SPACE-TIME IMAGE SEQUENCE ANALYSIS: OBJECT TUNNELS AND OCCLUSION VOLUMES

MIRKO RISTIVOJEVIĆ

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

BOSTON
UNIVERSITY

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

SPACE-TIME IMAGE SEQUENCE ANALYSIS: OBJECT TUNNELS AND OCCLUSION VOLUMES

by

MIRKO RISTIVOJEVIĆ

B.S., University of Belgrade, 1999
M.S., Boston University, 2002

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2006

Approved by

First Reader

Janusz Konrad, Ph.D.
Associate Professor of Electrical and Computer Engineering

Second Reader

W. Clem Karl, Ph.D.
Professor of Electrical and Computer Engineering and
Professor of Biomedical Engineering

Third Reader

Stan Sclaroff, Ph.D.
Associate Professor of Computer Science

Fourth Reader

Maja Bystrom, Ph.D.
Associate Professor of Electrical and Computer Engineering

To my wife Nataša,
without whose love and support
this work would not be possible.

Acknowledgments

When I started my graduate studies at the Boston University, I did not have an idea what my research topic would be. I was preoccupied with adjusting to the new environment, living in a foreign country, as well as my GTF responsibilities, some pretty difficult courses I was taking and the prospect of the upcoming Ph.D. qualifier exams looming ahead. I had serious doubts in the success of my studies, but somehow I carried on. One of the people that helped me tremendously in these early stages of my graduate studies was Tommaso Toffoli, for whom I was a teaching assistant. He is a great professor and a wonderful human being, and our discussions on science, engineering and life in general lifted my spirits many times.

One of the courses I took in my first semester at the Boston University was the image processing course with Janusz Konrad. Immediately, I was attracted to it, not only because of a very interesting subject matter, but also the engaging way Janusz was teaching. By the end of the semester I decided to pursue my research under his supervision, and he accepted, for which I am deeply grateful. His brilliant ideas and broad knowledge made my research possible. However, more than anything else, I would like to thank Janusz for his patience and for believing in me even when I did not believe in myself. His guidance and support helped me make all the necessary steps in the development of this dissertation. Finally, I would like to thank him for being such a wonderful person, passionate about our research, but also understanding of all the difficulties of graduate student life.

I would like to thank Clem Karl for the courses on stochastic processing and image reconstruction he teaches so wonderfully. What I learned there made a foundation for my research. I would also like to thank him for serving on my dissertation committee. I would like to thank Stan Sclaroff and Maja Bystrom for serving on my dissertation committee, and for many important comments and insights they provided, which greatly

improved the final dissertation document. I would also like to thank all the faculty at Boston University from whom I have taken classes.

Throughout my studies, my colleagues at the ISS lab contributed my research in various ways. Nikola Božinović, with whom I shared the office for the last five and a half years (and also an apartment for a year), helped me in more ways than I can try to describe. Our discussions on the problem of video processing helped me solve many issues in my research. His encyclopedic knowledge of all things PC helped me with many technical and computer problems. Our discussions on sports, politics, music, movies and life in general made my days in the office so much nicer. I want to thank Nikola and his wife Mina for our friendship outside of the office, which made me feel so much more at home in Boston. I would like to thank Julia Pavlovich for the help with classes we took together, and also for our friendship inside and outside of the office. I thank Andrew Litvin for many insightful discussions we had on our research problems, and for a wonderful month we spent together at the University of Nice. Yonggang Shi helped me numerous times with the motion segmentation and level-set method issues I had, for which I am deeply grateful. Also, I want to thank him for sharing his fast level-set implementation method with me, which helped tremendously with my experiments. I would like to thank Robert Weisenseel for helping the whole ISS group by maintaining the computer system of our lab. His extensive knowledge of Linux, Emacs, and Latex helped me on many occasions. I want to thank Peter McNerney for helping me with 3-D visualization of my results and shooting the *Car* sequence used in my experiments. I would also like to thank other ISS members, Serdar Ince, Philippe Agniel, Shuchin Aeron, Erhan Ermis, Onur Savas, George Atia, Karen Jenkins, and Andrej Cvetkovski for valuable insights and discussions, as well as for making PHO 401/446 a fun place to be.

I would like to thank my friends, Zoran Hadžibabić and Zoran Dimitrijević, for en-

couraging me to come to do my Ph.D. studies in United States. Zoran Hadžibabić is one of the main reasons I chose to come to Boston to study, and the wonderful years we spent living together in the Henry Street apartment will always remain unforgettable. Together with my other roommate from that apartment, Walter Rantner, he made that place a home away from home. I would like to thank my friends Bosiljka Tasić, Gojko Lalić, Selena Hadžibabić, and Jelena Veljković, with whom I spent many beautiful moments. Although he was studying across the continent in California, Zoran Dimitrijević's friendship and support helped me throughout my studies. Moreover, we grew up together and I feel that everything I accomplished I owe in part to him. I would also like to thank my roommate from the Union Street, Tiago Ribeiro, whose friendship made the two years I spent while my wife was studying in Syracuse less lonely.

I would like to thank my parents, Lazar and Danica, for their everlasting love and support for my education. I am grateful for my happy childhood, for their moral support and for directing me towards knowledge and education. I also want to thank my sister Mira, for her love and for being somebody I looked up to while growing up. She helped me tremendously by directing me towards mathematics and physics from the early age, introduced computers and electrical engineering to my life and always supported me with advice and example.

Finally, I want to mention a person without none of this would be possible. My wife Nataša, with whom I share my life for the past thirteen years, helped me complete not only this dissertation, but also my undergraduate studies. Her love and support got me through all the rough times, and I dedicate this dissertation to her.

SPACE-TIME IMAGE SEQUENCE ANALYSIS: OBJECT TUNNELS AND OCCLUSION VOLUMES

(Order No.)

MIRKO RISTIVOJEVIĆ

Boston University, College of Engineering, 2006

Major Professor: Janusz Konrad, Ph.D., Department of Electrical and Computer Engineering

ABSTRACT

We present a novel approach to joint space-time, motion-based video segmentation and occlusion detection. The segmentation of an image sequence into moving objects and estimation of object motion belong to the most important tasks in image sequence analysis. Image sequence segmentation is a very difficult problem, with numerous applications, including content-dependent video compression (e.g., MPEG-4), video processing (e.g., object-based frame-rate conversion and deinterlacing), surveillance, video database queries (event detection, tracking), and computer vision (scene analysis, structure from motion). In most studies to date, image sequences have been primarily analyzed and processed in groups of two frames; by differentiating one frame from the other, one is able to infer the dynamics occurring in an image sequence. These short-term dynamics (such as displacement between two frames, or occlusion/exposure areas) can be linked together or temporally constrained in order to reason about longer term dynamics. Although the two-frame approach has been very successful in some applications (e.g., MPEG compression standards), it is often inadequate for the analysis of non-constant velocity motion, detection of long-term innovation areas (occlusion and exposure), or

video segmentation.

In this dissertation, we propose to perform image sequence analysis jointly over multiple frames. We concentrate on spatio-temporal segmentation of image sequences and on the analysis of occlusion effects therein. The segmentation process is three-dimensional (3-D); we search for a volume carved out by each moving object in the image sequence domain, or "object tunnel", a new space-time concept. We pose the problem in variational framework by using only motion information (no intensity edges). The resulting problem can be viewed as volume competition, a 3-D generalization of region competition. We parameterize the unknown surface to be estimated, but rather than solving for it using an active-surface approach, we embed it into a higher-dimensional function and apply level-set methodology. We first develop simple models for the detection of moving objects over static background; no motion models are needed. Then, in order to improve segmentation accuracy, we incorporate parametric motion models (affine) for objects and background. We further extend the method by including explicit models for occluded and newly-exposed areas that lead to "occlusion volumes", another new space-time concept. Since in this case multiple volumes are sought, we apply a multiphase version of the level-set method. We extend our motion detection to account for camera motion and zoom-in (background is no longer static). In order to reduce computational complexity of our methods, we apply a recently-proposed fast level-set implementation and investigate its performance. We present various experimental results for synthetic and natural image sequences, including those from the VIVID Tracking Evaluation Web Site at Carnegie Mellon University.

Contents

List of Tables

List of Figures

List of Abbreviations

2-D	.....	Two-Dimensional
2-frame MCS	.....	Two-frame Motion-Compensated Segmentation
2-frame MD	.....	Two-frame Motion Detection
3-D	.....	Three-Dimensional
4-D	.....	Four-Dimensional
BFGS	.....	Broyden, Fletcher, Goldfarb, and Shanno
DWT	.....	Discrete Wavelet Transform
EM	.....	Expectation-Maximization
HCF	.....	Highest Confidence First
ICM	.....	Iterated Conditional Modes
ITU	.....	International Telecommunication Union
M-frame MCS-B	.....	Multi-frame MC Segmentation with Background occlusion modeling
M-frame MCS-BO	.....	Multi-frame MC Segmentation with Background and Object occlusion modeling
M-frame MD	.....	Multi-frame Motion Detection
M-frame MD-BMC	.....	Multi-frame Motion Detection with Background Motion Compensation
MAP	.....	Maximum A posteriori Probability
MC	.....	Motion-Compensated

MDL	.....	Minimum Description Length
MLE	.....	Maximum-Likelihood Estimation
MPEG	.....	Moving Picture Experts Group
MRF	.....	Markov Random Field
PDE	.....	Partial Differential Equation
PSNR	.....	Peak Signal-to-Noise Ratio
QCIF	.....	Quarter Common Intermediate Format
VIVID	.....	Video Verification of Identity

Chapter 1

Introduction

In this chapter we introduce the problem of video analysis and describe its main applications. We proceed with a brief discussion of our work on video analysis, followed by our main contributions. Finally, we give the layout of the rest of this dissertation.

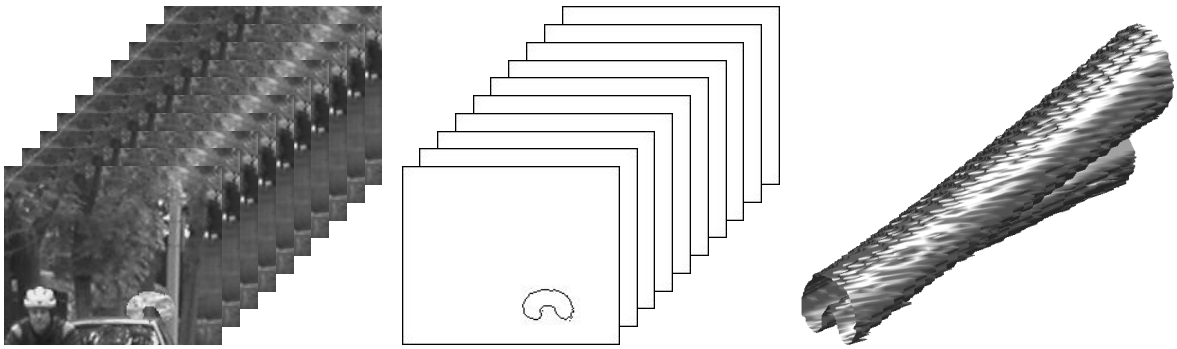
1.1 What is video analysis?

Analysis of image sequences consists of sequence segmentation into moving objects and static or moving background and estimation of the objects and background motion, as well as content analysis, understanding, recognition, and event detection in video sequences. In this dissertation, we will concentrate on several important areas of video analysis, including video segmentation, motion estimation and modeling, and detection of occlusion events.

Segmentation of an image sequence is a very important but difficult problem. Some methods use intensity information as the primary cue for segmentation while using motion only as an accessory. Other methods concentrate on motion information by segmenting dense motion fields calculated using some of the standard motion estimation methods. No matter which approach is used, video segmentation is closely related to the estimation of motion for each object in the sequence. Solving both problems jointly leads to joint motion estimation and segmentation, which is the road we will take in this dissertation.

In most studies to date, image sequences have been primarily analyzed and processed

in groups of two frames, which leads to separate segmentation of each frame of image sequence into 2-D regions (an example of resulting object contours is shown in Fig. 1·1 (b)). By differentiating one frame from the other, one is able to infer the dynamics occurring in an image sequence. These short-term dynamics (such as displacement between two frames, or occlusion/exposure areas) can be linked together or temporally constrained in order to reason about longer term dynamics. However, we believe that approach to be somewhat inadequate. Although the two-frame approach to video analysis has been very successful in some applications (e.g., MPEG and ITU compression standards), it is often inadequate for the analysis of non-constant velocity motion, detection of long-term innovation areas (occlusion and exposure), or video segmentation. In this dissertation, we propose to perform segmentation jointly over multiple frames which leads to partition of the three-dimensional image sequence domain into object and background volumes (example of the object tunnel is shown in Fig. 1·1 (c)).



(a) (b) (c)

Figure 1·1: Example of segmentation of an image sequence: (a) original frames of the sequence, (b) object contours resulting from the processing of two frame at a time, and (c) object tunnel resulting from joint processing of multiple frames.

Another important aspect of video analysis is modeling of motion fields for moving

objects and background. This is pretty straightforward in the case of two-frame processing where motion field has to be modeled only spatially, with models varying from dense (optical flow) to parametric (affine, perspective, etc.). However, if multiple frames are processed jointly, motion vectors become motion trajectories which need to be modeled both spatially and temporally. The simplest temporal model is piecewise linear where separate motion vectors between consecutive frames are pieced together to form motion trajectories. Parametric models, like constant velocity or constant acceleration models, were also investigated. We propose to use more flexible spline representation for both spatial and temporal motion trajectory modeling.

1.2 Applications

Segmentation of an image sequence has numerous applications, including content-dependent video compression (e.g., MPEG-4 (Brady, 1999)), object-based transcoding (Vetro et al., 2001), video processing (e.g., object-based frame-rate conversion (Han and Woods, 1997) and deinterlacing (Haan and Bellers, 1998), digital compositing (Chuang et al., 2001)), surveillance (Collins et al., 1999), video database queries (Chang et al., 1998) (event detection, tracking), semantic video analysis (used for adaptive video content delivery (A. Cavallaro, 2005)), and computer vision (scene analysis (Koch, 1993), structure from motion (Chiuso et al., 2002)).

An example of application which would benefit from the joint analysis and processing of multiple frames is video compression based on 3-D discrete wavelet transforms (DWTs), a research area with significant activity in the last decade. The most successful 3-D DWT video coders perform motion-compensated (MC) temporal DWT implemented through lifting and followed by a 2-D spatial DWT (Taubman and Zakhor, 1994; Ohm, 1994; Pesquet-Popescu and Bottreau, 2001; Bottreau et al., 2001). Depending on the order of the underlying temporal wavelet, the MC temporal DWT may require

motion representation over 2 frames (Haar wavelet), 3 frames (5/3 wavelet), 7 frames (9/7 wavelet) or even more. The temporal support of motion representation becomes even longer with the growing number of temporal decomposition levels used (temporal scalability). The current approaches use a simple frame-to-frame block matching motion estimation, and subsequent "stitching". In addition to the long-term motion representation, such coders face the problem of "innovations", such as occluded and newly-exposed areas, since they have no correspondence in the neighboring frames. Known as the "unconnected" pixels, such areas are currently handled in an ad-hoc fashion. We believe such coders could benefit from a reliable detection of occluded and newly-exposed image areas.

1.3 Proposed approach to video analysis

In our work on video segmentation we will use an iterative approach called *surface evolution*, which is an extension of curve evolution method to three-dimensional domain. Standard approach to curve evolution is through energy-minimizing active contours. Both parametric (B-splines, etc.) and non-parametric (level-set method) contour (or surface) representations have been used in the literature. Although active contours have been around for many years (Kass et al., 1988), only recently have they been applied to motion-based video segmentation. This is due to the development of effective implementation framework for active contours called *level-set* methods (Sethian, 1996b). The main idea is to represent a closed 2-D curve by a zero-level set of a 3-D dimensional function with two benefits over active-contour implementations: no need to re-parameterize the curve during evolution, and automatic handling of variable topology (objects can appear and disappear). Additionally, problems with curve self-intersections no longer exist and numerical schemes for the level-set method are general for arbitrary dimensions. The only drawback of the level-set implementation is its high computational

cost.

In this dissertation, we present a novel approach to joint space-time, motion-based video segmentation. We pose the problem in variational framework with respect to a 3-D surface that partitions the image sequence domain into inside and outside. The inside corresponds to a 3-D volume "carved out" by a moving object, that we shall call *object tunnel*, while the outside corresponds to background (possibly static). The problem is formulated as *volume competition*, i.e., the surface is adjusted in response to the competition between voxels inside and outside of it, a 3-D generalization of the region competition (Zhu and Yuille, 1996). The resulting active surface evolution equation is discretized and solved using standard level-set approach (Sethian, 1996b) on a 3-D domain (4-dimensional level-set function). In its simplest version, our algorithm performs motion detection based on models proposed earlier in the literature (Jehan-Besson et al., 2000).

This multi-frame motion detection algorithm results in fairly accurate object segmentations. However, it commits consistent errors due to the nature of the observation model used: segmented objects in each frame represent a union of object positions in consecutive frames. To address this issue, we extend the method by explicitly modeling the evolution of object and background using motion trajectories (Ristivojević and Konrad, 2004b). However, occluded and newly-exposed background areas (due to object motion) cannot be explained by these motion trajectories. To account for these regions, we include explicit models for the occluded and newly-exposed background areas. This leads to new space-time concepts of *occlusion volume* and *exposed volume*. As for motion trajectories, we use a parametric model associated either with object or background. Since we need to partition the image sequence domain into 4 volumes, we use the multiphase level-set framework (Vese and Chan, 2002) (to be explained in Section 2.3.4).

Although this approach significantly improves segmentation accuracy, it still does not handle well image sequences in which a moving object enters or leaves the scene, or is occluded by a static feature in the background. To solve that problem, we extend our formulation to include explicit models of the occluded and newly-exposed areas of the object (Ristivojević and Konrad, 2004a) (for a single moving object, resulting in partition of the image sequence domain into 6 volumes). Finally, we generalize our motion segmentation formulation to allow for multiple moving objects. General variational formulation is proposed, which can handle any number of objects and occlusion/exposure volumes. A solution method based on multiphase level-set algorithm is described and results for a natural sequence with two moving objects are presented.

Our simple multi-frame motion detection algorithm performs well only for image sequences with static backgrounds. Since we use that method to initialize our more advanced segmentation methods, it is very important to lift that constraint. We expand our motion detection formulation to account for camera motion and zoom. We use affine model to describe camera motion between frames, and calculate initial motion parameters using global motion estimation. Afterwards, evolution of the level-set surface that encloses moving objects and motion estimation in the background region only are performed simultaneously. For each of the methods described above, we present experimental results for synthetic and natural image sequences, and numerically compare results for the synthetic sequences (for which ground truth segmentation is known).

A major drawback of the level-set method is its computational complexity, which limits its usefulness for time-critical applications. Recently, a novel level-set implementation was developed by Shi (Shi, 2005) which dramatically reduces computational cost of the algorithm. We apply it to all of our motion detection and segmentation methods and investigate its performance by running experiments on different image sequences and comparing results to the ones obtained using standard level-set implementation.

Reduction of computation time by up to 200 times is achieved, while the segmentation accuracy of the standard level-set implementation is preserved. We also use this method to run experiments on larger image sequences from VIVID Tracking Evaluation Web Site at Carnegie Mellon University (Collins et al., 2005).

In our segmentation methods, we model evolution of moving objects and background using motion trajectories which are piece-wise linear, i.e., they consist of motion vectors calculated for each frame pair separately. These linear segments of the motion trajectory are not constrained temporally in any way. However, the underlying motion of moving objects is smooth in temporal direction and to model that we expand our motion estimation formulation with a term which penalizes the difference between consecutive motion vectors along trajectory.

In an effort to model motion of objects and background with parametric models over the whole span of the image sequence we investigate spatio-temporal modeling of motion trajectories using splines as a part of the future work. An important issue we address is how to calculate motion trajectories referenced around any frame of the sequence given spline representation of motion trajectories estimated with respect to one reference frame. We investigate motion trajectory modeling in detail, while only describing issues concerning its application to video segmentation.

To summarize, the contributions of this dissertation are:

- Development of a novel approach to joint spatio-temporal motion-based video detection and video segmentation based on volume competition and surface evolution.

- Development of new, space-time models for occluded and newly-exposed areas in a video sequence.

- Introduction of new space-time concepts of occlusion and newly-exposed volumes.

The rest of this dissertation is organized as follows. In Chapter 2, we give an overview of motion-based segmentation methods proposed in the literature, with an emphasis on multiple-frame methods. In Chapter 3, we formulate the problem of joint spatio-temporal image sequence segmentation, describe a solution, and show results for the simpler case of motion detection. We proceed to formulate motion-based segmentation method with background occlusion detection in Chapter 4, solve it using the multiphase level-set method and show results for synthetic and natural sequences. We extend the formulation to include object occlusions in Chapter 5, and to the general case of multiple moving objects in Chapter 6. In Chapter 7 we expand our motion detection method to include compensation for moving background. A novel fast level-set implementation developed by Shi is presented in Chapter 8, together with its application to our segmentation methods. Finally, in Chapter 9 we summarize the work presented in this dissertation and give some directions for future research.

Chapter 2

Prior work

Among numerous video segmentation methods proposed to date, we concentrate on the ones which use motion as a primary segmentation cue. First, we discuss methods in which segmentation of an image sequence is performed separately for each frame pair. In some cases, an additional temporal constraint is applied, as a post-processing step, to the resulting segmentation maps. Then, we present methods that jointly process groups of frames, where a group consists of minimum 3 frames. Finally, we discuss mathematical tools needed to implement segmentation algorithms proposed in this thesis.

2.1 Two-frame methods

The segmentation of an image sequence into moving objects is closely related to the estimation of motion for each object. A good estimate of an object's motion trajectories makes segmentation of this object much easier. However, in order to find good estimates for the object motion trajectories, one needs to localize the object in each frame of the sequence as closely as possible. This kind of "chicken and egg" problem can be solved adequately only by considering both problems jointly, which leads to joint motion estimation and segmentation. However, this joint segmentation/estimation problem is very difficult and ill-posed.

The early attempts to solve this problem involved simple thresholding over the inter-frame difference, where pixel-wise differences or block differences (to increase robustness) have been considered. Later, robust optical flow motion estimation with a discontinuity-

preserving smoothness constraint (implemented using discontinuity weights in the smoothness energy term) was proposed in (Konrad and Dubois, 1992; Mémin and Pérez, 1998). This model was extended to couple the motion estimation process with an object-based motion segmentation. The segmentation process interacts with motion estimation only through discontinuity weights. A combined motion estimation and segmentation method within a Bayesian framework was proposed in (Chang et al., 1997). Motion fields were modeled as the sum of a parametric field and a non-parametric residual field. Markov random field (MRF) models were applied to both motion and segmentation fields. The resulting energy function is minimized iteratively, with interleaved updates of motion and segmentation field performed using highest confidence first (HCF) optimization scheme.

Having multiple moving objects in an image sequence makes its segmentation even more difficult. We would like to avoid semiautomatic algorithms in which human intervention is needed to define the number of objects in an image sequence. In (Wang and Adelson, 1994) image sequences are represented with sets of overlapping layers, each layer corresponding to a moving object in the sequence and containing a set of maps specifying its intensity, opacity, and motion. Local motion estimates are obtained using multi-scale optical flow algorithm. An image is initially divided into small square regions and parameters of an affine motion model for each region are calculated. Similar motion models are grouped in the affine motion parameters space with a K-means clustering algorithm. Regions within the image are assigned to derived motion models in a way that minimizes motion distortion between estimated local motion field and the affine motion field corresponding to that model. The procedure is repeated iteratively together with additional processing which enforces local spatial connectivity of motion regions. Techniques for automatic decomposition of a video sequence into multiple motion models and their layers of support are presented in (Sawhney and Ayer, 1996). The authors

present a method for the simultaneous estimation of multiple 2D parametric models and their layers of support. The multiple model estimation problem is formulated as robust maximum-likelihood estimation (MLE) of mixture model parameters and of the layers of support represented as ownership probabilities. The adequate number of models is automatically decided using the minimum description length (MDL) principle that minimizes the encoding length of the model parameters and of the MLE residuals. The estimation uses a modified Expectation-Maximization (EM) algorithm.

Curve evolution methods and their implementation through level-set methods have recently become a modeling/solution tool of choice for image segmentation as well as spatial video segmentation, e.g., based on motion. Video segmentation methods based on that framework can be divided into two broad groups. In the first group, curve evolution stops at large gradients in the motion field (or both motion and intensity fields for "mixed" models). These approaches can be considered *edge-based*. In (Paragios and Deriche, 2000), the probability density function of inter-frame differences is statistically modeled using a mixture model, which is a combination of two components (background model and object model). Using the Bayes rule, the conditional object boundary probability given the observed data is estimated. The detection and the tracking problem are dealt with simultaneously using a geodesic active contour model with two terms: the detection term which forces the curve to converge towards the moving area, and the tracking term used for evolving the curve additionally until it coincides with the exact location of the moving object edge.

If the consecutive frames are stacked on top of each other, a video sequence can be represented as a 3-D volume $(x - y - t)$. Motion in the sequence can be estimated by analyzing orientations of local gray-value structures n this volume (gray values remain constant in the direction of motion). Moving and static parts on the image plane can be determined from the direction of minimal gray value change in the spatio-temporal

volume. This direction can be calculated using a 3-D structure tensor. At each point, structure tensor is defined based on the spatio-temporal image sequence derivatives measured in a region around that pixel (Wright and Pless, 2005). Eigenvalues of the three-dimensional structure tensor were used to create motion maps in (Zhang et al., 2001; Kühne et al., 2001). Tensor-based motion detection is incorporated into classical active-contour model (Caselles et al., 1997a). In (Kühne et al., 2001) results are improved by contour refinement using image gradient information, while global motion estimation is performed before tensor-based motion detection in (Zhang et al., 2001).

One of the limitations of the edge-based models is that they are easily attracted to local minima and the capture range of object's boundary contour is relatively small (requiring careful initialization). To alleviate these problems, the second group of approaches, which consider all intensities in a region (*region-based* approaches), have been proposed. In (Zhu and Yuille, 1996), a novel *region competition* formulation is derived by minimizing a generalized Bayes/MDL criterion. Intuitively, adjacent regions compete for ownership of pixels along their boundaries, subject to a smoothness constraint. This approach combines the most attractive aspects of active contours and region growing to minimize a global cost function. Region competition is applied to the problem of motion-based video segmentation in (Mansouri and Konrad, 1999; Mansouri and Konrad, 2003). A parametric motion model is assigned to each moving region and the problem is formulated in MAP framework using only motion information, with a prior on curve length. The resulting energy functional is minimized using the level-set method. This formulation is extended to account for the coincidence of motion and intensity boundaries in (Mansouri and Konrad, 2003). Furthermore, the approach is generalized to multiple-motion segmentation, using a system of coupled level-set partial differential equations. To initialize the segmentation algorithm, the number of motion classes and motion parameters (assumed to be affine) for each class are computed independently of

any motion segmentation using correspondence estimation and motion classification. A general framework for region-based active contours for moving object segmentation is proposed in (Jehan-Besson et al., 2003). That framework is applied to the detection of moving objects in a video sequence acquired by static camera. A variational method for the space-time segmentation of a moving object against still background over a sequence of two-dimensional or three-dimensional image frames is proposed in (Debreuve et al., 2001). This segmentation process is geometrically-constrained to be region-based, with information inside object/background region being integrated over the entire sequence. However, the evolution of object contour is performed separately for each frame, with a minimum-length regularization term. The method is implemented using level-set framework and applied to 3-D medical data.

2.1.1 Two-frame motion detection – a reference approach

In Chapter 3 we will present our spatio-temporal motion detection method. In order to evaluate performance of the new approach, we will use a two-frame motion detection method proposed by Jehan-Besson and Barlaud (Jehan-Besson et al., 2000). Our detection method (Chapter 3) can be viewed as a simple extension of this algorithm to space-time; the underlying models are the same in both approaches, but the reference algorithm is performed in spatial domain while the proposed approach is carried out in spatio-temporal domain. The reference algorithm partitions the spatial domain Ω by means of the following minimization:

$$\min_{\vec{\gamma},\boldsymbol{p},\bar{\boldsymbol{p}}} \alpha \iint_{\mathcal{R}} \xi(\boldsymbol{x},t;\boldsymbol{p})d\boldsymbol{x} + \iint_{\bar{\mathcal{R}}} \xi(\boldsymbol{x},t;\bar{\boldsymbol{p}})d\boldsymbol{x} + \lambda \int_{\mathcal{L}} d\vec{\gamma}, \tag{2.1}$$

where $\vec{\gamma} = \partial\mathcal{R}$ is a parameterized curve, region $\mathcal{R}$ is inside of $\vec{\gamma}$, its complement $\bar{\mathcal{R}}$ is outside of $\vec{\gamma}$ ($\Omega = \mathcal{R} \cup \bar{\mathcal{R}}$), and $\mathcal{L}$ is the length of the boundary curve $\vec{\gamma}$. In the formula above, $\boldsymbol{x}$ denotes spatial position and t denotes time, while $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$ are motion

parameters for the regions $\mathcal{R}$ and $\bar{\mathcal{R}}$, respectively.

Assuming stationary background, we propose, after Jehan-Besson and Barlaud (Jehan-Besson et al., 2000), absolute frame difference as the measure of background intensity variation:

$$\xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) = |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)|, \tag{2.2}$$

and a fixed penalty within the object:

$$\xi(\boldsymbol{x}, t; \boldsymbol{p}) = 1. \tag{2.3}$$

In order to attain the global minimum in (2.1), the contour $\vec{\gamma}$ must partition the domain so that points $(\boldsymbol{x}, t)$ with small frame difference (small $\xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$) are assigned to the outside ($\bar{\mathcal{R}}$), and those with large difference – to the inside ($\mathcal{R}$). The balance between such assignments is controlled by α. If an image sequence has a stationary background, frame difference in the background region will correspond to the noise level in the sequence. The parameter α serves as a threshold – points $(\boldsymbol{x}, t)$ with frame differences smaller than α will be assigned to the background ($\bar{\mathcal{R}}$), and those with differences larger than α to the object region ($\mathcal{R}$).

We can write active contour evolution equation for (2.1) as follows:

$$\frac{\partial \vec{\gamma}}{\partial \tau} = [\alpha - |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)| + \lambda\kappa]\vec{n}. \tag{2.4}$$

Ignoring the curvature and remembering that $\vec{n}$ is the inward unit normal, $\alpha > |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)|$ will result in the contour shrinking and thus relinquishing the point $(\boldsymbol{x}, t)$, while $\alpha < |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)$ will cause the contour to expand thus enclosing this point. As for the curvature κ, it plays the role of a smoothing filter with respect to curve point coordinates. By embedding active contour $\vec{\gamma}$ as the zero-level set of the surface ϕ, we

obtain the following level-set evolution equation:

$$\frac{\partial \phi}{\partial \tau} = [\alpha - |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)| + \lambda\kappa]\|\nabla\phi\|,$$

where τ is the evolution time and κ is a 2-D curvature of ϕ:

$$\kappa_m = \nabla\frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}.$$

Evolution is executed separately for each frame pair (i.e., each t and $t-1$).

2.1.2 Two-frame motion-based segmentation – a reference approach

Similarly to the motion detection algorithm, we need to validate the new video segmentation approach (to be presented in Chapter 4) against a method using motion compensation but only over two frames. We selected a recent method proposed by Mansouri and Konrad (Mansouri and Konrad, 2003). The basic formulation of their approach uses minimization (2.1) but with error measures $\xi(\boldsymbol{x}, t; \boldsymbol{p})$ redefined as follows:

$$\xi(\boldsymbol{x}, t; \boldsymbol{p}_i) = (I(\boldsymbol{p}_i\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1))^2,\ i = 1, 2,$$

where the motion parameters for the object and background are $\boldsymbol{p}_1 = \boldsymbol{p}$ and $\boldsymbol{p}_2 = \bar{\boldsymbol{p}}$. We use motion detection results calculated using previous, motion detection two-frame method to obtain estimates of motion parameters $\boldsymbol{p}_i$. We again minimize the energy functional (2.1), with $\alpha = 1$, and as a result obtain the following level-set evolution equation:

$$\frac{\partial \phi}{\partial t} = [\xi^2(\boldsymbol{x}, t; \boldsymbol{p}) - \xi^2(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) + \lambda\kappa]\|\nabla\phi\|.$$

2.2 Multiple-frame methods

In order to speed up the convergence and improve precision (by avoiding local minima), approaches based on two frames usually use previous-frame segmentation maps to initialize subsequent segmentations. Another approach is to perform tracking of segments between frames, either by including a tracking term into active contour model or by performing additional tracking step after the segmentation step is completed. However, an approach more interesting to us is to perform the segmentation jointly over several frames. Some early work using multiple frames includes motion detection using 3-D MRF models (Luthon et al., 1999). The authors propose two algorithms: a 3-D separable MRF model which uses motion information from three consecutive frames but only the current frame is processed at each time, and a 3-D non-separable model where observation and label fields are spatio-temporal 3-D random fields. Observations are still differences between consecutive frames and the neighborhood structure is a complete spatio-temporal cube. However, the resulting energy function is jointly optimized in spatio-temporal domain using a spatio-temporal version of the ICM algorithm. A multiresolution version of the 3-D non-separable algorithm has been also proposed to deal with uniform intensity moving areas and sub-pixel motion.

A video segmentation approach based on volume growing is proposed in (Porikli and Wang, 2001). Image frames between two scene cuts are organized in a 3-D "video-cube", marker points are selected as minimum gradient magnitude points and the volumes are enlarged iteratively from the markers using color/texture distance criteria. Self-descriptors for each volume and mutual descriptors for each pair of volumes are computed, capturing motion and spatial information of volumes. In the clustering stage, volumes are classified into objects in a fine-to-coarse clustering hierarchy. In each iteration, the pair of volumes with maximum descriptor-based similarity score is merged.

Parker and Magarey (Parker and Magarey, 2001) proposed a 3-D extension of the

2-D Mumford-Shah region-merging segmentation algorithm. In the 2-D merging algorithm, segmented image is represented as a simple, undirected, weighted graph, with vertices representing regions and the edges representing the boundaries between regions. Initially, each pixel is added to the graph as a separate region. In each iteration, two neighboring regions with minimum merging cost are merged and the graph is updated. This algorithm is easily generalized to 3-D segmentation by replacing pixels with voxels and four direct 2-D neighbors with six direct neighbors in 3-D. This 3-D segmentation algorithm is applied to a fixed-size spatio-temporal window sliding through time – new unsegmented frames are added to the graph while fully-segmented frames are removed from the back of the sliding window.

More recently, a novel concept of object tracking as spatio-temporal motion boundary detection has been investigated (Feghali et al., 2001; Mitiche et al., 2002). In (Feghali et al., 2001), a variational formulation for motion boundary detection in spatio-temporal space is proposed and implemented using the level-set method. The authors use the absolute value of normal component of the optical velocity as a measure of motion activity and the minimization of their energy criterion yields a level-set surface enfolding the volumes of motion activity. A similar method is proposed by the same authors in (Mitiche et al., 2002), where the estimation of the motion boundary surface is posed as MAP sequence partitioning problem. The same motion activity measure is used as in (Feghali et al., 2001), but with an explicit modeling of both static and moving regions. In addition to the standard minimum-surface assumption, the prior model encourages surfaces which are motion boundaries. Minimization of the resulting energy functional yields a spatio-temporal surface biased toward smooth closed surfaces which partition the image sequence into volumes of contrasting motion activity, coincide with motion boundaries, and have a small area. Both methods can handle moving cameras if the image motion field is locally approximately constant everywhere except at motion

boundaries.

Multiple-image framework has led to interesting space-time image sequence segmentation methods developed by Mansouri *et al.* (Mansouri et al., 2002; Mansouri and Mitiche, 2002) and, independently, by us (Konrad and Ristivojević, 2002). In (Mansouri et al., 2002), motion segmentation in spatio-temporal domain is formulated as a MAP estimation and solved using the level-set method. For the observation model, variation of intensities over object and background motion trajectories is represented using standard Brownian motion. Minimum boundary surface area is used as the prior model. The authors simplify the problem by considering only motion trajectories associated with constant-velocity translational motion, and estimate this motion prior to segmentation. This MAP formulation leads to a volume competition energy minimization problem, with two motion models competing for each boundary point through motion-compensated residuals. They proposed extension of the method to the estimation of multiple moving regions using a separate level-set function for each region. In (Mansouri and Mitiche, 2002), motion segmentation problem is formulated as a pursuit in the space of image segmentations. They propose an algorithm for spatial motion segmentation, but claim that it can be easily extended to the joint spatio-temporal case similarly to (Mansouri et al., 2002). Given a set of sparse correspondence points, a parametric transformation (e.g. translational, Euclidean, affine, ...) is estimated based on a projection of a square block around each correspondence point. Estimation of multiple object regions and the corresponding motion transformations is posed as a MAP problem, with standard minimum-length boundary prior. However, instead of solving the problem for all regions simultaneously, a transformation with the minimal estimation error is chosen and region corresponding to that transformation is estimated through level-set evolution. At the next step of the pursuit, only the residual of motion regions obtained in the previous step is segmented. The previously-estimated regions

can change since at each iteration a system of level-set equations is evolved, one for each region estimated thus far. The main difference of our approach is that we propose to estimate motion of the object simultaneously with its segmentation. Most recently, spatio-temporal motion detection method with background (camera) motion compensation has been developed by Feghali and Mitiche (Feghali and Mitiche, 2004). They pose the problem as joint estimation of motion boundary surface and background motion parameters through MAP framework. Similarly to (Mitiche et al., 2002) they use the normal component of optical velocity as motion measurement. The difference between that measurement and the normal component due to camera motion is used in the observation model for object and background region. With the standard minimal-boundary surface prior, this formulation leads to volume-competition energy functional, which is simultaneously minimized with respect to motion parameters and segmentation surface. Camera motion is assumed to be constant-velocity translational motion. This method is described in more detail in Section 7.1.

An extension of the active contour/level-set framework to multiple-object segmentation is very easy if the multiphase level-set methodology is used. This is in contrast to previous variational approaches to multiple-motion segmentation, where additional constraints are needed (Mansouri and Konrad, 2003). Since separate level-set function is used to describe each moving object, solutions obtained may not form a partition of the image domain – points belonging to two or more regions or not belonging to any region may exist. This problem cannot be completely removed even with additional constraints in level-set evolution equations, but it is handled naturally in the multiphase level-set method. Motion segmentation methods based on a multiphase level-set solution proposed in the literature consider only two frames of an image sequence at a time (Cremers, 2003; Shi et al., 2004; Mansouri et al., 2004). In (Cremers, 2003), the problem is formulated as extension of the Mumford-Shah functional from the case of gray-level

segmentation to the case of motion segmentation. Minimizing a functional with respect to the segmentation boundary and the set of motion vectors jointly solves the problem of segmentation and motion estimation. The authors assume constant-velocity field in each region. A variational framework for simultaneous segmentation of multiple motions and occlusions is proposed in (Shi et al., 2004). The proposed energy functional consists of a term that penalizes matching error for each motion transformation, a constant-cost occlusion term, and a minimum-length boundary term. In the first stage of algorithm, motion classification and estimation is performed using feature-based method, with motion transformation in each region assumed to be affine mappings. In the second stage, motion and occlusion segmentation is performed using the multiphase level-set method. A new representation of partition of an image domain into a fixed but arbitrary number of regions by explicit correspondence between the regions of segmentation and the regions defined by simple closed planar curves and their intersections is investigated in (Mansouri et al., 2004). This representation is used in the context of region competition for intensity, motion, and disparity based segmentation.

In terms of the detection of occlusions, methods proposed to date are primarily based on the analysis of 2-3 frames (Thoma and Bierling, 1989; Driessen and Biemond, 1991; Depommier and Dubois, 1992; Heitz and Bouthemy, 1993; Irani and Peleg, 1993; Lim et al., 2002). A method using more frames was proposed in (Chahine and Konrad, 1995). The authors propose to estimate motion trajectories over several frames using MAP estimation framework and MRF models. Each motion trajectory is modeled using quadratic parametric model (velocity and acceleration), and the parametric field is assumed to be spatially smooth. For structural model, sample variance is used as a measure of intensity variation over motion trajectories. The resulting energy minimization problem is solved using Gauss-Seidel relaxation. To improve motion estimation results, occlusion and motion discontinuity fields are introduced and estimated simultaneously

with the motion field. The structural model is changed to account for occlusion information, while the prior model is modified to incorporate motion discontinuities. Occlusion fields are estimated from five frames, and their model includes penalty for states different from visible, while it favors the creation of continuous occlusion regions only near motion discontinuities. It is very difficult to estimate temporal discontinuity based on just a few samples; using more frames improves the reliability of occlusion detection.

2.3 Optimization and implementation tools

Different optimization methods, both deterministic and stochastic, were successfully used in the literature. Algorithms like Highest Confidence First (HCF) (Chou and Brown, 1990), Iterated Conditional Modes (ICM) (Besag, 1986), or multiresolution iterative deterministic optimization belong to the former group. Simulated annealing (Geman and Geman, 1984) and other stochastic relaxation methods represent the latter group. However, we will concentrate on methods that use active-contour/level-set optimization framework.

2.3.1 Level-set method

Developed originally for the modeling of flame propagation, the level-set approach has found numerous applications in computer vision and image processing: object shape recovery (Malladi et al., 1995), 3-D object segmentation (Caselles et al., 1997b), Mumford-Shah problem of simultaneous image smoothing and segmentation (Tsai et al., 2000), and segmentation of a moving object against still background over a sequence of 2-D or 3-D image frames (Debreuve et al., 2001). In this section we will describe basic ideas behind the level-set method (detailed description can be found in (Sethian, 1996b)).

Consider a boundary, either a curve in two dimensions or a surface in three dimensions, separating one region from another, as shown on Fig. 2·1. This curve/surface

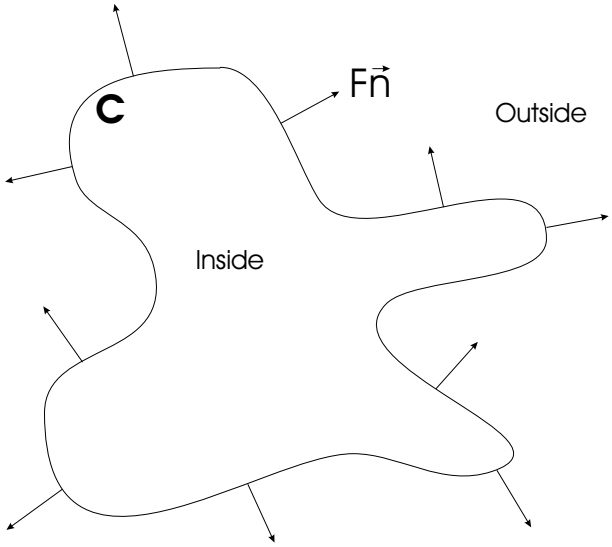


Figure 2·1: Curve propagating with speed F in the normal direction.

moves in a direction normal to itself with a known speed function F according to the following equation:

$$\frac{dC}{dt} = F\vec{n},$$

where $\vec{n}$ is the outward normal to the curve C. The goal is to track the motion of this interface as it evolves. In the level-set method, this evolution problem is formulated in an Eulerian framework, that is, one in which the underlying coordinate system remains fixed.

