2014

Probabilistic Models for Motion Segmentation in Image Sequences

Manjunath Narayana
University of Massachusetts - Amherst

PROBABILISTIC MODELS FOR MOTION SEGMENTATION IN IMAGE SEQUENCES

A Dissertation Presented

by

MANJUNATH NARAYANA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2014

School of Computer Science

PROBABILISTIC MODELS FOR MOTION SEGMENTATION IN IMAGE SEQUENCES

A Dissertation Presented

by

MANJUNATH NARAYANA

Approved as to style and content by:

Allen Hanson, Co-chair

Erik G. Learned-Miller, Co-chair

Rui Wang, Member

Aura Ganz, Member

Lori A. Clarke, Chair
School of Computer Science

Om.

ACKNOWLEDGMENTS

I would like to thank my advisors Allen and Erik for their guidance and support through the years. Allen's support during the initial years helped me find my areas of interest while Erik's efforts in the later years defined the final shape of my thesis. Their inputs have helped me polish my rough work into research that may be of interest to others. Special thanks to Erik for painstaking efforts into our publications. I have learned a lot in the process of writing papers with him.

Thanks to my committee members Dr. Rui Wang and Dr. Aura Ganz for their feedback on my thesis. My labmates and friends in Amherst - Dima, Gary, Moe, Andrew, Jackie, Laura, Ben, and Partha - Thank you for your company and friendship.

The greatest thanks to my mother Girijamma for her love, support, and sacrifices. My elder brothers Bhanu and Gopi have always been a source of support in my education and otherwise. I am fortunate to have caring sisters-in-law Savithri and Jaya. My nephews and niece - Suchi, Ved, Chaitu, and Raksha are the greatest source of joy.

Thanks to my in-laws Jayaram and Shobha for their support and because my higher studies have meant time away from their daughter for them.

Most of all, thanks to my wife Aruna for being the greatest source of strength and love. The arduous journey has been easier in her company.

Thank God, this is done.

ABSTRACT

PROBABILISTIC MODELS FOR MOTION SEGMENTATION IN IMAGE SEQUENCES

FEBRUARY 2014

MANJUNATH NARAYANA
B.E., B. M. S. COLLEGE OF ENGINEERING, BANGALORE, INDIA
M.S., UNIVERSITY OF KANSAS, LAWRENCE
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Allen Hanson and Professor Erik G. Learned-Miller

Motion segmentation is the task of assigning a binary label to every pixel in an image sequence specifying whether it is a moving *foreground* object or stationary *background*. It is often an important task in many computer vision applications such as automatic surveillance and tracking systems. Depending on whether the camera is stationary or moving, different approaches are possible for segmentation. Motion segmentation when the camera is stationary is a well studied problem with many effective algorithms and systems in use today. In contrast, the problem of segmentation with a moving camera is much more complex. In this thesis, we make contributions to the problem of motion segmentation in both camera settings. First for the stationary camera case, we develop a probabilistic model that intuitively combines the various aspects of the problem in a system that is easy to interpret and extend. In most stationary camera systems, a distribution over feature values for the background at each pixel location is learned from previous frames in the sequence and used for classification in the current frame. These pixelwise models fail to account for the influ-

ence of neighboring pixels on each other. We propose a model that by spatially spreading the information in the pixelwise distributions better reflects the spatial influence between pixels. Further, we show that existing algorithms that use a constant variance value for the distributions at every pixel location in the image are inaccurate and present an alternate pixelwise adaptive variance method. These improvements result in a system that outperforms all existing algorithms on a standard benchmark.

Compared to stationary camera videos, moving camera videos have fewer established solutions for motion segmentation. One of the contributions of this thesis is the development of a viable segmentation method that is effective on a wide range of videos and robust to complex background settings. In moving camera videos, motion segmentation is commonly performed using the image plane motion of pixels, or optical flow. However, objects that are at different depths from the camera can exhibit different optical flows, even if they share the same real-world motion. This can cause a depth-dependent segmentation of the scene. While such a segmentation is meaningful, it can be ineffective for the purpose of identifying independently moving objects. Our goal is to develop a segmentation algorithm that clusters pixels that have similar real-world motion. Our solution uses optical flow orientations instead of the complete vectors and exploits the well-known property that under translational camera motion, optical flow orientations are independent of object depth.

We introduce a non-parametric probabilistic model that automatically estimates the number of observed independent motions and results in a labeling that is consistent with real-world motion in the scene. Most importantly, static objects are correctly identified as one segment even if they are at different depths. Finally, a rotation compensation algorithm is proposed that can be applied to real-world videos taken with hand-held cameras. We benchmark the system on over thirty videos from multiple data sets containing videos taken in challenging scenarios. Our system is particularly robust on complex background scenes containing objects at significantly different depths.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

Motion is an important source of information in the world. The ability to detect motion is one of the most important functions performed in biological vision systems [3]. This ability in animals can be critical for their survival in the wild. For example, being able to detect and identify biological motion can help an animal recognize that a predator is approaching [66]. In many cases, otherwise ambiguous stimuli can be better understood when motion is considered. For instance, as shown in Figure 1.1, a well camouflaged animal can be mistaken for the foliage that it imitates and can be difficult to detect unless it starts moving. Because moving stimuli often represent biologically important objects, visual processing in animals has evolved to handle motion effectively [15].

The ability of animals to perceive motion has been the subject of much research [33, 14, 42, 12, 13, 66]. Motion perception studies have shown that humans are able to reliably recognize the animal subjects simply by observing the motion of the subjects' joints. Visualization of the motion of joints by the use of lights on the joints of subjects who are otherwise invisible due to dark clothing in a dark environment [33](see Figure 1.2) has led to several studies that highlight the importance of motion analysis in human vision systems [14, 42]. Ullman [73] showed that dots on a cylinder when projected on an image appeared random to observers when static, but were perceived as a cylinder when set in motion (Figure 1.3). Wallach and O'Connell [74] demonstrated that when unfamiliar objects were rotated behind a translucent screen and their shadows cast on the screen, viewers were able to describe the objects in motion. The static shadows were not recognizable, which shows the importance of motion segmentation and processing. Research from cognitive science has shown that the human visual system performs segmentation by combining motion cues with spatial information [20, 51]. Motion is hence a very important factor in our visual perception of the world around us.

Akin to animal vision systems, motion is very important in computer vision systems as well. The same cues that biological visual systems derive from motion are also useful in artificial vision systems. Many vision applications are built using motion processing as an

Figure 1.1. The camouflaged insect is very difficult to spot. If it were moving, it would be much easier to detect and identify. *image source: http:/www.indiastudychannel.com/resources/98707-Camouflage-The-Invisible-Illusion.aspx*

integral part of larger systems. One of the most common examples of motion processing is the grouping of image pixels into regions based on their motion. These regions often correspond to the different objects in the scene and such grouping can help in higher level tasks such as identifying the objects.

In this thesis, we address the problem of labeling of each pixel in an image sequence as belonging to either moving or stationary objects in the scene, or *motion segmentation*. Such a labeling is useful because objects that have moved tend to be involved in some interesting activity. Motion segmentation can help in vision systems by allowing computationally expensive algorithms to be applied to a much smaller region within an image instead of the entire image.

Figure 1.4 illustrates a typical pipeline in a video surveillance system that uses motion segmentation. An input video stream is typically processed using motion segmentation algorithms to identify potential moving regions in the scene. Subsequently, these moving regions may be processed to remove false detections and to classify the objects that are present in the genuinely moving regions. These objects may then be tracked and further analyzed to recognize the activities being performed. Although visualized as a linear pipeline, there is often feedback and significant interaction between the modules in real implementations of such systems. Processing of other cues such as color, texture, shape, etc. is also extremely useful in vision systems but not shown in the simple pipeline.

An illustration of the input and output at the various stages is given in Figure 1.5. An input frame from the video sequence after processing yields a mask which marks the

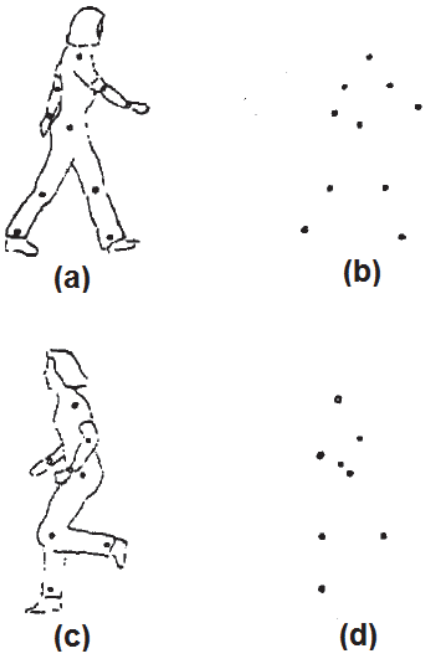

Figure 1.2. Figures (a) and (c) shows subjects with lights attached to their joints. Figures (b) and (d) show only the light points which were the input for various studies that show that humans can identify the subjects and their actions by looking at the motion of these points alone. *image source: Johansson [33]*

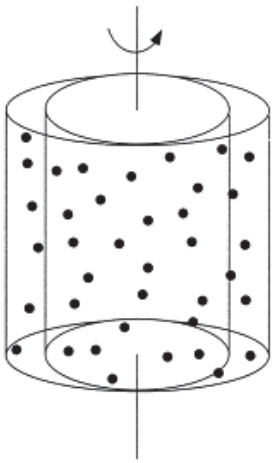

Figure 1.3. Ullman [73] showed that humans could infer the cylindrical surface from the motion of the projected dots on an image. *image source : Caudek and Rubin [7]*

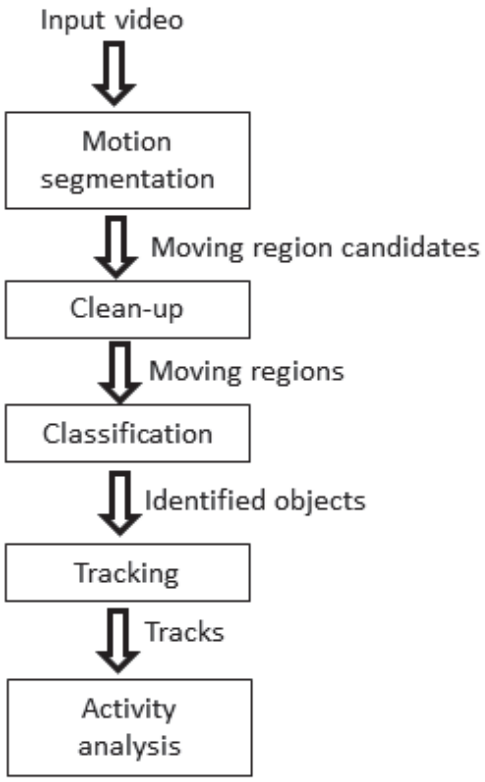

Figure 1.4. A typical pipeline for video surveillance and analysis

moving regions in the image. Small moving regions are removed during the clean-up phase (c). Various objects may then be identified and marked by bounding boxes while tracking them across many frames (d).

