tang, G. Medioni, M.-S. Lee, “Epipolar Geometry Estimation by Tensor Voting in 8D”, *ICCV*, pages 502-509, 1999.

[63] P.H.S. Torr, D.W. Murray, “A Review of Robust Methods to Estimate the Fundamental Matrix”, *IJCV*, 1997.

[64] S. Ullman, “The Interpretation of Visual Motion”, *MIT Press*, 1979.

[65] J. Wang, E. Adelson, “Representing Moving Images with Layers”, *IEEE Trans. on Image Processing Special Issue: Image Sequence Compression*, 3(5):625-638, Sept 1994.

[66] Y. Weiss, “Smoothness in Layers: Motion Estimation Using Nonparametric Mixture Estimation”, *CVPR*, pages 520-526, Puerto Rico, 1997.

[67] Y.-T. Wu, T. Kanade, J. Cohn, C.-C. Li, “Optical Flow Estimation Using Wavelet Motion Model”, *ICCV*, pages 992-998, Bombay, India, January 1998.

[68] Z. Zhang, “Determining the Epipolar Geometry and Its Uncertainty: A Review”, *IJCV*, 27(2), pp. 161-195, 1998.