We will assume for the moment that $F > 0$ (the front always moves "outward"). One way to characterize the position of this expanding front is to compute the arrival time $T(x, y)$ of the front as it crosses each point (x, y). The equation for this arrival function $T(x, y)$ is easily derived. Using the fact that $distance = rate * time$, in one dimensional case we have that

$$1 = F\frac{dT}{dx}.$$

In the multi-dimensional case, ∇T is orthogonal to the level sets of T and its magnitude

is inversely proportional to the speed:

$$|\nabla T|F = 1, \quad T = 0 \text{ on } \Gamma, \tag{2.5}$$

where Γ is the initial location of the interface. The front motion is characterized as the solution to a boundary value problem. If the speed F depends only on position, equation (2.5) reduces to what is known as the "Eikonal" equation.

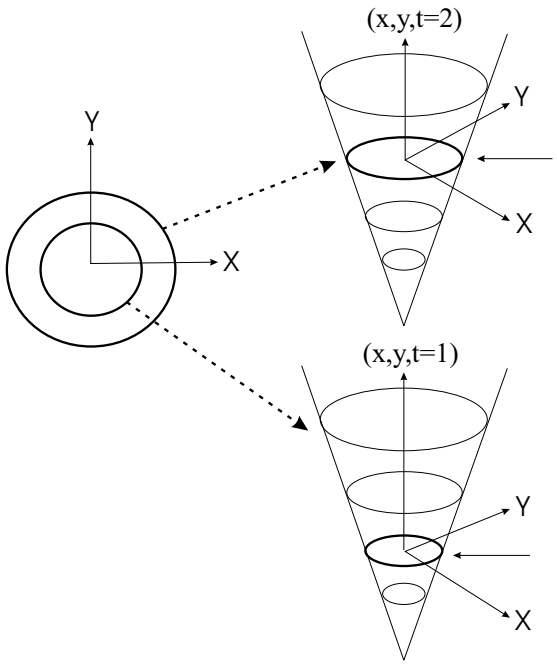


Figure 2·2: Transformation of front motion into the initial value problem.

Let's assume now that the front moves with speed F that is neither strictly positive nor negative. The front can move forward and backward, crossing a point (x, y) several times, so the crossing time $T(x, y)$ is not a single-valued function. One way to solve this problem is to embed the initial position of the front as the zero level-set of a higher-dimensional function ϕ. The evolution of this function ϕ can then be linked to the propagation of the front itself through a time-dependent initial value problem. At any time, the front is given by the zero-level set of the time-dependent level-set function ϕ

(see Fig. 2·2).

In order to derive an equation of the motion for this level-set function ϕ and match the zero-level set of ϕ with the evolving front, we first require that the level-set value of a particle on the front with path $x(t)$ must be always zero:

$$\phi(x(t), t) = 0.$$

Using the chain rule, we obtain:

$$\phi_t + \nabla\phi(x(t), t) \cdot x'(t) = 0.$$

The force F is the speed in the outward normal direction, so $x'(t) \cdot \vec{n} = F$, where $\vec{n} = \nabla\phi/|\nabla\phi|$. This yields an evolution equation for ϕ:

$$\phi_t + F|\nabla\phi| = 0, \text{ given } \phi(x, t = 0). \tag{2.6}$$

This is the level-set equation given by Osher and Sethian (Osher and Sethian, 1988). For certain forms of the speed function F, (2.6) becomes standard Hamilton-Jacobi equation.

Both of these level-set formulations, the initial-value and boundary-value perspective, have important advantages when compared to active-contour formulations.Topological changes in the evolving front are handled automatically. Both formulations are unchanged in higher dimensions, for hypersurfaces propagating in three dimensions or higher, and can be accurately approximated by computational schemes which exploit techniques borrowed from the numerical solutions of hyperbolic conservation laws. Both methods are made more efficient through the use of adaptive computational strategies, which will be described in Section 2.3.3. Finally, geometric properties of the front are easily determined in both formulations. For example, at any point of the front, the

outward normal vector is given by

$$\vec{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad \text{or} \quad \vec{n} = \frac{\nabla T}{|\nabla T|}, \tag{2.7}$$

and the curvature of the front at any point is easily obtained from the divergence of the normal vector, i.e.,

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} \quad \text{or} \quad \kappa = \nabla \cdot \frac{\nabla T}{|\nabla T|}, \tag{2.8}$$

2.3.2 Level-set methodology for active surfaces

Although the level-set approach implements curve evolution and has been known for a number of years, only recently has it been related to snakes. Caselles *et al.* (Caselles et al., 1997a) showed that energy-minimizing active contours (snakes) are related to the level set formalism by means of *geodesic active contours* (i.e., minimal distance paths) in a Riemannian space. In particular, if $\vec{\gamma}(p) = [x(p)\ y(p)]^T : [0,1] \rightarrow R^2$ is a parameterized planar curve and $I : \Omega \rightarrow R^+$ is an image intensity, it is shown that the snakes formulation, void of the (second-order) rigidity term, is equivalent to geodesic computation:

$$\min_{\vec{\gamma}} \int_0^1 |\dot{\vec{\gamma}}(p)|^2 dp - \lambda \int_0^1 |\nabla I(\vec{\gamma}(p))| dp \quad \rightsquigarrow \quad \min_{\vec{\gamma}} \int_0^1 g(|\nabla I(\vec{\gamma}(p))|)|\dot{\vec{\gamma}}(p)| dp \tag{2.9}$$

where $\dot{\vec{\gamma}} = \partial\vec{\gamma}/\partial p$, and $g(.)$ is a strictly decreasing function ($g(x) \rightarrow 0$ as $x \rightarrow 0$). Note, that in the geodesic formulation, the Euclidean length $|\dot{\vec{\gamma}}(p)|dp$ is weighted by $g(|\nabla I(\vec{\gamma}(p))|)$, and, therefore, instead of finding a classical minimal-length path (as achieved by minimizing $\int |\dot{\vec{\gamma}}(p)|dp$), one looks for a minimal-length path obeying image characteristics expressed through $g(|\nabla I(\vec{\gamma}(p))|)$.

Minimization of (2.9) is performed using the steepest-descent method. It is shown that initial curve $\vec{\gamma}(0,p)$ (t is evolution time) deforms towards (local) minimum of (2.9)

based on the following curve evolution equation:

$$\frac{\partial \vec{\gamma}(t,p)}{\partial t} = g(|\nabla I(\vec{\gamma}(t,p))|)\kappa\vec{n}_i - (\nabla g(|\nabla I(\vec{\gamma}(t,p))|) \cdot \vec{n}_i)\vec{n}_i, \tag{2.10}$$

where κ is the Euclidean curvature and $\vec{n}_i$ is the unit inward normal of $\vec{\gamma}$. Caselles *et al.* (Caselles et al., 1997a) proceeded to represent equation 2.10 using the level-set approach. We will follow their derivation. Starting from the following curve evolution equation (for a given function β)

$$\frac{\partial \gamma}{\partial t} = \vec{\gamma}_t = \beta\vec{n}_i, \tag{2.11}$$

they want to represent $\vec{\gamma}$ as the level-set of a function $\phi : R^2 \rightarrow R$ and to derive how ϕ should evolve. They give a very simple geometrical derivation of this embedding process. Assume that ϕ is negative in the interior of the zero-level set (usually, the signed distance function is used for ϕ) and consider a level-set, defined by

$$\{\Gamma \in R^2 : \phi(\Gamma, t) = 0\}. \tag{2.12}$$

They want to find the evolution of $\phi(t)$ such that the evolving curve $\vec{\gamma}(t)$ is represented by the evolving zero-level set $\Gamma(t)$ (i.e., $\vec{\gamma}(t) \equiv \Gamma(t)$). By differentiating (2.12) with respect to t they obtain

$$\nabla\phi \cdot \Gamma_t + \phi_t = 0. \tag{2.13}$$

Based on (2.7), for any level-set, the following relationship holds:

$$\frac{\nabla\phi}{|\nabla\phi|} = -\vec{n}_i. \tag{2.14}$$

Since $\vec{\gamma}(t) \equiv \Gamma(t)$, by combining relationships (2.11), (2.13), and (2.14) they obtain the level-set evolution equation:

$$\phi_t = \beta|\nabla\phi|. \tag{2.15}$$

Based on (2.9) and embedding (2.10) in ϕ for $\beta = g(|\nabla I|)\kappa - (\nabla g(|\nabla I|) \cdot \vec{n})$, Caselles *et al.* (Caselles et al., 1997a) obtained that solving the geodesic problem is equivalent to searching for the steady-state solution ($\frac{\partial \phi}{\partial t} = 0$) of the following evolution equation (with initial conditions $\phi(0, \vec{\gamma}) = \phi_0(\vec{\gamma})$):

$$\frac{\partial \phi}{\partial t} = g(|\nabla I|)\kappa|\nabla \phi| - \nabla g(|\nabla I| \cdot \nabla \phi, \tag{2.16}$$

where the curvature κ is computed on the level-sets of ϕ using (2.8).

In order to formulate image sequence segmentation jointly over several frames, it is natural to consider an extension of active contours (2-D) to active surfaces (3-D). This leads to minimal-surface formulations that have been applied to 3-D shape recovery (Malladi et al., 1995; Caselles et al., 1997b). A path particularly pertinent to this thesis has been undertaken by Caselles *et al.* (Caselles et al., 1997b). Let $I : \Omega \times \mathcal{T} \rightarrow R^+$ be intensity and let $\vec{\varsigma}$ be a surface in 3-D space with area $\mathcal{S}$. By parameterizing the surface, $\vec{\varsigma}(p, q) : [0, 1] \times [0, 1] \rightarrow R^3$ with $p = (x, y, z)$ and $q = q(x, y, z)$, Caselles *et al.* have proposed to compute the minimal surface enclosing a 3-D object as follows:

$$\min_{\vec{\varsigma}} \iint_{\mathcal{S}} g(\delta I) d\vec{\varsigma} \quad \rightsquigarrow \quad \frac{\partial \vec{\varsigma}}{\partial \tau} = [g(\delta I)\kappa_m - \nabla g(\delta I) \cdot \vec{n}_i]\vec{n}_i,$$

where $g(\cdot)$ is a strictly decreasing function, δI is a measure of intensity variation, $d\vec{\varsigma}$ is a Euclidean area element, κ_m is the mean curvature and $\vec{n}_i$ is the inward unit normal to $\vec{\varsigma}$. The term $g(\delta I)\kappa_m \vec{n}_i$ smoothes out the surface by reducing its curvature, unless $g(\delta I)$ is zero which means a large intensity change (e.g., perfect edge). The term $(\nabla g(\delta I) \cdot \vec{n}_i)\vec{n}_i$ "pushes" the contour toward an intensity edge as long as the orthogonal component of ∇g is non-zero. This term allows locking to edges with intensity variations or even gaps along the edge. This approach has been further extended by the same authors (Caselles

et al., 1997b):

$$\min_{\vec{\varsigma}} \iiint_{\mathcal{V}} f(\delta I) d\varpi + \lambda \iint_{\mathcal{S}} g(\delta I) d\vec{\varsigma} \quad \rightsquigarrow \quad \frac{\partial \vec{\varsigma}}{\partial \tau} = [f(\delta I) + g(\delta I)\kappa_m - \nabla g(\delta I) \cdot \vec{n}_i]\vec{n}_i, \quad (2.17)$$

where the new term is a measure of the Euclidean volume element $d\varpi$ weighted by $f(\delta I)$, and $f(\cdot)$ is another strictly decreasing function. The new term adds a constant "balloon" force $f(\delta I)\vec{n}_i$ to the surface evolution equation helping avoid local minima and speeding up convergence. Note that in the above formulation, the evolution force vanishes at $\delta I \to \infty$, i.e., at a perfect edge; objects with smooth intensity transitions cannot be detected.

Similarly to active contours, active-surface solutions (2.17) suffer from stability problems and fixed topology, which both can be overcome using level-set methodology. By embedding the active surface $\vec{\varsigma}$ into a hyper-surface ϕ (in 4-D space) leads to the following level-set evolution equation:

$$\frac{\partial \phi}{\partial \tau} = F\|\nabla\phi\| = [f(\delta I) + g(\delta I)\kappa_m - \nabla g(\delta I) \cdot \vec{n}_i]\|\nabla\phi\|.$$

2.3.3 Computationally-efficient level-set implementations

Efficiency of the level-set method was improved over the years, by the introduction of several algorithms. The straightforward approach to the level-set method is to solve the initial value partial differential equation (PDE) (2.6) for the level-set function ϕ in the entire computational domain. This is called a "full matrix approach" since one is updating all level sets, not just the zero-level set corresponding to the front itself. When only the zero-level set is of interest, the *Narrow Band* method has been proposed (Adalsteinsson and Sethian, 1995) to accelerate the evolution process. A tube is constructed around the zero-level set and the level-set function is initialized as a signed distance function within this tube. Only the values of ϕ within this narrow band are updated.

When the front moves near the edge of the tube boundary, the calculation is stopped, and a new tube is built with the zero-level set at the center. This rebuilding process is known as "re-initialization". One variant of the Narrow Band level-set method is to characterize the narrow band by the values of the level-set function ϕ, rather than by distance in the domain space, and dynamically add grid points to the narrow band as it moves based on the values of the level-set function. Points whose $|\phi|$ values go above a certain threshold level are removed, while neighbors are added around those that dip below the threshold. As new grid points are added, the ϕ function in the entire narrow band is reinitialized to the signed distance function, and the calculation is advanced one time step. Advantages of the Narrow Band level-set method compared to the standard "full matrix approach" are numerous. In addition to performing calculations only in the neighborhood of the zero-level set, extension of the speed function F has to be done only for the points lying near the front. Also, in the narrow band implementation, the time step can be adaptively chosen in response to the maximum velocity field achieved within the narrow band, not the entire domain.

The *Fast Marching* algorithm developed by Sethian (Sethian, 1996a) allows one to efficiently solve the boundary value problem (Eikonal equation (2.5)) without iterations. Using an upwind scheme, (2.5) can be discretized as follows:

$$\begin{bmatrix} \max(D_{ijk}^{-x}T, -D_{ijk}^{+x}T, 0)^2 \\ \max(D_{ijk}^{-y}T, -D_{ijk}^{+y}T, 0)^2 \\ \max(D_{ijk}^{-z}T, -D_{ijk}^{+z}T, 0)^2 \end{bmatrix}^{1/2} = \frac{1}{F_{ijk}}. \tag{2.18}$$

The central idea behind the Fast Marching method is to systematically construct the solution T using only upwind values. Upwind difference structure of (2.18), allows us to propagate information from smaller values of T to larger values. Starting from the boundary condition, we build the solution outward from the smallest T value. We sweep

the front along considering points in a thin zone around the existing front, marching this zone forward, freezing the values of existing points and bringing new ones into the narrow band structure. The grid point in this narrow band with the smallest value for T is located using a min-heap data structure and its value is fixed, new values of T at each of the four neighboring grid points are calculated using (2.18), and these points are included into the narrow band. The procedure is repeated until the whole domain has fixed values of T. There are few reasons Fast Marching method is usually even more efficient than Narrow Band method. Since there is no notion of time step in the Fast Marching method, the speed F of the front is irrelevant to the efficiency of the method. Also, the number of the elements of the heap depends on the length of the front; in most cases, this length is small enough that, for all practical purposes, the sort is very fast and essentially $O(1)$.

A combination of Fast Marching and Narrow Band methods was proposed by Paragios and Deriche (Paragios and Deriche, 2000), under the name *Hermes Algorithm*. It employs the idea of selective propagation (Fast Marching) over a relatively small window (Narrow Band). At each step, a pixel of the front with the highest absolute propagation velocity is chosen using heap sort, a centralized circular window is defined around this pixel and the level-set function is updated locally within this window. When the local level-set evolution is completed, the new front is extracted within the local window and the level-set function is reinitialized locally. The procedure is repeated until certain number of iterations is reached or the front does not further move. The proposed algorithm does not solve exactly the given PDE in terms of the intermediate levels of the curve evolution but the final solution is obtained very fast and corresponds to the solution of the original PDE.

More recently, even a more efficient algorithm for computing the numerical solution of Hamilton-Jacobi equations (example of which are Eikonal equations), the *Fast*

Sweeping method, was proposed by Tsai *et al.* (Tsai et al., 2003). It provides a way to compute solutions to a class of Hamilton-Jacobi equations for which the conventional fast marching method is not applicable. The basic idea of the "sweeping" approach is to calculate solution by visiting each grid node in some predefined order. There is no need for heap sort data structure to determine the grid point to be updated, and the solution is obtained after just few sweeps. Using this approach, the complexity of the algorithms drops from $O(N \log N)$ in fast marching to $O(N)$.

We will give more detailed description of the implementation of the level-set method we used in our detection and segmentation algorithms in Section 3.2. Recently, a novel level-set implementation was developed by Shi (Shi, 2005) which dramatically reduces computational cost of the algorithm and we will describe it in detail in Chapter 8.

2.3.4 Multiphase level-set methodology

For our multi-frame motion-compensated segmentation algorithms described in Chapters 4–6, we use *multiphase* framework (Vese and Chan, 2002), which allows efficient representation of up to 2^M regions with only M level-set functions. By design, the multiphase level-set method enforces a domain partition with no gaps and overlaps, thus simplifying energy formulation when compared to the "one object – one level-set surface" approach (Mansouri and Konrad, 2003). We will briefly describe the multiphase level-set representation introduced by Vese and Chan (Vese and Chan, 2002) (that paper also contains good overview of the development of multiphase level-set idea).

Using one level-set function, we can represent only two segments (phases) in an image (or an image sequence). The multiphase level-set model allows us to represent more than two segments, together with triple junctions and other complex topologies, in an efficient way. We need only $\log_2 N$ level-set functions to represent N segments (compared to $N-1$ level-set functions required when each segment is represented by a separate function).

In addition, this formulation automatically removes problems of vacuums and overlaps, because this partition is a disjoint decomposition and covering of an image (sequence) domain Ω by definition. We proceed to give more formal mathematical definition.

Let us consider $M = \log_2 N$ level-set functions $\phi_i : \Omega \to \mathcal{R}$. The union of the zero-level sets of ϕ_i will represent the edges in the segmented image (or image sequence). We also introduce the "vector level-set function" $\Phi = (\phi_1, \ldots, \phi_M)$ and the "vector Heaviside function" $H(\Phi) = (H(\phi_1), \ldots, H(\phi_M))$ whose components are only 1 or 0. We can now define segments in the domain Ω in the following way: two pixels (x_1, y_1) and (x_2, y_2) in Ω will belong to the same phase (or class), if and only if $H(\Phi(x_1, y_1)) = H(\Phi(x_2, y_2))$. In other words, the phases are given by the level sets of the function $H(\Phi)$, i.e., one phase is formed by the set

$$\{(x, y) \in \Omega | H(\Phi(x, y)) = \text{constant vector} \in H(\Phi(\Omega))\}.$$

There are up to $N = 2^M$ possibilities for the vector-values $H(\Phi(\Omega))$ in the image (image sequence). In this way, we can define up to $N = 2^M$ phases (classes) in the domain Ω. The phases defined in this manner form a disjoint decomposition and covering of Ω. Therefore, each pixel $(x, y) \in \Omega$ will belong to one, and only one class, by definition, and there is no vacuum or overlapping among the phases. The set of boundary curves (surfaces) is represented by the union of the zero-level sets of the functions ϕ_i.

Chapter 3

Multi-frame motion detection using active surfaces

Motion detection methods based on the processing of image sequence in groups of two frames at a time produce results with no temporal consistency. By differentiating one frame from the other they produce separate segmentations of each frame of image sequence into 2-D regions of motion activity. In order to improve precision, previous-frame detection maps are often used to initialize subsequent detections. Temporal constraining of the results can be introduced through post-processing by performing an additional tracking step after the detection is completed. Another approach is to causally constrain the current detection process with a previous-frames detection result. However, in any of these cases, future frames are not taken into account. Only by looking at multiple frames jointly the temporal consistency of moving regions can be fully exploited and lead to better detection results. In this chapter, we propose to perform segmentation jointly over multiple frames, which leads to a 3-D segmentation, i.e., search for a volume "carved out" by a moving object in the 3-D image sequence domain, which we call *object tunnel*. After developing general segmentation formulation in Section 3.1, we simplify it for the case of object moving against still background in Section 3.2, which leads to a method which detects moving areas in a sequence.

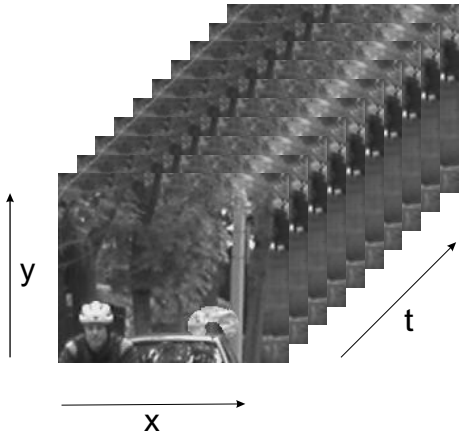


Figure 3·1: Example of image sequence and its spatio-temporal (x−y−t) domain.

3.1 General MAP formulation

We seek a parameterized surface $\vec{\varsigma}$ that delineates moving objects in an image sequence $I(\boldsymbol{x}, t)$. Let I_t, defined on Ω, be one frame of the image sequence captured at time t, and let $\mathcal{I}^t = \{I_\tau : \tau \in \mathcal{T}\}$ be a subset of the sequence $I(\boldsymbol{x}, t)$ based on which $\vec{\varsigma}$ is estimated (Fig. 3·1 shows an example of such image sequence). Clearly, $\Omega \times \mathcal{T}$ is the (spatio-temporal) domain of the sequence $I(\boldsymbol{x}, t)$ on which $\vec{\varsigma}$ is computed ($\boldsymbol{x} \in \Omega, t \in \mathcal{T}$). Let $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$ be motion parameters (e.g., affine with constant or slowly-varying velocity (Konrad and Stiller, 1997)) for the volume inside and outside of $\vec{\varsigma}$, respectively; we assume that motion trajectory for each image sequence point can be computed either from $\boldsymbol{p}$ or $\bar{\boldsymbol{p}}$.

We pose the problem of computing $\vec{\varsigma}$ in the framework of maximum *a posteriori* probability (MAP) estimation. Similarly to the two-frame formulation (Mansouri and Konrad, 2003), the MAP multiple-frame segmentation can be expressed as follows:

$$\max_{\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}} p(\vec{\varsigma}, \boldsymbol{p}, \bar{\boldsymbol{p}}|\mathcal{I}^t) = \tag{3.1}$$
$$\max_{\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}} \frac{p(I_t|\vec{\varsigma}, \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}) p(\vec{\varsigma}, \boldsymbol{p}, \bar{\boldsymbol{p}}|\mathcal{I}^t \backslash \{I_t\}) p(\mathcal{I}^t \backslash \{I_t\})}{p(\mathcal{I}^t)},$$

where p denotes probability density, and $\mathcal{I}^t \backslash \{I_t\}$ represents all frames of the sequence $\mathcal{I}^t$ except for the frame I_t at time t. Given that $p(\mathcal{I}^t \backslash \{I_t\})$ and $p(\mathcal{I}^t)$ are independent of $\vec{\varsigma}$, $\boldsymbol{p}$, $\bar{\boldsymbol{p}}$, optimization (3.1) will lead to the same maximum as:

$$\begin{aligned} &\max_{\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}} \; p(I_t|\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})p(\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}|\mathcal{I}^t\backslash\{I_t\}) = \\ &\max_{\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}} \; p(I_t|\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})p(\vec{\varsigma}|\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})p(\boldsymbol{p},\bar{\boldsymbol{p}}|\mathcal{I}^t\backslash\{I_t\}). \end{aligned} \tag{3.2}$$

The three terms in the above maximization are:

- $p(I_t|\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})$ – likelihood term: probability of the current frame I_t given all other frames of the sequence $\mathcal{I}^t\backslash\{I_t\}$, segmentation surface $\vec{\varsigma}$, and motion parameters $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. For example, if intensity over motion trajectory is assumed to be constant plus additive Gaussian noise, then the variation over motion trajectory can be modeled using independent identically-distributed zero-mean Gaussian random process.

- $p(\vec{\varsigma}|\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})$ – prior term on segmentation surface $\vec{\varsigma}$: models prior knowledge about the shape of the segmentation surface, as well as how it depends on the motion parameters $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. Assuming a smooth boundary surface leads to a minimal boundary surface prior term, while an additional term could incorporate consistency of motion trajectories and boundary surface shape.

- $p(\boldsymbol{p},\bar{\boldsymbol{p}}|\mathcal{I}^t\backslash\{I_t\})$ – prior term on motion parameters $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$: models prior knowledge about motion trajectories. This term depends on whether dense or parametric motion model is used, and usually models spatial and temporal smoothness of motion trajectories.

3.1.1 General energy formulation

Once the appropriate models are chosen for the likelihood and priors in MAP formulation (3.2), maximization can be transformed into the following energy-based minimization:

$$\min_{\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}}} \mathcal{E}_1(I_t;\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\}) + \mathcal{E}_2(\vec{\varsigma};\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\}) + \mathcal{E}_3(\boldsymbol{p},\bar{\boldsymbol{p}};\mathcal{I}^t\backslash\{I_t\}). \tag{3.3}$$

Terms in the above minimization correspond to terms in the MAP formulation (3.2):

- $\mathcal{E}_1(I_t;\vec{\varsigma},\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})$ – likelihood term. Different models are proposed for this likelihood term in the literature: standard assumption of constant intensity over motion trajectory, linear variation of intensity over motion trajectory (Negahdaripour et al., 1989), or constant spatial gradient of intensity over motion trajectory assumption (De Micheli et al., 1993).

- $\mathcal{E}_2(\vec{\varsigma};\boldsymbol{p},\bar{\boldsymbol{p}},\mathcal{I}^t\backslash\{I_t\})$ – prior term on segmentation surface $\vec{\varsigma}$. The segmentation surface is usually assumed to be smooth, yielding minimal boundary surface prior. An interesting issue is whether the segmentation surface $\vec{\varsigma}$ depends on motion parameters $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. That dependence can be expressed through some measure of consistency between motion trajectories and segmentation surface.

- $\mathcal{E}_3(\boldsymbol{p},\bar{\boldsymbol{p}};\mathcal{I}^t\backslash\{I_t\})$ – prior term on motion parameters $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. This term depends on the choice of spatial model for motion trajectories: sparse (calculated using block matching or feature correspondence), parametric (homography, projective, affine, bilinear, etc.), or dense (calculated using optical flow). Another issue is the temporal modeling of motion trajectories – whether they consist of individual displacement vectors or some temporal parametric model (quadratic, spline, etc.) is used. Usually, motion trajectories are modeled to be both spatially and temporally smooth, which can be enforced through a parametric model and/or an explicit smoothness constraint.

In order to formulate the segmentation problem, we make two standard assumptions: that image intensity along motion trajectory remains constant, and that the surface $\vec{\varsigma}$ is smooth, e.g., area $\mathcal{S}$ of $\vec{\varsigma}$ is small. We will decompose the minimization (3.3) into two interleaved minimizations: estimation of motion trajectories $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$ with segmentation surface $\vec{\varsigma}$ fixed, and estimation of the segmentation surface with motion trajectories fixed.

Estimation of motion parameters

If the segmentation surface $\vec{\varsigma}$ is fixed ($\vec{\varsigma} = \hat{\vec{\varsigma}}$), minimization (3.3) becomes:

$$\min_{\boldsymbol{p},\bar{\boldsymbol{p}}} \mathcal{E}_1(I_t; \hat{\vec{\varsigma}}, \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}) + \mathcal{E}_2(\hat{\vec{\varsigma}}; \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}) + \mathcal{E}_3(\boldsymbol{p}, \bar{\boldsymbol{p}}; \mathcal{I}^t \backslash \{I_t\}). \tag{3.4}$$

The first term in (3.4), $\mathcal{E}_1$, is a likelihood term, and it will be a measure of variability of image intensity along motion trajectories defined by $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. The second term, $\mathcal{E}_2$, measures only consistency of object tunnel and motion trajectories since the smoothness of segmentation surface term is constant for fixed $\vec{\varsigma}$. This consistency can be measured, for example, through angular difference between tangent to segmentation surface and instantaneous velocity on trajectory at each point on boundary surface. The term $\mathcal{E}_3$ is a prior term on motion trajectories (for example, a measure of their spatial and temporal smoothness).

Estimation of segmentation surfaces

When the motion trajectories are fixed ($\boldsymbol{p} = \widehat{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}} = \widehat{\bar{\boldsymbol{p}}}$), optimization (3.3) reduces to:

$$\min_{\vec{\varsigma}} \mathcal{E}_1(I_t; \vec{\varsigma}, \widehat{\boldsymbol{p}}, \widehat{\bar{\boldsymbol{p}}}, \mathcal{I}^t \backslash \{I_t\}) + \mathcal{E}_2(\vec{\varsigma}; \widehat{\boldsymbol{p}}, \widehat{\bar{\boldsymbol{p}}}, \mathcal{I}^t \backslash \{I_t\}). \tag{3.5}$$

The term $\mathcal{E}_1$ is a likelihood term described above, while the term $\mathcal{E}_2$ is a prior term describing our assumptions about the segmentation surface and its relationship to motion

trajectories. It measures smoothness of the segmentation surface $\vec{\varsigma}$ and how well the object tunnel defined by this surface corresponds to motion trajectories (for example, through angular measure described above).

For the likelihood term, we define $\xi(\boldsymbol{x}, t; \boldsymbol{p})$, for each point $(\boldsymbol{x}, t) \in \Omega \times \mathcal{T}$, to be a measure of variability of image intensity along motion trajectory defined by $\boldsymbol{p}$ and passing through $(\boldsymbol{x}, t)$. We propose to estimate the surface $\vec{\varsigma}$ (given motion trajectories $\boldsymbol{p} = \widehat{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}} = \widehat{\bar{\boldsymbol{p}}}$) delineating object motion by minimizing the following energy functional:

$$\min_{\vec{\varsigma}} \ \alpha \iiint_{\mathcal{V}} \xi(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}) d\boldsymbol{x} dt + \iiint_{\bar{\mathcal{V}}} \xi(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}) d\boldsymbol{x} dt + \lambda \iint_{\mathcal{S}} \zeta(\vec{\varsigma}, \widehat{\boldsymbol{p}}, \widehat{\bar{\boldsymbol{p}}}) d\vec{\varsigma}, \tag{3.6}$$

where $\vec{\varsigma} = \partial\mathcal{V}$, $\mathcal{V}$ is inside of $\vec{\varsigma}$ and $\bar{\mathcal{V}}$ is outside of $\vec{\varsigma}$, and λ associates a cost with the prior term $\zeta(\vec{\varsigma}, \widehat{\boldsymbol{p}}, \widehat{\bar{\boldsymbol{p}}})$. This prior term measures the smoothness of the segmentation surface $\vec{\varsigma}$, and its consistency with motion trajectories $\boldsymbol{p}$ and $\bar{\boldsymbol{p}}$. The parameter α enables us to give different weight to intensity variations over trajectories in the object and background. We leave $\xi(\boldsymbol{x}, t; \boldsymbol{p})$ generic for now; we will define it in detail for each segmentation case studied.

If we decide not to take into account the relationship between the shape of segmentation surface $\vec{\varsigma}$ and motion trajectories when estimating $\vec{\varsigma}$, the minimization (3.6) can be simplified to the likelihood and surface smoothness terms only. Segmentation surface $\vec{\varsigma}$ is estimated by minimizing a functional balancing intensity variability and surface roughness terms as follows:

$$\min_{\vec{\varsigma}} \ \alpha \iiint_{\mathcal{V}} \xi(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}) d\boldsymbol{x} dt + \iiint_{\bar{\mathcal{V}}} \xi(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}) d\boldsymbol{x} dt + \lambda \iint_{\mathcal{S}} d\vec{\varsigma}, \tag{3.7}$$

where λ now associates a cost with the Euclidean area element $d\vec{\varsigma}$. The minimization (3.7) can be interpreted as *volume competition*: the first term measures the compatibility

of image sequence point at $(\boldsymbol{x}, t)$ with the overall intensity and motion inside of $\vec{\varsigma}$, whereas the second term measures such compatibility outside of $\vec{\varsigma}$. The third term assures that a minimal area (smooth) surface is sought. Thus, the minimization process seeks as smooth a surface as possible that divides the domain into such $\mathcal{V}$ and $\bar{\mathcal{V}}$ that each is best explained by its own motion parameters and intensity.

In order to solve (3.7), various approaches are possible. We pursue solutions based on the level-set methodology that we reviewed in Section 2.3.2.

3.2 Motion detection formulation

The geodesic-surface formulation described in Section 2.3.2 has been used in the past for video segmentation. In particular, Feghali *et al.* (Feghali et al., 2001) used $\delta I = |I_t|/(I_x^2 + I_y^2)^{1/2}$, i.e., the normal component of optical velocity (where I_x, I_y, I_t are horizontal, vertical and temporal intensity derivatives, respectively), in (2.17) for motion detection against static background with very encouraging results.

Since a geodesic-surface approach, such as the one in (2.17), requires strong intensity edges, its usefulness is limited. We pursue motion detection based on the volume competition approach that we outlined in Section 3.1. We assume stationary background, a common scenario in security or monitoring applications. This restriction will be lifted in the motion segmentation algorithms in Chapters 4–6. Under the above assumptions, we propose, after Jehan-Besson and Barlaud (Jehan-Besson et al., 2000) (see Section 2.1.1 for more details of their method), absolute frame difference as the measure of background intensity variation (equation (2.2)) and a fixed penalty within the object (equation (2.3)). In order to attain the global minimum in (3.7), the surface $\vec{\varsigma}$ must partition the domain so that points $(\boldsymbol{x}, t)$ with small frame difference (small $\xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$) are assigned to the outside ($\bar{\mathcal{V}}$), and those with large difference – to the inside ($\mathcal{V}$). The balance between such assignments is controlled by α. If an image sequence

has a stationary background, frame difference in the background region will correspond to the noise level in the sequence. Parameter α serves as a threshold – points $(\boldsymbol{x}, t)$ with frame differences smaller than α will be assigned to the background ($\bar{\mathcal{V}}$), and those with differences larger than α to the object region ($\mathcal{V}$). Therefore, parameter α should be chosen based on the noise level in the sequence. The smoothness term with parameter λ controls how compact should be the object regions, and how smooth should be their boundaries. Larger values of λ ensure more compact objects, with smoother boundaries.

By replacing (2.2) and (2.3) into (3.7), that minimization reduces to:

$$\min_{\vec{\varsigma}} \iiint_{\Omega\times\mathcal{T}} h(I_t) d\boldsymbol{x} dt + \lambda \iint_{\mathcal{S}} d\vec{\varsigma}, \tag{3.8}$$

$$h(I_t) = \begin{cases} \alpha & \text{if } (\boldsymbol{x}, t) \in \mathcal{V}, \\ |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)| & \text{if } (\boldsymbol{x}, t) \in \bar{\mathcal{V}}. \end{cases}$$

Note that this form is related to minimization in (2.17): the Euclidean area element weight is 1 ($g(\delta I) = 1$), whereas the weighted volume measure is a discontinuous function $h(\cdot)$ that quantifies the competition between volumes. As we shall see, this will result in an additional force helping avoid local minima and speeding up the convergence.

Although formulations (3.8) and (2.17) look very similar, they differ, in fact, significantly. While the formulation (2.17) is edge-based and requires strong intensity edges δI in the data, our new formulation is volume-based and works well even in the presence of diffuse edges due to the inherent competition between volumes $\mathcal{V}$ and $\bar{\mathcal{V}}$. This has been nicely illustrated on still-image (2-D) segmentation by Chan and Vese (Chan and Vese, 2001). As already mentioned, similar spatio-temporal motion detection formulation is proposed by Feghali *et al.* (Feghali et al., 2001; Feghali and Mitiche, 2004). However, in their formulation segmentation surface is attracted towards high gradients ("edges") in the motion field, while in our formulation two volumes with different motion character-

istics compete for points in the image sequence domain. Furthermore, it cannot handle volume segmentation, i.e., explicit division into sub-volumes with different motion. The volume-competition formulation (3.7) can handle two different types of motion, and can be extended to multiple motions as we will show in Chapter 6.

In order to solve for $\vec{\varsigma}$ we simply note that (3.8) is basically identical to (2.17) except for $g(\delta I) = 1$ and $h(I_t)$ replacing $f(\delta I)$, while $\Omega \times \mathcal{T} = \mathcal{V} \cup \bar{\mathcal{V}}$. With this observation and following the path taken by Jehan-Besson *et al.* (Jehan-Besson et al., 2001; Jehan-Besson et al., 2003), we can write the evolution equation for (3.8) as follows:

$$\frac{\partial \vec{\varsigma}}{\partial \tau} = [\alpha - |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)| + \lambda \kappa_m]\vec{n}. \tag{3.9}$$

Ignoring the curvature and remembering that $\vec{n}$ is the inward unit normal, $\alpha > |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)|$ will result in the surface shrinking and thus relinquishing the point $(\boldsymbol{x}, t)$, while $\alpha < |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)$ will cause the surface to expand thus enclosing this point. As for the curvature κ_m, it plays the role of a smoothing filter with respect to surface point coordinates. For sufficiently large λ, the curvature term will assure simultaneously smooth boundaries of individual-frame segmentations and temporal continuity of such boundaries (no large changes in segment shapes between consecutive frames). This can be viewed as a generalization of tracking of individual-frame segmentations; our approach is based on a sound mathematical model rather than a sequence of ad hoc steps. The inherent space-time continuity of the volume $\mathcal{V}$ is what distinguishes this approach from methods supported on two frames only.

We solve the active-surface evolution equation (3.9) using the level-set methodology. By embedding the active surface $\vec{\varsigma}$ as the zero-level set of the hyper-surface ϕ (in 4-D space):

$$\vec{\varsigma}(\tau) = \{(x, y, t) | \phi(x, y, t, \tau) = 0\},$$

where τ is the evolution time, we obtain the following level-set evolution equation:

$$\frac{\partial \phi}{\partial \tau} = [\alpha - |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)| + \lambda \kappa_m] \|\nabla \phi\|. \tag{3.10}$$

This equation is implemented using standard discretization as described by Sethian (Sethian, 1996b). We can rewrite equation (3.10) in terms of force (speed) function as follows:

$$\frac{\partial \phi}{\partial \tau} = [F_{prop} + F_{curv}] \|\nabla \phi\|,$$

where $F_{prop} = \alpha - |I(\boldsymbol{x}, t) - I(\boldsymbol{x}, t-1)|$ is the propagation expansion/contraction force (image-based force term) and $F_{curv} = \lambda \kappa_m$ is the smoothing force dependent on the curvature (curvature force term). Following (Sethian, 1996b), we propose a discretization scheme to approximate equation (3.10):

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} + \Delta t \left[\begin{array}{c} -[\max(F_{prop_{ijk}}, 0)]\nabla^{+} + \min(F_{prop_{ijk}}, 0)\nabla^{-} \\ +[\lambda K_{i,j,k}^{n}((D_{ijk}^{0x})^2 + (D_{ijk}^{0y})^2 + (D_{ijk}^{0t})^2)^{1/2}] \end{array} \right], \tag{3.11}$$

where

$$\begin{aligned} \nabla^{+} &= [\quad \max(D_{ijk}^{-x}, 0)^2 + \min(D_{ijk}^{+x}, 0)^2 + \\ &\qquad \max(D_{ijk}^{-y}, 0)^2 + \min(D_{ijk}^{+y}, 0)^2 + \\ &\qquad \max(D_{ijk}^{-t}, 0)^2 + \min(D_{ijk}^{+t}, 0)^2]^{1/2}, \\ \nabla^{-} &= [\quad \max(D_{ijk}^{-x}, 0)^2 + \min(D_{ijk}^{+x}, 0)^2 + \\ &\qquad \max(D_{ijk}^{-y}, 0)^2 + \min(D_{ijk}^{+y}, 0)^2 + \\ &\qquad \max(D_{ijk}^{-t}, 0)^2 + \min(D_{ijk}^{+t}, 0)^2]^{1/2}, \end{aligned}$$

and $K^n_{i,j,k}$ is the central difference approximation to the mean curvature, κ_m:

$$\kappa_m = \nabla \frac{\nabla\phi}{|\nabla\phi|} = \frac{\left\{ \begin{array}{c} (\phi_{yy}+\phi_{tt})\phi_x^2 + (\phi_{xx}+\phi_{tt})\phi_y^2 + (\phi_{xx}+\phi_{yy})\phi_t^2 \\ -2\phi_x\phi_y\phi_{xy} - 2\phi_x\phi_t\phi_{xt} - 2\phi_y\phi_t\phi_{yt} \end{array} \right\}}{(\phi_x^2+\phi_y^2+\phi_t^2)^{3/2}}. \tag{3.12}$$

The spatio-temporal derivatives are discretized using forward, backward, and centered difference operators:

$$\begin{aligned} D^{+x}u &\equiv \quad \tfrac{u(x+h,y,t)-u(x,y,t)}{h}, \\ D^{-x}u &\equiv \quad \tfrac{u(x,y,t)-u(x-h,y,t)}{h}, \\ D^{0x}u &\equiv \quad \tfrac{u(x+h,y,t)-u(x-h,y,t)}{2h}, \end{aligned}$$

which we used in (3.11) in a shorthand notation in which $D^{+x}\phi^n_{ijk}$ is written as D^{+x}_{ijk}, etc.

In our implementation, we first initialize the level-set surface to a signed distance surface using the fast marching algorithm described in Section 2.3.3 by solving $\|\nabla\phi\|$=1. Once the level-set surfaces have been initialized, in each iteration we calculate the force F at zero-level set points and extend this force using the algorithm for fast construction of extension velocities described in (Adalsteinsson and Sethian, 1999), by solving $\nabla\phi\cdot\nabla F = 0$ for F. We update the surface ϕ using equation (3.11), and find new location of zero-level set surface. Every $10-100$ iterations we re-initialize the surface to signed distance function using the fast marching algorithm. Fig. 3·2 shows the block diagram of the algorithm.