1.1 Motion segmentation with a stationary camera

Motion segmentation for stationary cameras is a well-studied problem with many good solutions. The most common application of motion segmentation is in visual surveillance systems. Motion segmentation followed by selective processing, storage of information, and automatic analysis has led to the ubiquitous use of cameras for surveillance. Although the emphasis of this thesis is not on visual surveillance, a short discussion of several surveillance systems highlights the application of motion segmentation in many widely deployed vision systems.

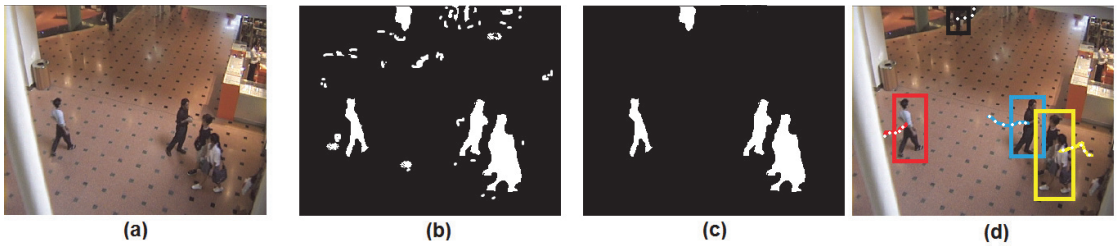

Figure 1.5. Figure (a) shows the input frame that is being processed. Figure (b) shows the result of a motion segmentation algorithm with white pixels corresponding to pixels that belong to moving objects. Removing small regions which are likely to be noise results in a relatively cleaner result in (c). Figure (d) visualizes the detected objects with bounding boxes. The detected objects may be tracked across multiple frames, visualized using dotted lines.

Pfinder [77] is a real-time system for tracking people and analyzing their behavior. Designed for analyzing a single person at a time with a stationary camera setup, the system uses motion segmentation to identify image regions on which detailed analysis is performed to locate the person and his/her body parts. More complex applications such as recognizing the gestures of the detected persons are enabled by the system.

W4 [24] is a surveillance system for detecting and tracking people to monitor their activities in an outdoor environment. Motion segmentation is performed by building a statistical model of the scene at each pixel. Moving regions are then classified into "people" or "other" objects for subsequent tracking, body pose estimation, and detection of objects that people may be carrying. Hydra [23] extends W4 to detect and analyze regions that consist of multiple people when they appear in a group.

Visual Surveillance and Monitoring (VSAM) [9] is another surveillance system and a multi-year testbed consisting of multiple cameras in different locations that are analyzed in a single control room on different monitors. Moving regions are classified into "people" and "vehicles". Actions such as "appearing", "moving", and "stopped" and interactions between objects such as "moving away from" and "moving towards" are then automatically determined.

The ADVISOR system [61] is a surveillance system designed for analyzing crowd behavior in train stations with the aim of using such analysis to enhance public safety. IBM S3 system [60] describes a system architecture with a focus on the storage and management of the video data for the purpose of behavior analysis and event-based retrieval.

More recent extensions to the above systems include multiple cameras which coordinate amongst themselves to achieve surveillance over a larger geographic area. As part of the VSAM project, extensions to multiple-camera systems are discussed by Collins *et al.* [10]. A system for surveillance in parking lots using a network of cameras is described by Micheloni *et al.* [44]. Nguyen *et al.* [49] discuss coordination between cameras for an indoor surveillance application. Algorithms for detection and tracking of objects in a network of cameras require special considerations. Some approaches for multi-camera systems are presented by Javed *et al.* [31, 30].

Thus, we see that a large number of systems have been developed and deployed for video analysis for stationary camera settings. Most existing systems use motion segmentation in some form to achieve the eventual goals of object identification and activity analysis.

1.2 Motion segmentation with a moving camera

Motion segmentation when the camera is moving is much more challenging technically. In comparison with stationary cameras, the main challenge with moving cameras is illustrated in Figure 1.6. Two subsequent frames and the difference between the intensities of the two frames at each pixel are shown in the figure. The first row is an example from a stationary camera video and the second row is from a moving camera video. We can see from the difference images that for the stationary camera, most of the background pixels exhibit very little change from one frame to the next. This observation is the basis for building statistical models for the background based on observed pixel intensities or color values. Pixels that deviate from the constructed statistical model at each pixel location are strong candidates for moving pixels. On the contrary, with a moving camera, both the background and moving objects exhibit a high difference in intensity values from one frame to the next. Before a statistical model can be built for each pixel, the two consecutive frames must be adjusted so that the background pixels in both frames are in correspondence [29, 25, 55]. This process is not straightforward and often not reliable enough to obtain a good model for the background.

Compared to the large number of deployed systems with stationary cameras, automatic moving camera systems are far less pervasive. Effective methods for motion segmentation for a moving camera can perhaps unlock the potential for many useful applications of the future. Moving camera segmentation algorithms can be useful in many interesting

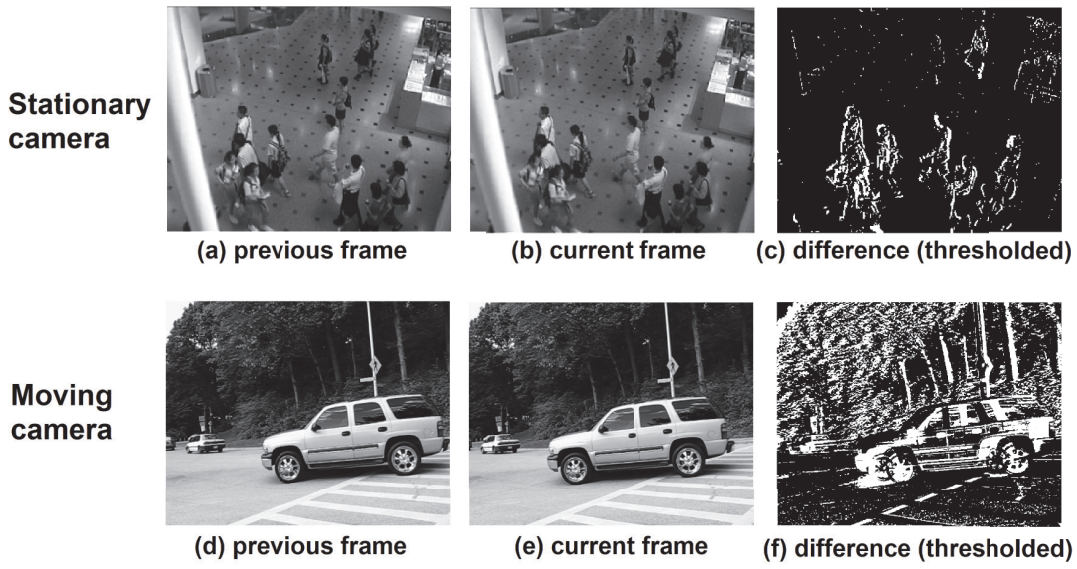

Figure 1.6. The first row shows a stationary camera example and the second row shows a moving camera example. Figures (a) and (d) are the previous frames. Figures (b) and (e) correspond to the current frame. Figure (c) is the difference between (a) and (b) with pixels that have a difference value larger than 10 colored in white. Most background pixels for the stationary camera case have a difference value less than 10. Figure (f) is the corresponding difference between (d) and (e). For the moving camera case, many background pixels have a difference value larger than 10.

platforms such as hand-held cameras, cameras on autonomous robots, and cameras on automobiles.

1.3 Algorithms for motion segmentation

Depending on whether the camera is stationary or moving, different algorithms may be used for motion segmentation. In this thesis, we present novel and effective algorithms for motion segmentation for both stationary and moving camera settings.

1.3.1 Stationary camera algorithms and our contributions

For a stationary camera such as a stationary surveillance camera in a building, one may assume that all the points on non-moving objects in the scene always maps to the same pixel location in the captured image for every frame in the video. Also called *background modeling*, the problem is extensively studied in the literature [69, 62, 17, 81, 40, 58, 26]. In its early implementations, background modeling was a process of building a model for the background of a video with a stationary camera, and identifying pixels that did not

conform well to this model. The pixels that were not well-described by the background model were assumed to be moving objects. Many systems today maintain models for the moving objects or *foreground* as well as the background, and these models compete to explain the pixels in a video. If the foreground model explains the pixels better, they are considered foreground. Otherwise they are considered background. In the first part of this thesis, we argue that the logical endpoint of this evolution is to simply use Bayes' rule to classify pixels. In particular, it is essential to have a background likelihood, a foreground likelihood, and a prior at each pixel. A simple application of Bayes' rule then gives a posterior probability over the label.

The only remaining question is the quality of the component models: the background likelihood, the foreground likelihood, and the prior. We describe a model for the likelihoods that is built by using not only the past observations at a given pixel location, but by also including observations in a spatial neighborhood around the location. This enables us to model the influence between neighboring pixels and is an improvement over earlier pixelwise models that do not allow for such influence. Further we use a spatially dependent prior for the background and foreground. The background and foreground labels from the previous frame, after spatial smoothing to account for movement of objects, are used to build the prior for the current frame.

These components are, by themselves, not novel aspects in background modeling. As we will show, many existing systems account for these aspects in different ways. We argue that separating these components, as suggested in this thesis, yields a very simple and effective model. Our intuitive description also isolates the model components from the classification or inference step. Improvements to each model component can be carried out without any changes to the inference or other components. The various components can hence be modeled effectively and their impact on the overall system understood more easily.

We discuss one such improvement in the likelihood estimation. We show that the use of a constant variance for all pixels in the image is not optimal and present a pixelwise adaptive variance method that results in higher segmentation accuracy. Our improvements result in a system that outperforms all existing algorithms on a standard benchmark data set.

1.3.2 Moving camera algorithms and our contributions

Motion segmentation with a moving camera is a more challenging problem. It is also more interesting because moving camera motion segmentation algorithms can be applied in a wide array of applications including robotics, automobile camera systems, and hand-held video cameras. Simple pixelwise models from stationary camera algorithms are inadequate for moving camera videos because both stationary and moving objects in the scene map to different pixel locations in each frame. For moving cameras, a common approach for motion segmentation involves computing the velocities of pixels in the image plane or *optical flow* [27]. Segmentation may be performed by clustering pixels based on the optical flow vectors. An alternative to using optical flow estimates is to track interesting points in the images and use long term trajectory information to model the motion observed in the background pixels. Several approaches combine motion information with color appearance information to perform motion segmentation. Chapter 4 describes these algorithms and their shortcomings.