3.3 Experimental results

We will test our multi-frame motion detection (M-frame MD) method on two synthetic test sequences: natural-texture, synthetic-motion, sequence *Bean* (176×144 pixels) in

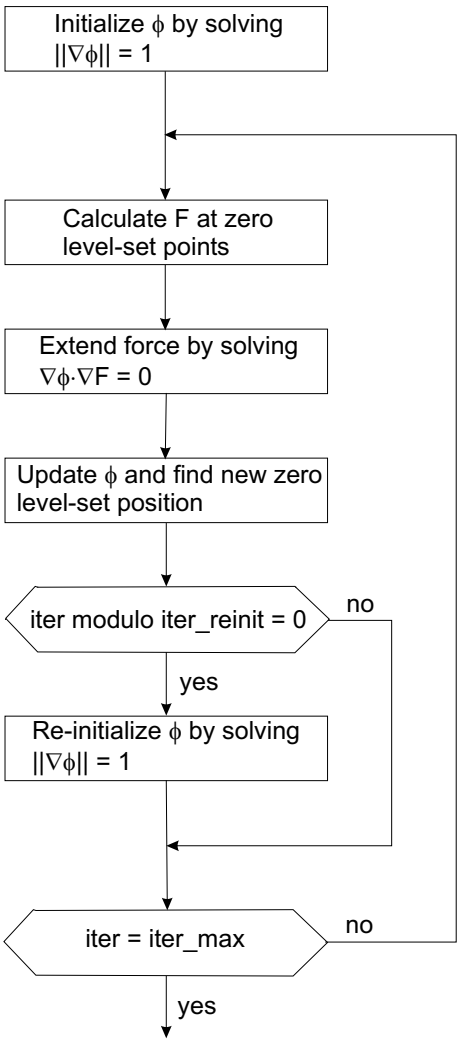


Figure 3·2: Block diagram of the level-set implementation of our motion detection algorithm. In the diagram, *iter_reinit* is the number of iterations after which the level-set surface is reinitialized and *max_iter* is the maximum number of iterations for which the algorithm is run.

which a bean-shaped object undergoes accelerated zoom and rotation, and the sequence *Bean_occl* where bean-shaped object undergoes the same motion as before, but is occluded by a static feature in the background in the second part of the sequence.

We first present results for the two-frame motion detection (2-frame MD) algorithm described in Section 2.1.1, i.e., our reference algorithm, for comparison. Results of the experiments on our two synthetic test sequences, *Bean* and *Bean_occl* are shown on Fig. 3·3. Table 3.1 presents a numerical comparison of this method with our algorithms.

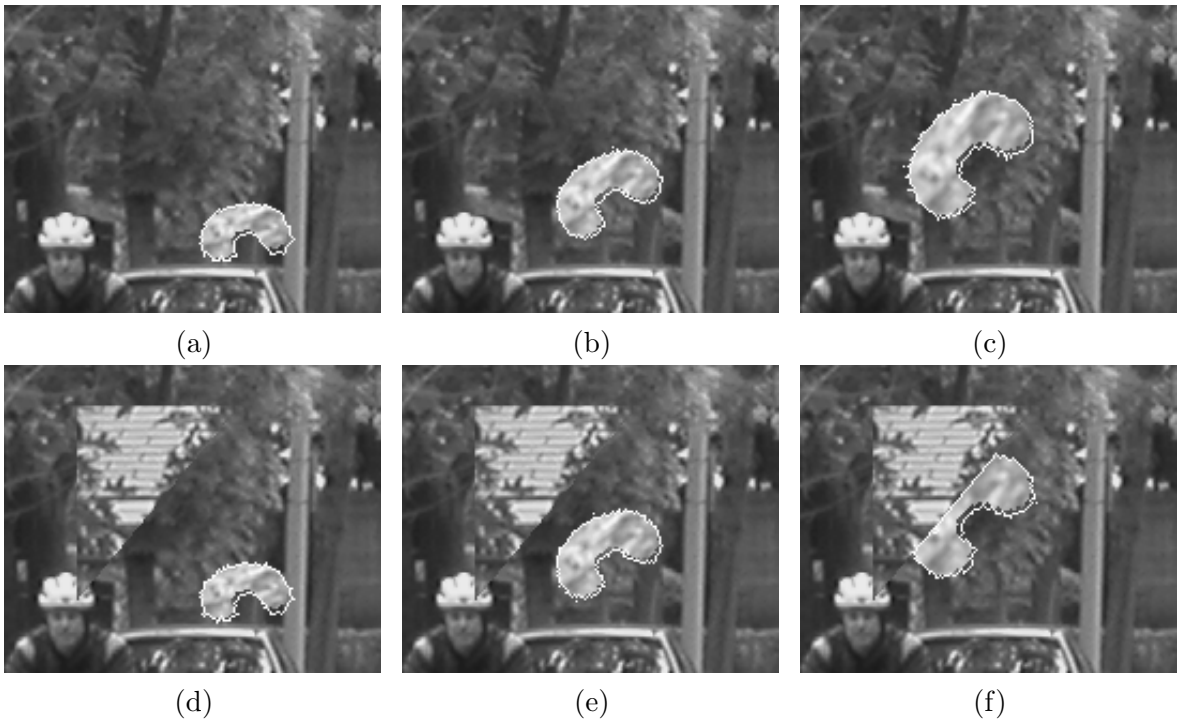


Figure 3·3: Results for the 2-frame MD algorithm: frames (a) #5, (b) #15, and (c) #25 from the synthetic image sequence *Bean*, and (d) #5, (e) #15 and (f) #25, from the synthetic image sequence *Bean_occl*, overlaid with final boundaries.

For both sequences we used the same set of parameters: $\alpha = 1$ and $\lambda = 0.1$. Given that both object trajectories and the object shape are smooth, we used small value for the curvature term parameter λ (i.e., there is no need to force smoothness with a large prior term). Also, given that the background is static and noise free, and we are trying to detect the moving areas, parameter α, which serves as a threshold, can be set to one. This 2-frame MD algorithm is semi-supervised: user is required to choose values for parameters α and λ. We run a batch of experiments with different values of parameters, and the results shown are the best we obtained (based on the segmentation error calculated using ground-truth data).

Fig. 3·4 shows results for our motion detection method, for the *Bean* image sequence. The segmentation was computed jointly over 30 frames, with α=1 and λ=0.1.

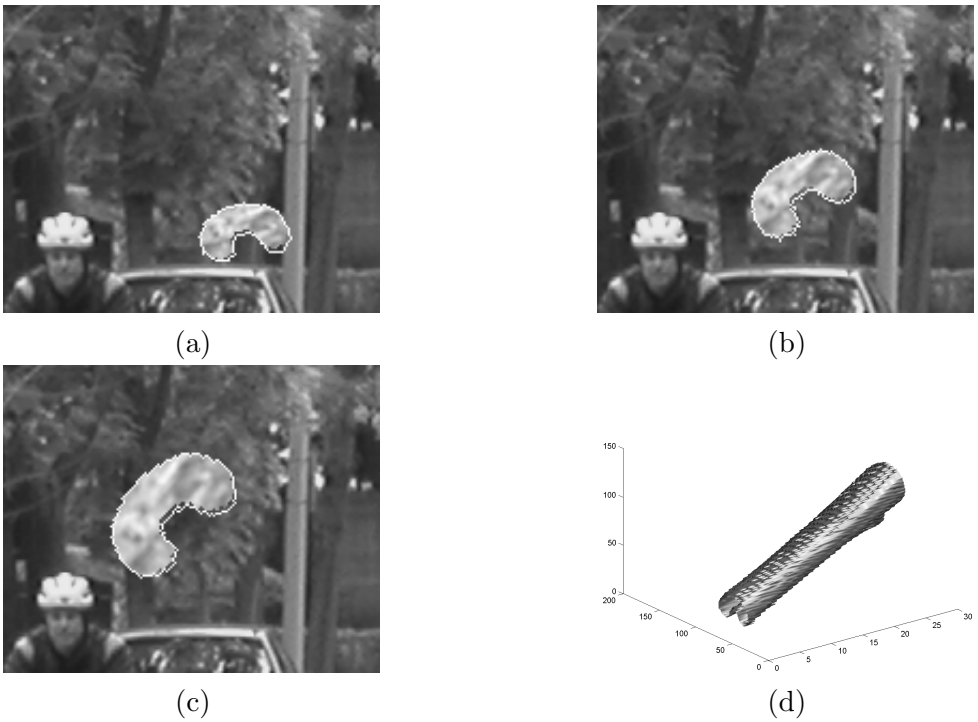


Figure 3·4: Results for M-frame MD algorithm, applied to synthetic image sequence *Bean*: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

Parameters are chosen for the same reasons as in two-frame motion detection case. Our multi-frame motion detection method is also semi-supervised, and the results we show are the best obtained from a batch of experiments with different parameter values. Fig. 3·4(a-c) shows three original frames with estimated boundaries superposed. These boundaries are cross-sections of the resulting object tunnel shown in Fig. 3·4(d) that partitions the sequence domain (30 frames) into voxels through which the object passes and those through which it does not. The results show very good recovery of the object shape as well as tracking between frames. The shape evolves consistently over time despite no explicit tracking, and in presence of significant motion (over 70 pixels, plus zoom-in). Note that this is achieved without explicit use of intensity edges. However, noticeable segmentation errors are present in areas lagging behind the moving object;

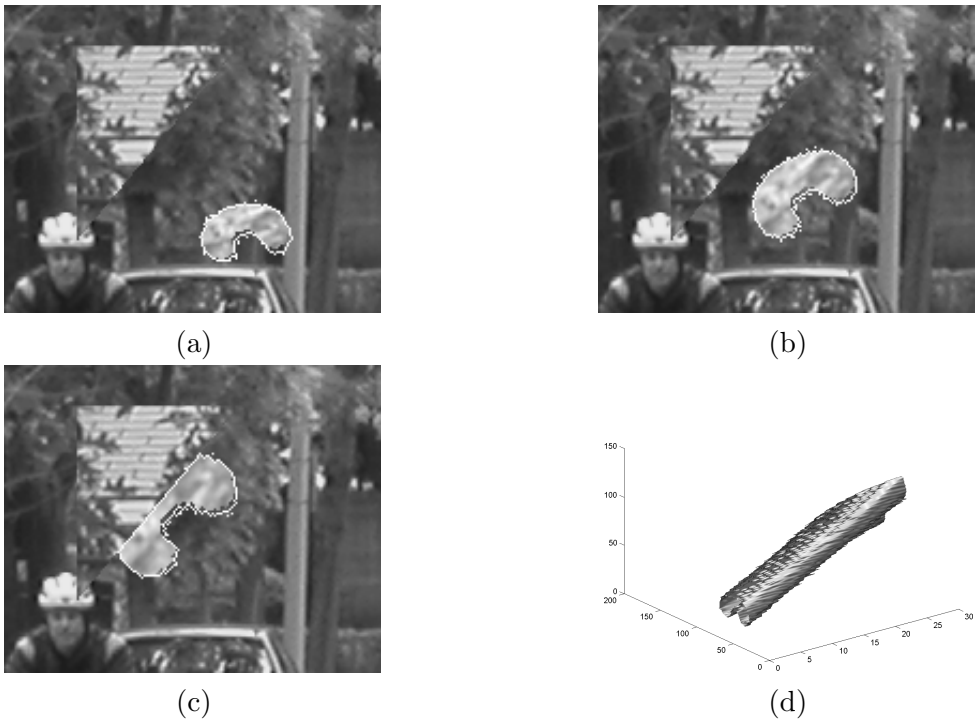


Figure 3·5: Results for M-frame MD algorithm, applied to synthetic image sequence *Bean_occl* with occluded object: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

since our method uses frame difference as the observation model, the resulting object shape is a union of object positions in consecutive frames. Similar results are obtained for the synthetic test sequence with occluded object, *Bean_occl* (Fig. 3·5). We used the same values for parameters α and λ as in the previous case. This example shows that our method handles object occlusions very well.

Fig. 3·6 presents results for a natural video sequence *Car* acquired with a static video camera (progressive, 180×120 pixels, 30 frames/s). Fig. 3·6(a–c) shows three frames of the 76-frame sequence used (frames #11 through #86 of the original sequence) overlaid with the estimated boundaries: a car enters the scene from the right, moves to the left with almost constant velocity and disappears behind a wall on the left. The segmentation was computed jointly over 76 frames, with α=2.5 and λ=1. In this

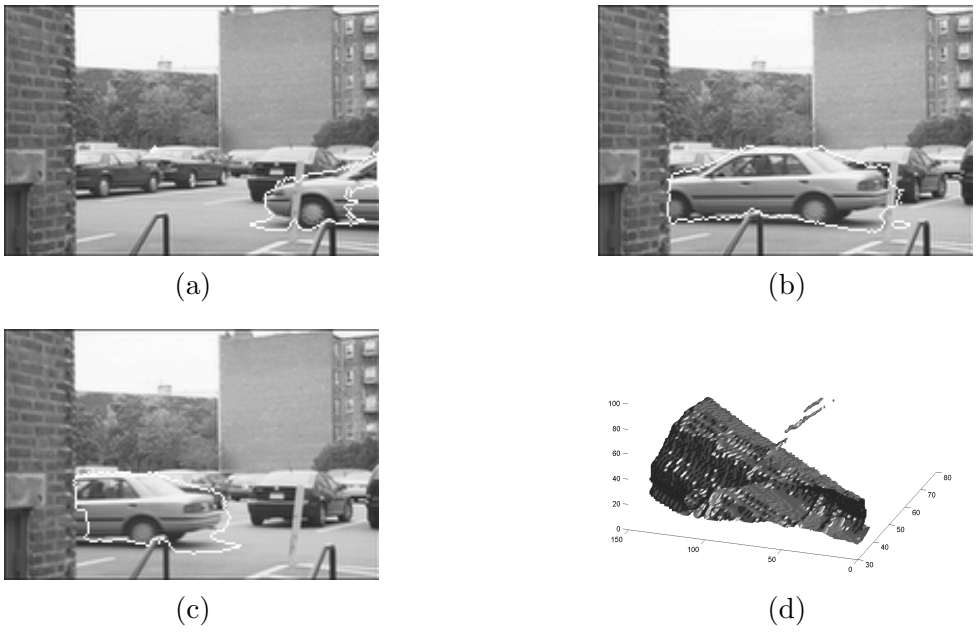
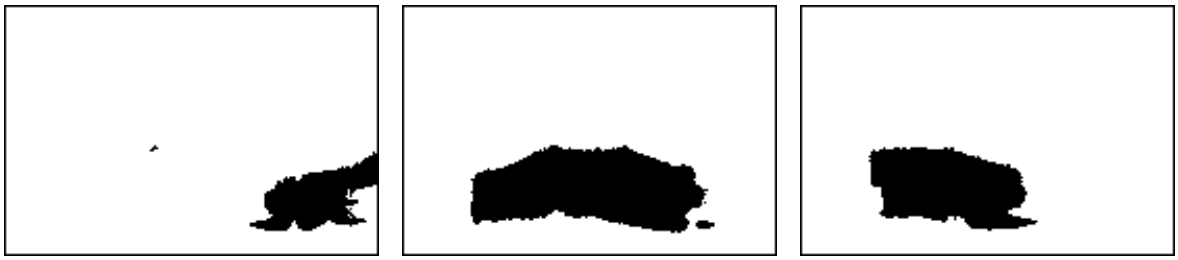


Figure 3·6: Results for M-frame MD algorithm, applied to natural image sequence *Car*: frames (a) #24, (b) #54, and (c) #64 from the sequence overlaid with final boundaries, and (d) corresponding object tunnel.

sequence, background is not perfectly still (there is a change in intensity and some small object motion in the background), so we need to use higher value for threshold α. Basically, the threshold is chosen so that the absolute frame difference is higher than the threshold only for pixels belonging to the object. However, a higher threshold leads to objects with some parts missing, so we use higher prior term to force smooth boundaries of the object. Fig. 3·6(d) shows the computed object tunnel; for visualization reasons it is shown from frame #30 to frame #76. Fig. 3·7 shows three frames from the sequence of estimated segmentation labels. The results demonstrate good object-shape recovery and tracking between frames, however in addition to segmentation errors noted in the synthetic sequence, here we have errors due to change of intensity throughout the sequence (real-life shooting), occlusion of the car by the wall, and moving shadows of the car. All this contributes to a less precise segmentation compared to the synthetic sequence. Still, the result is quite good for such simple models.



(a) (b) (c)

Figure 3·7: Estimated segmentation label fields corresponding to frames (a) #24, (b) #54, and (c) #64 from the *Car* image sequence.

Table 3.1: Object segmentation error ϵ_s (pixels per frame) for synthetic test sequences, for motion detection algorithms.

Sequence	Two-frame method 2-frame MD	Multi-frame method M-frame MD
Bean	166.0 (14%)	128.1 (10.8%)
Bean_occl	159.5 (17.1%)	133.9 (14.3%)

Using the ground-truth segmentation which is available to us for the two synthetic sequences, we compare numerically performance of our motion detection method to the two-frame motion detection method. For the segmentation error metric, we use symmetric difference between the ground-truth and computed segmentation:

$$\epsilon_s = \frac{1}{K} \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in \Omega} |O_g(\boldsymbol{x}, t_i) - O_e(\boldsymbol{x}, t_i)|, \tag{3.13}$$

where $K = card(\mathcal{T})$ is the number of frames under segmentation, and $O_g(\boldsymbol{x}, t)$ and $O_e(\boldsymbol{x}, t)$ are ground-truth and estimated binary segmentation maps at time t, respectively, defined as follows:

$$O(\boldsymbol{x}, t) = \begin{cases} 1 \ , \ \boldsymbol{x} \in \mathcal{R}_t, \\ 0 \ , \ \boldsymbol{x} \notin \mathcal{R}_t. \end{cases} \tag{3.14}$$

$\mathcal{R}_t$ is the object region in frame at time t, easily extracted from a cross-section of the object tunnel $\mathcal{V}$ at time t for the M-frame MD method, or directly obtained in the case of 2-frame MD method.

As Table 3.1 shows, our method clearly outperforms the two-frame method on both test sequences, with average improvement of the segmentation precision by 20%. Although it is hard to visually confirm these results, by comparing the results in Fig. 3·3 with the results in Fig. 3·4 and Fig. 3·5, we can see that for the same value of parameter λ, which controls smoothness of the solution, our method yields smoother object boundary, which is also closer to the true object boundary. However, due to limitations of our simple motion detection model (lack of motion modeling), each estimated object tunnel is a union of object positions in consecutive frames. This problem is inherent to the method, and will result in the segmentation error even for perfect, noiseless sequences. To alleviate this problem, we propose more advanced, motion segmentation methods in the following chapters.

Chapter 4

Multiframe motion-based video segmentation with background occlusion detection

In the previous chapter, we restricted video sequence analysis to the detection and tracking of moving objects against a stationary background. Although the method worked quite well, since motion was not accounted for significant segmentation errors occurred. By incorporating motion into the formulation, we can lift the stationary background restriction and also distinguish individual objects based on their dynamics. To this end, we propose to explicitly model the dynamics of objects and background using motion trajectories. Additionally, we explicitly model occluded and newly-exposed areas in the background (Ristivojević and Konrad, 2004b).

In particular, our goal is to partition the domain of an image sequence into four regions (volumes) as follows: moving object, moving or static background, background areas that were exposed in the past (background exposed volume) or that will be occluded in the future (background occlusion volume). For the first two volumes, we measure object and background intensity variations, respectively, along motion trajectories spanning the whole temporal support of the image sequence. Parts of the background that will be occluded or have been exposed within this support cannot be accurately modeled either by object or background motion trajectories. If not explicitly modeled, they will be randomly included in either object or background segmentation volumes,

thus creating errors. The more frames of the sequence are processed, the more of these errors are present. To address this, here we explicitly model these regions by a new space-time concepts of *occlusion volume* and *exposed volume*. Furthermore, we want, jointly with the segmentation, to estimate motion parameters of the objects and background. Since we need to partition the image sequence domain into four volumes, we use variational framework for the formulation and multiphase level-set method (Vese and Chan, 2002) for the solution.

4.1 Energy formulation

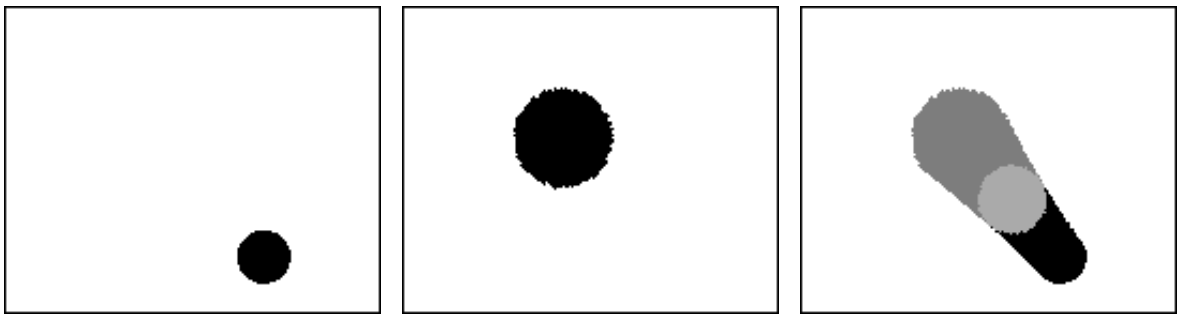
We would like to partition the domain of an image sequence into four volumes: moving *object tunnel* $\mathcal{V}_4$, static or moving *background tunnel* $\mathcal{V}_1$, *occlusion volume* $\mathcal{V}_2$, i.e., background pixels that are occluded by the object at any time within the sequence, and *exposed volume* $\mathcal{V}_3$, i.e., background pixels exposed by the object at some time in the sequence. The spatio-temporal domain of the image sequence can be divided into four volumes using two parameterized surfaces, $\vec{\varsigma}_1$ and $\vec{\varsigma}_2$, as described in Table 4.1.

Table 4.1: Segmentation volumes and associated energy terms for 2-surface example

Volume $\mathcal{V}_n$	j_n	$(\boldsymbol{x}, t)$	Level-set func. at $(\boldsymbol{x}, t)$	Energy terms
$\mathcal{V}_1$ = background	(00)	outside $\vec{\varsigma}_1, \vec{\varsigma}_2$	$\phi_1 < 0,\ \phi_2 < 0$	$\omega_1 \xi_{obj}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_2$ = occluded backgr.	(01)	inside $\vec{\varsigma}_1$, outside $\vec{\varsigma}_2$	$\phi_1 > 0,\ \phi_2 < 0$	$\omega_2 \xi_{occ}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_3$ = exposed backgr.	(10)	outside $\vec{\varsigma}_1$, inside $\vec{\varsigma}_2$	$\phi_1 < 0,\ \phi_2 > 0$	$\omega_3 \xi_{exp}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_4$ = object	(11)	inside $\vec{\varsigma}_1, \vec{\varsigma}_2$	$\phi_1 > 0,\ \phi_2 > 0$	$\omega_4 \xi_o bj(\boldsymbol{x}, t; \boldsymbol{p})$

An example of cross-section through such volumes for a simple binary image sequence is shown in Fig. 4·1(c). The portion of the background visible throughout the

sequence is represented in white, the moving object is in light gray, the portion of the background which is going to be occluded in the following frames is in dark gray, while the background region exposed in preceding frames is in black.



(a) (b) (c)

Figure 4·1: Frames (a) #1; and (b) #30 from a simple binary image sequence, and (c) segmentation regions in frame #15.

As described in Section 3.1 we decompose the problem of joint segmentation and estimation of motion parameters into two interleaved minimizations: estimation of motion parameters given segmentation surfaces, and estimation of segmentation surfaces with fixed motion parameters ($\boldsymbol{p} = \widehat{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}} = \widehat{\bar{\boldsymbol{p}}}$). For the estimation of segmentation surfaces, we extend the volume competition formulation (3.7) to take into account occluded and exposed background volumes, which leads to the following energy minimization:

$$\begin{aligned}\min_{\vec{\varsigma}_1,\vec{\varsigma}_2} \quad & \omega_4 \iiint_{\mathcal{V}_4} \xi_{obj}(\boldsymbol{x},t;\widehat{\boldsymbol{p}})d\boldsymbol{x}dt + \omega_1 \iiint_{\mathcal{V}_1} \xi_{obj}(\boldsymbol{x},t;\widehat{\bar{\boldsymbol{p}}})d\boldsymbol{x}dt + \\ & \omega_2 \iiint_{\mathcal{V}_2} \xi_{occ}(\boldsymbol{x},t;\widehat{\bar{\boldsymbol{p}}})d\boldsymbol{x}dt + \omega_3 \iiint_{\mathcal{V}_3} \xi_{exp}(\boldsymbol{x},t;\widehat{\bar{\boldsymbol{p}}})d\boldsymbol{x}dt + \\ & \lambda_1 \iint_{\mathcal{S}_1} d\vec{\varsigma}_1, + \lambda_2 \iint_{\mathcal{S}_2} d\vec{\varsigma}_2,\end{aligned} \tag{4.1}$$

where $\vec{\varsigma}_1 = \partial(\mathcal{V}_2 \cup \mathcal{V}_4)$, $\vec{\varsigma}_2 = \partial(\mathcal{V}_3 \cup \mathcal{V}_4)$, $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3 \cup \mathcal{V}_4 = \Omega \times \mathcal{T}$. The constants ω_1, ω_2, ω_3, and ω_4 are weights assigned to the volume energies and reflect the degree of uncertainty in measurements ξ_{obj}, ξ_{occ}, and ξ_{occ}. λ_1 and λ_2 associate a cost with the

Euclidean area elements $d\vec{\varsigma}_1$ and $d\vec{\varsigma}_2$, respectively.

Central to the segmentation performance are terms ξ_{obj}, ξ_{occ}, and ξ_{exp} in (4.1). Each of these terms must be designed to measure the consistency of image sequence voxels with one of the scenarios: visible, to-be-occluded or newly-exposed. Let $\boldsymbol{c_p}(t_i; \boldsymbol{x}, t)$ be a motion trajectory described by motion parameters $\boldsymbol{p}$, i.e., let $\boldsymbol{c_p}$ be a spatial position at time t_i of a feature that moved from position $\boldsymbol{x}$ at time t due to motion parameterized by $\boldsymbol{p}$ (Dubois, 1992). We define two measures of intensity along a motion trajectory between frames number k and l:

$$\begin{aligned} \mu_{k,l}(\boldsymbol{x}, t; \boldsymbol{p}) &= \frac{1}{\Delta} \sum_{i=k}^{l} \widetilde{I}(\boldsymbol{c_p}(t_i; \boldsymbol{x}, t), t_i), \\ \sigma^2_{k,l}(\boldsymbol{x}, t; \boldsymbol{p}) &= \frac{1}{\Delta} \sum_{i=k}^{l} (\widetilde{I}(\boldsymbol{c_p}(t_i; \boldsymbol{x}, t), t_i) - \mu_{k,l}(\boldsymbol{x}, t; \boldsymbol{p}))^2, \end{aligned}$$

where $\Delta = k - l + 1$ and $\widetilde{I}$ denotes interpolated intensity (e.g., bi-cubic interpolator (Keys, 1981)) because $(\boldsymbol{c_p}(t_i; \boldsymbol{x}, t), t_i)$ need not be on the sampling grid of the sequence. Clearly, $\mu_{k,l}(\boldsymbol{x}, t; \boldsymbol{p})$ is the sample mean and $\sigma^2_{k,l}(\boldsymbol{x}, t; \boldsymbol{p})$ is the sample variance of intensity along trajectory $\boldsymbol{c_p}$ between frames number k and l.

In order that a point be declared as visible in object or background throughout the sequence, intensity variation along motion trajectory passing through this point must be small (motion accurately explains the data). Thus, we define the *object/background volume* term as a sample variance along the whole trajectory:

$$\xi_{obj}(\boldsymbol{x}, t; \boldsymbol{p}) = \sigma^2_{1,K}(\boldsymbol{x}, t; \boldsymbol{p}), \tag{4.2}$$

where $K = card(\mathcal{T})$ is the number of frames under segmentation. This term will be small only for voxels for which a set of motion parameters exists that induces small intensity variations along the whole motion trajectory.

In order that a point be declared as background point to-be-occluded at a later time,

intensity variation along background motion trajectory passing through this point must be small up to this point and is expected to become large at some time in the future (as illustrated in Fig. 4·2 (a)). We propose the following *occluded background volume* term:

$$\xi_{occ}(\boldsymbol{x}, t = t_j; \bar{\boldsymbol{p}}) = \sigma^2_{1,j}(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) + \frac{\alpha_1}{\sigma^2_{j,K}(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) + 1}, \tag{4.3}$$

where j is such that $|t - t_j| \leq |t - t_l|, 1 \leq l \leq K$, i.e., t_j is the time instant of the nearest frame from the continuous time[1] t. In order that ξ_{occ} be small, motion $\bar{\boldsymbol{p}}$ must exist that induces small intensity variations up to t_j, while inducing large variations after t_j; the compromise between the two terms is adjusted by weight α_1.

Finally, for a point to be declared as background point exposed in the past, intensity variations along background motion trajectory are expected to be large prior to this point and small afterwards (example in Fig. 4·2 (b)). We propose the following *exposed background volume* term:

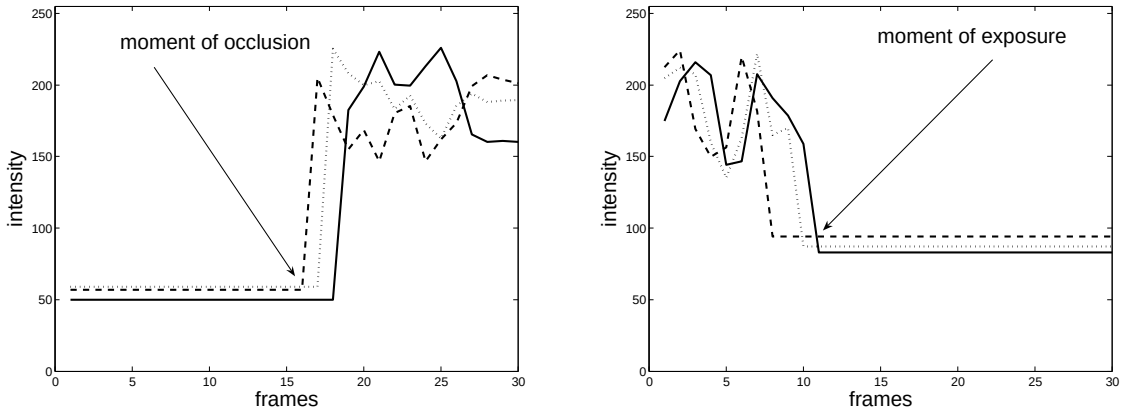
$$\xi_{exp}(\boldsymbol{x}, t = t_j; \bar{\boldsymbol{p}}) = \frac{\alpha_2}{\sigma^2_{1,j}(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) + 1} + \sigma^2_{j,K}(\boldsymbol{x}, t; \bar{\boldsymbol{p}}), \tag{4.4}$$

with j defined as above. Again, ξ_{exp} will be small if motion $\bar{\boldsymbol{p}}$ induces large intensity variations up to t_j and small variations afterwards; α_2 balances the impact of the two terms.

4.2 Solution method

We will first present implementation of the surface estimation formulation (4.1) using the multiphase level-set method, and then proceed to formulate and solve the motion parameters estimation problem.

[1] When the evolution equations are discretized, a discretization of time t is possible such that $t_j = t$.



(a) (b)

Figure 4·2: Example of intensity variation along motion trajectories in (a) occluded and (b) newly-exposed background areas taken from the *Bean_occl* sequence.

4.2.1 Estimation of segmentation surfaces

Following the multiphase level-set formalism described in Section 2.3.4, we represent the energy minimization (4.1) using two level-set functions, $\phi_1(\boldsymbol{x},t)$ and $\phi_2(\boldsymbol{x},t)$:

$$\begin{aligned}\min_{\phi_1,\phi_2}\mathcal{E}, \qquad \mathcal{E} = \;& \omega_4 \iiint_{\Omega\times\mathcal{T}} \xi_{obj}(\boldsymbol{x},t;\widehat{\boldsymbol{p}})H(\phi_1(\boldsymbol{x},t))H(\phi_2(\boldsymbol{x},t))d\boldsymbol{x}dt \\ &+ \omega_2 \iiint_{\Omega\times\mathcal{T}} \xi_{occ}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})H(\phi_1(\boldsymbol{x},t))(1-H(\phi_2(\boldsymbol{x},t)))d\boldsymbol{x}dt \\ &+ \omega_3 \iiint_{\Omega\times\mathcal{T}} \xi_{exp}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})(1-H(\phi_1(\boldsymbol{x},t)))H(\phi_2(\boldsymbol{x},t))d\boldsymbol{x}dt \\ &+ \omega_1 \iiint_{\Omega\times\mathcal{T}} \xi_{obj}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})(1-H(\phi_1(\boldsymbol{x},t)))(1-H(\phi_2(\boldsymbol{x},t)))d\boldsymbol{x}dt \\ &+ \lambda_1 \iiint_{\Omega\times\mathcal{T}} |\nabla H(\phi_1(\boldsymbol{x},t))|d\boldsymbol{x}dt + \lambda_2 \iiint_{\Omega\times\mathcal{T}} |\nabla H(\phi_2(\boldsymbol{x},t))|d\boldsymbol{x}dt,\end{aligned} \tag{4.5}$$

where $H(\phi)$ is the Heaviside function. The volumes that we seek are now defined through intersections of zero-level sets of surfaces ϕ_1 and ϕ_2, as described in Table 4.1, fourth column.

In order to carry out minimization (4.5), the level-set surfaces should evolve according to the direction of steepest descent, which is the direction of the negative gradient of energy $\mathcal{E}$ with respect to ϕ_1 and ϕ_2. As the result, we obtain the following level-set evolution equations (valid for all points $(\boldsymbol{x}, t)$, omitted for brevity of notation):

$$\begin{aligned}
\frac{\partial \phi_1(\tau)}{\partial \tau} &= \big\{\lambda_1 \kappa_{m_1} - [(\omega_4 \xi_{obj}(\widehat{\boldsymbol{p}}) - \omega_3 \xi_{exp}(\widehat{\boldsymbol{p}}))H(\phi_2(\tau)) + \\
&\quad (\omega_2 \xi_{occ}(\widehat{\boldsymbol{p}}) - \omega_1 \xi_{obj}(\widehat{\boldsymbol{p}}))(1 - H(\phi_2(\tau)))]\big\} \, \|\nabla \phi_1(\tau)\|, \\
\frac{\partial \phi_2(\tau)}{\partial \tau} &= \big\{\lambda_2 \kappa_{m_2} - [(\omega_4 \xi_{obj}(\widehat{\boldsymbol{p}}) - \omega_2 \xi_{occ}(\widehat{\boldsymbol{p}}))H(\phi_1(\tau)) + \\
&\quad (\omega_3 \xi_{exp}(\widehat{\boldsymbol{p}}) - \omega_1 \xi_{obj}(\widehat{\boldsymbol{p}}))(1 - H(\phi_1(\tau)))]\big\} \, \|\nabla \phi_2(\tau)\|,
\end{aligned} \tag{4.6}$$

where τ is the algorithmic evolution time, and κ_{m_1} and κ_{m_2} are mean curvatures (defined in (3.12)) of the level-set surfaces ϕ_1 and ϕ_2, respectively. We implement these equations iteratively using standard discretization as described in Section 3.2. In each iteration we calculate the forces F_1 and F_2 at zero level-set points of both surfaces, extend these forces using the fast marching algorithm by solving $\phi_i \cdot \nabla F_i = 0$ for F_i, $i = 1, 2$, and update the surfaces ϕ_1 and ϕ_2. Re-initialization of the surfaces using the fast marching algorithm by solving $\|\nabla \phi_i\|$=1 (signed distance) is performed every 100 iterations to keep surfaces as close as possible to signed distance functions. The algorithm is identical to the one shown in Fig. 3·2, except that two level-set surfaces are evolved simultaneously.

4.2.2 Estimation of motion parameters

In order to estimate motion given segmentation surfaces, first we have to define motion model we are going to use. Although ideally we would like to model motion trajectories over multiple frames with one set of parameters, we use here a simpler two-frame motion model, that describes displacement of each point in the image between two consecutive frames. In this case motion trajectory $\boldsymbol{c_p}(t_i; \boldsymbol{x}, t)$ simplifies to displacement vectors between each two consecutive frames of the sequence, i.e., it is piecewise-linear. To

model spatial variation of motion vectors inside each object (or background), we choose the affine spatial model, with six parameters $\boldsymbol{p} = [p_1, \ldots, p_6]$:

$$\boldsymbol{x}' = \boldsymbol{x} + \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} p_3 & p_4 \\ p_5 & p_6 \end{bmatrix} (\boldsymbol{x} - \boldsymbol{x}_0). \tag{4.7}$$

In Section 3.1 we formulated the estimation of motion parameters for a fixed segmentation surface through minimization (3.4). Extending that formulation to the case of two (fixed) segmentation surfaces ($\vec{\varsigma}_1 = \hat{\vec{\varsigma}}_1$ and $\vec{\varsigma}_2 = \hat{\vec{\varsigma}}_2$) leads to the following minimization:

$$\begin{aligned} \min_{\boldsymbol{p},\bar{\boldsymbol{p}}} \mathcal{E}_1(I_t, \hat{\vec{\varsigma}}_1, \hat{\vec{\varsigma}}_2, \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}) + \\ \mathcal{E}_2(\hat{\vec{\varsigma}}_1, \hat{\vec{\varsigma}}_2, \boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}) + \mathcal{E}_3(\boldsymbol{p}, \bar{\boldsymbol{p}}, \mathcal{I}^t \backslash \{I_t\}). \end{aligned} \tag{4.8}$$

We will assume that motion parameters of the object ($\boldsymbol{p}$) and background ($\bar{\boldsymbol{p}}$) are independent, and can be estimated through separate minimization processes. We proceed to describe the estimation procedure for the object motion parameters; background parameters are estimated exactly the same way (with object region/volume being replaced by background region/volume). Since we assumed a two-frame motion model, the first term of our energy functional, $\mathcal{E}_1$, simplifies to intensity mismatch between frame pairs but only within regions that belong to the moving object. Moreover, since the motion parameters are independent between different frame pairs, there is little that constrains motion trajectory to stay within object volume. In order to counteract this, we need to penalize shape mismatch between regions in the two frames; for the second energy term, $\mathcal{E}_2$, we use symmetric difference between projected object region from one frame and actual object region in the other frame. Energy term $\mathcal{E}_3$ is a prior term on motion trajectories and could measure their spatio-temporal smoothness. Given that we use affine motion model to represent trajectories, spatial smoothness is guaranteed, and there is no reason to enforce it with an additional energy term. For now, we will not impose any

temporal smoothness constraint on the motion trajectories – we will investigate that in the following section. Given all these assumptions and considering the discrete nature of images and computed volumes, in order to estimate motion parameters between each two consecutive frames of the image sequence we effectively minimize the following energy functional:

$$E = \sum_{\boldsymbol{x}_i \in \mathcal{R}_{t_1}} \rho\left[\widetilde{I}(\boldsymbol{x}'_i, t_2) - I(\boldsymbol{x}_i, t_1)\right] + \lambda \sum_{\boldsymbol{x}_i \in \Omega} \left[\widetilde{O}(\boldsymbol{x}'_i, t_2) - O(\boldsymbol{x}_i, t_1)\right]^2, \tag{4.9}$$

where $\mathcal{R}_t$ is the object region in frame at time t, $\rho(\cdot)$ is a robust estimator function, and $O(\boldsymbol{x}, t)$ is a binary segmentation map at time t defined as follows:

$$O(\boldsymbol{x}, t) = \begin{cases} 1\ ,\ \boldsymbol{x} \in \mathcal{R}_t, \\ 0\ ,\ \boldsymbol{x} \notin \mathcal{R}_t. \end{cases} \tag{4.10}$$

Since $(\boldsymbol{x}, t)$ does not have to belong to the sampling grid of the sequence, $\widetilde{I}$ and $\widetilde{O}$ denote interpolated values of intensity and segmentation map, respectively. Clearly, $\mathcal{R}_t$, and therefore $O(\cdot, t)$, is easily extracted from a cross-section of the object tunnel $\mathcal{V}$ at time t.

Since the segmentation surfaces $\vec{\varsigma_i}$ may be far from the underlying, true object boundaries, some points inside $\mathcal{R}_t$ may not belong to the object, and, therefore, their motion cannot be described by the object motion parameters. To deal with these outliers, we use a robust M-estimator (Black, 1992) ρ (e.g., Lorentzian, Geman-McClure, etc.) in the first term. The second term is a shape-matching penalty whose influence is controlled by parameter λ. In order to find solution to (4.9), we use MATLAB Optimization Toolbox's function `fminunc`, which uses BFGS quasi-Newton method with a mixed quadratic and cubic line search procedure.

4.2.3 Long-term temporal modeling of motion trajectories

As we described in previous section, we perform estimation of motion parameters separately for each frame pair in the sequence. Although each motion field is spatially constrained through an affine motion model, there is no explicit relationship between motion vectors at different time instants. We model motion trajectories $\boldsymbol{c}(t_i; \boldsymbol{x}, t)$ (defined in Section 4.1) as piecewise linear between images in each frame pair, but we do not constrain these linear segments temporally in any way; neighboring linear segments are computed independently of one another. However, the underlying motion of moving objects is smooth in the temporal direction, and a consistent spatio-temporal motion model is needed to match the spatio-temporal tunnel model we proposed.

The simplest way to ensure temporal smoothness in the estimated motion trajectories is to introduce an additional term into the motion estimation cost function, which will penalize the difference between motion field vectors at time t and corresponding motion vectors at time $t-1$. Starting with the energy functional (4.9), we add an additional term measuring the temporal smoothness of motion trajectories and corresponding to energy term $\mathcal{E}_3$ in the general motion estimation energy minimization (4.8). As the result, we obtain the following energy formulation:

$$\begin{aligned} E = \quad & \textstyle\sum_{\boldsymbol{x}_i \in \mathcal{R}_{t_1}} \rho \left[\widetilde{I}(\boldsymbol{x}'_i, t_2) - I(\boldsymbol{x}_i, t_1) \right] + \\ & \lambda_1 \textstyle\sum_{\boldsymbol{x}_i \in \Omega} \left[\widetilde{O}(\boldsymbol{x}'_i, t_2) - O(\boldsymbol{x}_i, t_1) \right]^2 + \\ & \lambda_2 \textstyle\sum_{\boldsymbol{x}_i \in \mathcal{R}_{t_1}} \left\| \boldsymbol{d}(\boldsymbol{x}_i, t_1) - \boldsymbol{d}(\boldsymbol{x}'_i, t_0) \right\|^2, \end{aligned} \tag{4.11}$$

where $\boldsymbol{x}'_i$ s related to $\boldsymbol{x}_i$ through affine projection (as defined in (4.7)) and points $\boldsymbol{x}_i$ are on a lattice at time t_1, while points $\boldsymbol{x}'_i$ may be off lattice at t_2 or t_0. We are estimating parameters of the motion field $\boldsymbol{d}(\boldsymbol{x}, t_1)$, between frames at time t_1, and t_2, while forcing it to be similar to the motion field $\boldsymbol{d}(\boldsymbol{x}, t_0)$, which was previously estimated between

frames t_0 and t_1. The similarity measure is applied only to points $\boldsymbol{x}_i \in \mathcal{R}_{t_1}$ for which the corresponding points $\boldsymbol{x}'_i$ belong to the moving object region at time t_0 ($\mathcal{R}_{t_0}$). For the first frame pair in the sequence, we should use the energy formulation (4.9), since there is no previous-frame motion field.

In order to test whether an introduction of additional smoothness constraint improves motion estimation results, we performed experiments on synthetic sequences *Bean* and *Bean_occl*, as well as on *Bean* sequence with added Gaussian white noise ($PSNR = 26dB$). We used the M-frame MD results from Chapter 3 (Figs. 3·4 and 3·5) as initial object segmentations. Based on that segmentation, we calculated motion parameters using the energy formulation (4.9) (without smoothness constraint) and energy formulation (4.11) (with smoothness constraint). Since we know the true object motion parameters used to generate the sequences, we were able to compare results numerically. As a motion error measure, we use the difference between estimated and true motion fields, averaged over all pixels in object regions in all frames of the sequence:

$$\epsilon_m = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{N_i}\sum_{\boldsymbol{x}_i \in \mathcal{R}_{t_i}} \left\| \boldsymbol{d}(\boldsymbol{x}_i, t_i) - \boldsymbol{d}_t(\boldsymbol{x}_i, t_i) \right\|, \tag{4.12}$$

where $K = card(\mathcal{T})$ is the number of frames under segmentation, N_i is the number of pixels inside the moving object region $\mathcal{R}_{t_i}$ at time t_i, $\boldsymbol{d}(\boldsymbol{x}, t_i)$ is the estimated motion field for the frame at time t_i, and $\boldsymbol{d}_t(\boldsymbol{x}, t_i)$ is the true motion field for the same frame.

Table 4.2 compares motion estimation errors obtained when estimation is performed with and without the temporal smoothness constraint on three synthetic sequences. It is obvious that the observed improvement in motion estimates, although exists, is not substantial. The reason for this lays in the second term of energy formulation (4.9). This shape mismatch penalty encourages motion trajectories to stay within the object tunnel, thus implicitly enforcing a smoothness constraint on the trajectories, given that the object tunnel is forced to be smooth in the segmentation process. This is especially

Table 4.2: Motion error, ϵ_m, for different synthetic sequences, for motion estimates calculated with and without smoothness constraint.

	Method		
Sequence	without smooth. prior	with smooth. prior	improvement
Bean	0.0129	0.0124	3.9%
Bean_occl	0.0308	0.0283	8.1 %
Bean with noise	0.3729	0.3396	8.9 %

true in the case of *Bean* sequence, where there is no object occlusion or noise, and the object tunnel is very smooth. Occlusion events and noise in the sequence make object tunnels less smooth, which explains a larger improvement in motion estimates due to temporal smoothness constraint in those cases. The computational complexity of the motion estimation algorithm does not increase much with the addition of the temporal smoothness constraint. Furthermore, even a small improvement in the accuracy of motion estimates can lead to substantially better segmentation results, which makes this method preferable to the simpler one introduced in Section 4.2.2.

4.2.4 Steps of the overall segmentation algorithm

The first step of the overall segmentation algorithm is to find motion parameters based on an initial segmentation of the moving object. In order to compute the initial segmentation, we use the motion detection algorithm proposed in Chapter 3. Using the resulting segmentation (single surface), we calculate motion parameters and initialize 2 surfaces needed by the segmentation step. Every voxel inside the object tunnel, estimated using the motion detection algorithm, is labeled as an object voxel for the motion-based segmentation algorithm. However, for every voxel outside the object tunnel (background), we verify whether its motion trajectory continues to stay outside (in which case it is

labeled as a background voxel) or is partially within the volume and partially outside (labeled as an occluded or exposed background voxel). Based on this sequence of labels, we initialize level-set functions ϕ_1 and ϕ_2, using rules defined in Table 4.1. After the segmentation step reaches convergence, we perform motion parameter estimation again, and start new segmentation step with the new motion parameters. This procedure is repeated until surfaces and motion parameters converge to a stable solution.

4.3 Experimental results

Results for the two-frame motion-compensated (MC) segmentation method (2-frame MCS method), serving as a reference algorithm and described in Section 2.1.2, for the two synthetic test sequences, are shown in Fig. 4·3. Numerical values of the segmentation errors are presented in Table 4.3. We will discuss these results further when we compare them to our multi-frame motion segmentation method. For the *Bean* sequence experiment we used $\lambda = 0.1$, given the smoothness of the object and its trajectory. However, for *Bean_occl* sequence we increased λ to 1, since object occlusion, which is not compensated for in the model, introduces some errors in trajectory estimation which can be somewhat alleviated by giving more emphasis to the smoothing term.

Our multi-frame MC segmentation algorithm is semi-supervised, and, as in the case of M-frame MD algorithm in the previous chapter, we run a batch of experiments with different parameter values and show the best obtained results: for synthetic sequences best results are chosen based on ground truth data, while in the case of natural sequences they are chosen by visual inspection. However, compared to the M-frame MD method where only two parameters need to be chosen, it is much more difficult to chose eight parameters for the segmentation algorithm. On the other hand, parameters α_i and ω_i are not very sensitive: they need to be changed by the order of magnitude to produce substantial change in segmentation results.

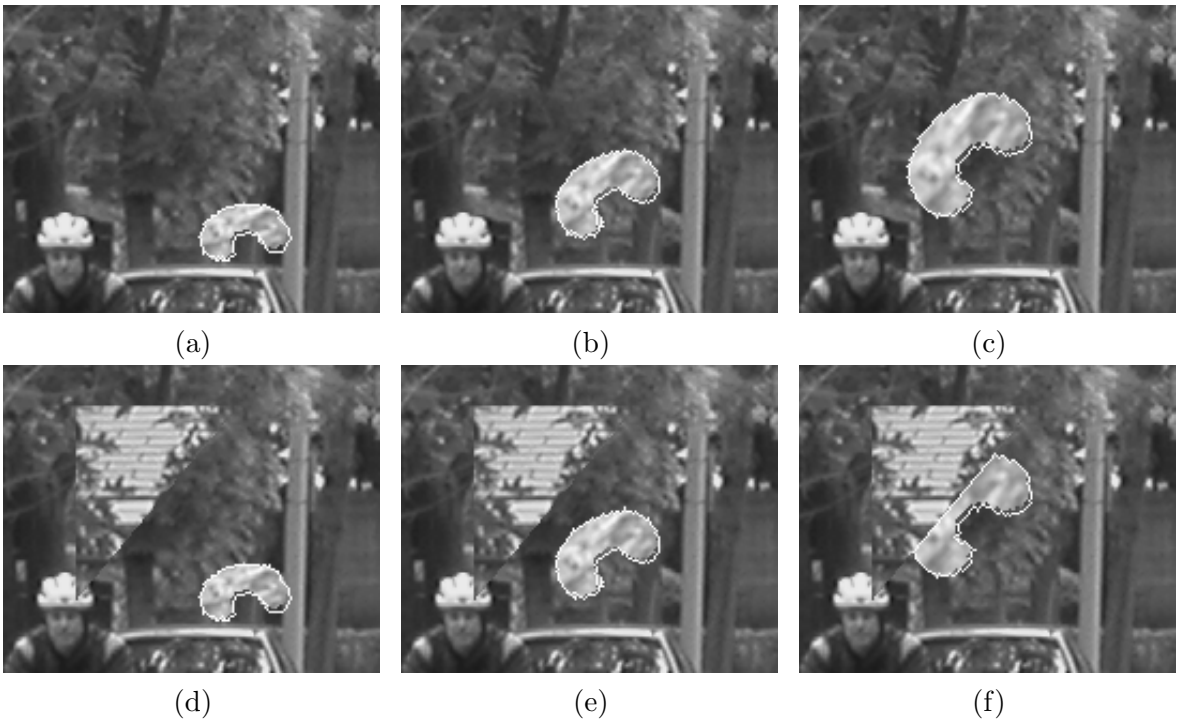


Figure 4·3: Results for the 2-frame MCS algorithm: frames (a) #5, (b) #15, and (c) #25 from the synthetic image sequence *Bean*, and (d) #5, (e) #15 and (f) #25, from the synthetic image sequence *Bean_occl*, overlaid with final boundaries.

First, we verified our multi-frame MC segmentation with background occlusion modeling (M-frame MCS-B) method on the *Bean* sequence. Fig. 4·4 shows two frames from the original sequence together with the final level-set contours (whose intersections define four regions), and the corresponding frames from the sequence of estimated labels. In this experiment we used $\alpha_1 = \alpha_2 = 1$, $\lambda_1 = \lambda_2 = 0.1$, $\omega_1 = \omega_2 = 1000$, $\omega_3 = 500$, and ω_4=1. Since the underlying segmentation is known in this ground-truth experiment, the parameters have been chosen empirically (we are showing the best results out of a batch of experiments). However, the set of weights ω_i used in experiments was chosen based on the following reasoning. Since the object is relatively small and undergoes significant motion, our confidence in the estimated object motion is lower than that in the background motion. This leads to higher values of ω_1, ω_2, ω_3. Although

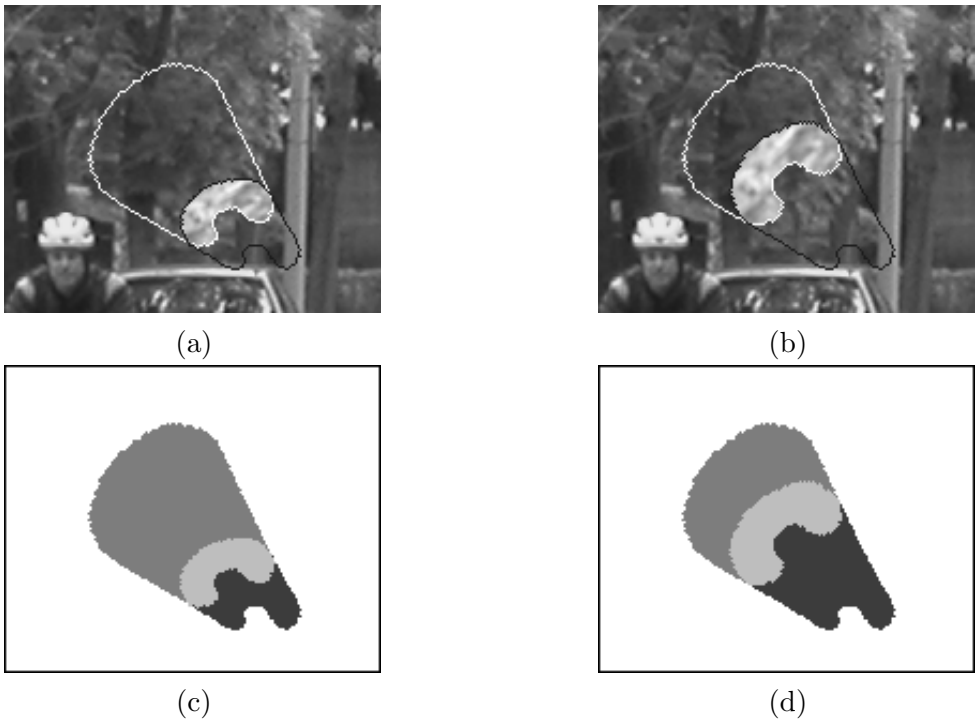


Figure 4·4: Results for the M-frame MCS-B algorithm, applied to synthetic image sequence *Bean*: frames (a) #10 and (b) #20 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

larger λ_1 and λ_2 would eliminate some of the spurious pixels at the object boundary, we keep them low because, given low resolution of the images ("blockiness" of the object boundary), it would also degrade segmentation accuracy. The algorithm converges after 250 iterations, and volumes corresponding to the four segmentation regions are shown in Fig. 4·5. Clearly, the object and background tunnels are very accurate, while the occlusion and exposed volumes, although contain some spurious, isolated voxels at the periphery, are still excellent renditions of the occlusion and uncovering effects occurring in the sequence. This result was obtained with a single segmentation step after the surfaces and motion had been initialized using multi-frame motion detection; no additional motion estimation/segmentation steps were necessary.

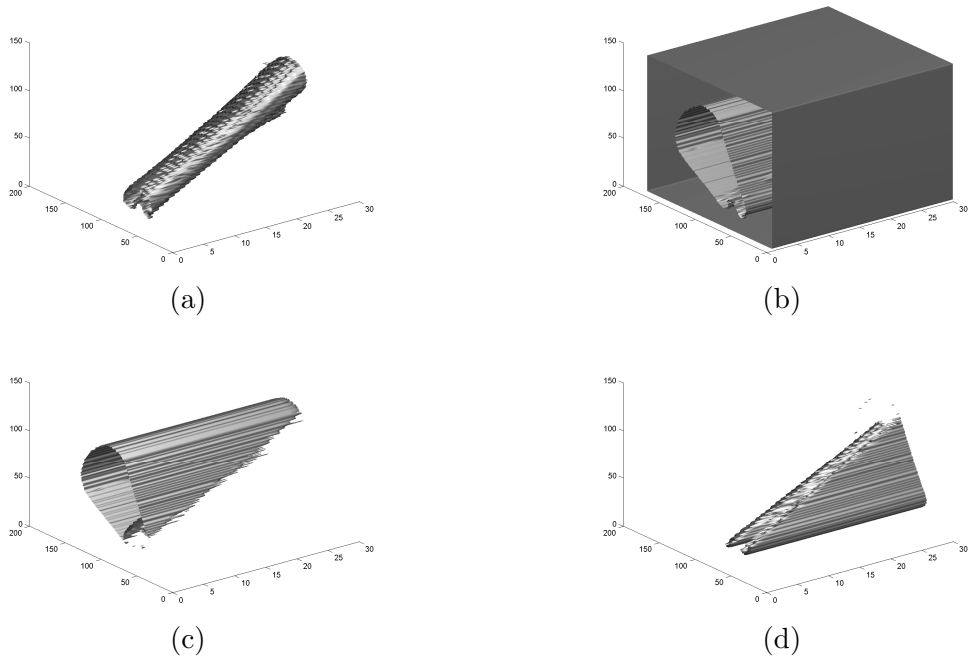


Figure 4·5: Volumes corresponding to results from Fig. 4·4: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

In order to evaluate the segmentation accuracy objectively, we computed the number of misclassified pixels. First, we compare our M-frame MCS-B method with its two-frame counterpart. As Table 4.3 shows, the number of misclassified pixels has dropped from 11.3% for the two-frame segmentation to 0.5% for the multi-frame segmentation that explicitly models background occlusion effects. Next, we compare our M-frame MCS-B method with its simpler cousin, M-frame MD method. Our motion detection partitioning (Fig. 3·4) resulted in 128 misclassified pixels per frame (or 10.8% of object pixels). After one pass of the multiphase segmentation algorithm, this error dropped to only 6 misclassified pixels per frame. This is due to motion compensation introduced in the segmentation algorithm. That solves the largest problem inherent to our motion detection method, i.e., object tunnel is no longer a convex hull of the union of object positions in consecutive frames. Since the *Bean* sequence is noiseless and there is no

object occlusion, all the sources of segmentation errors are removed (except interpolation errors inherent to our motion trajectory model) and solution is almost perfect.

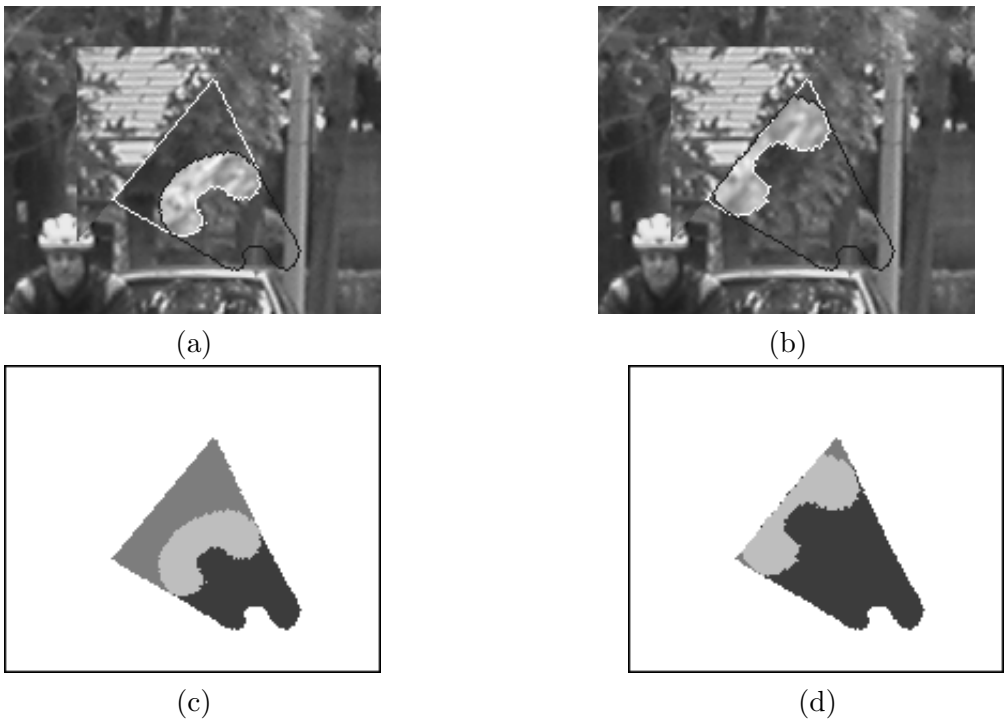


Figure 4·6: Results for the M-frame MCS-B algorithm, applied to synthetic image sequence *Bean_occl*: frames (a) #15 and (b) #25 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

We also applied the new algorithm to the *Bean_occl* sequence and results are shown in Fig. 4·6 and Fig. 4·7. We used the following parameters for the experiment: $\alpha_1 = \alpha_2 = 10$, $\lambda_1 = \lambda_2 = 2.5$, $\omega_4 = 1$, $\omega_1 = \omega_2 = 100$, $\omega_3 = 50$, and the algorithm converges after 1000 iterations. This result was obtained with a single segmentation step after motion estimation was performed and surfaces have been initialized using the motion detection result. No additional motion estimation/segmentation steps were necessary. Similarly to *Bean* sequence experiments, we assign higher values to weights ω_i corresponding to background terms because the object is being occluded by a static

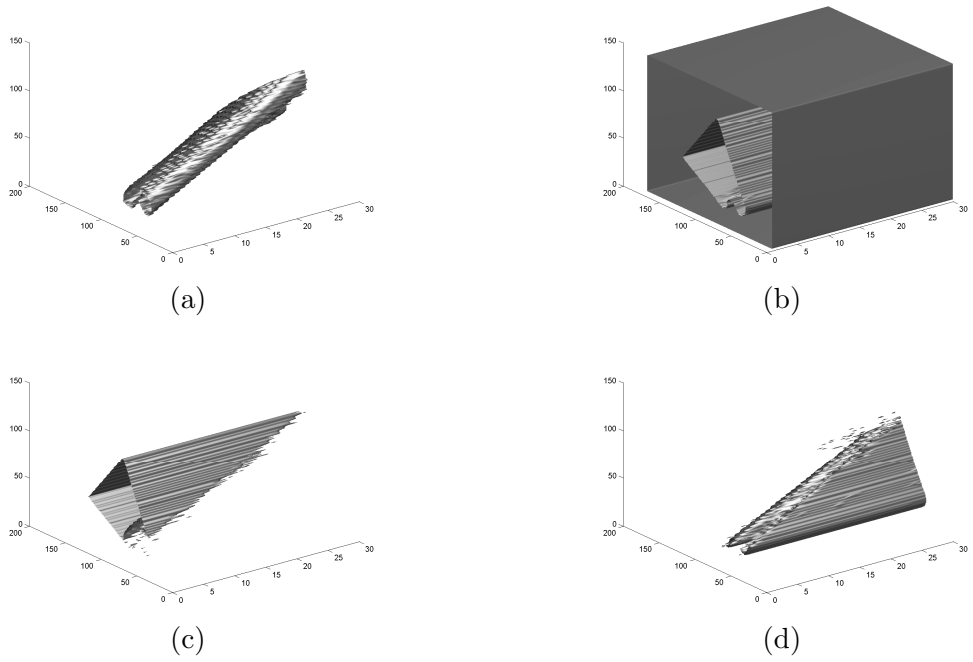


Figure 4·7: Volumes corresponding to results from Fig. 4·6: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

feature in the background and there is no provision for object occlusion in the model; object trajectories are not estimated as precisely thus leading to higher values of ξ_{obj} inside the object, and in order to avoid these voxels being classified as background, we compensate by increasing ω_i's of the background terms. For the same reason, prior terms get more weight through further increased values of λ_i. Similarly to the *Bean* sequence experiment, results are very good. Numerical comparison (Table 4.3) with the two-frame segmentation and multi-frame motion detection method confirms it: the number of misclassified pixels dropped from 130.1 pixels per frame (or 13.9% of object pixels) for the two-frame segmentation and 133.9 pixels per frame (or 14.3% of object pixels) in the case of motion detection to 15.7 pixels per frame (or just 1.7% of object pixels) for the new algorithm.

In another experiment, we applied this algorithm to the natural sequence *Car* over

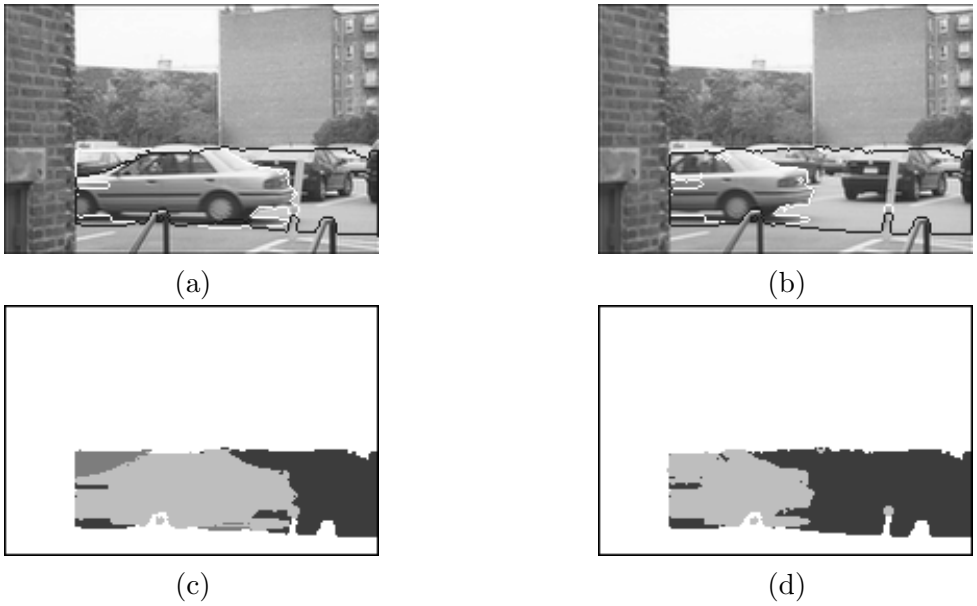
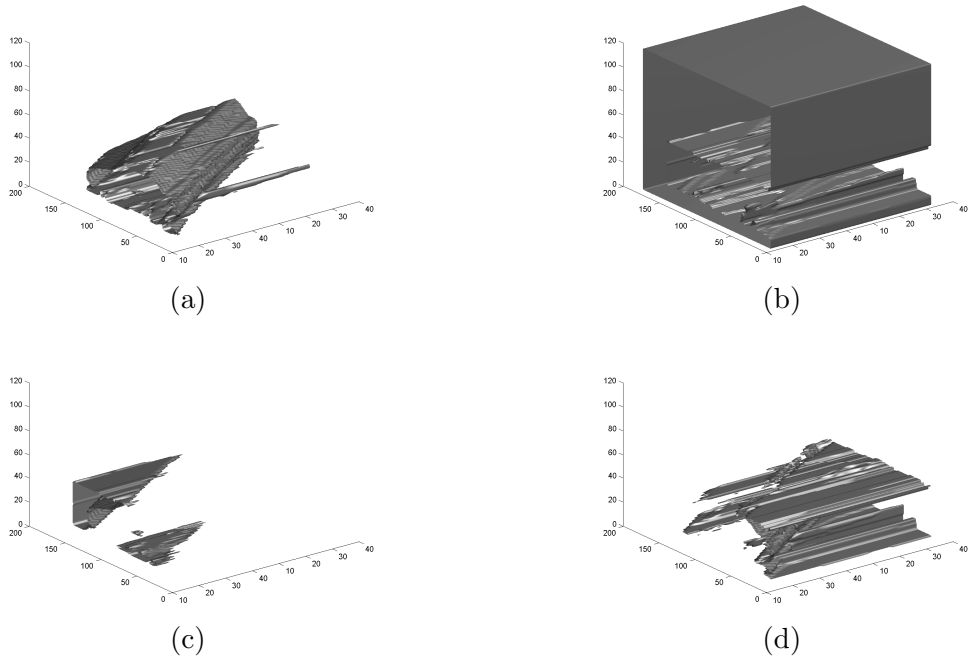


Figure 4·8: Results for the M-frame MCS-B algorithm, applied to *Car* image sequence: frames (a) #54 and (b) #64 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

40 frames (frames #35 through #74 of the original sequence). Fig. 4·8 shows two frames from the subsequence we used overlaid with the final level-set contours and the corresponding label fields. This time we used the following parameters: $\alpha_1 = \alpha_2 = 500$, $\lambda_1 = \lambda_2 = 2.5$, $\omega_1 = 3$, $\omega_2 = 40$, $\omega_3 = 10$, $\omega_4 = 1$ in a single segmentation step (after initialization of the volumes and motion using multi-frame detection result). We needed to adjust these parameters in comparison with the *Bean_occl* experiment, which also has object occlusion, because of stronger noise and background intensity variations in this camera-acquired sequence. Moreover, the background is not perfectly static anymore, so α_i's are increased to compensate for intensity variation which exists over background trajectories even if they are not occluded or exposed by the object. Fig. 4·9 shows volumes corresponding to the four segmentation regions obtained upon convergence at 1000 iterations. For visualization reasons the shown tunnels span the range from frame



(a) (b)

(c) (d)

Figure 4·9: Volumes corresponding to results from Fig. 4·8: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

#10 to frame #40 of the result.

The object tunnel clearly shows a forward horizontal motion and when viewed interactively in 3-D (Ristivojevic, 2004), its walls show a clear imprint of the car body. The occluded and exposed volumes quite accurately depict what will be occluded and what has been exposed; the occlusion region is in front of the car, and the uncovered region grows behind the car. Also, the static background visible throughout the sequence is almost perfectly recovered (note the detection of the static hand-rails in front of the car). Some errors at the car boundary, especially at sequence beginning and end, are due to the fact that the object is fully visible only in a small subset of frames in the sequence. In order to improve the segmentation, object occlusions should be explicitly modeled as well.

Compared to the synthetic sequence experiment (Fig. 4·4 – Fig. 4·7), the natural-data

results are not as good, but that is not unexpected given the noise, intensity variations, and complex motion, all present in the natural sequence. Although a numerical comparison is impossible due to the lack of ground-truth segmentation, visual inspection shows that object segmentation is improved compared to the motion-detection result (Fig. 3·6 and Fig. 3·6) used to initialize the algorithm. The boundary along the car roof-line is tight and accurate in the multiphase result, while the car shadow is accurately delineated below the car (even a tiny part of the shadow seen through the hand rail below the car is accurately classified as part of the moving object). Only the elongated cut-out on car's front fender is erroneous which is not so surprising given the very uniform intensity there.

Table 4.3: Object segmentation error ϵ_s (pixels per frame) for synthetic test sequences, for motion detection and motion compensated segmentation algorithms.

	Two-frame methods		Multi-frame methods	
Sequence	2-frame MD	2-frame MCS	M-frame MD	M-frame MCS-B
Bean	166.0 (14%)	134.0 (11.3%)	128.1 (10.8%)	6.0 (0.5%)
Bean_occl	159.5 (17.1%)	130.1 (13.9%)	133.9 (14.3%)	15.7 (1.7%)

We compare the performance of our multiphase motion segmentation method with the reference two-frame motion-compensated segmentation method presented in Section 2.1.2 numerically (using the error metric defined in (3.13)). Moreover, we compare these methods to the motion detection methods presented in Chapter 3. Table 4.3 shows segmentation errors for these four methods for the two synthetic test sequences, *Bean* and *Bean_occl*. Although the motion compensated two-frame segmentation method outperforms two-frame motion detection method by reducing the error for about 20%, it still performs just about as good as our multi-frame motion detection method. However, our

multiphase, motion segmentation method presented in this chapter outperforms other methods by far, reducing the error more than 20 times in the case of *Bean* sequence (where the object is fully visible), and more than 8 times in the case of *Bean_occl* sequence, where the object is occluded by a static feature in the background, which is not supported by our model, and thus introduces some segmentation errors.

We can conclude that in the case of sequences with only one moving object, fully visible throughout the sequence, our multiphase motion segmentation method is close to optimal, and better than all the other methods we presented so far. However, object occlusions and multiple moving objects are not supported in this model. We will try to solve these two issues with more advanced models in the next two chapters.

Chapter 5

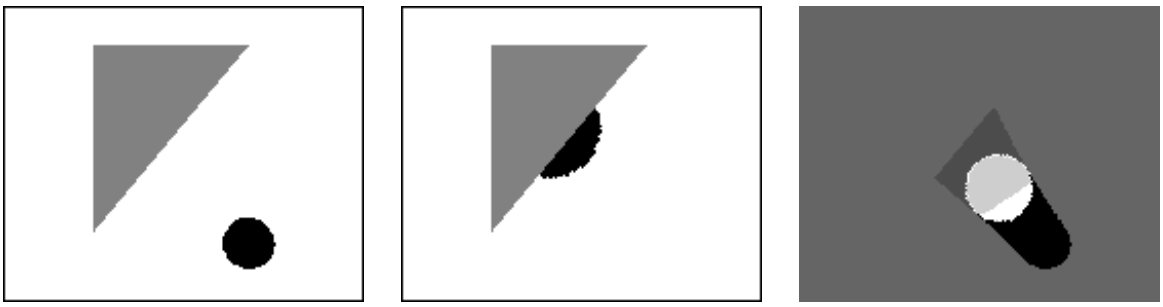
Multiframe motion-based video segmentation with object and background occlusion detection

In this chapter, we extend our video segmentation algorithm presented in the previous chapter by including explicit models of the occluded and newly-exposed areas for both the object and the background (Ristivojević and Konrad, 2004a). We measure object and background intensity variations along motion trajectories spanning the whole temporal support of the image sequence. Clearly, if parts of the object are occluded or exposed within this support, they cannot be accurately modeled either by object or background motion trajectories. As the result, they will be randomly included in either object or background segmentation volumes, thus creating errors. To solve that problem, we explicitly model these regions as occluded and newly-exposed object volumes. We use variational framework for the formulation and multiphase level-set methodology for the solution.

5.1 Energy formulation

We want to partition the domain of an image sequence into six regions (volumes). An example of spatial cross-section through such volumes for a simple binary image sequence with one moving object that is being occluded by a static feature is shown in Fig. 5·1. Compared to the method presented in Chapter 4, we add two more volumes: object area

that is going to be occluded by a feature in the background (object occlusion volume), and object area that was exposed in preceding frames (object exposed volume).



(a) (b) (c)

Figure 5·1: Frames (a) #1; and (b) #30 from a simple binary image sequence, and (c) segmentation regions for frame #15 (white – part of object visible throughout the sequence, light gray – part of object that is going to be occluded, medium gray – background visible throughout the sequence, dark gray – part of background that is going to be occluded, and black – part of background exposed in preceding frames).

Using the multiphase level-set method described in Section 2.3.4, we partition the spatio-temporal volume of $I(\boldsymbol{x}, t)$ into six volumes using three parameterized surfaces, $\vec{\varsigma}_1$, $\vec{\varsigma}_2$, and $\vec{\varsigma}_3$, as shown in Table 5.1, third column. Since three surfaces can partition the image sequence domain into 8 volumes, additional volumes, $\mathcal{V}_4$ and $\mathcal{V}_5$, are defined as "don't care" volumes that will be eliminated during optimization. For the estimation of segmentation surfaces, we extend volume competition formulation (4.1) to take into account occluded and exposed object volumes, which leads to the following energy

Table 5.1: Segmentation volumes and associated energy terms for 3-surface example

Volume $\mathcal{V}_n$	j_n	$(\boldsymbol{x}, t)$	Level-set func. at $(\boldsymbol{x}, t)$	Energy terms
$\mathcal{V}_1$ = background	(000)	outside $\vec{\varsigma}_1, \vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 < 0,\ \phi_2 < 0,\ \phi_3 < 0$	$\omega_1 \xi_{obj}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_2$ = occluded backgr.	(001)	inside $\vec{\varsigma}_1$, outside $\vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 < 0,\ \phi_3 < 0$	$\omega_2 \xi_{occ}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_3$ = exposed backgr.	(010)	inside $\vec{\varsigma}_2$, outside $\vec{\varsigma}_1, \vec{\varsigma}_3$	$\phi_1 < 0,\ \phi_2 > 0,\ \phi_3 < 0$	$\omega_3 \xi_{exp}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_4$ = unused	(011)	inside $\vec{\varsigma}_1, \vec{\varsigma}_2$, outside $\vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 > 0,\ \phi_3 < 0$	K_{pen}
$\mathcal{V}_5$ = unused	(100)	inside $\vec{\varsigma}_3$, outside $\vec{\varsigma}_1, \vec{\varsigma}_2$	$\phi_1 < 0,\ \phi_2 < 0,\ \phi_3 > 0$	K_{pen}
$\mathcal{V}_6$ = exposed object	(101)	inside $\vec{\varsigma}_1, \vec{\varsigma}_3$, outside $\vec{\varsigma}_2$	$\phi_1 > 0,\ \phi_2 < 0,\ \phi_3 > 0$	$\omega_6 \xi_{exp}(\boldsymbol{x}, t; \boldsymbol{p})$
$\mathcal{V}_7$ = occluded object	(110)	inside $\vec{\varsigma}_2, \vec{\varsigma}_3$, outside $\vec{\varsigma}_1$	$\phi_1 < 0,\ \phi_2 > 0,\ \phi_3 > 0$	$\omega_7 \xi_{occ}(\boldsymbol{x}, t; \boldsymbol{p})$
$\mathcal{V}_8$ = object	(111)	inside $\vec{\varsigma}_1, \vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 > 0,\ \phi_3 > 0$	$\omega_8 \xi_{obj}(\boldsymbol{x}, t; \boldsymbol{p})$

minimization:

$$\begin{aligned}\min_{\vec{\varsigma}_1, \vec{\varsigma}_2, \vec{\varsigma}_3}\ & \omega_1 \iiint_{\mathcal{V}_1} \xi_{obj}(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}) d\boldsymbol{x} dt + \omega_2 \iiint_{\mathcal{V}_2} \xi_{occ}(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}) d\boldsymbol{x} dt + \\ & \omega_3 \iiint_{\mathcal{V}_3} \xi_{exp}(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}) d\boldsymbol{x} dt + \omega_6 \iiint_{\mathcal{V}_6} \xi_{exp}(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}) d\boldsymbol{x} dt + \\ & \omega_7 \iiint_{\mathcal{V}_7} \xi_{occ}(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}) d\boldsymbol{x} dt + \omega_8 \iiint_{\mathcal{V}_8} \xi_{obj}(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}) d\boldsymbol{x} dt + \\ & \iiint_{\mathcal{V}_4} K_{pen} d\boldsymbol{x} dt + \iiint_{\mathcal{V}_5} K_{pen} d\boldsymbol{x} dt + \\ & \lambda_1 \iint_{\mathcal{S}_1} d\vec{\varsigma}_1, + \lambda_2 \iint_{\mathcal{S}_2} d\vec{\varsigma}_2, + \lambda_3 \iint_{\mathcal{S}_3} d\vec{\varsigma}_3,\end{aligned} \tag{5.1}$$

where $\vec{\varsigma}_1 = \partial(\mathcal{V}_1 \cup \mathcal{V}_3 \cup \mathcal{V}_6 \cup \mathcal{V}_7)$, $\vec{\varsigma}_2 = \partial(\mathcal{V}_1 \cup \mathcal{V}_4 \cup \mathcal{V}_5 \cup \mathcal{V}_7)$, $\vec{\varsigma}_3 = \partial(\mathcal{V}_1 \cup \mathcal{V}_5 \cup \mathcal{V}_6 \cup \mathcal{V}_8)$, and $\cup_{i=1}^{8} \mathcal{V}_i = \Omega \times \mathcal{T}$, while $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ are areas of the three surfaces. The weights ω_i reflect the uncertainty as to how accurately motion parameters can explain the dynamics occurring in individual volumes, while constants λ_1, λ_2, and λ_3 associate a cost with

the Euclidean area elements $d\vec{\varsigma_1}$, $d\vec{\varsigma_2}$, and $d\vec{\varsigma_3}$, respectively. Penalty terms, K_{pen}, are introduced to discourage assigning a point to the two unused volumes ($\mathcal{V}_4$ and $\mathcal{V}_5$). The individual terms of the above energy formulation measure cumulative consistency of image sequence voxels with models corresponding to different volumes we are estimating. Apart from the four terms we described in Section 4.1, there are two more additional terms: *occluded object volume* term and *exposed object volume* term.

In order that a point be declared as object point to-be-occluded at a later time, intensity variation along object motion trajectory passing through this point must be small up to this point and is expected to become large at some time in the future (as illustrated in Fig. 5·2 (a)). Similarly to (4.3), we propose the following *occluded object volume* term:

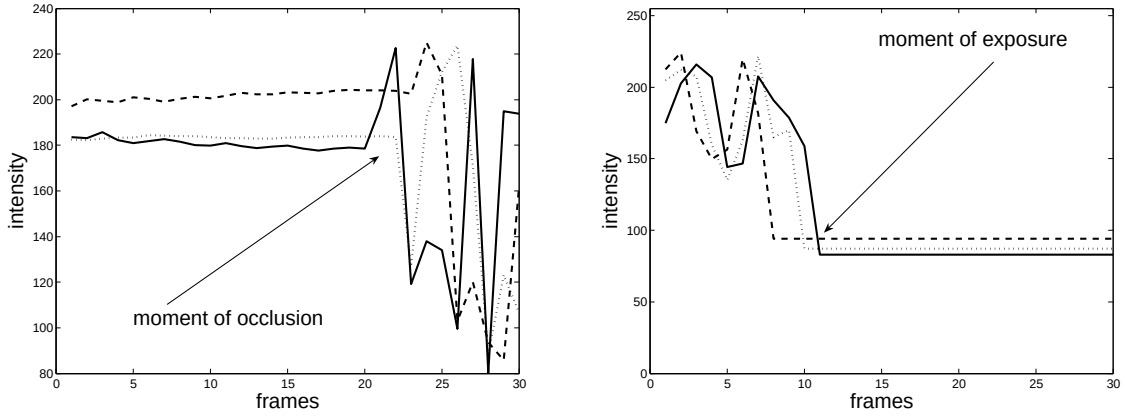
$$\xi_{occ}(\boldsymbol{x}, t = t_j; \boldsymbol{p}) = \sigma^2_{1,j}(\boldsymbol{x}, t; \boldsymbol{p}) + \frac{\alpha_3}{\sigma^2_{j,K}(\boldsymbol{x}, t; \boldsymbol{p}) + 1}, \tag{5.2}$$

where j is defined as in (4.3) In order that ξ_{occ} be small, motion $\boldsymbol{p}$ must exist that induces small intensity variations up to t_j, while inducing large variations after t_j; the compromise between the two terms is adjusted by weight α_3.

For a point to be declared as object point exposed in the past, intensity variations along object motion trajectory are expected to be large prior to this point and small afterwards (example in Fig. 5·2 (b)). Similarly to (4.4), we propose the following *exposed object volume* term:

$$\xi_{exp}(\boldsymbol{x}, t = t_j; \boldsymbol{p}) = \frac{\alpha_4}{\sigma^2_{1,j}(\boldsymbol{x}, t; \boldsymbol{p}) + 1} + \sigma^2_{j,K}(\boldsymbol{x}, t; \boldsymbol{p}), \tag{5.3}$$

with j defined as above. Again, ξ_{exp} will be small if motion $\boldsymbol{p}$ induces large intensity variations up to t_j and small variations afterwards; α_4 balances the impact of the two terms.



(a) (b)

Figure 5·2: Example of intensity variations along motion trajectories in (a) occluded and (b) newly-exposed object areas.

5.2 Solution method

As in Chapter 4, the problem of joint segmentation and estimation of motion parameters is decomposed into two interleaved minimizations: estimation of motion parameters given segmentation surfaces, and estimation of segmentation surfaces with fixed motion parameters. We describe the latter minimization; details of motion parameter estimation can be found in Section 4.2.2. Following the multiphase level-set method formalism, we represent energy minimization in (5.1) using three level-set functions, $\phi_1(\boldsymbol{x}, t)$, $\phi_2(\boldsymbol{x}, t)$,

and $\phi_3(\boldsymbol{x},t)$, as follows ($\phi_i(\boldsymbol{x},t)$, $i=1,2,3$, is replaced by ϕ_i for brevity):

$$\begin{aligned}
\min_{\phi_1,\phi_2,\phi_3} \mathcal{E}, \qquad \mathcal{E} = \quad & \omega_1 \iiint_{\Omega\times\mathcal{T}} \xi_{obj}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})(1-H(\phi_1))(1-H(\phi_2))(1-H(\phi_3))d\boldsymbol{x}dt \\
& + \omega_2 \iiint_{\Omega\times\mathcal{T}} \xi_{occ}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})H(\phi_1)(1-H(\phi_2))(1-H(\phi_3))d\boldsymbol{x}dt \\
& + \omega_3 \iiint_{\Omega\times\mathcal{T}} \xi_{exp}(\boldsymbol{x},t;\widehat{\widehat{\boldsymbol{p}}})(1-H(\phi_1))H(\phi_2)(1-H(\phi_3))d\boldsymbol{x}dt \\
& + \iiint_{\Omega\times\mathcal{T}} K_{pen}H(\phi_1)H(\phi_2)(1-H(\phi_3))d\boldsymbol{x}dt \\
& + \iiint_{\Omega\times\mathcal{T}} K_{pen}(1-H(\phi_1))(1-H(\phi_2))H(\phi_3)d\boldsymbol{x}dt \\
& + \omega_6 \iiint_{\Omega\times\mathcal{T}} \xi_{exp}(\boldsymbol{x},t;\widehat{\boldsymbol{p}})H(\phi_1)(1-H(\phi_2))H(\phi_3)d\boldsymbol{x}dt \\
& + \omega_7 \iiint_{\Omega\times\mathcal{T}} \xi_{occ}(\boldsymbol{x},t;\widehat{\boldsymbol{p}})(1-H(\phi_1))H(\phi_2)H(\phi_3)d\boldsymbol{x}dt \\
& + \omega_8 \iiint_{\Omega\times\mathcal{T}} \xi_{obj}(\boldsymbol{x},t;\widehat{\boldsymbol{p}})H(\phi_1)H(\phi_2)H(\phi_3)d\boldsymbol{x}dt \\
& + \lambda_1 \iiint_{\Omega\times\mathcal{T}} |\nabla H(\phi_1)|d\boldsymbol{x}dt + \lambda_2 \iiint_{\Omega\times\mathcal{T}} |\nabla H(\phi_2)|d\boldsymbol{x}dt + \\
& + \lambda_3 \iiint_{\Omega\times\mathcal{T}} |\nabla H(\phi_3)|d\boldsymbol{x}dt.
\end{aligned} \tag{5.4}$$

Volumes we are estimating are now defined through intersections of zero-level sets of surfaces $\phi_1(\boldsymbol{x},t)$, $\phi_2(\boldsymbol{x},t)$, and $\phi_3(\boldsymbol{x},t)$, as shown in Table 5.1, fourth column.