We propose to perform motion segmentation with two significant differences from existing approaches. Firstly, we use the orientation of the optical flow vectors, instead of the complete vectors or their magnitudes. For camera translation, the orientation of optical flow vectors are invariant to the depth of the objects in the scene. Thus, in comparison to object depth-dependent optical flow magnitudes which could result in characterizing the background objects at different depths as separate entities, the optical flow orientations result in a more robust representation. This is illustrated in Figure 1.7. Secondly, we consider the 3-d motion of the camera and make use of the strong constraints that motion enforces on optical flow orientations.

In order to perform segmentation, we estimate the 3-d motions that best explain the observed orientations in the image sequence. The 3-d motion that best explains a majority of the pixels in the image is considered the camera's motion and the pixels that are consistent with the camera motion are classified as background or non-moving. We introduce a probabilistic model that automatically estimates the number of independently moving objects, computes the motion parameters for the camera as well as the moving objects, and labels the image pixels as either background or foreground.

The assumption of pure translation motion in the camera is often violated in real videos. To address the issue of rotation in the camera, we present an algorithm for rotation compensation. Camera rotation is handled by using the property that the optical flow due to a camera's rotation does not depend on the scene structure. Similarly to optical flow orienta-

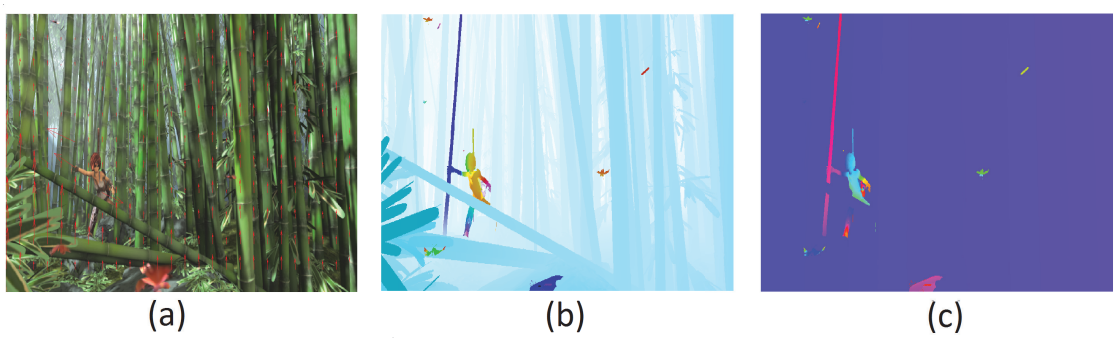

Figure 1.7. Illustration of optical flow orientations. (a) A forest scene with a moving person (from Sintel [6] data set). The person is holding on to a bamboo tree, which moves with the person. There are also a few leaves falling in the scene. (b) Visualization of the optical flow vectors (using code from [63]). (c) Orientation of the optical flow vectors. The optical flow vectors and magnitudes on the trees depend on the distance of the trees from the camera. The orientations are not depth-dependent and can much more reliably predict that all the trees are part of the coherently moving background entity.

tions for camera translation, optical flow vectors due to camera rotation are independent of the depth of objects in the scene. Given the camera's rotation parameters, rotational optical flow vectors can be predicted exactly. We iteratively estimate the translation and rotation parameters such that the error between the translational flow orientations and the orientations of the observed flow vectors, after the rotational flow components have been removed from them, is minimized. With the rotation component removed, the assumption of pure translation holds good and results in reliable segmentation in videos where the original algorithm failed due to camera rotation.

Our system is shown to work on more than thirty videos from multiple benchmark data sets. Ours is the first system to report results on such a large set of videos. In comparison to existing approaches, our system does not require any special initialization step to detect objects in the first frame or motion information from multiple frames to achieve reliable segmentation. The system presents a viable approach for motion segmentation in complex background scenarios.

1.3.3 Contributions of the thesis

The main contributions of this thesis are:

1. Description of a probabilistic background model that intuitively combines location-specific priors and spatially smoothed appearance likelihoods for the background and foreground.

2. Improvements to stationary camera motion segmentation algorithms by using pixel-wise adaptive kernel variances resulting in the state of the art accuracy on benchmark data.

3. Development of an accurate moving camera motion segmentation algorithm that addresses current challenges in segmentation by using optical flow orientations in a non-parametric probabilistic model.

4. Introduction of a rotation compensation algorithm that enables the application of the segmentation algorithm to videos that contain both translation and rotation motion of the camera.

The thesis is organized as follows. Chapter 2 discusses algorithms for motion segmentation in a stationary camera setting and our complete probabilistic model where the various components are intuitively separated. In chapter 3, we address the important problem of variance selection for background modeling. Chapter 4 describes our model that uses optical flow orientations for motion segmentation in moving camera systems. Modeling of camera rotation for the purpose of segmentation is discussed in Chapter 5. We conclude with a discussion in Chapter 6.

1.3.4 Ambiguity in motion segmentation

Although motion segmentation is defined as the labeling of pixels as either moving or stationary, in many scenarios, it is an ill-defined problem. A few examples where a human may or may not consider the object to be moving are:

1. Leaves and branches waving in the wind

2. Waves on a water surface

3. Shadows cast by moving objects on stationary surfaces

4. Reflections on shiny surfaces

5. Periodic motion or vibrations on objects such as escalators

6. Occasional motion in objects such as curtains

7. Natural phenomena such as snow fall, rain, and waterfalls

8. A moving object that has stopped for a brief period, such as a car at an intersection.

It is not easy to arrive at a definition of motion segmentation that encompasses all ambiguous scenarios like those listed above. Our system identifies objects as moving if their physical location in the scene has changed more than was expected under the statistical model built from the history of the scene.

CHAPTER 2

A COMPLETE MODEL FOR MOTION SEGMENTATION IN STATIONARY CAMERA SYSTEMS

2.1 Introduction

Motion segmentation for stationary camera videos, also called background subtraction, is a well researched problem. Algorithms have evolved from early approaches modeling the background at each pixel [77, 69, 62, 17] to methods that include an explicit model for the foreground [40, 58], and finally to more recent models that incorporate spatial dependence between neighboring pixels [58].

In early algorithms [77, 62], a probability distribution $p_{\mathbf{x}}(\mathbf{c}|\mathtt{bg})$ over *background* colors $\mathbf{c}$ is defined and learned for each location $\mathbf{x}$ in the image. These distributions are essentially the background likelihood at each pixel location. Pixels that are well explained by the background likelihood are classified as *background* and the remaining pixels in the image are labeled as *foreground*. Toyama *et al.* [69] use a Weiner filter to predict the intensities of the background pixels in the current frame using the observed values from the previous frames and to identify non-conforming pixels as foreground. Wren *et al.* [77] model the background as a Gaussian distribution at each pixel. To account for the multiple intensities often displayed by background phenomena such as leaves waving in the wind or waves on water surfaces, Stauffer and Grimson [62] learn a parametric mixture of Gaussians (MoG) model at each pixel. The MoG model update procedure as described by Stauffer and Grimson can be unreliable during initialization when not enough data have been observed. To improve the performance during model initialization, Kaewtrakulpong and Bowden [35] suggest a slightly different model update procedure. Porikli and Tuzel [52] obtain the background likelihood by using a Bayesian approach to model the mean and variance values of the Gaussian mixtures. Elgammal *et al.* [17] avoid the drawbacks of using a parametric MoG model by instead building the background likelihoods with non-parametric kernel density estimation (KDE) using data samples from previous frames in history.

While they are still called "backgrounding" systems, later systems maintain a model for the foreground as well as the background [40, 58]. Explicit modeling of the foreground has

been shown to improve the accuracy of background subtraction [58]. In these models, pixel labeling is performed in a competitive manner by labeling as foreground the pixels that are better explained by the foreground model. The remaining pixels are labeled as background.

Although it is natural to think about priors along with likelihoods, the use of an explicit prior for the background and foreground is less common. In the object tracking literature, Aeschliman *et al.* [2] use priors for the background and foreground objects for segmentation of tracked objects. In background modeling algorithms that do not explicitly model the prior, the foreground-background likelihood ratio is used for classification. Pixels that have a likelihood ratio greater than some predefined threshold value are labeled as foreground. This method is equivalent to using an implicit prior that is the same at all pixel locations.

Thus, existing algorithms make use of some subset of the three natural components for background modeling - the background likelihood, the foreground likelihood, and the prior. They make up for the missing components by including effective model-specific procedures at the classification stage. For instance, Elgammal *et al.* [17] and Stauffer and Grimson [62] use only the background likelihood, but, during classification, consider a likelihood threshold below which pixels are considered as foreground. Zivkovic [81] applies Bayes' rule to the problem of computing background posteriors, but since neither the foreground likelihood nor the priors are explicitly modeled, the classification is essentially based on a threshold on background likelihood values. Sheikh and Shah [58] utilize both foreground and background likelihoods, but do not use an explicit prior. Instead, by using a foreground-background likelihood ratio as the classification criterion, they effectively use a uniform prior.

We argue that the logical endpoint of the model evolution for backgrounding is a system where all three components are explicitly modeled and Bayes' rule is applied for classification. Such a system has the advantage of being a simpler model where the modeling of the individual components is isolated from the inference step. This separation allows us to describe the components without any relation to the classification procedure. Our motivation behind this approach is that the components can individually be improved, as we will show in later sections, without affecting each other or the final inference procedure.

In this chapter of the thesis, the components of our background system are described and placed in the context of existing algorithms where possible. Section 2.2 discusses the evolution of the background likelihood models and our improvements to the most successful models. In Section 2.3, we discuss the modifications to the likelihood for modeling the foreground. Modeling of the prior is described in Section 2.4. Computation of posterior

probabilities by using the above components is explained in Section 2.5. Results comparing our system to earlier methods on a benchmark data set are given in Section 2.6.

2.2 Background likelihood

The background likelihood, which is a distribution over feature values, is a common aspect among many backgrounding systems. Stauffer and Grimson [62] model the background likelihood at each pixel using a mixture of Gaussians (MoG) approach. The requirement of specifying the number of mixture components in the MoG model is removed in the non-parametric kernel density estimation (KDE) model [17]. In the KDE model, the distributions at each pixel location are estimated by summing up contributions from the observed background data samples at that location from previous frames in history. For each pixel location $\mathbf{x} = [x, y]$, both these models maintain a distribution $p_{\mathbf{x}}(\mathbf{c})$ that is independent of the neighboring pixels. Here, $\mathbf{c} = [r, g, b]$ is a vector that represents color. These neighbor-independent distributions have the drawback of not being able to account for the influence of neighboring pixels on each other's color distributions.