In order to carry out minimization (5.4), level-set surfaces should be evolved along the direction of steepest descent, which is the direction of the negative gradient of energy $\mathcal{E}$ with respect to ϕ_1, ϕ_2, and ϕ_3. As a result of minimization, we obtain the following

level-set evolution equations (valid for all $(\boldsymbol{x}, t)$ that are omitted for brevity):

$$
\begin{aligned}
\frac{\partial \phi_1(\tau)}{\partial \tau} &= F_1 \|\nabla \phi_1(\tau)\| = \|\nabla \phi_1(\tau)\| \\
&\quad \{\lambda_1 \kappa_{m_1} - [(\omega_8 \xi_{obj}(\widehat{\boldsymbol{p}}) - \omega_7 \xi_{occ}(\widehat{\boldsymbol{p}})) H(\phi_2(\tau)) H(\phi_3(\tau)) + \\
&\quad (\omega_6 \xi_{exp}(\widehat{\boldsymbol{p}}) - K_{pen})(1 - H(\phi_2(\tau))) H(\phi_3(\tau)) + \\
&\quad (K_{pen} - \omega_3 \xi_{exp}(\widehat{\widehat{\boldsymbol{p}}})) H(\phi_2(\tau))(1 - H(\phi_3(\tau))) + \\
&\quad (\omega_2 \xi_{occ}(\widehat{\widehat{\boldsymbol{p}}}) - \omega_1 \xi_{obj}(\widehat{\widehat{\boldsymbol{p}}}))(1 - H(\phi_2(\tau)))(1 - H(\phi_3(\tau)))]\} \\
\frac{\partial \phi_2(\tau)}{\partial \tau} &= F_2 \|\nabla \phi_2(\tau)\| = \|\nabla \phi_2(\tau)\| \\
&\quad \{\lambda_2 \kappa_{m_2} - [(\omega_8 \xi_{obj}(\widehat{\boldsymbol{p}}) - \omega_6 \xi_{exp}(\widehat{\boldsymbol{p}})) H(\phi_1(\tau)) H(\phi_3(\tau)) + \\
&\quad (\omega_7 \xi_{occ}(\widehat{\boldsymbol{p}}) - K_{pen})(1 - H(\phi_1(\tau))) H(\phi_3(\tau)) + \\
&\quad (K_{pen} - \omega_2 \xi_{occ}(\widehat{\widehat{\boldsymbol{p}}})) H(\phi_1(\tau))(1 - H(\phi_3(\tau))) + \\
&\quad (\omega_3 \xi_{exp}(\widehat{\widehat{\boldsymbol{p}}}) - \omega_1 \xi_{obj}(\widehat{\widehat{\boldsymbol{p}}}))(1 - H(\phi_1(\tau)))(1 - H(\phi_3(\tau)))]\} \\
\frac{\partial \phi_3(\tau)}{\partial \tau} &= F_3 \|\nabla \phi_3(\tau)\| = \|\nabla \phi_3(\tau)\| \\
&\quad \{\lambda_3 \kappa_{m_3} - [(\omega_8 \xi(\widehat{\boldsymbol{p}}) - K_{pen}) H(\phi_1(\tau)) H(\phi_2(\tau)) + \\
&\quad (\omega_7 \xi_{occ}(\widehat{\boldsymbol{p}}) - \omega_3 \xi_{exp}(\widehat{\widehat{\boldsymbol{p}}}))(1 - H(\phi_1(\tau))) H(\phi_2(\tau)) + \\
&\quad (\omega_6 \xi_{exp}(\widehat{\boldsymbol{p}}) - \omega_2 \xi_{occ}(\widehat{\widehat{\boldsymbol{p}}})) H(\phi_1(\tau))(1 - H(\phi_2(\tau))) + \\
&\quad (K_{pen} - \omega_1 \xi_{obj}(\widehat{\widehat{\boldsymbol{p}}}))(1 - H(\phi_1(\tau)))(1 - H(\phi_2(\tau)))]\},
\end{aligned}
$$

where τ is the algorithmic evolution time, and κ_{m_1}, κ_{m_2}, and κ_{m_3} are mean curvatures (defined in (3.12)) of level-set surfaces ϕ_1, ϕ_2, and ϕ_3, respectively. We implement these equations iteratively using standard discretization as described in Section 3.2. In each iteration we calculate the forces F_1, F_2, and F_3 at zero level-set points of all surfaces, extend these forces using the fast marching algorithm by solving $\phi_i \cdot \nabla F_i = 0$ for F_i, $i = 1, 2, 3$, and update the surfaces ϕ_1, ϕ_2, and ϕ_3. Re-initialization of the surfaces using the fast marching algorithm by solving $\|\nabla \phi_i\|$=1 is performed every 100 iterations to

keep surfaces as close as possible to the signed distance function.

5.3 Experimental results

We again initialize the algorithm with volumes and motion parameters resulting from the multi-frame motion detection, as described in Section 4.2.2. First, we applied our multi-frame MC segmentation with background and object occlusion modeling (M-frame MCS-BO) algorithm to the synthetic test sequence *Bean_occl*. Fig. 5·3(a–b) shows two frames of the original sequence overlaid with the estimated boundaries. We used the following parameters: $\alpha_1 = \alpha_2 = 1$, $\alpha_3 = \alpha_4 = 100$, $\lambda_1 = \lambda_2 = \lambda_3 = 2.5$, $\omega_1 = \omega_2 = \omega_3 = \omega_6 = 10$, $\omega_7 = \omega_8 = 1$, $K_{pen} = 100$. Since there are six forces competing in this

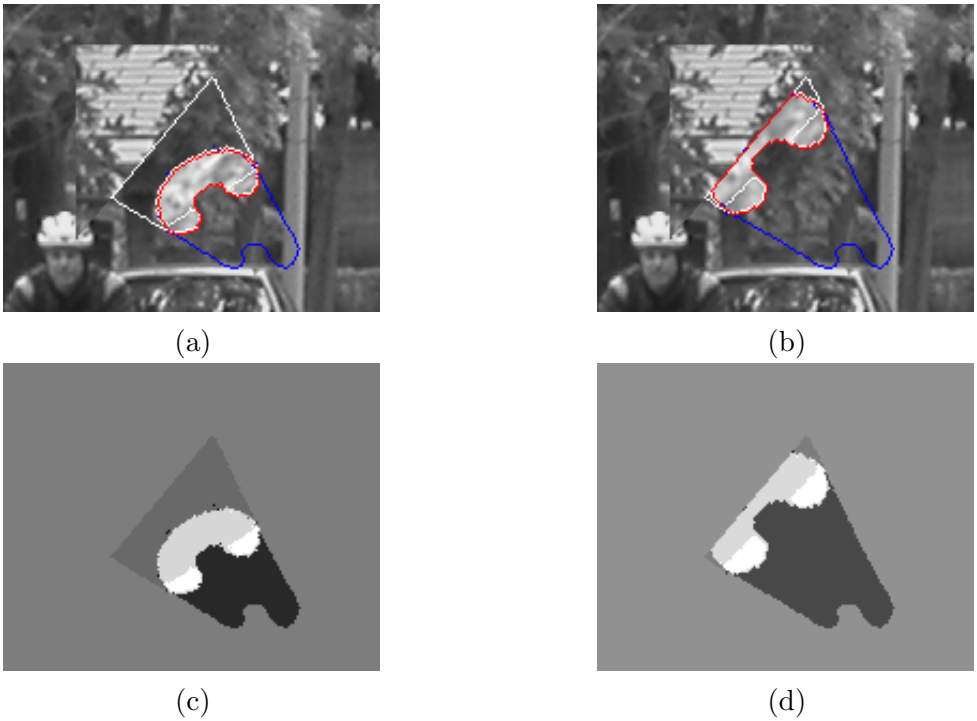


Figure 5·3: Results for the M-frame MCS-BO algorithm, applied to synthetic image sequence *Bean_occl*: frames (a) #15 and (b) #25 from the sequence overlaid with final boundaries, and (c-d) corresponding label fields (white – object, light gray to dark gray: object occlusion, background, background occlusion, and background exposed).

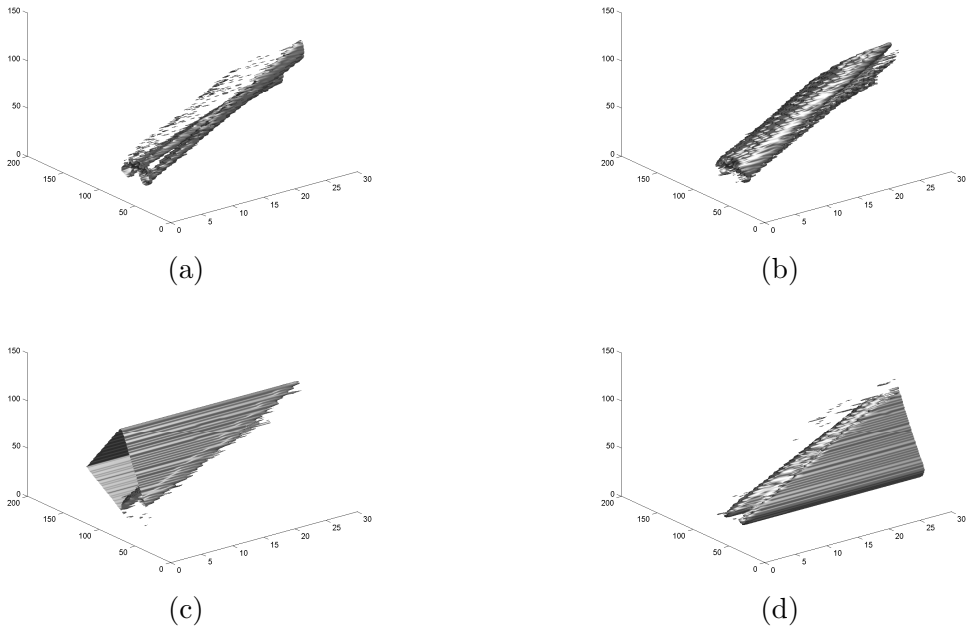


Figure 5·4: Volumes corresponding to results from Fig. 5·3: (a) object tunnel, (b) object occlusion volume, (c) background occlusion volume , and (d) background exposed volume.

case, we further increased values of λ_i's compared to previous experiments on the same sequence. Also, α_3 and α_4 are larger than α_1 and α_2 because of much higher variation of intensity over object trajectories compared to background trajectories (background is perfectly still). These parameters have to be large enough to make object's occlusion (exposed) terms small only if intensity variation over trajectory is larger after (before) the current voxel due to object occlusion (exposure), and not due to a variation of intensity over fully-visible object trajectory. Similarly, since the intensity variations over object trajectories are larger than over background trajectories, ω_8 is smaller than ω_1. Finally, weights are increased for background occlusion and exposed terms to obtain the best possible results. We used two steps of motion estimation and segmentation (in the first step algorithm converges after 3000 iterations, with the following parameters: $\alpha_1 = \alpha_2 = 5$, $\alpha_3 = \alpha_4 = 500$, $\lambda_1 = \lambda_2 = \lambda_3 = 2.5$, $\omega_1 = \omega_2 = \omega_3 = 10$, $\omega_6 = \omega_7 = 5$,

$\omega_8 = 1$, $K_{pen} = 100$). The algorithm converges after 1900 iterations of the second step. Fig. 5·3(c–d) shows two frames of the resulting segmentation labels. Clearly, the object and background as well as to-be-occluded and exposed parts of the background are accurately estimated. Also, to-be-occluded and exposed parts of the object are very well recovered, although they contain some spurious, isolated voxels at the boundary of the object. Four of the final six tunnels are shown in Fig. 5·4. The segmentation error is further reduced as is shown in the Table 5.2, although only slightly.

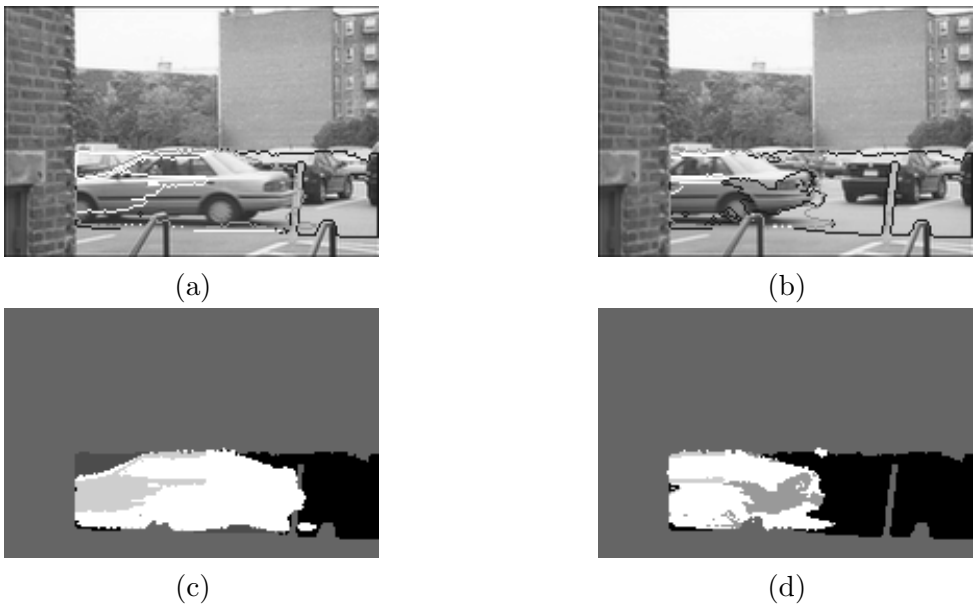


Figure 5·5: Results for the M-frame MCS-BO algorithm, applied to *Car* image sequence: frames (a) #54 and (b) #64 from the sequence overlaid with final boundaries, and (c–d) corresponding label fields (white – object, light gray to dark gray: object occlusion, object exposed, background, background occlusion, and background exposed).

We also applied the algorithm to the *Car* sequence, again over 40 frames (frames #35 through #74 of the original sequence). Fig. 5·5(a–b) shows two frames of the original sequence overlaid with the final boundaries. This time we used the following parameters: $\alpha_1 = \alpha_2 = 500$, $\alpha_3 = \alpha_4 = 2 * 10^4$, $\lambda_1 = \lambda_2 = \lambda_3 = 2.5$, $\omega_1 = 2, \omega_2 = \omega_3 = 10$, $\omega_6 = \omega_7 = 4$, $\omega_8 = 1$, $K_{pen} = 100$. Here, we used the same values for parameters α_1, α_2,

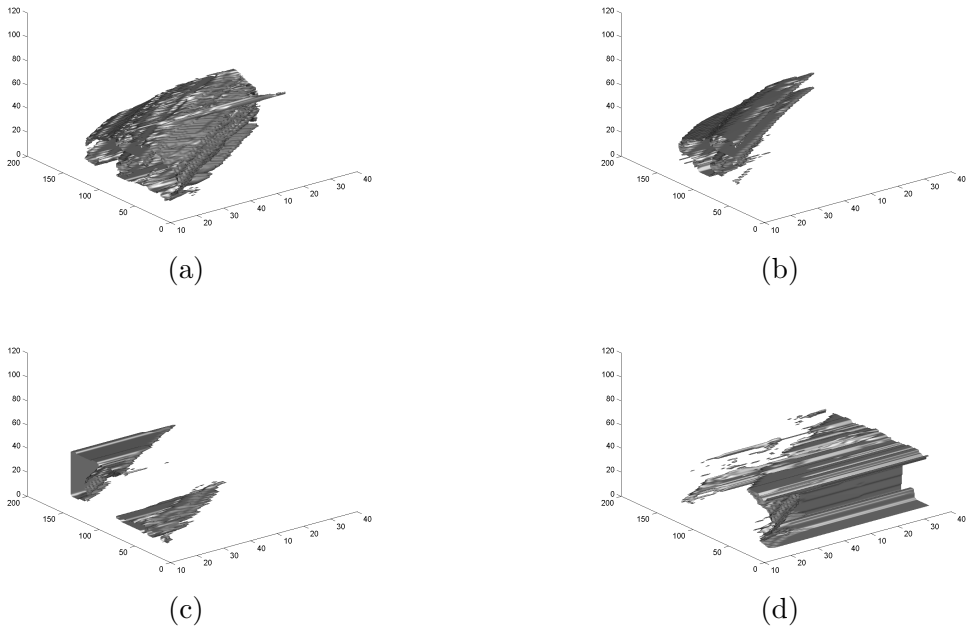


Figure 5·6: Volumes corresponding to results from Fig. 5·5: (a) object tunnel, (b) object occlusion volume, (c) background occlusion volume , and (d) background exposed volume.

λ_1 and λ_2 as in the case of the algorithm modeling occlusions in the background only. However, due to much larger variations of intensity over object trajectories in comparison with background trajectories, we needed to increase α_3 and α_4. The weights ω_i have similar values as in the experiment with synthetic *Bean_occl* sequence. Fig. 5·5(c–d) shows two frames of the resulting segmentation labels upon algorithm convergence after 1000 iterations. Clearly, all background regions are well estimated: the background area to be occluded is in front of the car while the exposed one grows behind the car. Object occlusion and uncovered regions are in the right place (front and rear of the car, respectively) but they are not precise. This is, we believe, due to inaccuracies in the estimated motion and variations of object intensity over time. Fig. 5·6 shows the object tunnel, object occlusion volume and background occlusion/exposed volumes corresponding to the four segmentation regions . Again, for visualization reasons the

shown tunnels are from frame #10 to frame #40.

Despite imprecise occluded and exposed object regions, errors in object boundary, clear in Fig. 4·8(c–d), have now been corrected. Overall, the car is segmented more accurately (considering the union of fully-visible, to-be-occluded and exposed areas of the car), and even the static hand rails are very precisely recovered. However, in order to improve the accuracy of occluded and exposed object areas, more advanced models are needed.

Table 5.2: Object segmentation error ϵ_s (pixels per frame) for synthetic test sequences.

	Two-frame methods		Multi-frame methods		
Sequence	2-frame MD	2-frame MCS	M-frame MD	M-frame MCS-B	M-frame MCS-BO
Bean	166.0 (14%)	134.0 (11.3%)	128.1 (10.8%)	6.0 (0.5%)	-[1]
Bean_occl	159.5 (17.1%)	130.1 (13.9%)	133.9 (14.3%)	15.7 (1.7%)	12.9 (1.4%)

We now compare all the methods presented so far, starting with the two-frame motion detection and finishing with the multi-frame MC segmentation with background and object occlusion modeling. We show segmentation errors for each of these methods, for two synthetic test sequences, in Table 5.2. Comparing results in Fig. 3·3 and Fig. 4·3, we can see that both two-frame methods perform reasonably well, with the motion-compensated result being somewhat better. However, even the simplest multi-frame method, based on motion detection (Fig. 3·4 and Fig. 3·5) outperforms the two-frame methods. A huge improvement is obtained by the introduction of motion models and background occlusion modeling in the M-frame MCS-B method (Fig. 4·4 and Fig. 4·6). Error drops from 10-15% to less than 2%. In the case of *Bean_occl* sequence, a fur-

[1]This experiment is not performed since there are no occlusions in *Bean* sequence.

ther improvement is obtained by explicit modeling of object occlusions in the M-frame MCS-BO method (Fig. 5·3). We can conclude that performing segmentation jointly over a group of frames, in the spatio-temporal domain, indeed improves segmentation results, while the introduction of motion models and handling of background and object occlusions reduces errors to almost negligible levels (in the case of synthetic sequences).

Chapter 6

Generalization to multiple moving objects

In the previous two chapters, we presented motion-based segmentation methods which explicitly model occlusion and newly-exposed regions in the object and the background. However, these formulations allowed for only one moving object and possibly moving background. We would like to extend these models to include any number of moving objects with their occluded and newly-exposed areas. Using the multiphase methodology, we develop general variational formulation which allows for multiple moving objects, some of which can be occluded or exposed, moving background, and occluded and exposed regions in the background. In order to achieve that, we use several level-set surfaces whose intersections define all these regions. We minimize that formulation using optimization methods described in previous chapters, and present results for a natural video sequence with two moving objects.

6.1 Energy formulation

Using the multiphase level-set framework described in Section 2.3.4, we partition the domain of the image sequence into N volumes using M parameterized surfaces, $\vec{\varsigma}_i$, $i = 1, 2, ..., M$, where $N \leq 2^M$. An example of spatial cross-section through such volumes for a simple binary image sequence with one moving object that is being occluded by a static feature is shown in Fig. 5·1. Since M surfaces partition the image sequence domain into 2^M volumes, whenever $N < 2^M$ we define unused $2^M - N$ volumes as "don't care" volumes that will be eliminated during optimization.

Let $\boldsymbol{p}, ..., \boldsymbol{p}_L, \bar{\boldsymbol{p}}$ be motion parameters (e.g., affine) associated with L objects and background, respectively. These L+1 sets are associated with up to $3(L$+1) different areas (visible, exposed and to be occluded). Clearly, $N \geq 3(L+1)$ must hold. For the estimation of segmentation surfaces with fixed motion parameters ($\boldsymbol{p}_i = \widehat{\boldsymbol{p}}_i$), we extend volume competition formulation (5.1) to take into account multiple moving objects and associated occluded and exposed object volumes, which leads to the following energy minimization:

$$\min_{\vec{\varsigma}_1,...,\vec{\varsigma}_M} \sum_{i=1}^{N} \omega_i \iiint_{\mathcal{V}_i} \xi_i(\boldsymbol{x}, t; \widehat{\boldsymbol{p}}_i) d\boldsymbol{x} dt + \sum_{i=N+1}^{2^M} \iiint_{\mathcal{V}_i} K_{pen} d\boldsymbol{x} dt + \sum_{i=1}^{M} \lambda_i \iint_{\mathcal{S}_i} d\vec{\varsigma}_i, \qquad (6.1)$$

where $\mathcal{S}_i$ is the area of surface $\vec{\varsigma}_i$. The weights ω_i reflect uncertainty as to how accurately motion parameters can explain the dynamics occurring in individual volumes, while constants λ_i associate a cost with the Euclidean areas $d\vec{\varsigma}_i$. The penalty K_{pen} discourages assigning a point to unused volumes. As we mentioned before, our M-frame MC segmentation method is semi-supervised: user is required to choose values for parameters λ_i and ω_i (K_{pen} is also a parameter, but the same large value can be used in all experiments). The more volumes we are estimating, the more parameters have to be chosen and the more difficult it gets. Based on the experiments we performed, segmentation results are not very sensitive to the change in the weights ω_i: they need to be changed by the order of magnitude before any substantial change in the segmentation results can be observed. Still, finding an automatic way to estimate good parameter values is an important issue for future work. Individual terms $\xi_i(\boldsymbol{x}, t; \boldsymbol{p}_i)$ of the first sum in the above energy minimization are designed to measure the consistency of image sequence voxels with one of the scenarios: visible ($\xi_{vis}(\boldsymbol{x}, t; \boldsymbol{p}_i)$), to-be-occluded ($\xi_{occ}(\boldsymbol{x}, t; \boldsymbol{p}_i)$) or newly-exposed($\xi_{exp}(\boldsymbol{x}, t; \boldsymbol{p}_i)$). These terms are introduced in Section 4.1, for the case of background occluded and exposed volumes (equations (4.2)–(4.4)). By replacing $\bar{\boldsymbol{p}}$ with

$\boldsymbol{p}_i$ we define the *object/background volume* term:

$$\xi_{vis}(\boldsymbol{x}, t; \boldsymbol{p}_i) = \sigma^2_{1,K}(\boldsymbol{x}, t; \boldsymbol{p}_i),$$

occluded volume term:

$$\xi_{occ}(\boldsymbol{x}, t; \boldsymbol{p}_i) = \sigma^2_{1,j}(\boldsymbol{x}, t; \boldsymbol{p}_i) + \frac{\alpha_1}{\sigma^2_{j,K}(\boldsymbol{x}, t; \boldsymbol{p}_i) + 1},$$

and *exposed volume* term:

$$\xi_{exp}(\boldsymbol{x}, t; \boldsymbol{p}_i) = \frac{\alpha_2}{\sigma^2_{1,j}(\boldsymbol{x}, t; \boldsymbol{p}_i) + 1} + \sigma^2_{j,K}(\boldsymbol{x}, t; \boldsymbol{p}_i)$$

6.2 Solution method

As in Chapter 4 and Chapter 5, we decouple estimation of motion parameters and estimation of segmentation surfaces into two interleaved minimizations. We describe the latter minimization; details of motion parameter estimation can be found in Section 4.2.2.

Following the multiphase level-set method formalism (Vese and Chan, 2002), we accomplish energy minimization in (6.1) using M level-set functions, $\phi_i(\boldsymbol{x}, t)$, $i = 1, ..., M$. Volumes that we seek are now defined through intersections of zero-level sets of surfaces ϕ_i. In order to carry out minimization (6.1), level-set functions should evolve along the direction of steepest descent, which is the direction of negative gradient of the total energy with respect to ϕ_i. As the result of minimization, we obtain a set of level-set evolution equations. We identify each volume sought by the following binary label: $j_n = (a^1_n, ..., a^M_n)$ for $1 \leq n \leq 2^M$, with each a^m_n $(m = 1, ..., M)$ being either 0 or 1. With this notation, volume #1 is identified by the binary label $j_1 = (1, 0, ..., 0)$. Each voxel's

membership in a volume can be established using the following indicator function:

$$\chi_{j_n}(\boldsymbol{x},t) = \prod_{i=1}^{M}[(1-a_n^i) + (2a_n^i - 1)H(\phi_i(\boldsymbol{x},t))],$$

where $H(\cdot)$ is the Heaviside step function. Using this notation, evolution equation for one surface valid at all $(\boldsymbol{x},t)$, that are omitted for brevity, is:

$$\begin{aligned}\frac{\partial\phi_l(\tau)}{\partial\tau} &= F_l\|\nabla\phi_l(\tau)\| = \|\nabla\phi_l(\tau)\|\{\lambda_l\kappa_{m_l}+ \\ &\sum_{n=1}^{2^M}[(1-2a_n^l)g_n \prod_{i=1,i\neq l}^{M}((1-a_n^i) + (2a_n^i-1)H(\phi_i))]\},\end{aligned} \tag{6.2}$$

where τ is the algorithmic evolution time, κ_{m_l} is the mean curvature of ϕ_l, and the function $g_n(\boldsymbol{x},t)$ is defined as follows:

$$g_n(\boldsymbol{x},t) = \begin{cases} \omega_n\xi_n(\boldsymbol{x},t;\widehat{\boldsymbol{p}}_n) & 1 \leq n \leq N, \\ K_{pen} & N+1 \leq n \leq 2^M. \end{cases}$$

Although the second term in (6.2) looks complicated, it is simply a sum of terms $\omega_n\xi_n$ and K_{pen} with suitable signs (depending whether $(\boldsymbol{x},t)$ is inside or outside of ϕ_i's). At each $(\boldsymbol{x},t)$ the evolution force thus combines a curvature term and a term whose sign and amplitude depend on a comparison of error terms ξ_i among each other and also against K_{pen}.

The first step of the overall algorithm is to find motion parameters based on an initial segmentation, such as the one computed using the motion detection algorithm of Chapter 3. Using the initial segmentation (single surface), we estimate the number of separate moving objects in the sequence. We use MATLAB Image Processing Toolbox's function `bwlabel` to estimate the number of connected object regions in each frame of the motion detection result. Then, we track these regions through time to create separate tunnels for each object. Using the method described in Section 4.2.2, we

calculate motion parameters for each of the moving objects and background.

For L moving objects, we need up to $3(L+1)$ different volumes (visible, exposed and to be occluded). We calculate the necessary number of level-set surfaces $M = \lceil \log_2 N \rceil$, where N is the number of volumes we are going to estimate in the segmentation algorithm ($N \leq 3(L+1)$). We proceed to create a sequence of labels, using the following rules:

- For every voxel inside one of the object tunnels, we verify whether the whole of its motion trajectory is inside the tunnel (in which case it is labeled as one of the object voxels) or is partially outside the tunnel (when it is labeled as an occluded or exposed object voxel).

- For every voxel outside all of the object tunnels (background voxel), we verify whether the whole of its motion trajectory is outside of all the object tunnels (in which case it is labeled as a background voxel) or is partially within some of the object tunnels (when it is labeled as an occluded or exposed background voxel).

Based on this sequence of labels, we initialize M level-set functions needed by the segmentation step. After this initialization, segmentation steps are interleaved with motion parameter estimation; the procedure is repeated until surfaces and motion parameters converge to a stable solution.

In each iteration of the segmentation step we calculate the forces F_i at zero level-set points of all surfaces, extend these forces using the fast marching algorithm by solving $\phi_i \cdot \nabla F_i = 0$ for F_i, $i = 1, ..., M$, and update the surfaces ϕ_i. The re-initialization of the surfaces using the fast marching algorithm by solving $\|\nabla \phi_i\|$=1 is performed after every 10 to 100 iterations (depending on the value of smoothness parameter λ: for values of $\lambda < 1$ we perform reinitialization every 100 iterations, otherwise we do it every 10 iteration) to keep surfaces as close as possible to a signed distance function.

6.3 Experimental results

We have tested the proposed algorithm on sequences with multiple moving objects obtained from the Universität Karlsruhe web site (Karlsruhe, 1997). We extracted a 176×256-pixel window from one of the sequences (Karl-Wilhelm-Straße), that we shall call *Traffic* and that contains two cars moving down a street (Fig. 6·1), captured by a static camera (no background motion). We first applied the multi-frame motion detection algorithm, with α=5 and λ=10. The parameter α is chosen based on the noise level in the background (since it serves as a sort of threshold), while the high value of parameter λ suppresses spurious, false noisy objects in the background and keeps the objects boundaries smooth. The algorithm converges after 4000 iterations and, as we can see from Fig. 6·1, both object shapes are very accurate. Fig. 6·2 shows object and background tunnels obtained in the experiment. Noticeable segmentation errors are present due to inherent problems with the motion detection method (the result is a union of object positions in consecutive frames). The noise in the background creates additional problems, while the high value of λ tends to straighten out the object tunnels, which leads to inclusion of parts of the background behind and in front of the cars into the object regions.

In order to alleviate some of the segmentation inaccuracies, we applied the multi-frame segmentation algorithm with occlusion modeling of the background. Since there are no object occlusions, we used 5 volumes: two objects, background, as well as occluded and uncovered background areas. In the motion estimation step we calculated motion parameters for the left car (left object, $\boldsymbol{p}_1$), right car (right object, $\boldsymbol{p}_2$) and background ($\bar{\boldsymbol{p}} = 0$, static background). We used three level-set functions, with assignments shown in Table 6.1 (three volumes were left unused). We used the following parameters: $\alpha_1 = \alpha_2 = 100$, $\lambda_1 = \lambda_2 = \lambda_3 = 5$, $\omega_1 = 1$, $\omega_2 = \omega_3 = 3$, $\omega_7 = 5$, $\omega_8 = 4$, $K_{pen} = 100$, and performed 400 iterations. Fig. 6·3 shows two frames from the sequence overlaid with the

Table 6.1: Segmentation volumes and associated energy terms for the multi-frame motion-compensated segmentation method in case of two objects and background occlusion modeling.

Volume $\mathcal{V}_n$	j_n	$(\boldsymbol{x}, t)$	Level-set func. at $(\boldsymbol{x}, t)$	Energy terms
$\mathcal{V}_1$ = background	(000)	outside $\vec{\varsigma}_1, \vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 < 0,\ \phi_2 < 0,\ \phi_3 < 0$	$\omega_1 \xi_{obj}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_2$ = occluded backgr.	(001)	inside $\vec{\varsigma}_1$, outside $\vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 < 0,\ \phi_3 < 0$	$\omega_2 \xi_{occ}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_3$ = exposed backgr.	(010)	inside $\vec{\varsigma}_2$, outside $\vec{\varsigma}_1, \vec{\varsigma}_3$	$\phi_1 < 0,\ \phi_2 > 0,\ \phi_3 < 0$	$\omega_3 \xi_{exp}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$
$\mathcal{V}_4$ = unused	(011)	inside $\vec{\varsigma}_1, \vec{\varsigma}_2$, outside $\vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 > 0,\ \phi_3 < 0$	K_{pen}
$\mathcal{V}_5$ = unused	(100)	inside $\vec{\varsigma}_3$, outside $\vec{\varsigma}_1, \vec{\varsigma}_2$	$\phi_1 < 0,\ \phi_2 < 0,\ \phi_3 > 0$	K_{pen}
$\mathcal{V}_6$ = unused	(101)	inside $\vec{\varsigma}_1, \vec{\varsigma}_3$, outside $\vec{\varsigma}_2$	$\phi_1 > 0,\ \phi_2 < 0,\ \phi_3 > 0$	K_{pen}
$\mathcal{V}_7$ = right object	(110)	inside $\vec{\varsigma}_2, \vec{\varsigma}_3$, outside $\vec{\varsigma}_1$	$\phi_1 < 0,\ \phi_2 > 0,\ \phi_3 > 0$	$\omega_7 \xi_{obj}(\boldsymbol{x}, t; \boldsymbol{p}_2)$
$\mathcal{V}_8$ = left object	(111)	inside $\vec{\varsigma}_1, \vec{\varsigma}_2, \vec{\varsigma}_3$	$\phi_1 > 0,\ \phi_2 > 0,\ \phi_3 > 0$	$\omega_8 \xi_{obj}(\boldsymbol{x}, t; \boldsymbol{p}_1)$

final level-set contours, and the corresponding label fields. Four volumes corresponding to four segmentation regions are shown in Fig. 6·4. All background regions (visible, to-be-occluded, and exposed) are accurately estimated. The moving cars are estimated somewhat better compared to the detection result (boundaries are tighter around the objects), but some inaccuracies are still present. Apart from noise and brightness variation over time, another problem is the very similar gray level of the right car (especially in the second part of the sequence) and of the pavement behind it. Also, there is little texture in the moving objects and background. Still, results are very good given all the difficulties this sequence presents. We clearly demonstrated how our method can be used for segmentation of complex sequences with multiple moving objects, shot in real-life conditions.

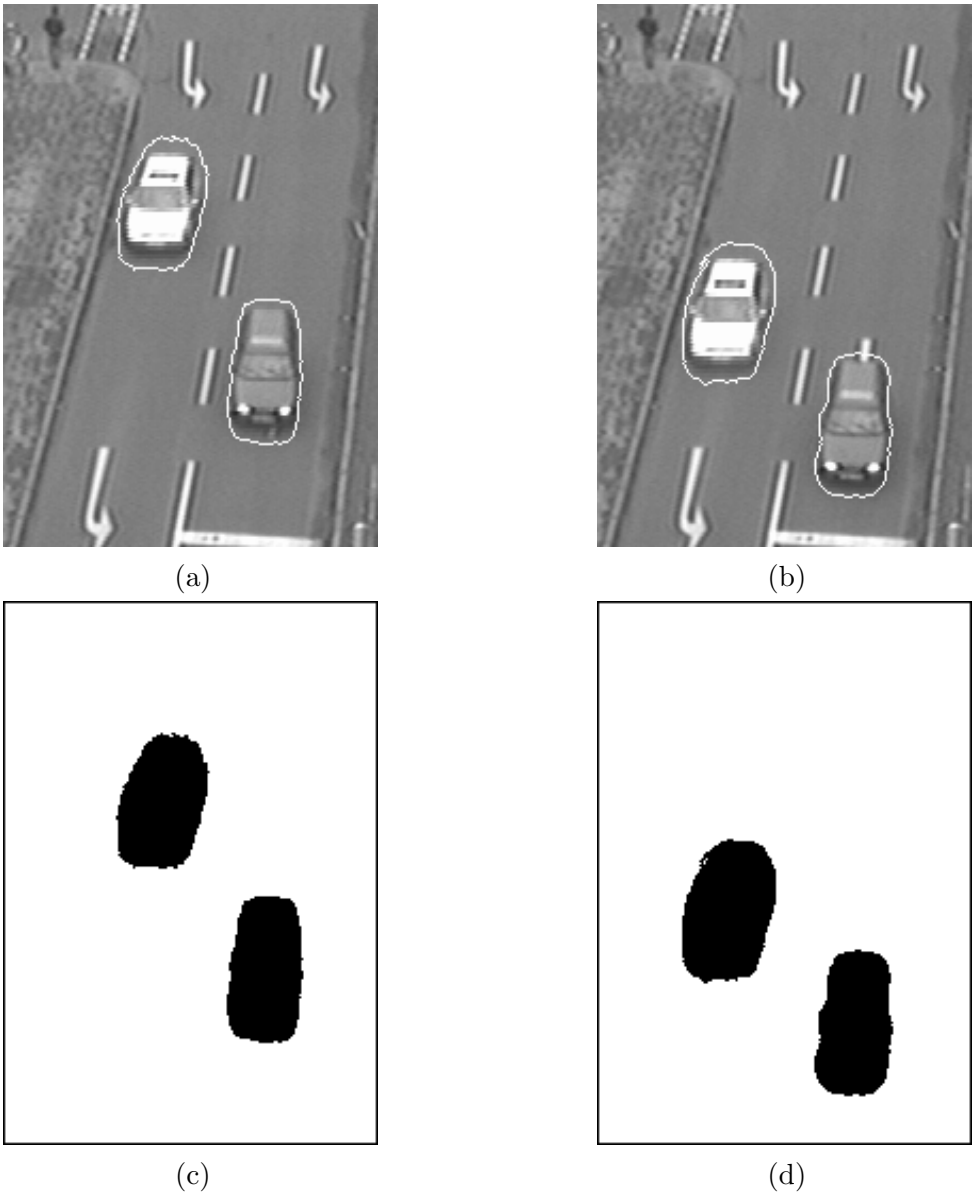
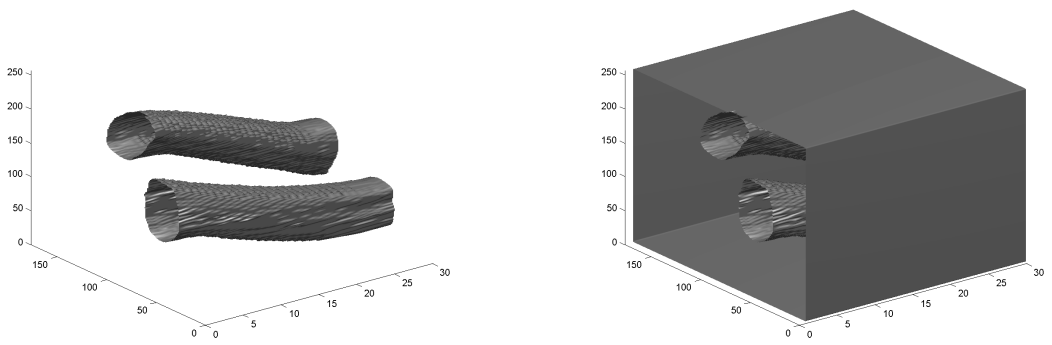
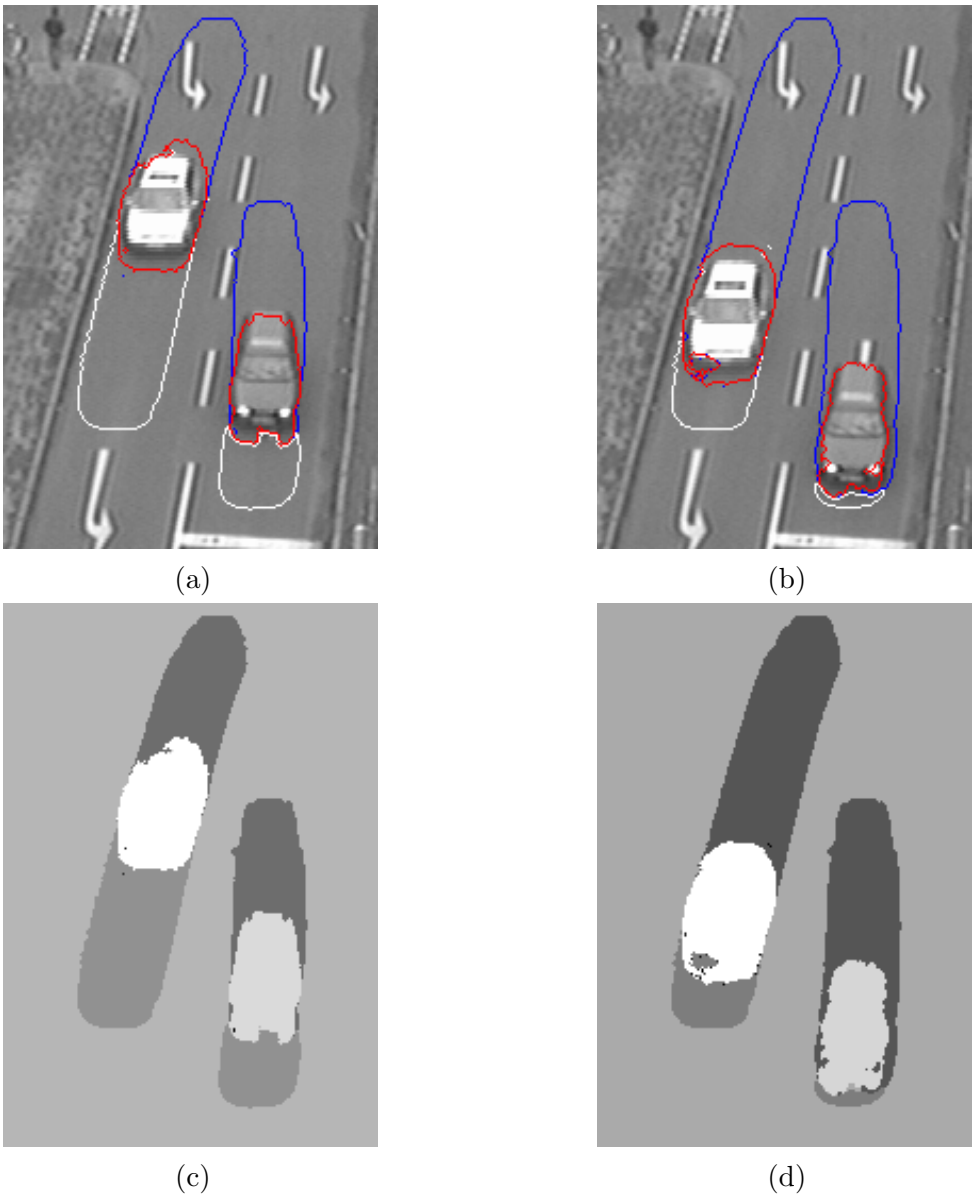


Figure 6·1: Results for the M-frame MD algorithm, applied to *Traffic* image sequence: frames (a) #15 and (b) #25 from the sequence overlaid with final boundaries, and (c–d) corresponding label fields (white – background and black – object).