To allow neighboring pixels to influence the background likelihood at a given pixel location, we model the likelihood at a particular pixel location to be a weighted sum of distributions from its spatial neighbors. Our *smoothed* background likelihood $P_{\mathbf{x}}(\mathbf{c})$ for each pixel location $\mathbf{x}$ is a weighted sum of distributions from a spatial neighborhood $\mathcal{N}_{\mathcal{B}}$ around $\mathbf{x}$. Each neighboring likelihood is weighted by its spatial distance (i.e., distance in the image coordinates) from $\mathbf{x}$:

$$P_{\mathbf{x}}(\mathbf{c}|\mathtt{bg}; \Sigma^{\mathrm{B}}_{\mathbf{S}}) = \frac{1}{\mathrm{Z}} \sum_{\Delta \in \mathcal{N}_{\mathcal{B}}} \mathrm{p}_{\mathbf{x}+\Delta}(\mathbf{c}|\mathtt{bg}) \times \mathrm{G}(\Delta; \mathbf{0}, \Sigma^{\mathrm{B}}_{\mathbf{S}}). \tag{2.1}$$

Here Δ is a spatial displacement that defines a spatial neighborhood $\mathcal{N}_{\mathcal{B}}$ around the pixel location $\mathbf{x}$ at which the likelihood is being computed. $G(\cdot; \mathbf{0}, \Sigma^{B}_{\mathbf{S}})$ is a zero-mean multivariate Gaussian with covariance $\Sigma^{B}_{\mathbf{S}}$. B indicates that the covariance is for the background model and $\mathbf{S}$ denotes the spatial dimensions. The normalization constant Z is

$$Z = \sum_{\Delta \in \mathcal{N}_{\mathcal{B}}} G(\Delta; \mathbf{0}, \Sigma^{B}_{\mathbf{S}}). \tag{2.2}$$

The weighted sum results in a spatial smoothing of the distributions as shown in Figure 2.1. This spreading of information is useful in modeling spatial uncertainty of background

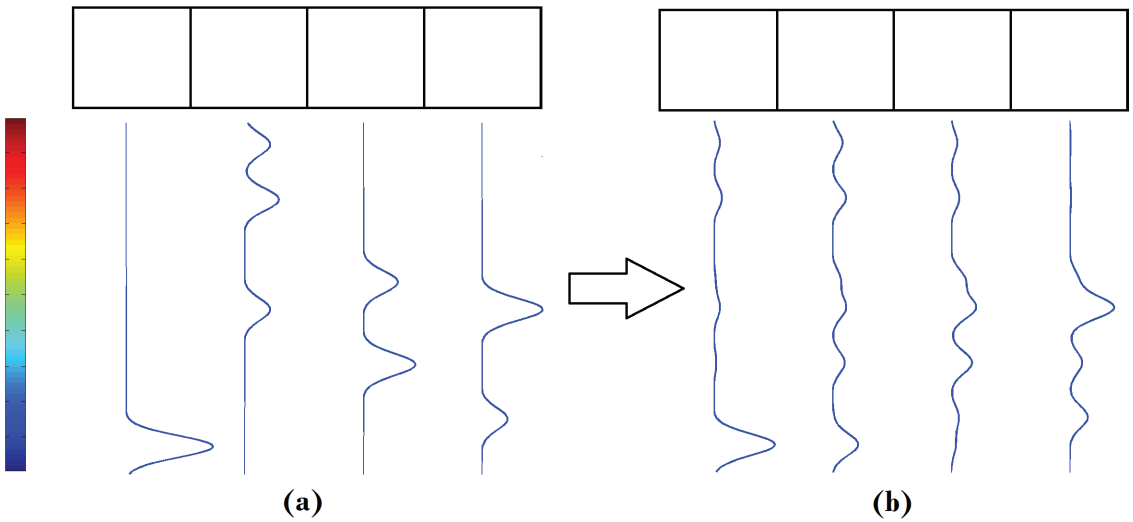

Figure 2.1. Influence of neighboring pixels on each other is modeled by spreading information spatially. Figure (a) shows some example likelihoods for each pixel in a single-dimensional (row) image. The distributions shown below each pixel are the estimated background likelihoods. The vertical axis corresponds to color values which are visualized in the color map on the left side of the image. The horizontal axis corresponds to the probability of the corresponding color. Figure (b) shows the smoothed likelihood at each pixel, which is a weighted sum of the likelihoods in the pixel's neighborhood. The effect of smoothing is clearly visible in the first pixel. The distribution in the first pixel clearly influences the distributions at the second and third pixels. The distance-dependent nature of the weights results in the first pixel influencing the third pixel less than it does the second pixel.

pixels. $\Sigma_{\mathbf{S}}^{B}$ controls the amount of smoothing and spreading of information in the spatial dimensions.

Explicitly maintaining a distribution at each pixel location is impractical for color features which can take one of 256^3 values if each of the three color channels have a range between 0 and 255. Instead, we compute likelihoods with KDE using the data samples from the previous frame. Let $\mathbf{b}_{\mathbf{x}}^{t-1}$ be the observed background color at pixel location $\mathbf{x}$ in the previous frame. Using a Gaussian kernel with covariance $\Sigma_{\mathbf{C}}^{B}$ in the color dimensions, our KDE background likelihood in the current frame (at time t) is given by

$$P_{\mathbf{x}}^{t}(\mathbf{c}|\mathsf{bg};\Sigma_{\mathbf{C}}^{\mathrm{B}},\Sigma_{\mathbf{S}}^{\mathrm{B}}) = \frac{1}{\mathrm{Z}}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}} \mathrm{G}(\mathbf{c}-\mathbf{b}_{\mathbf{x}+\Delta}^{\mathrm{t}-1};\mathbf{0},\Sigma_{\mathbf{C}}^{\mathrm{B}}) \times \mathrm{G}(\Delta;\mathbf{0},\Sigma_{\mathbf{S}}^{\mathrm{B}}). \tag{2.3}$$

Figure 2.2 illustrates the process of computing the background likelihood using the observed background colors in one image. It may be noted that the covariance matrix $\Sigma_{\mathbf{S}}^{B}$

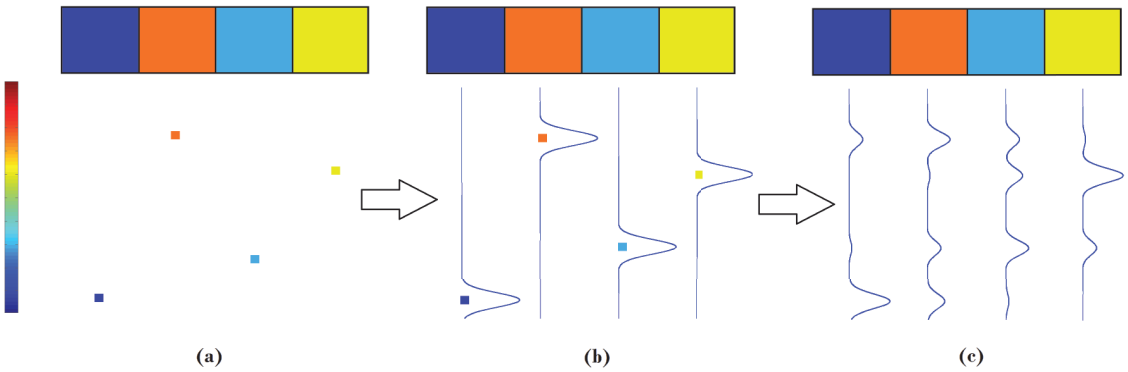

Figure 2.2. Modeling the likelihoods using pixel data samples and KDE. Figure (a) shows the colors at each pixel. The corresponding color and its location with respect to the vertical color axis is shown under each pixel. Figure (b) shows the likelihood at each pixel estimated using KDE with a Gaussian kernel. Figure (c) shows the effect of spatial smoothing of the KDE-based likelihoods. Again, the illustration uses a one-dimensional row image in which a pixel's color is also represented in one dimension. It is straightforward to extend the example to two-dimensional spatial coordinates and three-dimensional color space.

controls the amount of spatial influence from neighboring pixels. The covariance matrix $\Sigma_{\mathbf{C}}^{B}$ controls the amount of variation allowed in the color values of the background pixels.

Finally, we consider background data samples not just from the previous frame, but from the previous T frames in order to obtain a more accurate likelihood. We also allow probabilistic contribution from the previous frames' pixels by weighting each pixel according to its probability of belonging to the background:

$$P_{\mathbf{x}}^{t}(\mathbf{c}|\mathtt{bg};\Sigma^{\mathrm{B}})=\frac{1}{\mathrm{K}_{\mathtt{bg}}}\sum_{\mathrm{i}\in 1:\mathrm{T}}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}}\mathrm{G}(\mathbf{c}-\mathbf{b}_{\mathbf{x}+\Delta}^{\mathrm{t-i}};\mathbf{0},\Sigma_{\mathbf{C}}^{\mathrm{B}})\times\mathrm{G}(\Delta;\mathbf{0},\Sigma_{\mathbf{S}}^{\mathrm{B}})\times\mathrm{P}_{\mathbf{x}}^{\mathrm{t-i}}(\mathtt{bg}|\mathbf{b}_{\mathbf{x}+\Delta}^{\mathrm{t-i}}). \quad (2.4)$$

Σ^{B} represents the covariance matrices for the background model and consists of the color dimensions covariance matrix $\Sigma_{\mathbf{C}}^{B}$ and the spatial dimensions covariance matrix $\Sigma_{\mathbf{S}}^{B}$. $P_{\mathbf{x}}^{t}(\mathtt{bg}|\mathbf{b}_{\mathbf{x}}^{\mathrm{t}})$ is the probability that pixel at location $\mathbf{x}$ in the frame t is background. $K_{\mathtt{bg}}$ is the appropriate normalization factor:

$$K_{\mathtt{bg}}=\sum_{i\in 1:T}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}}G(\Delta;\mathbf{0},\Sigma_{\mathbf{S}}^{B})\times P_{\mathbf{x}}^{t-i}(\mathtt{bg}|\mathbf{b}_{\mathbf{x}+\Delta}^{\mathrm{t-i}}). \quad (2.5)$$

For efficiency, we restrict the covariance matrices to be diagonal and hence parameterize them by their diagonal elements.

2.2.1 Existing work on spatial smoothing of distributions

The use of spatial smoothing of distributions is not entirely new. Sheikh and Shah [58] use a joint domain-range model that combines the pixels' position values and color observations into a joint five-dimensional space. By modeling the likelihoods in the joint space, they allow pixels in one location to influence the distribution in another location. Their background likelihood is defined as:[1]

$$P^t(\mathbf{c},\mathbf{x}|\mathtt{bg};\Sigma^{\mathrm{B}}) = \frac{1}{\mathrm{K}}\sum_{\mathrm{i}\in 1:\mathrm{T}}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}}\mathrm{G}(\mathbf{c}-\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta};\mathbf{0},\Sigma^{\mathrm{B}}_{\mathbf{C}})\mathrm{G}(\Delta;\mathbf{0},\Sigma^{\mathrm{B}}_{\mathbf{S}})\mathrm{P}^{\mathrm{t-i}}_{\mathbf{x}}(\mathtt{bg}|\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta}). \quad (2.6)$$

The normalization constant, K, is given by

$$K = \sum_{i\in 1:T}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}} P^{t-i}_{\mathbf{x}}(\mathtt{bg}|\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta}). \quad (2.7)$$

The key difference between their model and ours is that theirs is, for the entire image, essentially a *single* distribution in the joint domain-range space whereas ours consists of a different location-dependent distribution at each pixel. This difference has a big effect on the classification stage. As we will see later, their classification criterion, based on the ratio of foreground and background likelihoods in this five-dimensional space, has an undesirable dependence on the size of the image. By replacing the single joint distribution with a *field* of distributions dependent on image location, we avoid the dependence on image size and achieve better results.

The joint domain-range model has been used earlier in the object tracking literature. Elgammal *et al.* [16] use a joint domain-range model that is almost identical to the background model of Sheikh and Shah [58]. A scheme very similar to our Equation 2.1 was used in a tracking system by Han and Davis [22] to interpolate the pixelwise appearance distributions for an object whose size has changed during the tracking process. The close resemblance between these models suggests that tracking and background modeling share similar fundamental principles and can be achieved under the same framework. One such framework that integrates segmentation and tracking has been described by Aeschliman *et al.* [2].

---

[1] We have modified their equation to allow probabilistic contributions from the pixels and changed the notation to make it easily comparable to ours.

Ko *et al.* [36] use a histogram-based variant of the Sheikh and Shah [58] background model which is built from observations in a spatial neighborhood around each pixel from previous frames in history. However, they do not consider the spatial distance between a pixel and its neighbor when summing up the contributions. In addition, they build another distribution, which can be interpreted as the "texture" at each pixel, by using only the current frame observations in each pixel's spatial neighborhood. Their classification criterion for foreground pixels is to threshold the Bhattacharya distance between the background distribution and the "texture" distribution. Our model is different because of our classification criterion that uses foreground likelihoods and explicit priors for the background and foreground which we discuss in subsequent sections.

2.3 Foreground likelihood

Explicit modeling of the foreground likelihood has been shown to result in more accurate systems [40, 58]. Our foreground likelihood is very similar to our background likelihood. However, it is important to consider in the foreground likelihood the possibility of hitherto unseen color values appearing as foreground. This may happen because a new foreground object enters the scene or an existing foreground object either changes color or, by moving, exposes a previously unseen part of it. We find it useful to separate the foreground process into two different sub-processes: previously seen foreground, which we shall refer to as *seen* foreground, and previously unseen foreground, which we shall refer to as *unseen* foreground. The likelihood for the seen foreground process is computed using a KDE procedure similar to the background likelihood estimation:

$$P^t_{\mathbf{x}}(\mathbf{c}|\mathtt{fg};\Sigma^{\mathrm{F}}) = \frac{1}{\mathrm{K}_{\mathtt{fg}}} \sum_{\mathrm{i}\in 1:\mathrm{T}} \sum_{\Delta\in\mathcal{N}_{\mathcal{F}}} \mathrm{G}(\mathbf{c} - \mathbf{f}^{\mathrm{t-i}}_{\mathbf{x}+\Delta}; \mathbf{0}, \Sigma^{\mathrm{F}}_{\mathbf{C}}) \times \mathrm{G}(\Delta; \mathbf{0}, \Sigma^{\mathrm{F}}_{\mathbf{S}}) \times \mathrm{P}^{\mathrm{t-i}}_{\mathbf{x}}(\mathtt{fg}|\mathbf{f}^{\mathrm{t-i}}_{\mathbf{x}+\Delta}). \quad (2.8)$$

Similar to Equation 2.4, $\mathbf{f}^t_{\mathbf{x}}$ is the observed foreground color at pixel location $\mathbf{x}$ in frame t. Σ^F is the covariance matrix for the foreground model, and $K_{\mathtt{fg}}$ is the normalization factor, analogous to $K_{\mathtt{bg}}$. $P^t_{\mathbf{x}}(\mathtt{fg}|\mathbf{f}^{\mathrm{t}}_{\mathbf{x}})$ is the probability that pixel at location $\mathbf{x}$ in the frame t is foreground.

Since foreground objects typically move more than background objects in stationary camera videos and also exhibit more variation in their color appearance, we typically use higher covariance values for the foreground than for the background. Compared to the

background, we use a larger spatial neighborhood for the foreground to account for larger expected motion in the foreground.

The likelihood for the unseen foreground process is simply a uniform distribution over the color space.

$$P^t_{\mathbf{x}}(\mathbf{c}|\mathtt{fu}) = \frac{1}{\mathrm{R} \times \mathrm{G} \times \mathrm{B}} \tag{2.9}$$

for all locations $\mathbf{x}$ in the image, where R, G, and B, are the number of possible intensity values for red, green, and blue colors respectively.

The unseen foreground process constantly tries to account as foreground any colors not reasonably explained by both the background and the seen foreground likelihood.

The concept of using a uniform likelihood is not new. For instance, Sheikh and Shah [58] mix a uniform distribution (in five-dimensional space) to their foreground likelihoods to explain the appearance of new foreground colors in the scene. Separation of the foreground process into two sub-processes, as we have done, is equivalent to the mixing of the likelihoods into one combined likelihood. The advantage of considering them as separate sub-processes is that when combined with a separate prior for each, greater modeling flexibility can be achieved. For instance, at image boundaries where new objects tend to enter the scene, a higher prior can be used for the unseen foreground process.

2.4 Priors

In addition to modeling the likelihoods, we explicitly model spatially varying priors for the background and foreground processes. Such spatial priors have recently been used for segmentation of objects being followed in a tracking algorithm [2]. Background modeling systems that use a likelihood ratio as the classification criterion are implicitly assuming a uniform prior for the entire image. In such systems, if the foreground-background likelihood ratio at a given pixel is greater than some predefined threshold L, then the pixel is labeled as foreground. Using a value of 1 for L means that the background and foreground processes have a uniform and equal prior value at every pixel location. Other values of L imply using a uniform but unequal prior for the background and foreground.

We generalize the notion of the prior by considering a spatially varying prior. The uniform prior is simply a special case of our model. We define pixelwise priors for the three processes involved - background, previously seen foreground, and unseen foreground. The classified pixel labels from the previous frame are used as a starting point for building the priors for the current frame. We assume that a pixel that is classified as background in the

previous frame has a 95% probability of belonging to the background in the current frame as well. The pixel has a 2.5% probability of belonging to a seen foreground object, and a 2.5% probability of coming from a previously unseen foreground object. For a foreground pixel in the previous frame, we assume that due to object motion, there is a 50% probability of this pixel becoming background, a 25% probability of this pixel belonging to the same foreground object as in the previous frame, and a 25% probability that it becomes a new unseen object. The choice of these values is discussed later.

There are hence essentially two settings for the prior at each pixel depending on whether the pixel was labeled background or foreground in the previous frame. Instead of using the hard thresholds described above, we use the pixel's background label probability from the previous frame when computing the prior. For instance, a pixel that has probability p of being background in the previous frame will have a background prior equal to $p \times .95 + (1-p) \times .5$. Also, since objects typically move by a few pixels from the previous frame to the current frame, we apply a smoothing (7×7 Gaussian filter with a standard deviation value of 1.75) to the classification results from the previous frame before computing the priors for the current frame. Let $\tilde{P}_{\mathbf{x}}^{t-1}(\mathtt{bg})$ be the smoothed background posterior image from the previous frame. The priors for the current frame are

$$\begin{aligned} P_{\mathbf{x}}^{t}(\mathtt{bg}) =& \tilde{P}_{\mathbf{x}}^{t-1}(\mathtt{bg}) \times .950 + (1 - \tilde{\mathrm{P}}_{\mathbf{x}}^{\mathrm{t-1}}(\mathtt{bg})) \times .500, \\ P_{\mathbf{x}}^{t}(\mathtt{fg}) =& \tilde{P}_{\mathbf{x}}^{t-1}(\mathtt{bg}) \times .025 + (1 - \tilde{\mathrm{P}}_{\mathbf{x}}^{\mathrm{t-1}}(\mathtt{bg})) \times .250, \\ P_{\mathbf{x}}^{t}(\mathtt{fu}) =& \tilde{P}_{\mathbf{x}}^{t-1}(\mathtt{bg}) \times .025 + (1 - \tilde{\mathrm{P}}_{\mathbf{x}}^{\mathrm{t-1}}(\mathtt{bg})) \times .250. \end{aligned} \tag{2.10}$$

Figure 2.3 is an illustration of the prior computation process. Figure 2.3(a) shows the previous frame for which the background label probabilities at each pixel have been computed in (b). The background probabilities are smoothed with a Gaussian filter in (c). Using Equation 2.10, the background prior (d), the foreground prior (e), the unseen foreground prior (f) are computed. These priors are then used for computing the posterior probabilities in the current frame, as we explain in the next section.

In our implementation, although the likelihoods for the foreground and unseen foreground processes are different, the priors for the two processes are equal at every pixel. It is not necessary that the priors for the seen foreground and the unseen foreground be the same in all background modeling systems. For instance, at image boundaries, using a higher prior value for the unseen foreground could result in better detection of new objects that enter the scene in these regions.