(c) (d)

Figure 6·2: Volumes corresponding to results from Fig. 6·1: (a) object tunnels and (b) background tunnel.



(a) (b)

(c) (d)

Figure 6·3: Results for the M-frame MCS-B algorithm, applied to *Traffic* image sequence: frames (a) #15 and (b) #25 from the sequence overlaid with final boundaries, and (c–d) corresponding label fields (white – left object, light gray to dark gray: right object, background, to-be-occluded background, and exposed background).

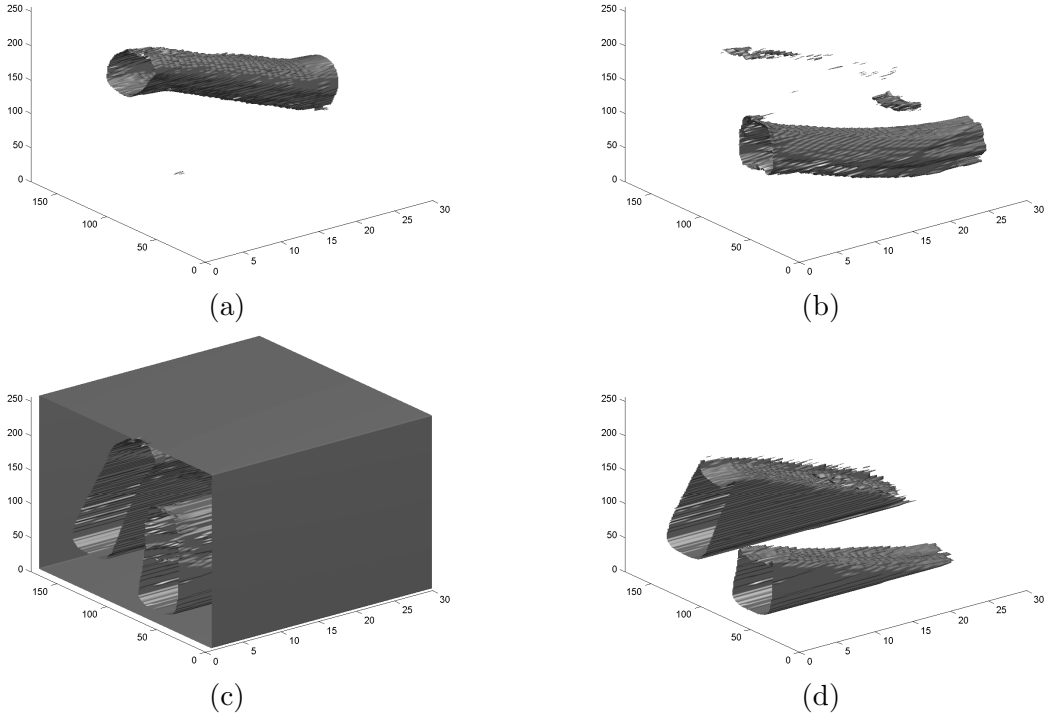


Figure 6·4: Volumes corresponding to results in Fig. 5·3: (a) left-car tunnel, (b) right-car tunnel, (c) background tunnel, and (d) background occlusion volume.

Chapter 7

Motion detection for active cameras

In Chapter 3, we proposed a video segmentation method based on motion detection, which has one serious limitation – the model we used requires static background throughout the sequence (we are minimizing frame difference in the background volume). If the background is not static, for example due to camera motion or zoom, our algorithm cannot differentiate between the moving object and the background. Since we use that method to initialize our more advanced methods (Chapters 4 – 6), it is very important to lift that constraint. To that end, we propose to account for camera motion/zoom in our motion detection formulation, and to estimate it simultaneously with the evolution of the level-set surface that encloses moving objects. From now on, when we say camera motion we will also mean zoom-in and zoom-out.

The simplest approach to camera motion estimation is to treat that motion as global motion in a sequence and use any of the global motion estimation algorithms to calculate it (an overview can be found, for example, in (Wang et al., 2002)). However, that approach would work accurately only if moving objects in the sequence are very small. Otherwise, apparent motion of the objects would corrupt the camera motion estimate. In order to avoid this, we propose to perform motion estimation only on the current background region, which is defined by the position of the level-set segmentation surface that is evolved simultaneously with motion estimation.

The next issue is the choice of models for describing camera motion between frames of a sequence. The simplest solution would be to assume a translational motion in

the background, but that is a very coarse approximation which would account only for horizontal and vertical camera translation parallel to the scene, i.e., track and boom, respectively. If we want to model pan, tilt, roll (i.e., turning around the vertical axis, turning around the horizontal axis, and rotation around the optical axis, respectively), and zoom (for definitions of these terms and examples of camera motion see (Wang et al., 2002)) of the camera, we need to use the affine motion model. It describes apparent motion in the background induced by any small camera motion, under orthographic projection, quite accurately when the background scene is planar or very far away from the camera.

7.1 Method of Feghali and Mitiche

An interesting approach to image sequence segmentation in the presence of camera motion was recently proposed by Feghali and Mitiche (Feghali and Mitiche, 2004). They proposed to use the normal component of optical flow velocity as a measure of motion activity:

$$\omega_\perp = \begin{cases} \frac{-I_t}{||\nabla I||}, & \text{for} \quad ||\nabla I|| \neq 0 \\ 0, & \text{for} \quad ||\nabla I|| = 0 \end{cases},$$

where ∇I and I_t are the spatial gradient and temporal derivative of image I, respectively. They define $\omega_\perp^* = \omega_\perp - \omega_{c\perp}$, where $\omega_{c\perp}$ is a component along $\omega_\perp$ of a local velocity computed from the global motion estimate (camera motion) and is a function of $\vec{\theta}$, the parameters of velocity due to camera motion. Subsequently, they seek such a partition of the image sequence domain, $\mathcal{P}_S = R_S \cup R_S^c$, that points inside the object ($\boldsymbol{x} \in R_S$, S is a closed segmentation surface) have significant normal motion ($|\omega_\perp^*| \gg 0$), while points in the background ($\boldsymbol{x} \in R_S^c$) have negligible normal motion ($\omega_\perp^* \approx 0$). Using a suitable observation model and standard regularization term, they convert MAP estimation of

$(S, \vec{\theta})$ into minimization of the following energy functional:

$$E(S, \vec{\theta}) = \alpha \int\limits_{R_S} e^{-(\omega_\perp^*(\vec{\theta}))^2} d\rho + \beta \int\limits_{R_S^c} (\omega_\perp^*(\vec{\theta}))^2 d\rho + \lambda \int\limits_{S} d\sigma.$$

Since they use the translational motion model, and additionally assume that camera motion is constant during the span of an image sequence, only two motion parameters are needed, $\vec{\theta} = (a, b)$. Interleaved minimization is performed using gradient descent for motion parameters $\vec{\theta}$ and level-set PDE descent for segmentation surface S. A block diagram of the algorithm is presented in Fig. 7·1 (a).

7.2 Extension of the method of Feghali and Mitiche

Inspired by the approach described in the previous section, we propose to improve our motion detection (Chapter 3) by incorporating background (camera) motion. We start from the general volume-competition energy minimization:

$$\min_{\vec{\varsigma}, \boldsymbol{p}, \bar{\boldsymbol{p}}} \alpha \iiint\limits_{\mathcal{V}} \xi(\boldsymbol{x}, t; \boldsymbol{p}) d\boldsymbol{x} dt + \iiint\limits_{\bar{\mathcal{V}}} \xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) d\boldsymbol{x} dt + \lambda \iint\limits_{\mathcal{S}} d\vec{\varsigma}, \tag{7.1}$$

Since our goal is only the detection of moving regions in an image sequence (not a segmentation into separately-moving objects), we simplify this formulation, similarly to Section 3.2, by assuming a fixed penalty α within the object ($\xi(\boldsymbol{x}, t; \boldsymbol{p}) = 1$). However, we are leaving the background parametric motion $\bar{\boldsymbol{p}}$ as an unknown in our minimization. We propose motion-compensated frame difference as the measure of background intensity variation:

$$\xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}}) = |I(\boldsymbol{x}, t) - I(\boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x}, t; \bar{\boldsymbol{p}}), t - 1)|,$$

where $\boldsymbol{d}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$ is a displacement vector, calculated from motion parameters $\bar{\boldsymbol{p}}$, that describes background motion between frames at times $t - 1$ and t. In order to reach a

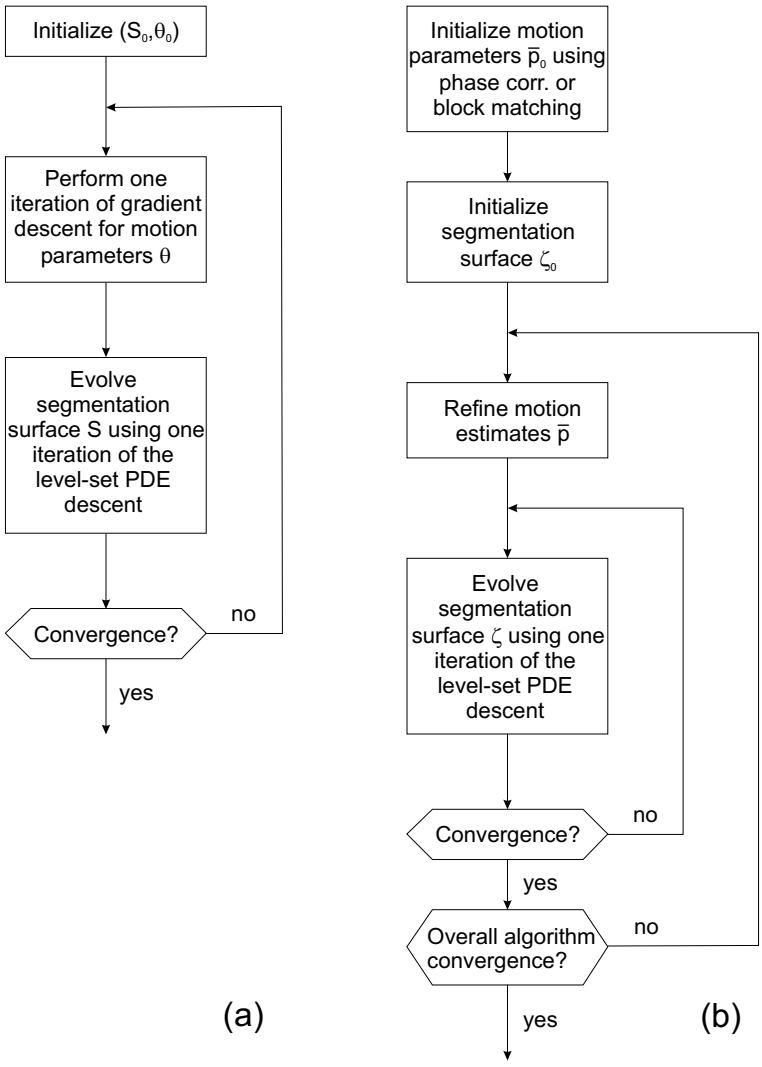


Figure 7·1: Block diagram of (a) Feghali and Mitiche method and (b) proposed new method.

minimum in (7.1), the surface $\vec{\varsigma}$ must partition the domain $\Omega \times \mathcal{T}$ so that points $(\boldsymbol{x}, t)$ with small motion-compensated frame difference are assigned to the outside ($\bar{\mathcal{V}}$), and those with large difference – to the inside ($\mathcal{V}$). The balance between such assignments is controlled by α. In order to increase robustness, we replace the absolute value as the measure of background intensity variation in $\xi(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$ with a robust M-estimator (Black, 1992) ρ (more on this in the next section). Then, the minimization (7.1) reduces

to:

$$\min_{\vec{\varsigma},\bar{\boldsymbol{p}}} \iiint_{\Omega\times\mathcal{T}} h(I_t)d\boldsymbol{x}dt + \lambda \iint_{\mathcal{S}} d\vec{\varsigma}, \tag{7.2}$$

$$h(I_t) = \begin{cases} \alpha & \text{if } (\boldsymbol{x},t) \in \mathcal{V}, \\ \rho(I(\boldsymbol{x},t) - I(\boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x},t;\bar{\boldsymbol{p}}),t-1)) & \text{if } (\boldsymbol{x},t) \in \bar{\mathcal{V}}. \end{cases}$$

While Feghali and Mitiche (Feghali and Mitiche, 2004) use a constant translational model for camera motion throughout the sequence, we will use an affine model, with parameters of motion model changing from frame to frame, a closer approximation to real-life scenarios.

7.3 Solution method

In order to carry out minimization (7.2), we decompose the problem into two interleaved minimizations: estimation of motion parameters given segmentation surfaces, and estimation of segmentation surfaces with fixed motion parameters.

7.3.1 Estimation of motion parameters

When the segmentation surface $\vec{\varsigma}$ is fixed ($\vec{\varsigma} = \widehat{\vec{\varsigma}}$), i.e., volume $\bar{\mathcal{V}}$ is defined ($\bar{\mathcal{V}} = \widehat{\bar{\mathcal{V}}}$), minimization (7.2) reduces to:

$$\min_{\bar{\boldsymbol{p}}} \iiint_{\widehat{\bar{\mathcal{V}}}} \rho(I(\boldsymbol{x},t) - I(\boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x},t;\bar{\boldsymbol{p}}),t-1))d\boldsymbol{x}dt. \tag{7.3}$$

Since the segmentation surface $\widehat{\vec{\varsigma}}$ may be far away from the true, underlying object boundary, some points inside $\widehat{\bar{\mathcal{V}}}$ may not belong to the background and cannot be explained by the background motion model. To deal with these outliers, we replaced the absolute norm with a robust M-estimator (Black, 1992) ρ (e.g., Lorentzian, Geman-McClure, etc.) in the energy formulation (7.2).

When a constant-translation motion model is used, like in (Feghali and Mitiche, 2004), there are only three variables to be estimated: two motion parameters and a segmentation surface. In that case, it is easy to perform joint minimization (7.2) with respect to all three variables, and to do simultaneous evolution of the segmentation surface and estimation of the motion parameters by interleaving gradient descent for motion parameters $\bar{\boldsymbol{p}}$ and level-set PDE descent for segmentation surface $\vec{\varsigma}$. However, when more complex, affine motion model is used, with parameters changing from frame to frame, another approach may be more suitable: estimate parameters $\bar{\boldsymbol{p}}$ for each frame pair by minimizing the energy in equation (7.3), and then use estimated parameters to perform several iterations of surface evolution. Since between PDE evolution iterations little changes, it does not make sense to re-estimate motion parameters until changes in the segmentation surface $\vec{\varsigma}$ accumulate after a number of iterations.

Instead of minimizing (7.3) directly, we will decompose it into separate minimizations for each frame pair (I_{t-1}, I_t). After discretization, we are solving the following minimization for each frame t:

$$\min_{\bar{\boldsymbol{p}}_t} \sum_{\boldsymbol{x}_i \in \bar{\mathcal{R}}_t} \rho(I(\boldsymbol{x}_i, t) - I(\boldsymbol{x}_i - \boldsymbol{d}(\boldsymbol{x}_i, t; \bar{\boldsymbol{p}}_t), t-1)), \tag{7.4}$$

where $\bar{\boldsymbol{p}}_t$ is the set of affine motion parameters at time t, while $\bar{\mathcal{R}}_t$ is the background region in frame I_t (cross-section of volume $\bar{\mathcal{V}}$ at time t).

The first step of the overall, multi-frame motion detection with background motion compensation (M-frame MD-BMC) algorithm (Fig. 7·1 (b) shows block diagram of the algorithm) is to initially estimate global motion without any information about object location. In other words, since we do not know what is the background region $\bar{\mathcal{R}}_t$, the minimization (7.4) is performed over the whole frame, I_t. To find a solution to (7.4), we use MATLAB Optimization Toolbox's function `fminunc`, which applies BFGS quasi-Newton method with a mixed quadratic and cubic line search procedure. In order to

avoid local minima and to handle large background motions, we do not initialize the optimization procedure with zero motion parameters. Instead, we apply simple phase correlation (Kuglin and Hines, 1975) or block matching (Wang et al., 2002) algorithm to the image sequence to obtain initial values for the translation parameters.

7.3.2 Segmentation surface estimation

For the surface evolution step of our M-frame MD-BMC algorithm we are solving (7.2) with the background motion $\bar{\boldsymbol{p}}$ parameters fixed ($\bar{\boldsymbol{p}} = \widehat{\bar{\boldsymbol{p}}}$, estimated in the motion estimation phase). That leads to the following minimization:

$$\min_{\vec{\varsigma}} \iiint_{\Omega\times\mathcal{I}} h(I_t) d\boldsymbol{x} dt + \lambda \iint_{\mathcal{S}} d\vec{\varsigma}, \tag{7.5}$$

$$h(I_t) = \begin{cases} \alpha & \text{if } (\boldsymbol{x}, t) \in \mathcal{V}, \\ \rho(I(\boldsymbol{x}, t) - I(\boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x}, t; \widehat{\bar{\boldsymbol{p}}}), t-1)) & \text{if } (\boldsymbol{x}, t) \in \bar{\mathcal{V}}. \end{cases}$$

This minimization is very similar to the one in (3.8) in Section 3.2 and is solved using the level-set methodology presented there.

Once the surface evolution step of the algorithm is performed and initial segmentation surface is estimated, motion estimation in the second pass of the overall algorithm is performed on the background region, with previous pass motion estimates used as initial parameters for this pass. It is followed by another segmentation step in which newly-calculated motion parameters are used. This procedure is repeated until no further improvement in the estimated motion can be obtained.

7.4 Experimental results

As before, we will first test our M-frame MD-BMC method on synthetic sequences and compare it to the M-frame MD method (described in Chapter 3) applied to the same sequences. Additionally, we will compare the detection results when translational and

affine motion models are used to describe background motion. Next, we will apply our method to camera-acquired test sequences. Finally, we will use both the M-frame MD-BMC and M-frame MD methods' results to initialize motion-compensated segmentation with background occlusion modeling (M-frame MCS-B method, introduced in Chapter 4) and compare the resulting segmentations.

7.4.1 Results for synthetic sequences

We will use two test sequences: natural-texture, synthetic-motion, sequence *Bean Small* (176×144 pixels) in which both a bean-shaped object and background undergo accelerated zoom and rotation, and a similar sequence *Bean Normal* with the same object motion but larger zoom and rotation in the background.

We first investigate what would happen should we apply our M-frame MD algorithm, with no compensation of background motion, to these sequences. For both sequences we used the same parameters: $\alpha = 20$ and $\lambda = 1$; results after 800 iterations are shown in Fig. 7·2. A large value of parameter α is chosen to avoid object estimate covering the whole sequence domain (given that all the pixels in the sequence are moving). It is obvious that the algorithm is nowhere close to estimating true object position, and instead finds false object regions in the background wherever there is some texture (low-texture regions are not affected by the background motion). This was to be expected and is a motivation to introduce compensation of the background motion in the algorithm.

As we mentioned before, we decided to use affine instead of translational motion model to describe the background motion. In order to test whether this more complicated motion model brings any improvement to the detection results, we run a set of experiments on another synthetic sequence, *Bean Big*. This sequence is similar to the other two, but has much larger affine component of background motion (object motion is the same). We run several M-frame MD-BMC experiments for various values of pa-

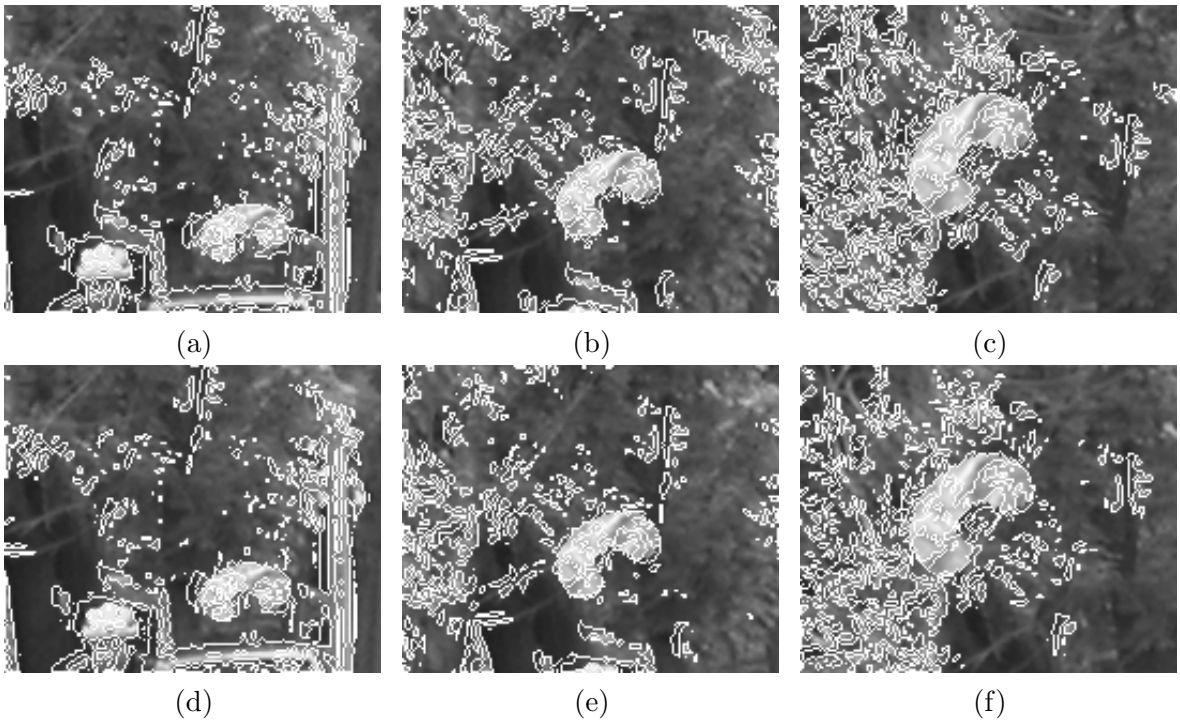


Figure 7·2: Results for the M-frame MD method, applied to two sequences with moving background: final contours for frames (a) #5, (b) #15, and (c) #25 from the synthetic image sequence *Bean Small*, and frames (d) #5, (e) #15 and (f) #25, from the synthetic image sequence *Bean Normal*.

rameter α, using translational or affine motion models in the motion estimation phase. Fig. 7·3 shows object segmentation error (in pixels per frame) for these experiments. It is obvious that affine model outperforms translational model by a wide margin for values of $\alpha < 0.9$. For $\alpha \geq 0.9$ the object cannot be detected no matter which motion model is used so the segmentation error is close to the size of the object. The object is accurately detected only if affine model is used with α around 0.75–0.8. Overall, we can conclude that the introduction of affine model is justified especially when background motion is more complicated than simple pan and tilt.

Similarly to our other detection and segmentation algorithms, M-frame MD-BMC method is semi-supervised: user is required to choose values of parameters α, σ, and

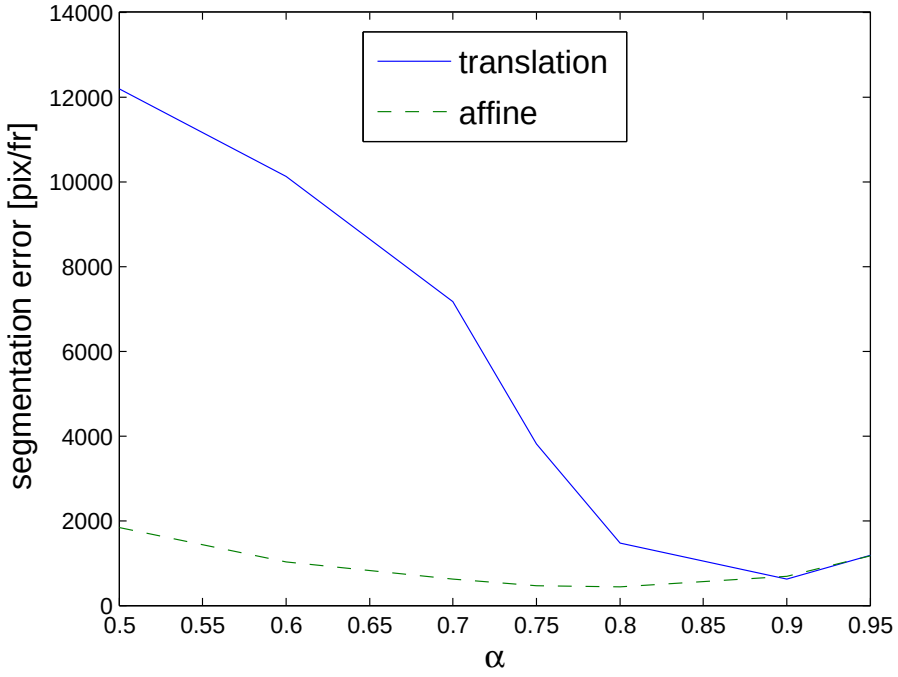


Figure 7·3: Segmentation error ϵ_m for experiments with different values of parameter α, on sequence *Bean Big*, with translational or affine motion model used for background motion.

λ. We again run a batch of experiments with different parameter values for each of the sequences tested. We show the best results obtained, either based on ground-truth data (for synthetic sequences) or visual inspection (for natural sequences).

We apply the M-frame MD-BMC method to two synthetic sequences, *Bean Small* and *Bean Normal*. Fig. 7·4 shows results for the *Bean Small* sequence after 500 iterations, obtained using the following parameters: $\alpha = 0.25$ and $\lambda = 0.1$ (we also used the Geman-McClure robust estimator (Black, 1992), with $\sigma = 15$). Given that α serves as a threshold to which we compare the output of the robust estimator, the range of values for α is restricted to $[0, 1]$. The value of α also depends on the robust estimator parameter σ (higher σ means lower α will be necessary). Fig. 7·4(a–c) shows three original frames with estimated boundaries superposed, while Fig. 7·4(d) shows the corresponding object tunnel. The results show a good recovery of object shape and tracking between frames.

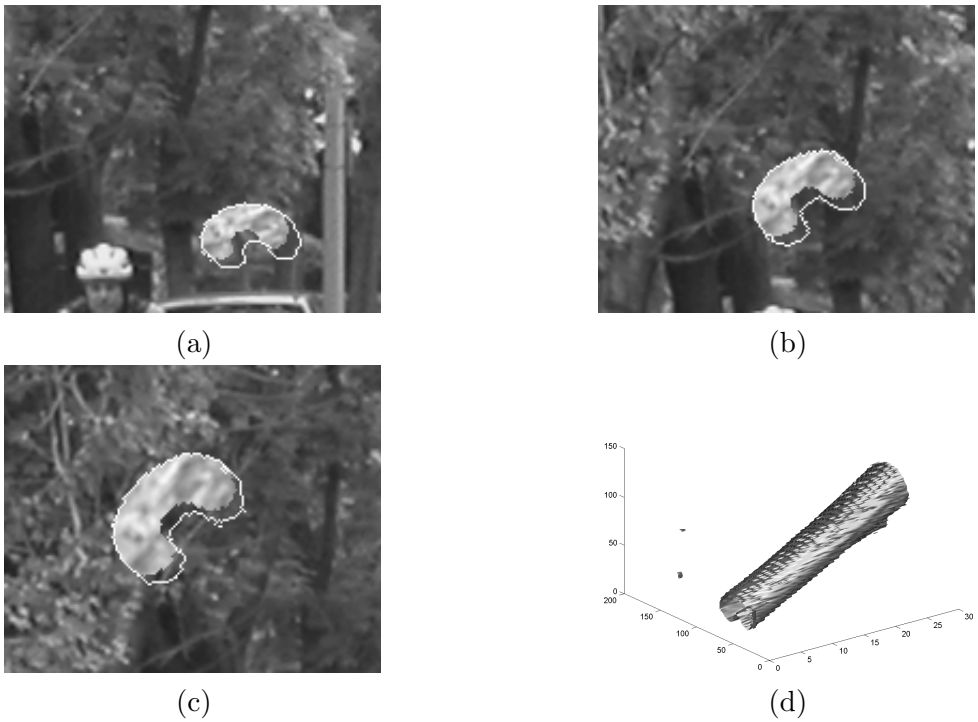


Figure 7·4: Results for the M-frame MD-BMC algorithm, applied to synthetic image sequence *Bean Small*: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

However, noticeable segmentation errors are present in areas lagging behind the moving object. When compared to the results obtained for the sequence with the same object motion but with static background (*Bean*) in Fig. 3·4, these erroneous areas are much larger. That is due to the moving background and the fact that the object encompasses all the points not well-explained by the background motion model. In addition to the union of object positions in consecutive frames (result for the case of static background), the object includes parts of background which were occluded (or will be exposed) in the other frame. The larger the background motion, the larger these areas. Using the ground-truth segmentation, we calculated object segmentation error to be 391.1 pixels per frame (or 32.99%). Fig. 7·5 shows similar results obtained for *Bean Normal* sequence after 200 iterations. We used the following parameters: $\alpha = 0.75$, $\lambda = 5$, and $\sigma = 5$ (we

used the Geman-McClure robust estimator in all experiments). The object segmentation error is 383.6 pixels per frame (32.36%). This example shows that our method works as well for larger background motions, however to further reduce the segmentation error it needs to explicitly account for occlusions.

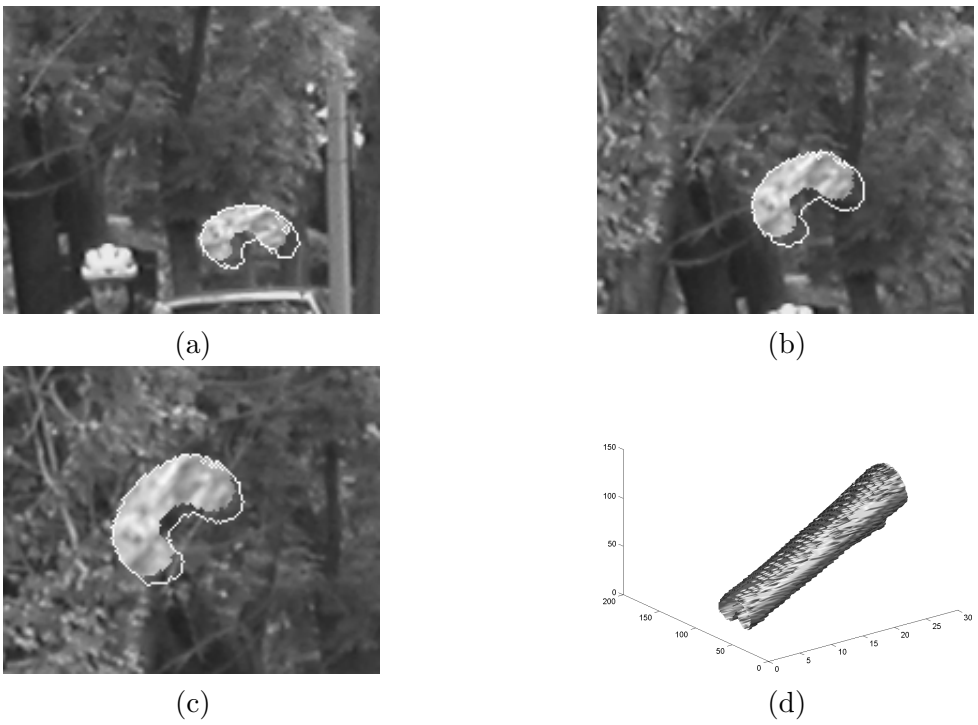


Figure 7·5: Results for the M-frame MD-BMC algorithm, applied to synthetic image sequence *Bean Normal*: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

7.4.2 Results for natural sequences

After verifying our method on the synthetic sequences, we proceed to test it on a couple of natural sequences. Fig. 7·6 presents results for standard test video sequence *Foreman*. We applied our method to 30 frames of the sequence, using $\alpha = 0.25$, $\lambda = 0.2$, and $\sigma = 5$, with the algorithm converging after 1000 iterations. Given that data in the natural image sequence, like *Foreman*, adheres less to affine motion models for the object and

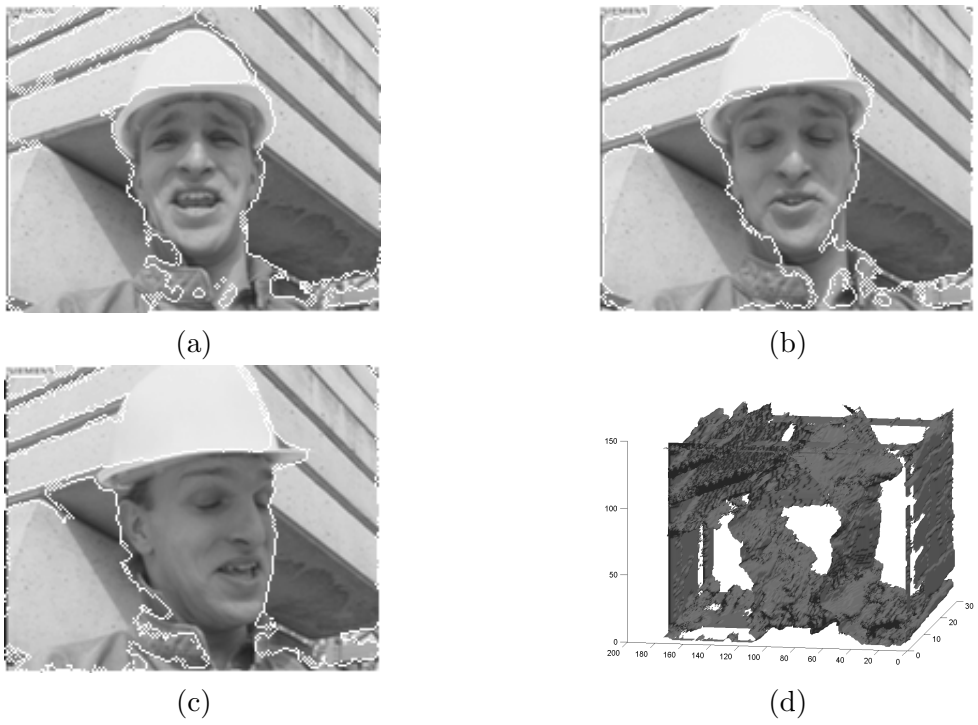


Figure 7·6: Results for the M-frame MD-BMC algorithm, applied to the *Foreman* image sequence: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

background, we increased the influence of the prior term by using a slightly larger λ. As can be seen in Fig. 7·6, worker's face is well estimated. However, his helmet, which is a very uniform region and therefore satisfies any motion model, is included in the background region, with only its edge being labeled as an object. Additionally, strong edges in the background on the left side of the frames are falsely included in the object region. A possible reason for this is that the affine motion model explains well the motion of planar surfaces in the scene. However, we have two distinctive planes in the background of this sequence, and only one can be explained with one set of affine parameters. This leads to erroneous regions in the other plane.

Results for another standard test sequence, *Stefan*, are shown in Fig. 7·7. The algorithm was applied to 30 frames of the sequence, and converged after 500 iterations.

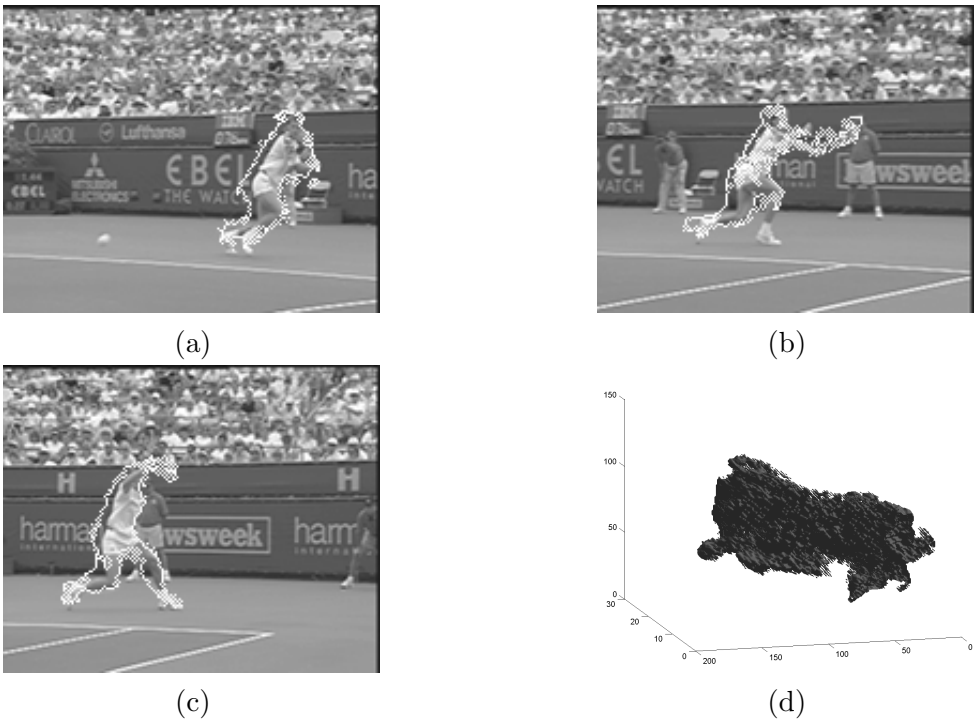


Figure 7·7: Results for the M-frame MD-BMC algorithm, applied to the *Stefan* image sequence: frames (a) #5, (b) #15, and (c) #25 from the sequence overlaid with final boundaries, and (d) the corresponding object tunnel.

The following parameters were used: $\alpha = 0.5$, $\lambda = 0.2$, and $\sigma = 10$, and were chosen following the same logic as for the previous sequences. The object shape is recovered very well, although some of its parts (legs), with similar texture to the background, are missing in some frames. The object is also tracked well across the sequence, especially given the fast motion both of the object and the background. A product of that large motion is an erroneous area lagging behind the object, but that is inherent to the model we use.

7.4.3 Results for segmentation with background occlusion detection

We use motion detection methods (with or without background motion compensation) to initialize our more advanced segmentation methods. In order to check how much

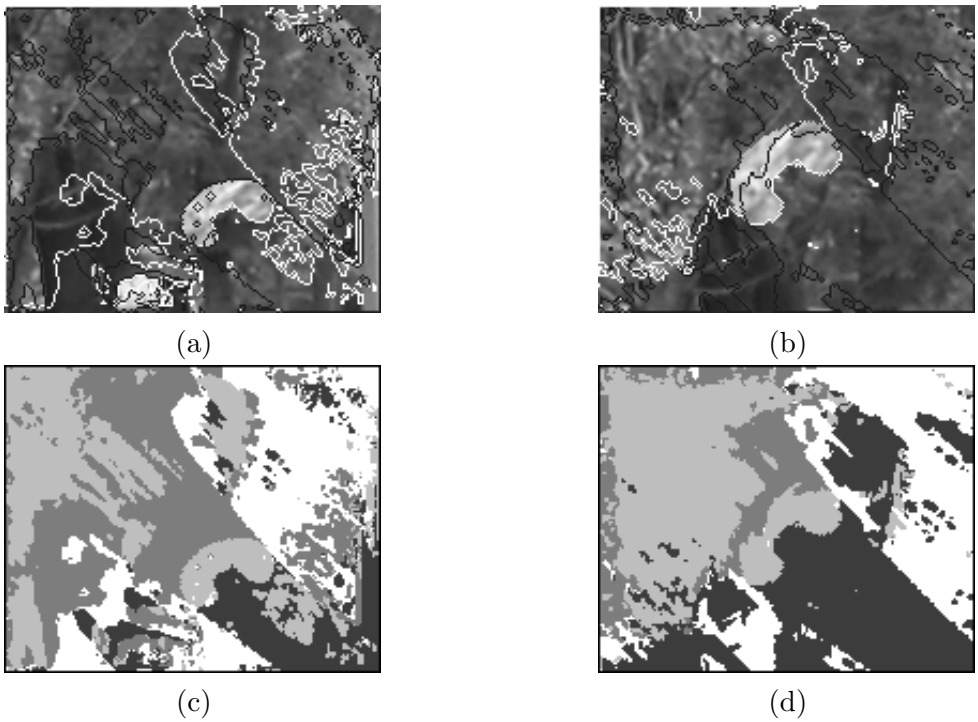


Figure 7·8: Results for the M-frame MCS-B algorithm initialized with the M-frame MD result (Fig. 7·2 (a–c)), applied to the synthetic image sequence *Bean Small*: frames (a) #10 and (b) #20 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

the final segmentation result depends on the proper initialization, we will first perform segmentation based on the detection result without background motion compensation. We will use the detection results from Fig. 7·2 to initialize our M-frame MCS-B method. Segmentation results for two synthetic sequences, *Bean Small* and *Bean Normal*, are shown in Figs. 7·8– 7·11. It is obvious that the results are very poor: Fig. 7·8 and Fig. 7·10 show large false object and occlusion/exposed areas which make tunnels in Fig. 7·9 and Fig. 7·11 very far from the underlying true segmentation volumes. This is quite expected since a poor initial detection results lead to erroneous object and background motion estimates, which in turn produce these poor segmentation results.

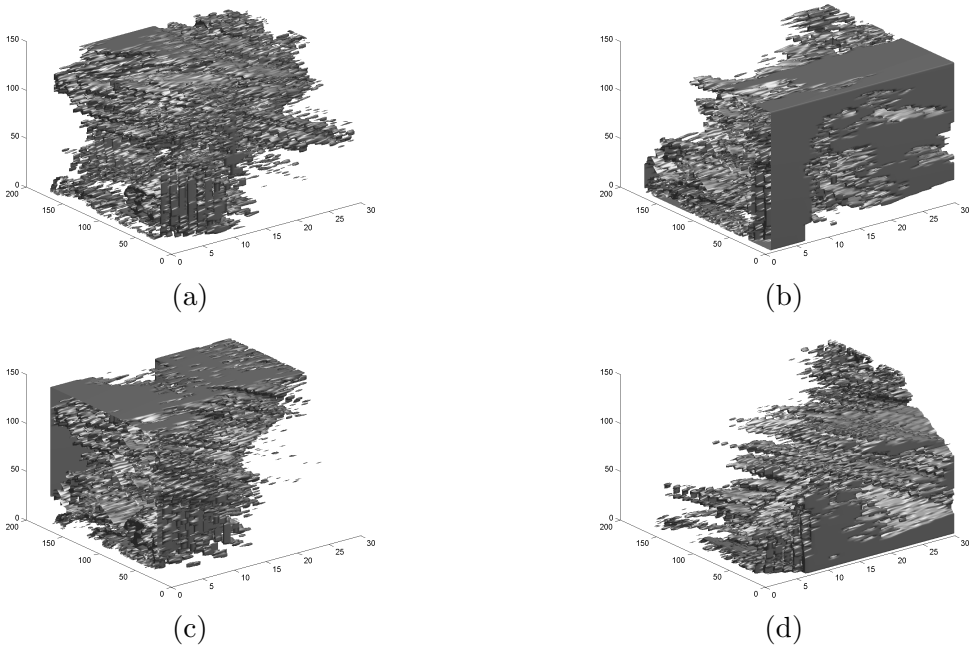


Figure 7·9: Volumes corresponding to results from Fig. 7·8: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

To alleviate these problems, we use motion detection with background motion compensation results (Fig. 7·4 and Fig. 7·5) to initialize the M-frame MCS-B algorithm. The first step is to calculate motion of the object and background based on the detection result obtained using the M-frame MD-BMC method. Next, we perform multiphase level-set evolution (4.6) (segmentation step) with the following parameters: $\omega_1 = 1$, $\omega_2 = \omega_3 = 5$, $\alpha_1 = \alpha_2 = 1000$, and $\lambda_1 = \lambda_2 = 0.8$. Parameters are the same for both sequences. They are chosen to suppress false occluded and exposed background regions in the result. The algorithm converges after 100 iterations and results for both sequences are shown in Figs. 7·12–7·15. Fig. 7·12 and Fig. 7·14 show two frames from the original sequence together with the final level-set contours, and the corresponding frames from the sequence of estimated labels. Volumes corresponding to the four segmentation regions are shown in Fig. 7·13 and Fig. 7·15 for the sequence *Bean Small* and *Bean Normal*,

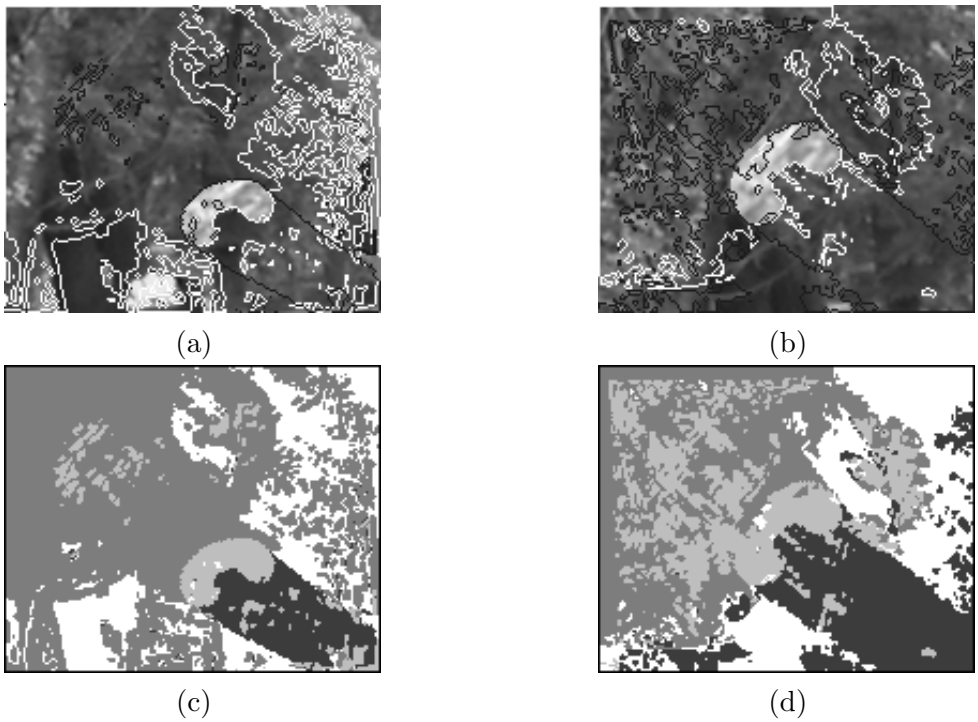


Figure 7·10: Results for the M-frame MCS-B algorithm initialized with the M-frame MD result (Fig. 7·2 (d–f)), applied to the synthetic image sequence *Bean Normal*: frames (a) #10 and (b) #20 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

respectively. This result was obtained with a single segmentation step initialized using the M-frame MD-BMC result; no additional motion estimation/segmentation steps were necessary. Object and background tunnels are very accurate, while occlusion and exposed volumes, although contain some spurious voxels at the periphery, still represent very well occlusion and uncovering effects in the background. Overall, we retained the accuracy of the segmentation results reported in Chapter 4 while expanding the set of sequences for which our segmentation method works from static ones to those with a moving background.

Since we know the ground-truth data, we can evaluate the segmentation accuracy

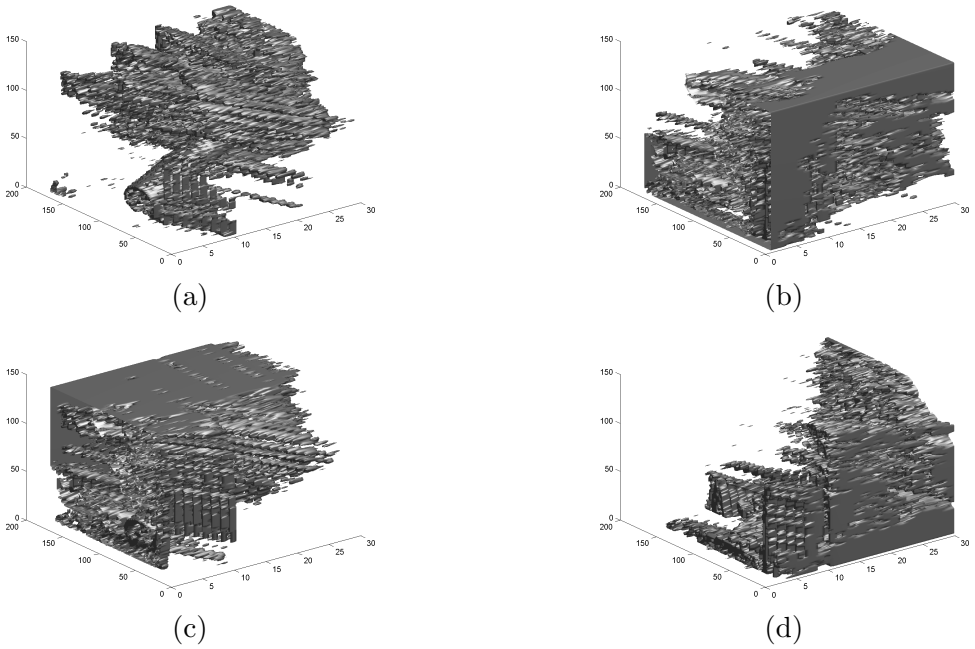


Figure 7·11: Volumes corresponding to results from Fig. 7·10: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

objectively, and compare solutions for different methods. In Table 7.1 we compare the M-frame MCS-B segmentation method with the simpler M-frame MD-BMC method. It is obvious that the error drops significantly when object motion compensation and explicit modeling of background occlusion volumes is introduced in the segmentation algorithm. That solves the largest problem inherent to the detection algorithm, i.e., the object tunnel is no longer a union of object positions and background occlusion regions in consecutive frames. The remaining small error is due to imperfect object and background motion estimates.

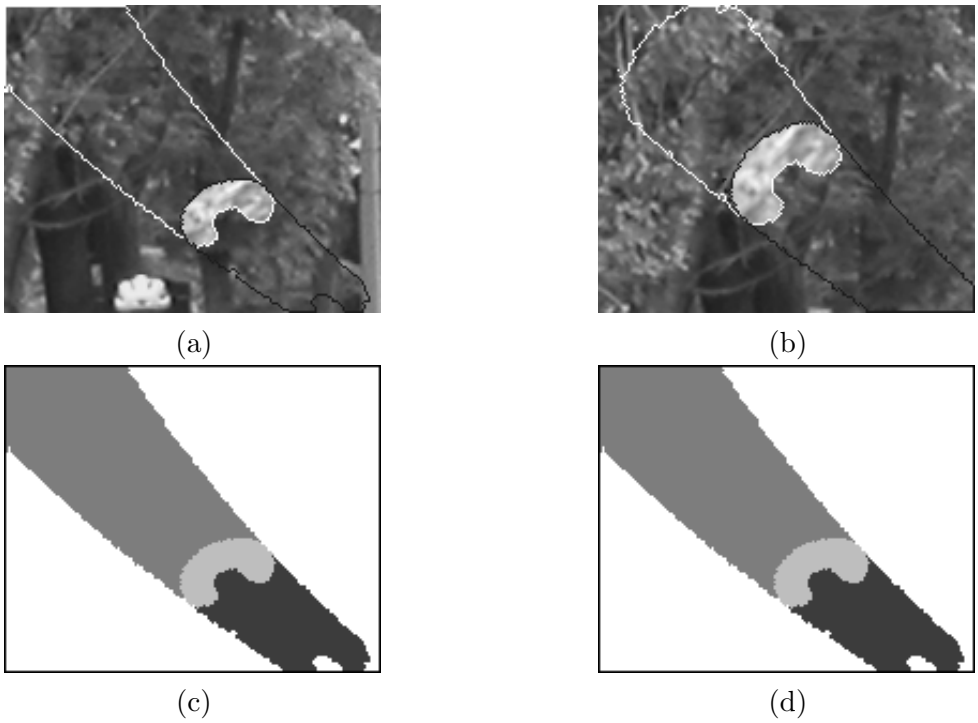


Figure 7·12: Results for the M-frame MCS-B algorithm initialized with the M-frame MD-BMC result (Fig. 7·4), applied to the synthetic image sequence *Bean Small*: frames (a) #10 and (b) #20 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).

Table 7.1: Object segmentation error ϵ_s (pixels per frame) for synthetic test sequences, for motion detection and motion compensated segmentation algorithms.

Sequence	M-frame MD-BMC	M-frame MCS-B
Bean Small	391.1 (33%)	31.7 (2.7%)
Bean Normal	383.6 (32.4%)	41.1 (3.5%)

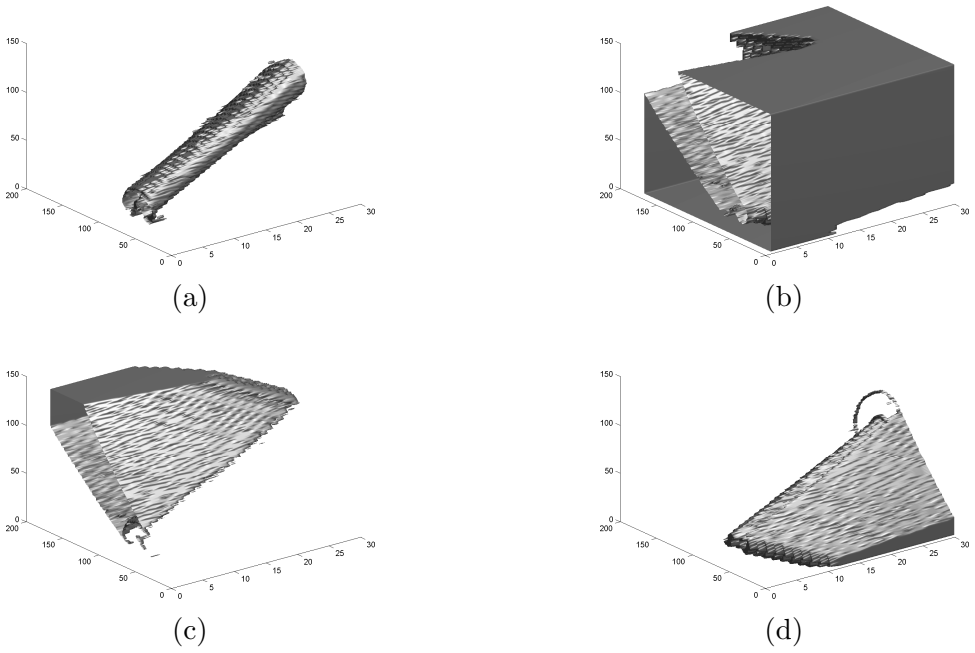


Figure 7·13: Volumes corresponding to results from Fig. 7·12: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

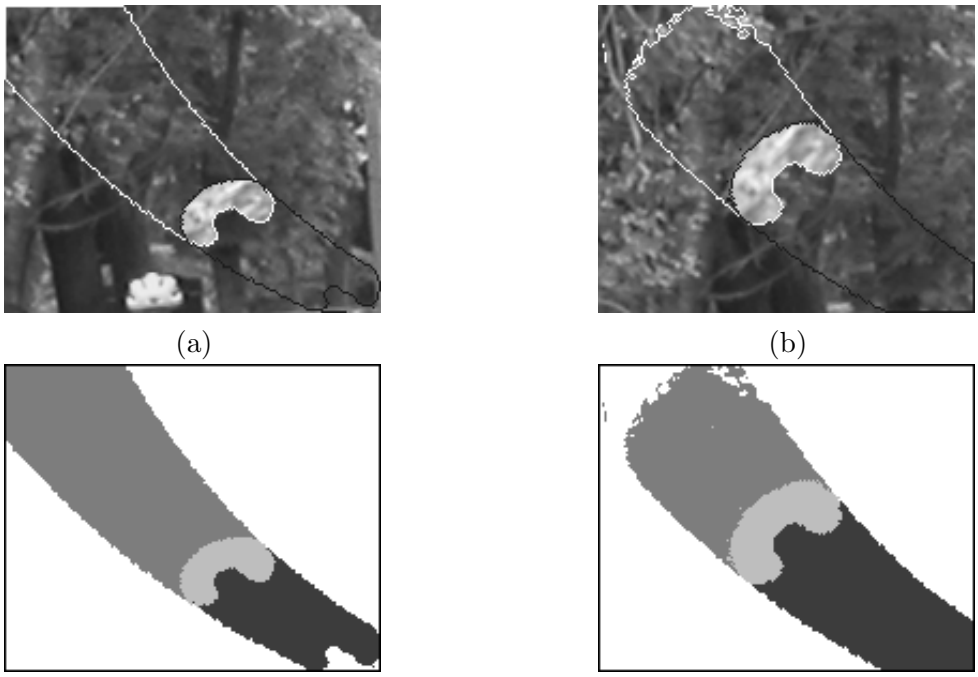
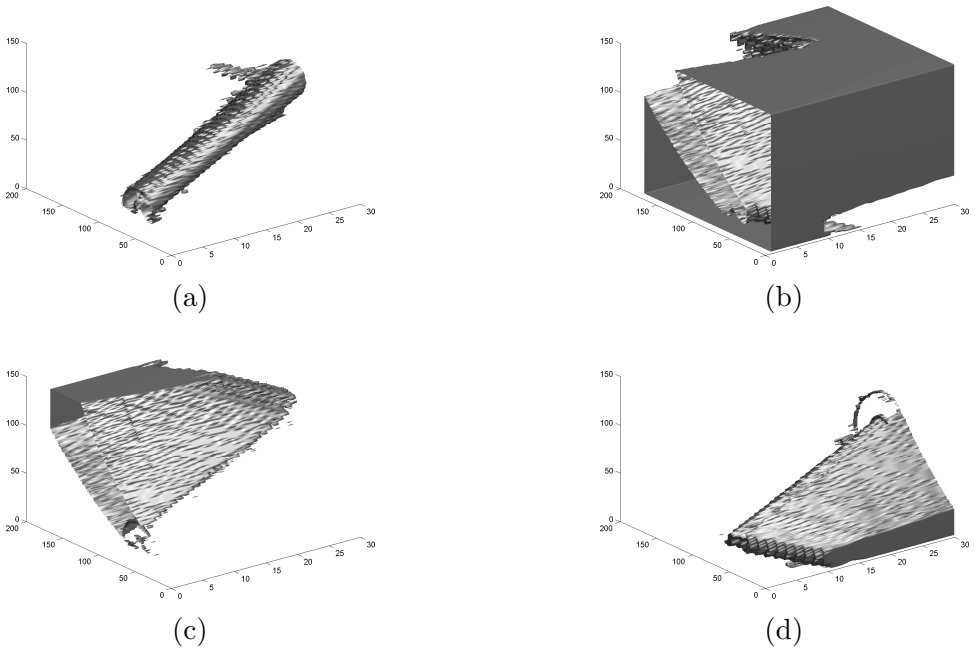


Figure 7·14: Results for the M-frame MCS-B algorithm initialized with the M-frame MD-BMC result (Fig. 7·5), applied to the synthetic image sequence *Bean Normal*: frames (a) #10 and (b) #20 from the sequence overlaid with final boundaries (intersections define four regions), and (c–d) corresponding label fields (white – background, light gray – object, dark gray – occluded background, and black – exposed background).



(a) (b) (c) (d)

Figure 7·15: Volumes corresponding to results from Fig. 7·14: (a) object tunnel, (b) background tunnel, (c) background occlusion volume , and (d) background exposed volume.

Chapter 8

Fast level-set implementation

The level-set approach to image and video segmentation problems is very attractive since it handles topological changes in the solution and has a simple numerical implementation for arbitrary dimensions. However, it is very involved computationally (especially for higher dimensional problems) and, therefore, not very practical for any time-critical application (e.g., real-time tracking or segmentation). In this chapter, we describe a novel implementation of the level-set method developed by Shi (Shi, 2005) which dramatically reduces the computational complexity of the algorithm, and we apply his approach to our problem of video segmentation.

8.1 Background

In the level-set method, a curve C is represented implicitly as the zero-level set of a function ϕ defined over a uniform grid. Following a method proposed by Shi (Shi, 2005), we choose ϕ to be negative inside the contour C and positive outside C. We assume that ϕ is defined over the domain $D \subset \mathcal{R}^K$ and discretized on a uniform grid with the sampling interval of one. Also, by curve C we mean a surface in 3-D case and a hypersurface in K-dimensional case. Given this implicit representation for the curve C, we can define two lists of grid points adjacent to contour C, as follows:

$$\begin{aligned} L_{out} &= \{\boldsymbol{x} | \phi(\boldsymbol{x}) > 0 \; and \; \exists \boldsymbol{y} \in N(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{y}) < 0\}, \\ L_{in} &= \{\boldsymbol{x} | \phi(\boldsymbol{x}) < 0 \; and \; \exists \boldsymbol{y} \in N(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{y}) > 0\}, \end{aligned} \tag{8.1}$$

where $\boldsymbol{x} = (x_1, x_2, ..., x_K)$ and $N(\boldsymbol{x})$ is a discrete neighborhood of $\boldsymbol{x}$ defined as follows:

$$N(\boldsymbol{x}) = \{\boldsymbol{y} \in D | \sum_{k=1}^{K} |y_k - x_k| = 1\} \; \forall \boldsymbol{x} \in D.$$

L_{in} is the list of grid points adjacent to the contour C that are just inside C, while L_{out} is the list of adjacent points that are just outside C. For a given curve C, the two lists of neighboring grid points L_{in} and L_{out} are unique and, vice versa, the location of the curve can be determined up to the accuracy of the grid sampling interval if the two lists are given.

Let us follow with an analysis of the relationship between the motion of the level-set curve and the two lists L_{in} and L_{out} presented by Shi (Shi, 2005). In order to move the curve C outside a certain grid point A at position $\boldsymbol{x}_A$ (which means that $\phi(\boldsymbol{x}_A)$ becomes negative) it is enough to switch it from the list L_{out} to L_{in}. Similarly, to move the curve inward, inside of some point B, we just need to switch the grid point B membership from L_{in} to L_{out}. Switching points between L_{in} and L_{out} requires only to set ϕ to some negative or positive values at those points. By applying such a procedure to all points in L_{in} and L_{out}, we can move the contour C inward or outward one grid point at a time everywhere along the curve with minimal computation. This is the basis for the fast level-set method proposed by Shi (Shi, 2005).

8.2 Implementation

In the standard level-set method, the curve C is evolved under a force field F by solving the following PDE numerically:

$$\phi_t + F|\nabla\phi| = 0. \tag{8.2}$$

However, in Shi's method, the curve C is evolved by simply switching points from L_{out} to L_{in} or back, without explicitly solving equation (8.2). There are two implementations of the fast level-set method proposed by Shi (Shi, 2005): for general evolution forces F and using a fast, smoothing regularization. In the latter implementation, Shi proposed to separate the evolution due to the data-dependent force and the smoothness regularization into two cycles. The smoothness regularization is realized with simple Gaussian filtering that is approximated with integer operations. The smoothing procedure is performed much less frequently if the noise level is low. In the first cycle of the algorithm, several iterations of the curve evolution with the data-dependent force are performed using the fast algorithm for the general evolution speeds. In the second cycle, several iterations of Gaussian filtering are applied to the level-set function to smooth the curve. We will concentrate on the fast level-set method implementation for the general evolution forces.

The following data structure is used in the fast implementation:

- ϕ – an array for the level-set function;
- F – an array for the force;
- L_{in} and L_{out} – lists of inside and outside neighboring grid points.

The values for the function ϕ are chosen from a limited set of integers to achieve faster computation. This function is defined to locally approximate the signed distance function:

$$\phi(\boldsymbol{x}) = \begin{cases} 3 & \text{if } \boldsymbol{x} \text{ is an exterior point;} \\ 1 & \text{if } \boldsymbol{x} \in L_{out}; \\ -1 & \text{if } \boldsymbol{x} \in L_{in}; \\ -3 & \text{if } \boldsymbol{x} \text{ is an interior point.} \end{cases} \tag{8.3}$$

In Shi's algorithm, the magnitude of the evolution force is ignored, only the sign is used.

Hence, F is also an integer array with entries 1, 0, or -1. The two lists L_{in} and L_{out} are standard bi-directionally linked lists.

In the implementation proposed by Shi (Shi, 2005), two basic operations on the data structure defined above are used. The procedure *switch_in()* for a point $\boldsymbol{x} \in L_{out}$ switches $\boldsymbol{x}$ from L_{out} to L_{in}. With $\boldsymbol{x} \in L_{in}$ now, all its neighbors that were exterior points before become neighboring grid points now and are added to L_{out} in the second step of the operation. By applying the *switch_in()* procedure to any point in L_{out}, the boundary is moved outward by one grid point at that location. Similarly, the procedure *switch_out()* for a point $\boldsymbol{x} \in L_{in}$ switches $\boldsymbol{x}$ from L_{in} to L_{out} and adds all its neighbors which were interior points to L_{in}. By applying the *switch_out()* procedure to an inside neighboring grid point, we move the boundary inward by one grid point.

Now, we will describe the fast level-set implementation for general evolution forces. At every iteration, the speed at all points in L_{in} and L_{out} is computed and its sign is stored in F. Next, we scan through the list L_{out} and apply the *switch_in()* procedure at points for which $F > 0$. This scan moves parts of the curve with positive speed outward by one grid point. After this scan, some of the points in L_{in} become interior points due to newly-added inside neighboring grid points, and they are deleted. We then scan through the list L_{in} and apply the *switch_out()* procedure for each point with $F < 0$, moving those parts of the curve with negative speed inward by one grid point. Similarly to the previous scan, points in L_{out} which became exterior points, are deleted from the list. After a scan through both lists, a stopping criterion is checked; for example, the evolution stops if either of the following conditions is satisfied:

- the force at each neighboring pixel satisfies:

$$\begin{aligned} &F(\boldsymbol{x}) \leq 0 \; \forall \boldsymbol{x} \in L_{out} \text{ and} \\ &F(\boldsymbol{x}) \geq 0 \; \forall \boldsymbol{x} \in L_{in}. \end{aligned} \tag{8.4}$$

- a pre-defined maximum number of iterations is reached.

If the force F is non-positive for all points in L_{out}, the *switch_in()* procedure is not performed anywhere on the curve C (the curve is not moving outward). Similarly, if the force F is non-negative for all points in L_{in}, the *switch_out()* procedure is not performed anywhere on the curve C (the curve is not moving inward). If both of these conditions are satisfied, which is expressed in (8.4), the curve C does not move at all in the current iteration and the values in the arrays L_{in}, L_{out}, and ϕ do not change. That means that the curve would not move in the following iterations either, so the evolution process can be stopped.

When the level-set function ϕ is chosen to be the signed distance function, the curvature of C equals the Laplacian of ϕ. In the implementation proposed by Shi, ϕ is chosen to locally (around zero-level set) approximate the signed distance function, and given that only the sign of the force function is important, we decided to approximate the curvature in the force calculation by the Laplacian of the level-set function ϕ:

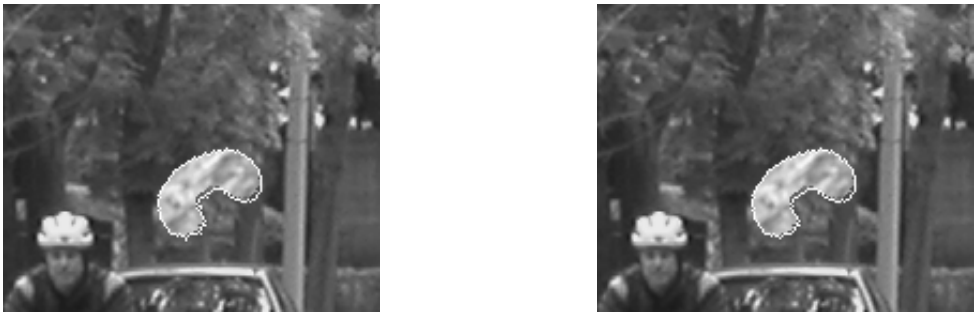
$$\kappa_m \approx \Delta\phi = \phi_{xx} + \phi_{yy} + \phi_{tt}. \tag{8.5}$$

This algorithm can be viewed as an extreme Narrow Banding since it limits the computation to only grid points neighboring the curve. However, the curve is evolved without solving the level-set PDE, while the major advantages of the level-set method are kept, such as the automatic handling of topological changes and the generality of the numerical scheme for arbitrary dimensions. The computations are on integers which makes the algorithm even more efficient. Also, since no PDE is solved, there is no need for careful step size control to maintain numerical stability in this method and reinitialization is not an issue. However, intermediate positions of the curve C obtained after each iteration of the fast level-set algorithm do not correspond to the ones obtained by solving the level-set PDE (8.2). The level-set function is approximated by integer

values defined in (8.3), which means that the position of zero-level set which defines the curve C is also approximate. Furthermore, the curvature of C which is calculated from this integer-valued approximation of ϕ is only an approximation of the curvature in the standard level-set method. Although Shi claims that the final position of curve C calculated using the fast level-set method corresponds to the one obtained by the standard method, we think that all these approximations may introduce errors in the final solution.

8.3 Experimental comparison with the standard level-set method

In order to assess how much the approximations described in the previous section influence the results of the fast level-set method, we use it in several of our motion detection and segmentation methods described earlier. Then, we compare those results to the ones obtained using the standard level-set method.

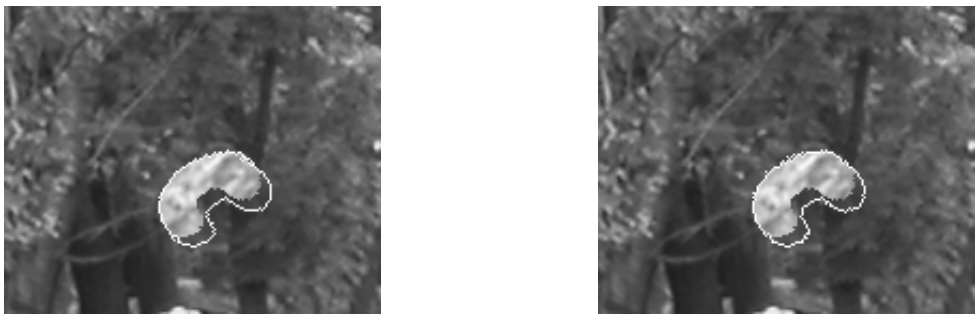


(a) (b)

Figure 8·1: Results for the M-frame MD algorithm, applied to the synthetic image sequence *Bean*: frame #15 from the sequence overlaid with final boundary calculated using the (a) standard level-set implementation and (b) fast level-set implementation.

Fig. 8·1 shows one frame of the *Bean* sequence overlaid with the final boundary calculated using the M-frame MD method with the standard and fast level-set implementations. It is obvious that the results are very similar, which is further confirmed by the segmentation error values shown in the first row of Table 8.1. However, the fast

level-set method requires only 100 iterations, and the result is computed in less than a minute (less than 2s per frame since we are processing 30 frames at once), compared to hours required by the standard method (Section 3.2).



(a) (b)

Figure 8·2: Results for the M-frame MD-BMC algorithm, applied to the synthetic image sequence *Bean Small*: frame #15 from the sequence overlaid with final boundary calculated using the (a) standard level-set implementation and (b) fast level-set implementation.

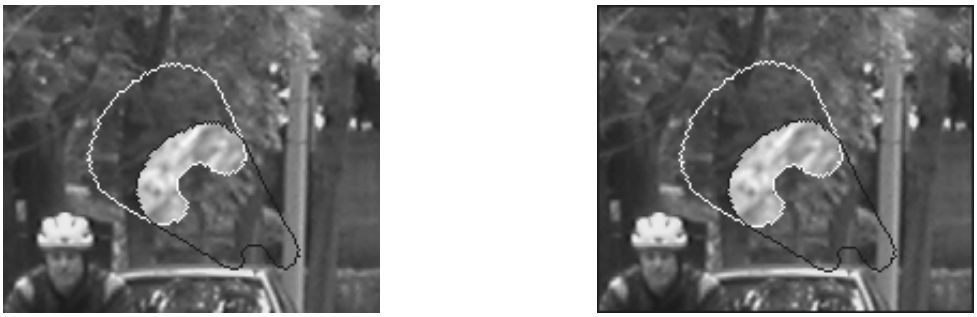
A similar conclusion can be drawn from Fig. 8·2, which shows results obtained by applying the M-frame MD-BMC method on the *Bean Small* sequence. The fast implementation results in somewhat larger segmentation error, but that is due to the choice of detection parameters. For the standard level-set implementation experiment, we used the optimal set of parameters, which is no longer optimal when used with the fast method. If another set of parameters is used with the fast method, object segmentation error is much closer to that of the standard method (the error is 409.97 pixels per frame).

Different results are obtained when the same detection method is applied to the standard *Foreman* sequence (Fig. 8·3). Even visually, the resulting level-set boundaries for the standard (Fig. 8·3 (a)) and fast level-set method (Fig. 8·3 (b)) differ significantly. Again, if we choose a different set of parameters for the fast level-set implementation, adjusted for that method, we are able to achieve a result (Fig. 8·3 (c)) similar to the one in Fig. 8·3 (a). However, there is one more aspect in which this experiment differs from the others: the value of the regularization parameter λ was set to 0.2, while in all other

(a) (b) (c)

Figure 8·3: Results for the M-frame MD-BMC algorithm, applied to the *Foreman* image sequence: frame #15 from the sequence overlaid with final boundary calculated using the (a) standard level-set implementation, (b) fast level-set implementation, and (c) fast level-set implementation with adjusted set of parameters.

experiments we used $\lambda = 0.1$. Obviously, higher values of the curvature parameter λ influence standard and fast level-set method solutions differently. It seems that for the same value of λ the fast method solution is smoother which means that the curvature term influence is larger.



(a) (b)

Figure 8·4: Results for the M-frame MCS-B algorithm, applied to the synthetic image sequence *Bean*: frame #20 from the sequence overlaid with final boundaries calculated using the (a) standard level-set implementation and (b) fast level-set implementation.

Finally, we applied our M-frame MCS-B method to the *Bean* sequence and the results for the two implementations are shown in Fig. 8·4. Although we used the same set of parameters in the segmentation step, we initialized it using the optimal detection

results and optimal motion estimates which are different for standard and fast level-set implementation (i.e., different parameters are used in M-frame MD method implemented using standard and fast level-set implementations to calculate the initial segmentation surfaces; furthermore, different parameters are used in the initial estimation of motion parameters based on the initial surfaces calculated using the standard and fast level-set methods). As the result, we obtained very small object segmentation errors (Table 8.1, third row), and almost perfect results.

Table 8.1: Object segmentation error ϵ_s (pixels per frame) for synthetic test sequences, for different detection and segmentation algorithms implemented using the standard and fast level-set methods.

Sequence (Method)	standard level-set	fast level-set
Bean (M-frame MD)	128.1 (10.8%)	120.5 (10.2%)
Bean Small (M-frame MD-BMC)	391.1 (33%)	479.2 (42.3%)
Bean (M-frame MCS-B)	6.0 (0.5%)	3.7 (0.3%)

Based on these experiments, we can conclude that the fast level-set method produces results with the accuracy and flexibility of the standard level-set method, at a fraction of the computational cost. The only major difference is how each method handles higher values of the regularization parameter λ. The influence of the curvature term grows much faster with λ for the fast method, and that should be taken into account when the parameters for an experiment are being chosen.

8.4 Experimental results for the VIVID dataset

The fast level-set method allows us to run our segmentation algorithms on much larger sequences (spatially and temporally) in a reasonable amount of processing time. An interesting database of video sequences can be found on VIVID (Video Verification

of Identity) Tracking Evaluation Web Site (Collins et al., 2005) at Carnegie Mellon University. For each sequence in the database, ground-truth segmentation data are provided for one moving object. We chose two VIVID sequences to further test our M-frame MD-BMC algorithm.

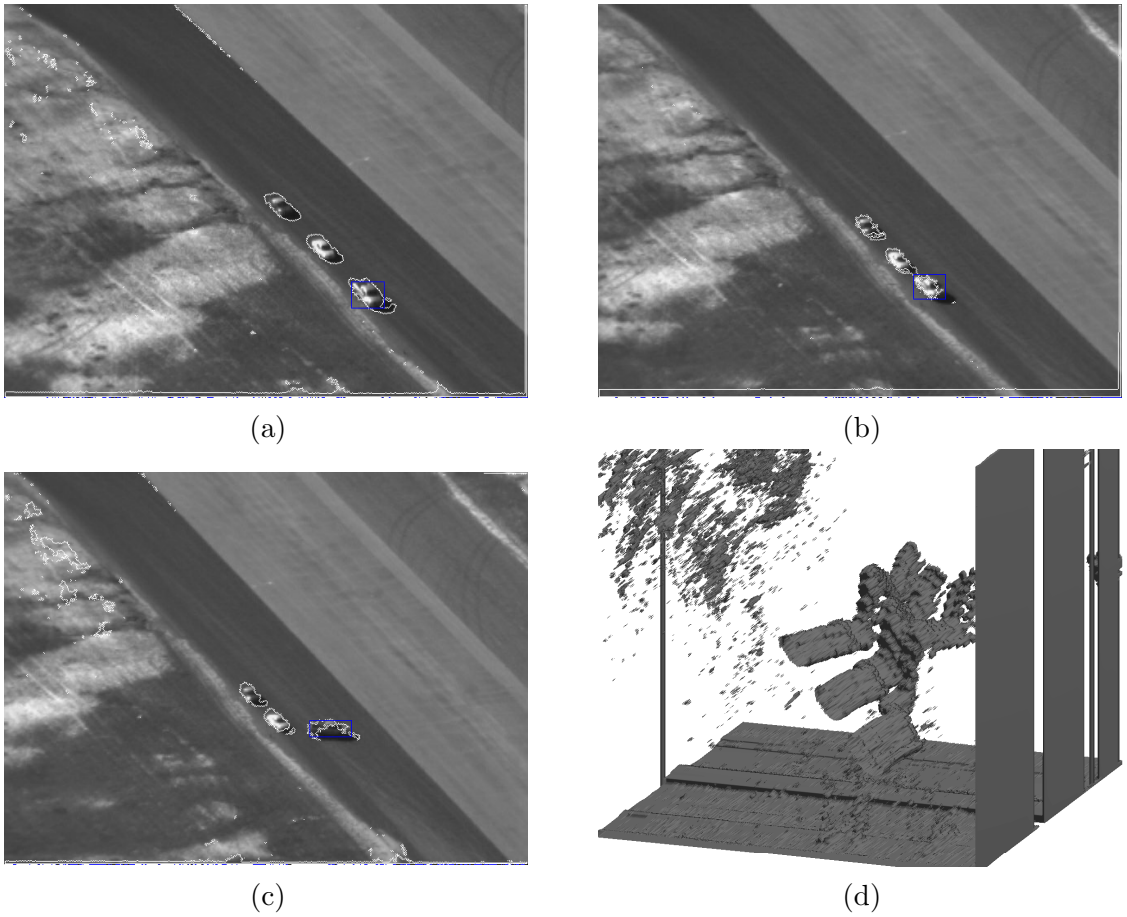
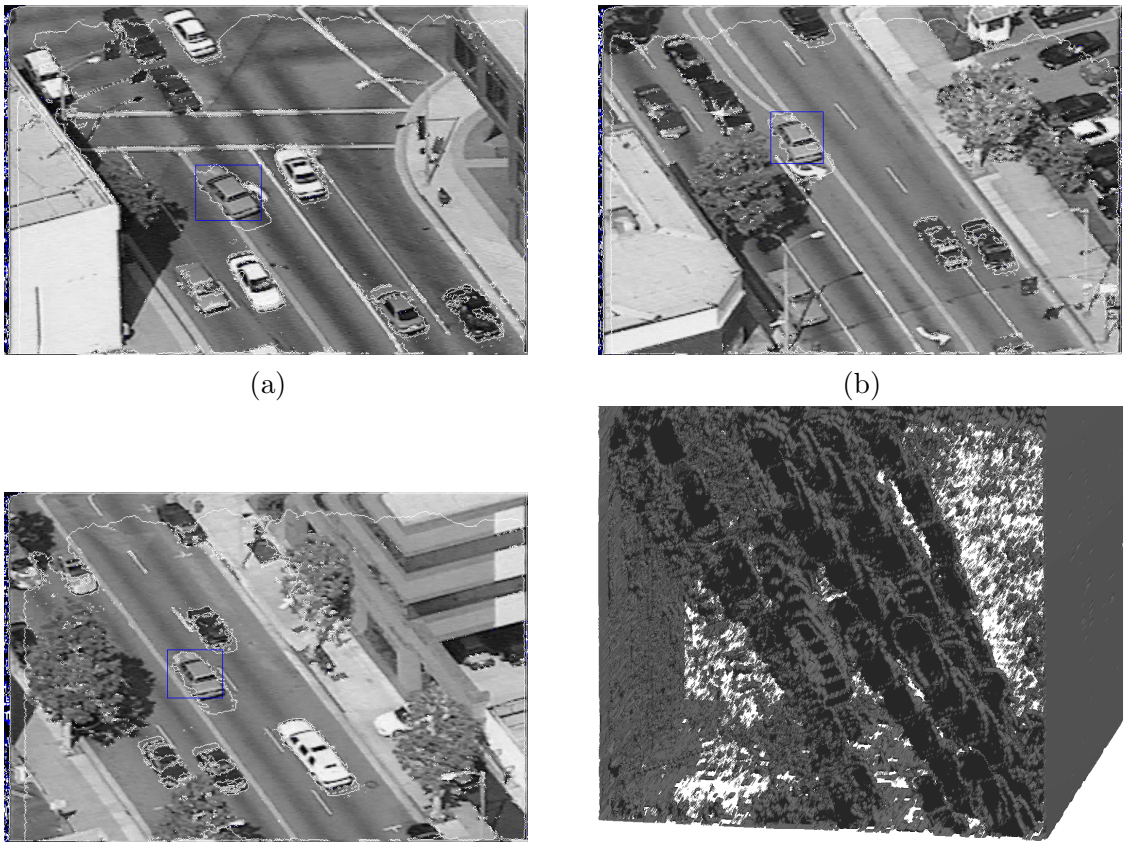


Figure 8·5: Results for the M-frame MD-BMC algorithm, applied to the *Egtest02* image sequence: frames (a) #10, (b) #30, and (c) #50 from the sequence overlaid with ground-truth tracking rectangle and final boundaries, and (d) the corresponding object tunnel.

Figs. 8·5 (a)–(c) show three frames of the sequence *Egtest02* (640×480 pixels, 60 frames) overlaid with the ground-truth tracking rectangle and final boundaries calculated using our M-frame MD-BMC method. The algorithm converges after 400 iterations

with the following parameters: $\alpha = 0.4$ and $\lambda = 0.1$. Fig. 8·5 (d) shows the corresponding object tunnels. The three moving cars are pretty well estimated throughout the sequence. When the first car in the column turns left in the second part of the sequence, it is not estimated as well anymore due to a change in illumination which makes it more similar to the background. Given the moving background, some noise present in the upper-left corner of the sequence is not unexpected – that area has more texture and cannot be perfectly compensated for. False object areas along right and bottom edges of each frame are newly uncovered background areas with no corresponding regions in previous frames. The ground-truth segmentation is given only for the first car in the column with respect to which our result gives 35.12% segmentation error (averaged over 30 frames for which the ground-truth data are known). As explained above, this error is much larger once the vehicle turns in the second part of the sequence. In order to compare our results with the baseline methods, we used results from the VIVID web site (Collins et al., 2005). For the comparison measure, we used average accuracy in terms of percentage of similarity between ground-truth bitmap and estimated bitmap within the ground-truth tracking rectangle area. The best baseline results reported are obtained using Basic Mean Shift method (Comaniciu et al., 2003), which produces 25.31% segmentation error compared to 35.12% we obtained. Given that the baseline methods are specifically developed for tracking ground vehicles from airborne sensor platforms, performance of our detection method is quite satisfactory.

Fig. 8·6 shows results for the sequence *Hollywood* (720×480, 150 frames). Results are obtained after 500 iterations for the following detection parameters: $\alpha = 0.4$ and $\lambda = 0.1$. There are several moving cars in the scene which are fairly well estimated. There is also large background motion present, which yields false object regions similarly to the *Egtest02* sequence: noise in the background and newly-exposed background regions on the right and top edges of each frame. The ground-truth data are given for



(a) (b) (c) (d)

Figure 8·6: Results for the M-frame MD-BMC algorithm, applied to the *Hollywood* image sequence: frames (a) #25, (b) #75, and (c) #125 from the sequence overlaid with ground-truth tracking rectangle and final boundaries, and (d) the corresponding object tunnel.

the car bounded by the tracking rectangle in Fig. 8·6 (a)–(c), with respect to which our method produces segmentation error of 20.96% (again averaged over 30 frames for which the ground-truth data is known, using the same accuracy measure as for the previous sequence). The best baseline results reported on the VIVID web site are for the Enhanced Meanshift with Foreground/Background Ratio method (Comaniciu et al., 2003), which produces segmentation error of 27.88%. Although our method outperformed the best baseline result, it is important to note that we performed tracking over a much smaller subsequence of the *Hollywood* sequence compared to the baseline method.

Chapter 9

Conclusions and future work

In this dissertation we presented our work on several important areas of video analysis, including video segmentation, motion estimation and modeling, and detection of occlusion events. We proposed a novel approach to joint space-time, motion-based video segmentation. The problem is posed in variational framework with respect to a 3-D surface that partitions the image sequence domain into inside and outside. The inside corresponds to *object tunnel*, a 3-D volume "carved out" by a moving object, while the outside corresponds to background. The problem is formulated as *volume competition*, a 3-D generalization of region competition (Zhu and Yuille, 1996). The resulting active surface evolution equation is discretized and solved using standard level-set approach (Sethian, 1996b). We used this framework to develop simple algorithm for motion detection based on models proposed earlier in the literature (Jehan-Besson et al., 2000).