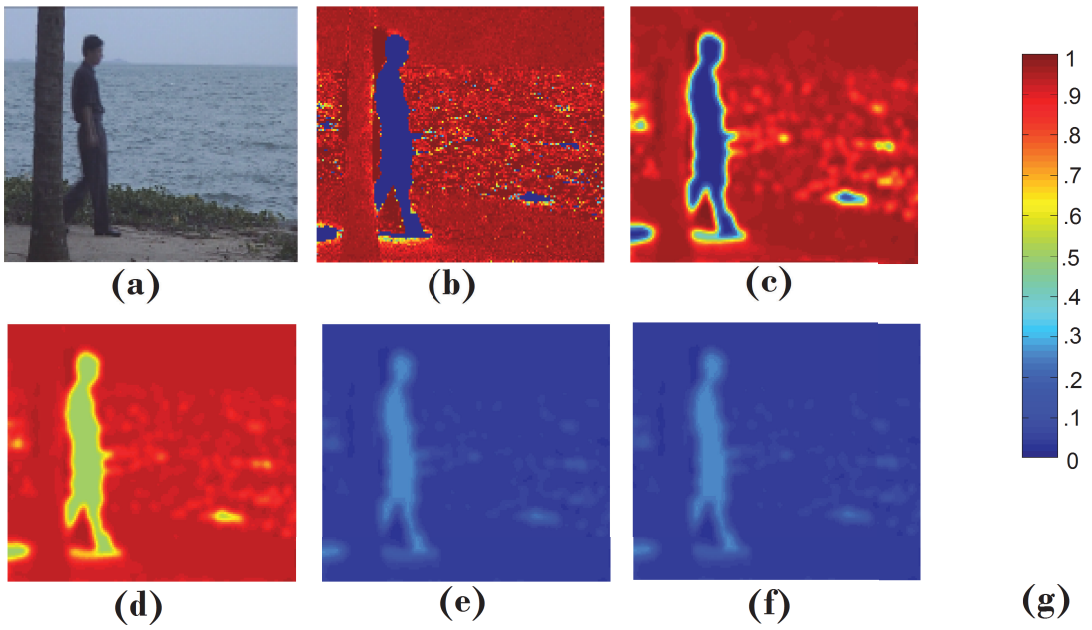

Figure 2.3. Illustration of computation of the spatially dependent prior. The image from the previous frame is shown in (a). The background probabilities in (b) are first smoothed with a Gaussian filter to allow for some amount of object motion in the scene. The smoothed probabilities are shown in (c), from which the background prior (d), the foreground prior (e), and the unseen foreground prior (f) are computed. The mapping from color to probability values is given in (g). We use equivalent equations for the foreground and unseen foreground priors which result in (e) and (f) being identical.

Our choice of the values .95 and .50 for the background prior for pixels that have been labeled as background and foreground in the previous frame respectively is guided by the intuition that background pixels change their label from one frame to the next very rarely and foreground objects that are moving have a moderate chance of revealing the background in the next frame. That these values are set by hand is a weakness of our current system.[2] The advantage of our approach is that these values can easily be learned automatically by accumulating statistics from the scene over a long period of time. Although the effect of using different priors for the background and foreground is equivalent to using

[2]Observations from the ground truth labels from videos in the change detection data set [21] show that between 95 and 100 percent of all pixels labeled as background in each frame retain their background label in the next frame. We believe the use of the value .95 for background prior is justified in light of this observation. The use of .50 for the background prior in pixel locations that were labeled as foreground in the previous frame essentially allows the likelihood to decide the labels of these pixels in the current frame.

a decision threshold on the foreground-background likelihood ratio, the priors are easier to understand and update. For example, if the priors at each pixel are updated using the pixel labels from long term scene history, the statistics could reveal a higher foreground prior near doors in the scene and at image borders. A similar scheme to update a decision threshold at these locations is far less natural.

We use a Gaussian filter of size 7 because the foreground objects in these videos typically move by 5 to 10 pixels. The size of the filter can potentially be learned by tracking the foreground objects. If there is a significant depth variation in different parts of the scene, a different parameter can be learned for the corresponding image regions by using tracking information [46].

2.5 Computing the posteriors - putting the components together during inference

Given the likelihoods and the priors as described in the previous sections, the only thing left to do is to compute the posterior probability of background and foreground, conditioned on the observed pixel values using Bayes' rule.

Given an observed color vector $\mathbf{c}$ at pixel location $\mathbf{x}$ in frame t, the probability of background and foreground are

$$\begin{aligned} P^t_{\mathbf{x}}(\mathtt{bg}|\mathbf{c}) &= \frac{\mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathtt{bg};\Sigma^{\mathrm{B}}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathtt{bg})}{\sum_{\mathrm{l}=\mathtt{bg},\mathtt{fg}} \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathrm{l};\Sigma^{\mathrm{l}}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathrm{l}) + \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathtt{fu}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathtt{fu})} \\ P^t_{\mathbf{x}}(\mathtt{fg}) &= 1 - \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathtt{bg}|\mathbf{c}). \end{aligned} \tag{2.11}$$

When the ideal likelihoods and priors are known, classification based on Bayes' rule gives the minimum possible error. A common alternative classification criterion is the ratio of the foreground likelihood to the background likelihood. The likelihood ratio classification in the joint domain-range model deserves special consideration because it implicitly includes a notion of a prior. However, as we show in the next section, the implicit prior involved causes a peculiar dependence on the image size. Our model does not exhibit this undesired consequence.

2.5.1 Likelihood ratio-based classification in the joint domain-range model

In the Sheikh and Shah joint domain-range model [58], the classification of pixels is done based on the likelihood ratios of the background and foreground processes. The decision criterion based on the ratios of the five-dimensional background and foreground likelihoods can be represented as

$$\begin{aligned} P^t(\mathbf{c},\mathbf{x}|\mathtt{bg}) &\gtrless^{?} P^t(\mathbf{c},\mathbf{x}|\mathtt{fg}) \\ P^t(\mathbf{c}|\mathbf{x},\mathtt{bg}) \times \mathrm{P^t}(\mathbf{x}|\mathtt{bg}) &\gtrless^{?} P^t(\mathbf{c}|\mathbf{x},\mathtt{fg}) \times \mathrm{P^t}(\mathbf{x}|\mathtt{fg}). \end{aligned} \tag{2.12}$$

The classification decision hence depends on the factors $P^t(\mathbf{x}|\mathtt{bg})$ and $P^t(\mathbf{x}|\mathtt{fg})$. These factors are the prior probability of a particular pixel location given the background or foreground process. For any pixel location $\mathbf{x}$, these factors can depend upon parts of the image that are arbitrarily far away. This is because the prior likelihood of a given pixel location being foreground will be smaller if more pixels from another part of the image are detected as foreground, and larger if fewer pixels elsewhere are detected as foreground (since $P^t(\mathbf{x}|\mathtt{fg})$ must integrate to 1). Furthermore, these factors will change when the image size is changed, hence affecting the classification [47]. By separating the system components and bringing them together during the posterior computation, we avoid this arbitrary dependence on the size of the image.

2.5.2 Dependence of the joint domain-range model on spatial neighborhood extent

Another undesirable effect in the Sheikh and Shah model is that the likelihood computation depends on the size of the spatial neighborhoods considered for the background and foreground processes. Their likelihood model (Equation 2.6) is biased towards whichever process has a smaller spatial neighborhood - typically the background process in our system. If the neighborhood is large, pixel samples that are spatially far away contribute little to the numerator, but heavily to the denominator. Figure 2.4 illustrates this phenomenon with a synthetic example. Consider that a red foreground object was present in front of a pink background in the previous frame and that the foreground pixel samples from this image are used to compute the foreground likelihoods at each pixel in the current frame. For simplicity, we consider the case where the object has not moved from the previous frame to the current. Applying the Sheikh and Shah normalization scheme, we see that as the size of the neighborhood for foreground samples is increased from 1 to 3, the likelihood values

for the foreground pixels decrease dramatically (compare Figures 2.4c and d). Using our normalization method, the dependence between the spatial neighborhood and likelihood values is eliminated (Figures 2.4e and f).

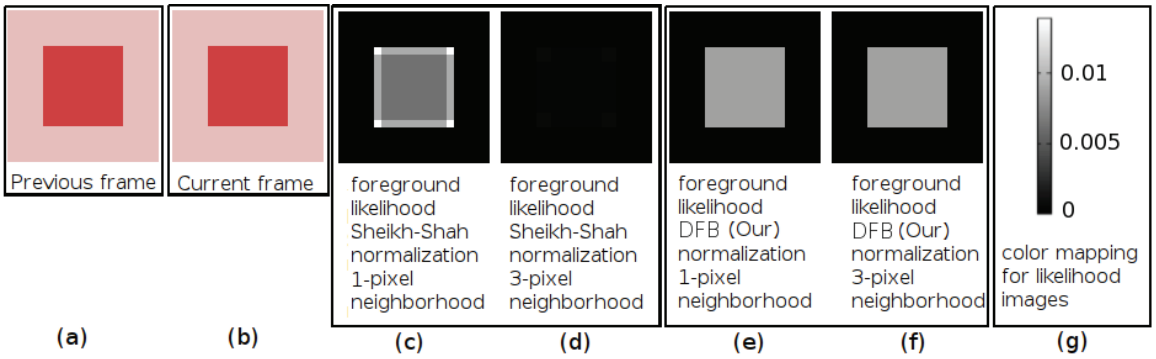

Figure 2.4. Sheikh and Shah normalization equation leads to a dependency between neighborhood size and likelihood values. Our normalization does not.

2.5.3 Model initialization and update

To initialize the models, it is assumed that the first few frames (typically 50) are all background pixels. The background model is populated using pixel observations from these frames. In order to improve efficiency, we sample 5 frames at equal time intervals from these 50 frames. The foreground model is initialized to have no observations. The foreground likelihood (Equation 2.8) enables colors that are not well explained by the background model to be classified as foreground, thus bootstrapping the foreground model. Once the observed pixel $c_{\mathbf{x}}$ at location $\mathbf{x}$ in a new frame is classified using Equation 2.11, the background and foreground models at the location $\mathbf{x}$ can then be updated with the new observation. Background and foreground observations at location $\mathbf{x}$ from the oldest frame in the models are replaced by $c_{\mathbf{x}}$. Observations from the previous 5 frames are maintained in memory as the foreground model data samples. The label probabilities of the background/foreground from Equation 2.11 are also saved along with the sample values for subsequent use in the Equations 2.4 and 2.8.

One consequence of the update procedure described above is that when a large foreground object occludes a background pixel at $\mathbf{x}$ for more than 50 frames, all the background observations in the spatial neighborhood of $\mathbf{x}$ are replaced by these foreground observations that have very low $P(\mathrm{bg}|\mathrm{b}_{\mathbf{x}})$ values. This causes the pixel at $\mathbf{x}$ to be misclas-

sified as foreground even when the occluding foreground object has moved away (because the background likelihood will be extremely low due to the influence of $P(\mathtt{bg}|\mathrm{b}_{\mathbf{x}})$ in Equation 2.4). To avoid this problem, we replace the background observations from location $\mathbf{x}$ in the oldest frame in the background model with the new observation $c_{\mathbf{x}}$ from the current frame only if $P(\mathtt{bg}|\mathrm{c}_{\mathbf{x}})$ estimated from Equation 2.11 is greater than 0.5.

In our chosen evaluation data set, there are several videos with moving objects in the first 50 frames. The assumption that all these pixels are background is not severely limiting even in these videos. The model update procedure allows us to recover from any errors that are caused by the presence of foreground objects in the initialization frames.

2.6 Comparison to earlier systems

In this section, we compare our system to the various earlier systems described. We use the I2R benchmark data set [40] with nine videos taken in different settings. The videos have several challenging features like moving leaves and waves, strong object shadows, and moving objects becoming stationary for a long duration. The videos are between 500 and 3000 frames in length and typically 128 x 160 pixels in size. Each video has 20 frames for which the ground truth has been marked. We use the F-measure to judge accuracy [41]; the higher the F-measure, the better the system:

$$F = \frac{2 \times recall \times precision}{recall + precision}. \tag{2.13}$$

To evaluate our results, the posterior probability of the background label is thresholded at a value of 0.5 to get the foreground pixels. Following the same procedure as Liao et al. [41], any foreground 4-connected components smaller than a size threshold of 15 pixels are ignored. The various systems compared are the MoG model of Stauffer and Grimson [62], the KDE model of Elgammal *et al.* [17], the complex background-foreground model of Li *et al.* (ACMMM03) [40], the joint domain-range model of Sheikh and Shah (jKDE) [58], [3] and our model, which we call the distribution field background (DFB) model. The naming reflects the fact that our model is a *field* of distributions with one distribution at each pixel location and was inspired by the description of such models in the tracking literature by Sevilla-Lara and Learned-Miller [56].

[3]The KDE and jKDE models are our own implementations and include spatially-dependent priors and Bayes' classification criterion in order to make a fair comparison.

Results in Table 2.1 show that systems that model the spatial influence of pixels, namely the jKDE model and our DFB model yield significantly higher accuracy. The table shows that the jKDE system is most accurate for our chosen parameter setting. It shows that very effective systems can be built even if the underlying model has certain deficiencies (as we showed in Section 2.5.2 for the jKDE). Mere separation of the model components as we have done and computing posterior probabilities for the labels does not guarantee better results. The usefulness of our system description is in that it enables a clear understanding of the different components and allows for better modeling of the components without having to tweak the inference procedure. To illustrate this aspect of our system, we describe one specific example of improving the background likelihood model by identifying a shortcoming in the model and developing a strategy to fix it in Chapter 3. The improved background likelihood model yields significantly more accurate results than the jKDE model as Section 3.2 will show.

Video	MoG	KDE	ACMMM03	jKDE	DFB
AirportHall	57.86	62.46	50.18	**70.13**	67.95
Bootstrap	54.07	61.15	60.46	**71.77**	69.17
Curtain	50.53	61.83	56.08	**87.34**	85.66
Escalator	36.64	40.84	32.95	53.70	**54.01**
Fountain	**77.85**	52.76	56.49	57.35	77.11
ShoppingMall	66.95	63.05	67.84	**74.12**	70.95
Lobby	**68.42**	22.78	20.35	27.88	21.64
Trees	55.37	64.01	75.40	**85.80**	82.61
WaterSurface	63.52	51.16	63.66	**78.16**	75.80
Average	59.02	53.34	53.71	**67.36**	67.21

Table 2.1. *F-measure* comparison between various existing algorithms on I2R data. Modeling the spatial influence of pixels (jKDE and DFB) significantly improves accuracy. MoG and ACMMM03 results are as reported by Li *et al.* [41]. For KDE, jKDE, and DFB, we use color dimension covariance value of $45/4$ for both the background and foreground models. For jKDE and DFB, we use spatial dimension covariance values of $3/4$ and $12/4$ for the background and foreground models respectively.

2.7 Discussion

We argue that the view of background modeling described in this chapter is, from a probabilistic perspective, clean and complete for the purpose of background modeling. By separating the various aspects of a background modeling system, namely the background likelihood, the foreground likelihood, and a prior, into distinct components, we have presented a simple view of background modeling. For inference, these separate components are brought together to compute the posterior probability for background.

Previous backgrounding systems have also modeled the components that we have described, but have often combined them with each other or caused dependence between the components and the inference. The separation of the components from each other and their isolation from the inference step makes the system easy to understand and extend. The individual components can be improved without having to consider their interdependence and effect on the inference. In the next chapter, we will show one example of improving the background likelihood model and its positive impact on the system's accuracy.

We use a spatially varying prior that depends on the labels from the previous frame. The model can further be improved by using a different prior at the image boundaries where new foreground objects are more likely. The modeling of the prior can also be improved by the explicit use of object tracking information.

We also believe that isolation of the model components can help in the development of effective learning methods for each of them. For example, the prior can be learned simply by counting the number of times each pixel is labeled as background or foreground. Maintaining a record of the number of times a pixel changes its label from background to foreground and vice-versa is one possible scheme to learn the prior values described in Section 2.4. Such a learning scheme can help build a dynamic model for the priors at different regions in the image.

CHAPTER 3

PIXELWISE ADAPTIVE VARIANCES FOR STATIONARY CAMERA SYSTEMS

Chapter 2 explains background modeling by clearly separating the various components involved and combining them during inference. In this chapter, we discuss improvements to the likelihood part of the model. The clear separation of likelihoods and priors means that the prior modeling and final classification procedures discussed earlier remain unchanged - only the likelihood component changes.

In this chapter, the effect of kernel variance in the KDE for likelihood estimation is discussed. Using a uniform kernel variance for the entire image is not effective. A pixelwise adaptive kernel variance enables significantly better likelihood estimates. As explained earlier, the likelihood distributions at each pixel may be modeled in a parametric manner using a mixture of Gaussians [62] (MoG) or using non-parametric kernel density estimation [17] (KDE). Sheikh and Shah [58] allow a pixel's spatial neighbors to influence its distribution via a joint domain-range density estimation [58] resulting in a more accurate system compared to earlier neighbor-independent models. Sheikh and Shah also show that the use of an explicit foreground model along with a background model can be useful. In a manner similar to theirs, we used kernel density estimates to obtain the background and foreground likelihoods at each pixel location using data samples from a spatial neighborhood around that location from previous frames. The variance used in the estimation kernel reflects the spatial and appearance uncertainties in the scene. On applying our method to a data set with wide variations across the videos, we found that choosing suitable kernel variances during the estimation process is very important. With various experiments, we establish that the best kernel variance could vary for different videos and more importantly, even within a single video, different regions in the image should be treated with different variance values. For example, in a scene with a stationary tree trunk and leaves that are waving in the wind, the trunk region can be explained with a small amount of spatial variance. The leaf regions may be better explained by a process with a large variance. Interestingly, when there is no wind, the leaf regions may also be explained with a low variance. The

optimal variance hence changes for each region in the video and also across time. This phenomenon is captured reasonably in MoG [62] by the use of a different parameter for each pixel which adapts dynamically to the scene statistics, but the pixelwise model does not allow a pixel's neighbors to affect its distribution. KDE-based models are updated with data samples from the most recent frame to better model a scene's dynamic nature. We show that using location-specific variances in addition to updating the model greatly improves background modeling. Our approach with pixelwise variances, which we call adaptive distribution fields backgrounding (DFBA) results in significant improvement over uniform variance models and state of the art backgrounding systems.

Although KDE is a non-parametric approach for estimating probability densities, the choice of the kernel variance or the bandwidth is an important one. Using large variance values can result in a very smooth density function while low variance values result in insufficient smoothing of the density function. The idea of using a pixelwise variance for background modeling is not new. Although Sheikh and Shah [58] use a uniform variance, they discuss the use of variances that change as a function of the data samples or as a function of the point at which the estimation is made. (called *sample-point estimator* and *balloon estimator* in the KDE literature respectively [34, 45]). Variance selection for KDE is a well studied problem [72] with common solutions including mean integrated square error (MISE), asymptotic MISE (AMISE), and the leave-one-out-estimator based solutions. In the background subtraction context, there has been work on using a different covariance at each pixel [45, 65]. Zivkovic and Heijden [82] use a balloon estimator to adapt the kernel variance. Mittal and Paragios [45] use a hybrid approach but require that the uncertainty in the features be known. Tavakkoli et al. [65] learn the covariances for each pixel from a training set of frames and keep the learned covariances fixed for the entire classification phase. We use a maximum-likelihood approach to select the best variance at each pixel location. For every frame of the video, at each pixel location, the best variance is picked from a set of variance values by maximizing the likelihood of the pixel's observation under the different variances. This makes our method a *balloon estimator* [45]. By explicitly selecting the best variance from a range of variance values, we do not require the covariances to be calculable in closed-form and also allow for more flexibility at the classification stage.

Selecting the best of many kernel variances for each pixel means increased computation. One possible trade-off between accuracy and speed can be achieved by a caching scheme where the best kernel variances from the previous frame are used to calculate the likelihoods for the current frame pixels. If the resulting classification is overwhelmingly in

favor of either label, there is no need to perform a search for the best kernel variance for that pixel. The expensive variance selection procedure can be applied only to pixels where there is some contention between the two labels. We present a heuristic that achieves significant reduction in computation compared to our full implementation while maintaining the benefits of adaptive variance.

Development and improvement of the probabilistic models is one of the two main themes in background modeling research in recent years. The other theme is the development of complex features like local binary [26] and ternary patterns [41] that are more robust than color features for the task of background modeling. Scale-invariant local ternary patterns [41] (SILTP) are recently developed features that have been shown to be very robust to lighting changes and shadows in the scene. By combining color features with SILTP features in our adaptive variance kernel model, we bring together the best ideas from both themes in the field and achieve state of the art results on a benchmark data set.

The main contributions of this chapter are:

1. A practical scheme for pixelwise variance selection for background
2. A heuristic for selectively updating variances to improve speed further.
3. Incorporation of complex SILTP features into the joint domain-range kernel framework to achieve state of the art results.

The chapter is organized as follows. Dynamic adaptation of kernel variances is discussed in Section 3.1. Results and comparisons are in Section 3.2. An efficient algorithm involving caching is discussed in Section 3.3. We end with a discussion in Section 3.4.

3.1 Pixelwise adaptive kernel variance selection

A few issues with the choice of kernel variance values are discussed along with our proposed solution of using a different kernel variance for each pixel in the image.