This multi-frame motion detection algorithm commits consistent errors due to the nature of the observation model used: segmented objects in each frame represent a union of object positions in consecutive frames. To address this issue, we extend the method by explicitly modeling the evolution of object and background using motion trajectories (Ristivojević and Konrad, 2004b). However, occluded and newly-exposed background areas (due to object motion) cannot be explained by these motion trajectories. To account for these regions, we include explicit models for the occluded and newly-exposed background areas, which leads to new space-time concepts of *occlusion volume* and *exposed volume*. As for motion trajectories, we use a parametric model associated either

with object or background. Since we need to partition the image sequence domain into 4 volumes, we use the multiphase level-set framework (Vese and Chan, 2002). Although this approach significantly improves segmentation accuracy, it does not handle image sequences in which object occlusion is present. To alleviate that, we extend our formulation to include explicit models of the occluded and newly-exposed areas of the object (Ristivojević and Konrad, 2004a). Finally, we generalize our motion segmentation formulation to allow for any number of moving objects and occlusion/exposure volumes.

Our simple multi-frame motion detection algorithm performs well only for image sequences with static backgrounds, limiting the set of image sequences it can be applied to. We expand our motion detection formulation to account for camera motion and zoom. We use affine model to describe camera motion between frames and perform simultaneous evolution of the level-set surface that encloses moving objects and motion estimation in the background region.

To reduce computational complexity of the level-set method, we applied a novel level-set implementation, recently developed by Shi (Shi, 2005), to our motion detection and segmentation methods and investigated its performance. We obtained reduction of computation time by up to 200 times, while preserving segmentation accuracy of the standard level-set implementation.

In our segmentation methods, we modeled evolution of moving objects and background using motion trajectories which are piece-wise linear, i.e., they consist of motion vectors calculated for each frame pair separately. Given that the underlying motion of moving objects is smooth in temporal direction, we expanded our motion estimation formulation with a term that penalizes the difference between consecutive motion vectors along trajectory.

We can conclude that active-surface level-set framework looks very promising for video analysis. Using our multi-frame motion detection and segmentation algorithms

(a) (b) (c)

Figure 9·1: Results for M-frame MD algorithm, applied to a natural image sequence with two cars occluding one another: frames (a) #1, and (b) #15 from the sequence overlaid with final boundaries, and (c) the corresponding object tunnel.

complex tunnels can be calculated describing interesting events. For example, Fig. 9·1 shows results for an image sequence in which two cars are moving one behind the other. At the start of the sequence, the first car occludes the second car, but as they turn in different directions, both cars become visible. That is very well illustrated by the fork in the tunnel in Fig. 9·1 (c). Based on such tunnels accurate motion over many frames could perhaps be computed, as well as different interesting events in the sequence (occlusions, moments of collision between objects, etc.).

The main contributions of this dissertation are:

- Development of a novel approach to joint spatio-temporal motion-based video detection and video segmentation based on volume competition and surface evolution.

- Development of new, space-time models for occluded and newly-exposed areas in a video sequence.

- Introduction of new space-time concepts of occlusion and newly-exposed volumes.

9.1 Future work

In this section we will briefly describe several directions in which our work on video segmentation, motion estimation and occlusion detection could be extended.

9.1.1 Better modeling of occluded and newly-exposed areas

In Chapter 5 we saw that estimated occluded and newly-exposed volumes of an object are not very accurate for the case of the natural image sequence *Car*. We believe that this is due to inaccuracies in the estimated motion and variations of object intensity over time. However, these problems are likely to arise for any natural (camera-acquired) sequence, in case of occluded/newly-exposed volumes of moving objects or non-static background, i.e., whenever it is impossible to estimate motion with perfect accuracy. It is necessary to address this problem by proposing better models for occluded and newly-exposed areas as well as for fully-visible areas. Models proposed in equations (4.2)–(4.4) assume that intensity variations over the part of motion trajectories where object is visible must be small. However, if object intensity changes over time and/or motion trajectories are not estimated accurately enough, this condition will not be satisfied. The assumption of constant intensity over motion trajectory has to be replaced with a different approach, and investigation of that problem is an important part of future work. Negahdaripour and Gennert (Negahdaripour et al., 1989) propose to replace the brightness constancy constraint with a more general constraint, which permits a linear transformation between image brightness values. The transformation parameters are allowed to vary smoothly so that inexact matching is allowed. De Micheli *et al.* (De Micheli et al., 1993) replaced constant brightness assumption with the assumption of constant spatial intensity gradient. These and other similar ideas should be investigated and applied to the modeling of occluded and newly-exposed areas.

9.1.2 Spline modeling of motion trajectories

Our approach to the estimation of motion parameters, as described in Section 4.2.2, was to estimate them separately for each frame pair in the sequence, with no explicit relationship between motion vectors at different time instants. In Section 4.2.3, we introduced a temporal smoothness constraint in the estimation of motion parameters, to account for the smooth underlying motion of moving objects in the temporal direction. However, in both approaches we model motion trajectories $\boldsymbol{c}(t_i; \boldsymbol{x}, t)$ (defined in Section 4.1) as piecewise-linear between images in each frame pair. Ideally, one would like a consistent spatio-temporal motion model to match the spatio-temporal tunnel model we proposed. Although a dense representation (optical flow) with suitable constraints could serve this purpose, it requires very many parameters and does not support explicit continuous representation (motion between grid points is computed through arbitrary interpolation). An implicit continuous-space motion representation (such as affine) is needed in order to recover motion at arbitrary grid point from a single set of parameters. Clearly, a parametric motion model supporting explicit continuous representation would be more suitable.

The ultimate goal here is to model motion trajectories over the whole span of an image sequence. In this context, it would be interesting to introduce a continuous temporal parametric model which will describe moving object trajectories across the image sequence. Such a model would allow more accurate motion-compensated temporal interpolation of a video sequence. Moreover, a parametric temporal model would introduce strong constraint on motion trajectories in the direction of time, which, in turn, may allow one to replace the parametric spatial motion model (affine) with a less constrained smooth motion field.

A quadratic trajectory model was originally proposed in (Dubois and Konrad, 1993), and developed in detail by Chahine and Konrad (Chahine and Konrad, 1995). This

model worked well, but only for the trajectories with up to 7 frames, and with simple spatial smoothness model. We feel that more flexible model is required and we think that polynomial spline model (Unser, 1999; Szeliski and Shum, 1996; Szeliski and Coughlan, 1997) would be more appropriate to describe motion trajectories over sequences of 30 and more frames. Thus, we propose to model motion trajectories temporally using splines.

In the current implementation of our video segmentation algorithm, we need motion defined on all grid points for each of the different motion trajectories (objects, background). For example, if we consider level-set evolution equations for segmentation surfaces in the M-frame MCS-B method, (4.6), it is obvious that the terms $\xi_{obj}(\boldsymbol{x}, t; \boldsymbol{p})$ and $\xi_{obj}(\boldsymbol{x}, t; \bar{\boldsymbol{p}})$ (defined in (4.2)) need to be calculated for all points of an image sequence. In order to calculate these terms at any point, we have to measure intensity variation over object and background trajectories passing through that point, which means that those trajectories need to be defined for all points of an image sequence (i.e., both object and background trajectories need to be defined inside the object volume, as well as background volume). In general, motion trajectories passing through neighboring grid points are related (except at object boundaries). Thus, any temporal motion model needs to be combined with some form of spatial model. A simple solution would be to enforce spatial smoothness of motion trajectories, however, only a spatially-parametric model would allow us to compute displacements at any spatial position in a consistent fashion. One option would be to keep the affine model in spatial domain, but it is not clear how to combine it with the temporal spline model. Furthermore, spatial affine models are restrictive and describe accurately only projection of the motion of a planar patch in 3-D space under orthographic projection. For more complicated objects, e.g., human body motion, we need less restrictive spatial model. An alternative is to use a spatial spline model (especially smoothing splines (Precioso et al., 2003)) in all three

dimensions.

We propose to represent motion trajectories using splines in the following manner. For each point in the first frame of the sequence (the first frame is chosen as the reference frame), $(\boldsymbol{x}, t_1)$, a trajectory passing through this point is defined as follows

$$\boldsymbol{c}(t; \boldsymbol{x}, t_1) = \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x}, t), \tag{9.1}$$

where we model the displacement $\boldsymbol{d}(\boldsymbol{x}, t)$ using *B-splines* (Unser, 1999):

$$\begin{aligned} \boldsymbol{d}(\boldsymbol{x}, t) &= (d_x(\boldsymbol{x}, t), d_y(\boldsymbol{x}, t)) \\ d_x(\boldsymbol{x}, t) &= \sum_{k \in Z} \sum_{l \in Z} \sum_{m \in Z} c_x(k, l, m) \beta^3(x - k) \beta^3(y - l) \beta^3(t - m) \\ d_y(\boldsymbol{x}, t) &= \sum_{k \in Z} \sum_{l \in Z} \sum_{m \in Z} c_y(k, l, m) \beta^3(x - k) \beta^3(y - l) \beta^3(t - m). \end{aligned} \tag{9.2}$$

In the equations above, the spatial position vector $\boldsymbol{x}$ has two components, $\boldsymbol{x} = (x, y)$, while the B-spline of degree 3, denoted by $\beta^3(x)$, is defined as

$$\beta^3(x) = \begin{cases} \frac{2}{3} - |x|^2 + \frac{|x|^3}{2}, & 0 \leq |x| < 1 \\ \frac{(2 - |x|)^3}{6}, & 1 \leq |x| < 2 \\ 0, & 2 \leq |x|. \end{cases} \tag{9.3}$$

The above model offers a spline representation of motion trajectories passing through any point in the first frame, including grid points. However, those trajectories do not necessarily pass through grid points in other frames of the sequence. Fig. 9·2 illustrates this problem for 2-D case $(x - t)$. In the segmentation methods we proposed thus far, we need trajectories passing through all grid points in all frames in order to calculate energy terms ξ_{obj}, ξ_{occ}, and ξ_{exp} (for example, in the case of M-frame MCS-B method, defined in equations (4.2) – (4.4)). Therefore, we need to calculate trajectory passing through any point in the sequence domain based on the proposed spline representation

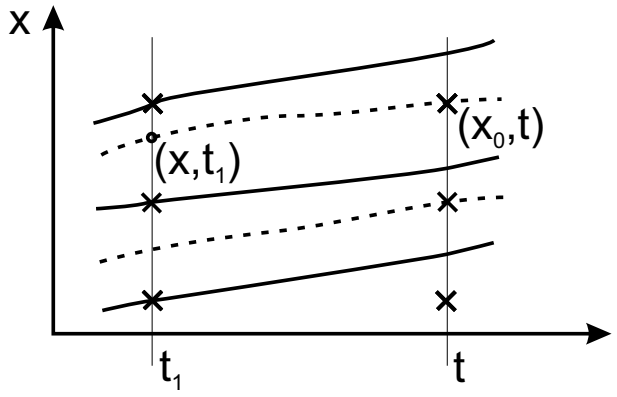


Figure 9·2: Example of motion trajectories in 2-D case: the known motion trajectories passing through grid points in frame at t_1 are represented with solid lines; the unknown trajectories passing through grid points in frame at t are dashed.

anchored in the first frame. In order to do that, we need to find a point $(\boldsymbol{x}, t_1)$ in the reference frame from which originates a trajectory passing through grid point $(\boldsymbol{x}_0, t)$ in frame at time t (see Fig 9·2 for an example of such trajectory). This can be formulated as follows

$$\text{find } \boldsymbol{x} \text{ such that } \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x}, t) = \boldsymbol{x}_0. \tag{9.4}$$

Given the nonlinearity of $\boldsymbol{d}(\boldsymbol{x}, t) = (d_x(\boldsymbol{x}, t), d_y(\boldsymbol{x}, t))$, equation (9.4) cannot be solved analytically. We propose to estimate $\boldsymbol{x} = (x, y)$ using the following minimization:

$$\begin{aligned} (x, y) &= \arg\min_{\boldsymbol{x} \in \Omega} f(x, y) \\ f(x, y) &= (x_0 - (x + d_x(\boldsymbol{x}, t)))^2 + (y_0 - (y + d_y(\boldsymbol{x}, t)))^2. \end{aligned} \tag{9.5}$$

We solve this minimization using steepest descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda_k \nabla f(\boldsymbol{x}_k). \tag{9.6}$$

Using this method, only one set of spline coefficients, describing motion trajectories passing through the reference frame, is enough to obtain motion trajectories passing

through any grid (or non-grid) point, in any frame, which makes this spline representation suitable for our segmentation problem. To demonstrate how this method works, we test it on a simple synthetic 1-D example. In this case, we have

$$d(x) = \sum_{k \in Z} c(k)\beta^3(x-k), \tag{9.7}$$

and we are looking for x such that $x + d(x) = x_0$. Minimization (9.5) reduces to:

$$x \quad = \arg\min_{x \in \mathcal{L}} (x_0 - (x + d(x)))^2, \tag{9.8}$$

where $\mathcal{L}$ is the domain (line segment) on which x is defined. Fig. 9·3 (a) shows an example of displacement function $d(x)$ for a particular set of coefficients $c(k)$ within spline representation (9.7). Fig. 9·3 (c) shows estimated values of x for which $x + d(x) = x_0$, where x_0 lays on 1-D grid (x_0 is an integer value in this case). These values are obtained after 100 iterations of the steepest descent algorithm, with $\lambda_k = 0.2$. The associated estimation error is shown in Fig. 9·3 (d). This error is very small, proving that our algorithm works correctly. In this example, the function $x + d(x)$ is monotonic, which corresponds to situations where each point in one frame maps to a different point in the next frame. If this condition is not satisfied (due to occlusion, for example), we would not be able to uniquely recover values of x. However, this condition can be enforced in the process of motion estimation.

The case of 3-D spline representation of motion trajectories (9.2) is a little bit more involved. We want to calculate trajectories passing through grid points in frame at time $t = \tau$ using minimization (9.5). It is important to note that we are solving this minimization for a single frame at time τ, in which case the displacement $\boldsymbol{d}(\boldsymbol{x}, \tau)$ is a function of only two variables, x and y. Figs. 9·4 (a) and (b) show displacement functions $d_x(x, y, \tau)$ and $d_y(x, y, \tau)$ which are calculated using spline representation (9.2). Figs. 9·4 (e) and (f) show the estimated values of x and y for which $\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x}, \tau) = \boldsymbol{x}_0$ holds, where

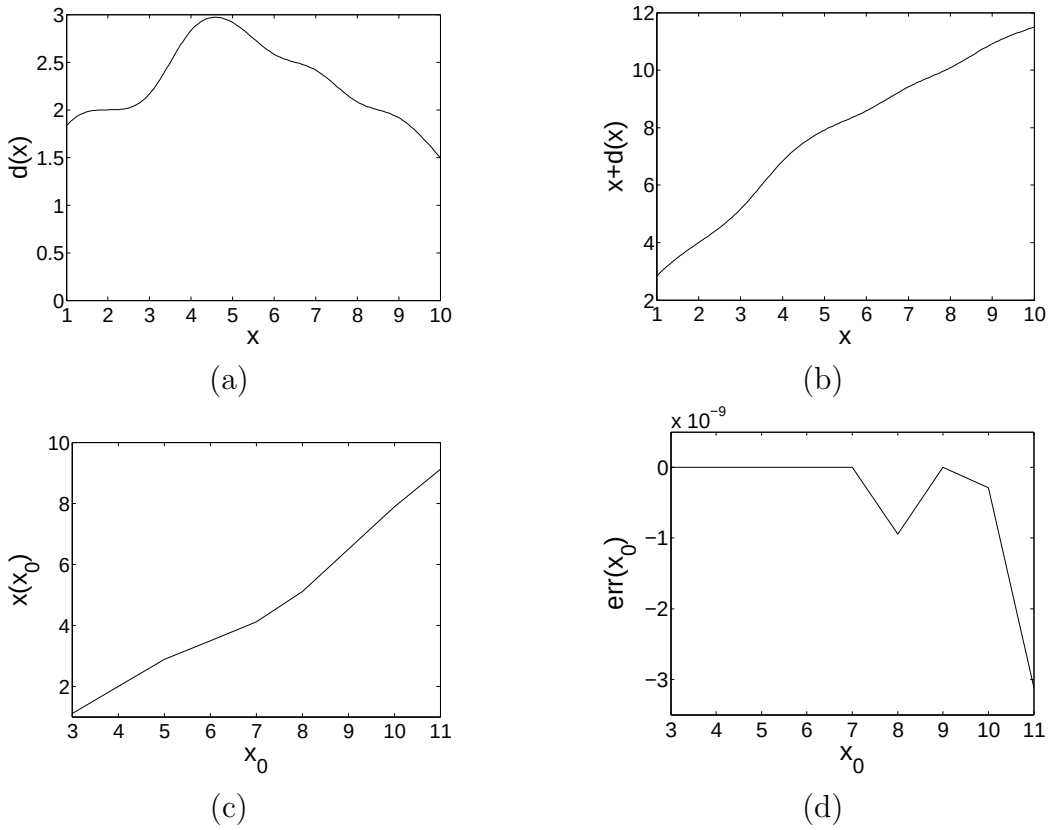


Figure 9·3: Results for 1-D example: (a) displacement $d(x)$, (b) $x+d(x)$, (c) estimated values of x such that $x+d(x) = x_0$, and (d) estimation error, $\text{err}(x_0) = x_0 - (x + d(x))$.

point $\boldsymbol{x}_0 = (x_0, y_0)$ is on a 2-D grid (x_0 and y_0 are integer values in this case). These values are obtained after 100 iterations of the steepest descent algorithm, with $\lambda_k = 0.2$. Fig. 9·4 (g) shows the estimation error which is again very small. This example shows that the proposed method is quite adequate for the calculation of motion trajectories passing through grid points in any frame of the sequence given a spline representation of motion trajectories passing through reference frame. For the small example presented here (10×10 image), execution time of a MATLAB implementation of the algorithm is around $30s$. This means that a more efficient C implementation is required.

The main benefit of this new spatio-temporal spline representation of motion trajectories lays in its flexibility. It allows us to find a balance between accuracy of a dense

(optical flow) representation (which requires too many parameters) and simplicity of global parametric representation (such as affine). This can be achieved by changing the number of spline coefficients $c(k, l, m)$ used (changing the density of spline control grid). This flexibility enables us to more accurately capture complex motion of the objects in an image sequence, and better motion estimates lead to more accurate segmentation results. This new spline motion model could also find application in video coding as an alternative to currently predominant block-based models (better motion accuracy with little additional transmission penalty).

An important area of future work is incorporation of spline spatio-temporal models of motion trajectories into our video segmentation algorithms. A related issue is the estimation of spline coefficients of the proposed model, or motion estimation is spline spaces. One approach is to estimate dense motion vector fields between frames of the sequence using optical flow techniques and then fit a spline model to those vector fields. An alternative approach is to directly estimate the spline coefficients using a method similar to the one proposed in Section 4.2.2, by extending the energy functional (4.9) from two consecutive frames to the whole span of the sequence and minimizing it over all coefficients of the spline model. Clearly, instead of the displaced frame difference, a measure of intensity variation over motion trajectories would need to be minimized.

The estimated parameters of a spatio-temporal spline trajectory model would be valid only within the associated object or background volume. In principle, the formulation of our volume-competition segmentation requires that motion of each object and background be defined everywhere in the image sequence. However, in practice motion trajectory information is required only at zero-level sets where the evolution force is calculated (i.e., at the boundary of two different volumes only motion for these two objects needs to be defined). Note that in our segmentation algorithms we perform numerous iterations of segmentation surface evolution between motion estimation steps. Even if

motion of all relevant objects and background were defined at the boundary of these volumes initially, after the motion estimation step, as the segmentation surfaces move, that would be no longer true. One solution is to re-estimate motion parameters after each iteration of the surface evolution. However, that is very expensive computationally and extremely inefficient given that the segmentation surfaces change very little between iterations; the newly-estimated motion parameters would be virtually identical to the previous ones. A better solution would be to extrapolate motion models for each volume in a narrow band around the volume boundary, so that a re-estimation of motion parameters would be necessary only after a significant change in segmentation surfaces. It is, however, necessary to investigate how inaccuracies in motion trajectories in the extrapolated region would influence the segmentation result.

Another approach would be to have a single spatio-temporal trajectory model describing trajectories in the whole image sequence domain (all the object and background tunnels). In that case modeling of motion discontinuities within the spline model becomes a very important issue.

9.1.3 Real-time implementation of our video segmentation algorithms

Currently, our segmentation and motion estimation algorithms are implemented using a combination of MATLAB and C code. The evolution of segmentation surfaces using standard level-set method is implemented in MATLAB, with some computationally involved portions of code written as C functions called from MATLAB. However, this implementation takes hours to reach a solution even for a 30-frame QCIF sequence. Even with a more optimized implementation, written exclusively in C, standard level-set implementation of our detection and segmentation algorithms would be far from real-time. On the other hand, the fast level-set implementation takes only 2s per frame to evolve segmentation surface until convergence, even though it is currently written in

MATLAB. An optimized version of this algorithm implemented in C would probably result in near real-time performance.

The motion estimation part of the overall video segmentation algorithm is also written in MATLAB, and takes less than a minute to calculate motion estimates for one frame of an image sequence. A C implementation of the BFGS quasi-Newton method used in MATLAB version would definitely reduce the computation time, however probably not to a real-time level. A faster method to minimize (4.9) is required ((Press et al., 1992) is a good source for minimization algorithms). Overall, with efficient C implementations of both surface evolution and motion estimation algorithms, nearly real-time performance of the overall video segmentation algorithm could be achieved even today. Given the rapid rate of increase in the computing power of personal computers, a real-time implementation will be available in the near future.

A real-time performance would require that our video segmentation method be also autonomous. Currently, our method is semi-automatic; user interaction is still necessary in the domain of choosing segmentation and motion estimation parameters (α, λ, ω's, etc.) and deciding how many segmentation surfaces need to be evolved. For a real-time implementation, we need to find a way to choose or estimate these parameters automatically. For example, value of the parameter α in the M-frame MD algorithm depends on the level of noise in the background. Values of weights ω_i in the M-frame MCS algorithms depend on the variation of intensity over object/background trajectories. In order to estimate those parameters automatically, one could use expectation-maximization (EM) algorithm (Moon, 1996; Zhang et al., 1994).

The number of segmentation surfaces needed in our video segmentation algorithm directly depends on the number of moving objects in the sequence. The M-frame MD algorithm gives us the number of moving regions in each frame of the sequence (after spurious, isolated pixels are removed using morphological operations). However, in

order to be able to perform initial estimation of motion parameters for each moving object we need to track these moving regions throughout the sequence. An interesting approach for finding the number of moving objects based on correspondence estimation is proposed in (Mansouri and Konrad, 2003). They estimate correspondence points and group them into separate classes, each with its own distinct motion, which gives them the number of distinct motion classes and their parameters. One could adapt their approach in the following manner: for each moving region estimated in the detection step of the algorithm, feature points are extracted. For each pair of frames in the image sequence, correspondence between feature points is established, thus establishing correspondence between detected moving regions in those frames. That would allow tracking of moving regions throughout the sequence, and also eliminate larger erroneous regions in the detection result. The number of moving objects in the sequence, L, would be the number of moving regions which were successfully tracked using feature points correspondence. The number of surfaces would be $M = \lceil \log_2 N \rceil$, where $N = 3(L + 1)$, to account for occluded/newly-exposed regions in the objects and background.

Another important issue is the causality of the proposed video segmentation algorithm. Since in our segmentation algorithm multiple frames are processed jointly, including several future frames, the algorithm is inherently non-causal and would cause delay in any real-time implementation. An interesting question for the future work would be whether it is possible to approximate our algorithm with a causal one. The obvious difficulty would be with the estimation of to-be-occluded volumes – in order to check whether certain region will be occluded in the future we need information from the future image frames.

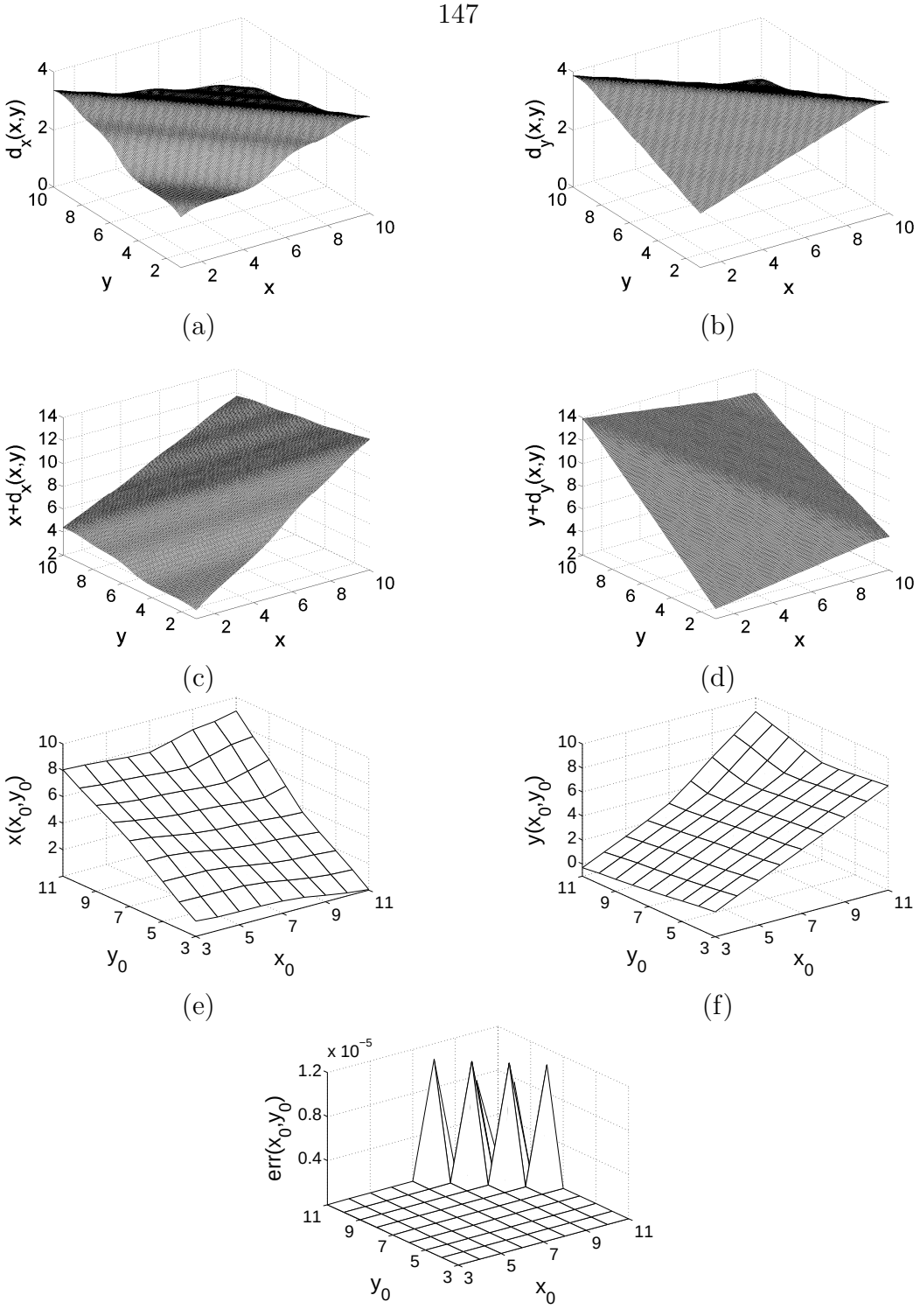


Figure 9·4: Results for 3-D example: displacements (a) $d_x(x, y, \tau)$ and (b) $d_y(x, y, \tau)$, (c) $x + d_x(x, y, \tau)$, (d) $y + d_y(x, y, \tau)$, (e),(f) estimated values of x and y such that $x + d_x(x, y, \tau) = x_0$ and $y + d_y(x, y, \tau) = y_0$, and (g) estimation error, $\mathrm{err}(x_0, y_0) = \sqrt{(x_0 - (x + d_x(x, y, \tau)))^2 + (y_0 - (y + d_y(x, y, \tau)))^2}$.

References

A. Cavallaro, O. Steiger, T. E. (2005). Semantic video analysis for adaptive content delivery and automatic description. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1200–1209.

Adalsteinsson, D. and Sethian, J. (1995). A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269–277.

Adalsteinsson, D. and Sethian, J. (1999). The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148:2–22.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, B 48:259–279.

Black, M. (1992). *Robust incremental optical flow*. PhD dissertation, Yale University, Department of Computer Science.

Bottreau, V., Bénetière, M., Felts, B., and Pesquet-Popescu, B. (2001). A fully scalable 3D subband video codec. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1793–1796.

Brady, N. (1999). Mpeg-4 standardized methods for the compression of arbitrarily shaped video objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1170–1189.

Caselles, V., Kimmel, R., and Sapiro, G. (1997a). Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79.

Caselles, V., Kimmel, R., Sapiro, G., and Sbert, C. (1997b). Minimal surfaces based object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):394–398.

Chahine, M. and Konrad, J. (1995). Estimation and compensation of accelerated motion for temporal sequence interpolation. *Signal Processing, Image Communication*, 7(4–6):503–527.

Chan, T. and Vese, L. (2001). Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277.

Chang, M., Tekalp, A., and Sezan, M. (1997). Simultaneous motion estimation and segmentation. *IEEE Transactions on Image Processing*, 6(9):1326–1333.

Chang, S.-F., Horace, W. C. H., Sundaram, H., and Zhong, D. (1998). A fully automated content based video search engine supporting spatio-temporal queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):602–615.

Chiuso, A., Favaro, P., Jin, H., and Soatto, S. (2002). Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535.

Chou, P. and Brown, C. (1990). The theory and practice of Bayesian image labelling. *International Journal of Computer Vision*, 4:185–210.

Chuang, Y.-Y., Curless, B., Salesin, D., and Szeliski, R. (2001). A bayesian approach to digital matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II264–II271.

Collins, R., Lipton, A. J., and Kanade, T. (1999). A system for video surveillance and monitoring. In *American Nuclear Society Eight International Topical Meeting on Robotics and Remote Systems*.

Collins, R., Zhou, X., and Teh, S. (2005). An open source tracking testbed and evaluation web site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2005)*.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577.

Cremers, D. (2003). A multiphase level set framework for motion segmentation. In Griffin, L. and Lillholm, M., editors, *4th International Conference on Scale Space Theories in Computer Vision*, pages 599–614, Isle of Skye, Scotland. Springer.

De Micheli, E., Torre, V., and Uras, S. (1993). The accuracy of the computation of optical flow and of the recovery of motion parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):434–447.

Debreuve, E., Barlaud, M., Aubert, G., Laurette, I., and Darcourt, J. (2001). Space-time segmentation using level set active contours applied to myocardial gated SPECT. *IEEE Transactions on Medical Imaging*, 20(7):643–659.

Depommier, R. and Dubois, E. (1992). Motion estimation with detection of occlusion areas. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages III.269–III.272.

Driessen, J. and Biemond, J. (1991). Motion field estimation for complex scenes. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 511–521.

Dubois, E. (1992). Motion-compensated filtering of time-varying images. *Multidimensional Systems and Signal Processing*, 3:211–239.

Dubois, E. and Konrad, J. (1993). Estimation of 2-D motion fields from image sequences with application to motion-compensated processing. In Sezan, M. and Lagendijk, R., editors, *Motion Analysis and Image Sequence Processing*, chapter 3, pages 53–87. Kluwer Academic Publishers.

Feghali, R. and Mitiche, A. (2004). Spatiotemporal motion boundary detection and motion boundary velocity estimation for tracking moving objects with a moving camera: A levels sets PDEs approach with concurrent camera motion compensation. *IEEE Transactions on Image Processing*, 13(11):1473–1490.

Feghali, R., Mitiche, A., and Mansouri, A.-R. (2001). Tracking as motion boundary detection in spatio-temporal space. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, pages 600–604.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Haan, G. D. and Bellers, E. (1998). Deinterlacing - an overview. *Proceedings of the IEEE*, 86(9):1839–1857.

Han, S.-C. and Woods, J. (1997). Frame-rate up-conversion using transmitted motion and segmentation fields for very low bit-rate video coding. In *Proceedings of the IEEE International Conference on Image Processing*, volume I, pages 747–750.

Heitz, F. and Bouthemy, P. (1993). Multimodal estimation of discontinuous optical flow using Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232.

Irani, M. and Peleg, S. (1993). Motion analysis for image enhancement: resolution, occlusion and transparency. *Journal of Visual Communication and Image Representation*, 4(4):324–335.

Jehan-Besson, S., Barlaud, M., and Aubert, G. (2000). Detection and tracking of moving objects using a new level set based method. In *Proceedings of the International Conference on Pattern Recognition*, pages 1112–1117.

Jehan-Besson, S., Barlaud, M., and Aubert, G. (2001). Video object segmentation using Eulerian region-based active contours. In *Proceedings of the IEEE International Conference on Computer Vision*.

Jehan-Besson, S., Barlaud, M., and Aubert, G. (2003). DREAM^{2}S: Deformable regions driven by an Eulerian accurate minimization method for image and video segmentation. *International Journal of Computer Vision*, 53(1):45–70.

Karlsruhe, U. (1997). Web site. `http://i21www.ira.uka.de/image_sequences/`.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331.

Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160.

Koch, R. (1993). Dynamic 3-d scene analysis through synthesis feedback control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):556–568.

Konrad, J. and Dubois, E. (1992). Bayesian estimation of motion vector fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):910–927.

Konrad, J. and Ristivojević, M. (2002). Joint space-time image sequence segmentation based on volume competition and level sets. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 573–576.

Konrad, J. and Stiller, C. (1997). On Gibbs-Markov models for motion computation. In Li, H., Sun, S., and Derin, H., editors, *Video Compression for Multimedia Computing – Statistically Based and Biologically Inspired Techniques*, chapter 4, pages 121–154. Kluwer Academic Publishers.

Kuglin, C. and Hines, D. (1975). The phase correlation image alignment method. In *Proceedings of the International Conference on Cybernetics and Society*, pages 163–165.

Kühne, G., Weickert, J., Schuster, O., and Richter, S. (2001). A tensor-driven active contour model for moving object segmentation. In *Proceedings of the IEEE International Conference on Image Processing*, pages 73–76.

Lim, K., Das, A., and Chong, M. (2002). Estimation of occlusion and dense motion fields in a bidirectional Bayesian framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):712–718.

Luthon, F., Caplier, A., and Liévin, M. (1999). Spatiotemporal MRF approach with application to motion detection and lip segmentation in video sequences. *Signal Processing*, 76:61–80.

Malladi, R., Sethian, J., and Vemuri, B. (1995). Shape modeling with front propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–176.

Mansouri, A.-R. and Konrad, J. (1999). Motion segmentation with level sets. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, pages 126–130.

Mansouri, A.-R. and Konrad, J. (2003). Multiple motion segmentation with level sets. *IEEE Transactions on Image Processing*, 12(2):201–220.

Mansouri, A.-R. and Mitiche, A. (2002). Spatial/joint space-time motion segmentation of image sequences by level set pursuit. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, pages 265–268.

Mansouri, A.-R., Mitiche, A., and Feghali, R. (2002). Spatio-temporal motion segmentation via level set partial differential equations. In *Proceedings of the Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 243–247.

Mansouri, A.-R., Mitiche, A., and Vázquez, C. (March 2004). Multiregion competition: A level set extension of region competition to multiple region partitioning of images and image sequences. *Computer Vision and Image Understanding*.

Mémin, E. and Pérez, P. (1998). Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing*, 7(5):703–719.

Mitiche, A., Feghali, R., and Mansouri, A.-R. (2002). Tracking moving objects as spatio-temporal boundary detection. In *Proceedings of the Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 206–110.

Moon, T. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60.

Negahdaripour, S., Shokrollahi, A., and Gennert, M. (1989). Relaxing the brightness constancy assumption in computing optical flow. In *Proceedings of the IEEE International Conference on Image Processing*, pages 806–810.

Ohm, J. (1994). Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571.

Osher, S. and Sethian, J. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on the hamilton-jacobi formulation. *Journal of Computational Physics*, 79:12–49.

Paragios, N. and Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280.

Parker, B. and Magarey, J. (2001). Three-dimensional video segmentation using a variational method. In *Proceedings of the IEEE International Conference on Image Processing*, pages 765–768.

Pesquet-Popescu, B. and Bottreau, V. (2001). Three-dimensional lifting schemes for motion compensated video compression. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1793–1796.

Porikli, F.-M. and Wang, Y. (2001). An unsupervised multi-resolution object extraction algorithm using video-cube. In *Proceedings of the IEEE International Conference on Image Processing*, pages 359–362.

Precioso, F., Barlaud, M., Blu, T., and Unser, M. (2003). Smoothing B-Spline active contour for fast and robust image and video segmentation. In *Proceedings of the IEEE International Conference on Image Processing*, volume I, pages 137–140.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical recipes in C: The art of scientific computing*. Cambridge University Press, 2-nd edition.

Ristivojevic, M. (2004). Web site. `http://iss.bu.edu/mirko/Research/research.html`.

Ristivojević, M. and Konrad, J. (2004a). Joint space-time image sequence segmentation: object tunnels and occlusion volumes. In *Proceedings of the*

IEEE International Conference on Acoustics, Speech, and Signal Processing, volume III, pages 9–12.

Ristivojević, M. and Konrad, J. (2004b). Joint space-time motion-based video segmentation and occlusion detection using multi-phase level sets. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5308, pages 156–167.

Sawhney, H. and Ayer, S. (1996). Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830.

Sethian, J. (1996a). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595.

Sethian, J. (1996b). *Level Set Methods*. Cambridge University Press.

Shi, Y. (2005). *Dynamic imaging with fast level set method*. PhD dissertation, Boston University.

Shi, Y., Konrad, J., and Karl, W. (2004). Multiple motion and occlusion segmentation with a multiphase level set method. In *Proceedings of SPIE Visual Communications and Image Processing*, volume 5308, pages 189–198.

Szeliski, R. and Coughlan, J. (1997). Spline-based image registration. *International Journal of Computer Vision*, 22(3):199–218.

Szeliski, R. and Shum, H.-Y. (1996). Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210.

Taubman, D. and Zakhor, A. (1994). Multirate 3-D subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588.

Thoma, R. and Bierling, M. (1989). Motion compensating interpolation considering covered and uncovered background. *Signal Processing, Image Communication*, 1:191–212.

Tsai, A., Yezzi, A., and Willsky, A. (2000). A curve evolution approach to smoothing and segmentation using the Mumford-Shah functional. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 119–124.

Tsai, Y., Cheng, L., Osher, S., and Zhao, H. (2003). Fast sweeping algorithms for a class of hamilton-jacobi equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694.

Unser, M. (1999). Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38.

Vese, L. and Chan, T. (2002). A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293.

Vetro, A., Sun, H., and Wang, Y. (2001). Object-based transcoding for adaptable video content delivery. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):387–401.

Wang, J. and Adelson, E. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638.

Wang, Y., Ostermann, J., and Zhang, Y.-Q. (2002). *Video Processing and Communciations*. Prentice Hall.

Wright, J. and Pless, R. (2005). Analysis of persistent motion patterns using the 3d structure tensor. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 14–19.

Zhang, J., Gao, J., and Liu, W. (2001). Image sequence segmentation using 3-D structure tensor and curve evolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(5):629–641.

Zhang, J., Modestino, J., and Langan, D. (1994). Maximum-likelihood parameter estimation for unsupervised stochastic model-based image segmentation. *IEEE Transactions on Image Processing*, 3(4):404–420.

Zhu, S. and Yuille, A. (1996). Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900.

CURRICULUM VITAE

Mirko Ristivojević

1125 Commonwealth Avenue, Apt. 28
Allston, MA 02134
mirko@bu.edu
(617) 642-2733
http:/iss.bu.edu/mirko

Vita

Birth year:	1974.
Birthplace:	Šabac, Serbia, Yugoslavia.

Education

Current	**Ph.D. in Electrical and Computer Engineering**, Boston University (GPA: 3.43/4.00). Advisor: Prof. Janusz Konrad; expected graduation: January 2006.
May 2002	**M.S. in Electrical and Computer Engineering**, Boston University (GPA: 3.90/4.00).
November 1999	**Dipl.Ing. in Electrical Engineering**, School of Electrical Engineering, University of Belgrade, Serbia, Yugoslavia (GPA: 8.93/10.00).

Experience

May 2001 -	**Research Assistant** with Prof. Janusz Konrad, Boston University.
Spring 2005	**Teaching Assistant** for Prof. T. Toffoli and R. Giles, Boston University: SC450 Microprocessors.
Summer 2004	**Research visit**, University of Nice at Sophia Antipolis, France.
Summer 2002	**Quality Engineer**, Mathworks Inc., Natick, MA.
2000 - 2001	**Teaching Assistant** for Prof. Tommaso Toffoli, Boston University: SC450 Microprocessors.
1999 - 2000	**Development Engineer**, GVS Company, Belgrade, Yugoslavia.
1993 - 1995	**Teaching instructor** in Electronics, Petnica Science Center, Yugoslavia.

Honors

2000	2000 Graduate Teaching Fellowship at BU.
March 1998	Invited visitor, National Technical University of Athens, Greece.
1993 - 1999	Prize scholarship for college from my home town of Sabac.
1993	Participated in the summer school of science at the Weizmann Institute, Israel as one of the two representatives from Yugoslavia.
1993	Elected to the five-member Yugoslavian team for the International Physics Olympiad.
1993	Second award on Yugoslavian federal Physics Competition.
1989	First award on Yugoslavian federal Physics Competition.

Professional Societies

Member of the IEEE.

Publications

Journal Papers

[1] M. Ristivojević and J. Konrad. **Space-time image sequence analysis: Object tunnels and occlusion volumes.** *IEEE Trans. Image Processing*, pages 364–376, February 2006.

Conference Papers

[1] M. Ristivojević and J. Konrad. **Joint space-time image sequence segmentation: object tunnels and occlusion volumes.** *Proc. IEEE Int. Conf. Accoustic Speech Signal Processing*, vol. III, pp 9-12, Montreal, Canada, May 2004.

[2] M. Ristivojević and J. Konrad. **Joint space-time motion-based video segmentation and occlusion detection using multi-phase level sets.** *Proc. SPIE Visual Communications and Image Processing*, vol. 5308, San Jose, California, January 2004.

[3] J. Konrad and M. Ristivojević. **Video segmentation and occlusion detection over multiple frames.** *Proc. SPIE Image and Video Communications and Process.*, vol. 5202, San Jose, California, January 2003.

[4] J. Konrad and M. Ristivojević. **Joint space-time image sequence segmentation based on volume-competition and level sets.** *Proc. IEEE Int. Conf. Image Processing*, vol. 1, Rochester, New York, September 2002.

Other Publications

[1] M. Ristivojević. **Fingerprint image compression using wavelet transformation and scalar quantization.** Engineering Diploma Thesis (in Serbian), School of Electrical Engineering, University of Belgrade, Serbia, July 1999.