3.1.1 A single global variance value for all pixels in an image

As explained in the introduction, different parts of the scene may have different statistics and hence need different kernel variance values. For example, consider the examples in Figure 3.1 where the same variance value is used for every pixel in the image. In Figure 3.1a to 3.1d, using a high spatial dimension kernel variance helps in accurate classification

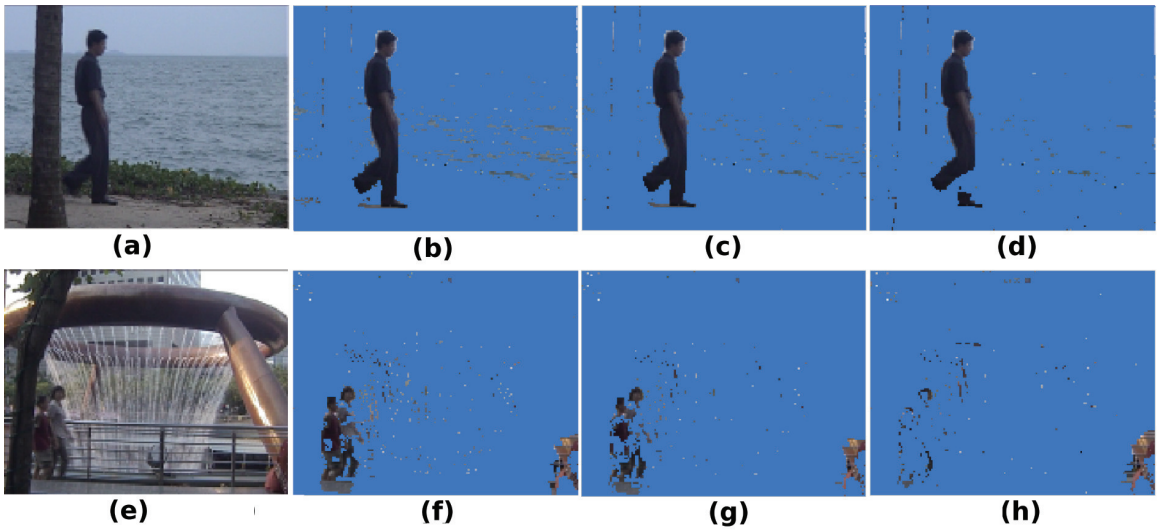

Figure 3.1. Two video sequences classified using increasing values of spatial kernel variance. **Column 1:** Original image. **Column 2:** $\sigma_{\mathrm{S}}^{B} = 1/4$. **Column 3:** $\sigma_{\mathrm{S}}^{B} = 3/4$. **Column 4:** $\sigma_{\mathrm{S}}^{B} = 12/4$. With a low value for spatial variance (b and f), many background pixels are misclassified as foreground. Increasing the spatial variance helps correct these errors, but can lead to foreground pixels being incorrectly classified as background (for example, the person's leg in d and the persons in h are lost).

of the water surface pixels, but doing so causes some pixels on the person's leg to become part of the background. Ideally, we would have different kernel variances for the water surface pixels and the rest of the pixels. Similarly in the second video (Figure 3.1e to 3.1h), having a high spatial kernel variance allows accurate classification of some of the fountain pixels as background at the cost of misclassifying many foreground pixels.

3.1.2 Optimal kernel variance across different videos

Figure 3.1 also shows that while the medium kernel variance may be the best choice for the first video, the low kernel variance may be best for the second video. In the results section, we show that for a data set with large variations like I2R [40], a single value for kernel variance for all videos is not sufficient to capture the variability in all the videos.

3.1.3 Background and foreground variances

Sheikh and Shah use the same kernel parameters for background and foreground models. Given the different nature of the two processes, it is reasonable to use different kernel parameters. For instance, foreground objects typically move between 5 and 10 pixels per

frame in the I2R [40] data set, whereas background pixels are either stationary or move very little. Hence, it is useful to have a larger spatial variance for the foreground model than for the background model.

3.1.4 Optimal kernel variances for classification

Having different variances for the background and foreground models reflects the differences between the expected uncertainty in the two processes. However, having different variances for the two processes could cause erroneous classification of pixels. Figure 3.2 shows a 1-dimensional example where using a very wide kernel (high variance) or very narrow kernel for the background process causes misclassification. Assuming that the red point (square) is a background sample and the blue point (triangle) is a foreground sample, it is reasonable to infer that the center point 'x' is equally likely to belong to the background or the foreground. Using the same variance for the background and foreground kernels would result in an equal likelihood for both at the center point. However, having a very low variance kernel (dashed red line) or a very high variance (solid red line) for the background process makes the background likelihood of the center point 'x' lower than the foreground likelihood. Thus, it is important to pick the optimal kernel variance for each process during classification.

In order to address all four issues discussed above, we propose the use of location-specific variances. For each location in the image, a range of kernel variances is tried and the variance which results in the highest likelihood is chosen for the background and the foreground models separately.

Mathematically, the likelihood and normalization factor Equations 2.4 and 2.5 now include a location-specific covariance matrix:

$$P^t_{\mathbf{x}}(\mathbf{c}|\mathsf{bg};\Sigma^{\mathrm{B}}_{\mathbf{x}})=\frac{1}{\mathrm{K}_{\mathsf{bg}}}\sum_{\mathrm{i}\in 1:\mathrm{T}}\sum_{\Delta\in\mathcal{N}_{\mathcal{B}}}\mathrm{G}(\mathbf{c}-\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta};\mathbf{0},\sigma^{\mathrm{B}}_{\mathbf{C},\mathbf{x}})\times\mathrm{G}(\Delta;\mathbf{0},\sigma^{\mathrm{B}}_{\mathbf{S},\mathbf{x}})\times\mathrm{P}^{\mathrm{t-i}}_{\mathbf{x}}(\mathsf{bg}|\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta}). \quad (3.1)$$

where $\sigma^B_{\mathbf{C},\mathbf{x}}$ and $\sigma^B_{\mathbf{S},\mathbf{x}}$ represent the location-specific color and spatial dimension variances at location $\mathbf{x}$. For each pixel location $\mathbf{x}$, the optimal variance for the background process is selected by maximizing the likelihood of the background at pixel $\mathbf{x}$ under different variance values:

$$\{\hat{\sigma}^B_{\mathbf{C},\mathbf{x}},\hat{\sigma}^B_{\mathbf{S},\mathbf{x}}\}=\underset{\sigma^B_{\mathbf{C},\mathbf{x}},\sigma^B_{\mathbf{S},\mathbf{x}}}{\arg\max}\,P(\mathbf{c}_{\mathbf{x}}|\mathsf{bg};\sigma^{\mathrm{B}}_{\mathbf{C},\mathbf{x}},\sigma^{\mathrm{B}}_{\mathbf{S},\mathbf{x}}). \quad (3.2)$$

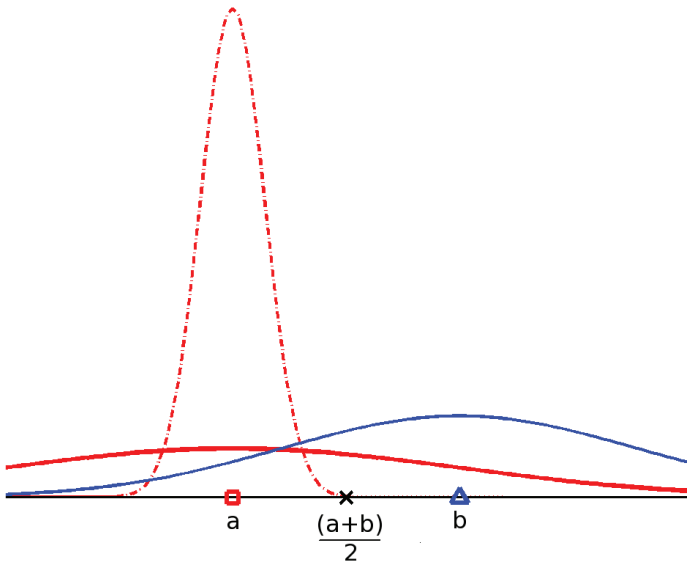

Figure 3.2. 1-dimensional example shows the effect of the kernel variance in classification. Using a higher or lower variance at point 'a' compared to point 'b' can cause misclassification of the intermediate point between them.

Here, $\sigma^B_{\mathbf{C},\mathbf{x}} \in R^B_{\mathbf{c}}$ and $\sigma^B_{\mathbf{S},\mathbf{x}} \in R^B_{\mathbf{x}}$. $R^B_{\mathbf{c}}$ and $R^B_{\mathbf{x}}$ represent the set of color and spatial dimension variances from which to choose the optimal variance. These constitute the diagonal elements in the covariance matrices $\Sigma^B_{\mathbf{C}}$ and $\Sigma^B_{\mathbf{S}}$.

Figures 3.3 and 3.4 illustrate the effect of using adaptive kernels. Consider a synthetic scene with no foreground objects, but in which the colors in the central greenish part of the background have been displaced at random by one or two pixel locations to simulate spatial uncertainty. As shown in Figure 3.3, the adaptive kernel variance method models the scene better by applying a high spatial variance for pixels that have moved and a low spatial variance for pixels that have not moved. Similarly, for color variance, Figure 3.4 shows the resulting likelihoods when uniformly sampled noise is added to the color values in the central part of the image. A small color variance value results in low likelihoods for pixels whose colors have changed. A large color variance results in low likelihoods for pixels that have not changed. The adaptive kernel variance method performs well in both kinds of pixels.

A similar variance selection procedure may be followed for the foreground likelihood. However, in practice, it was found that the variance selection procedure yielded large improvements when applied to the background model and little improvement in the fore-

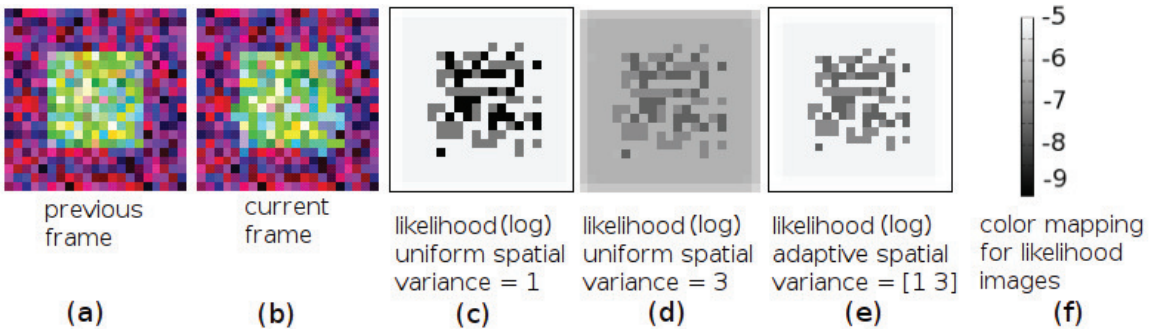

Figure 3.3. (a) and (b) Spatial uncertainty in the central part of the background. (c) Small uniform variance results in low likelihoods for pixels that have moved. (d) Large uniform variance results in higher likelihoods of the moved pixels at the expense of lowering the likelihoods of stationary pixels. (e) Adaptive variance results in high likelihoods for both the moved and stationary pixels.

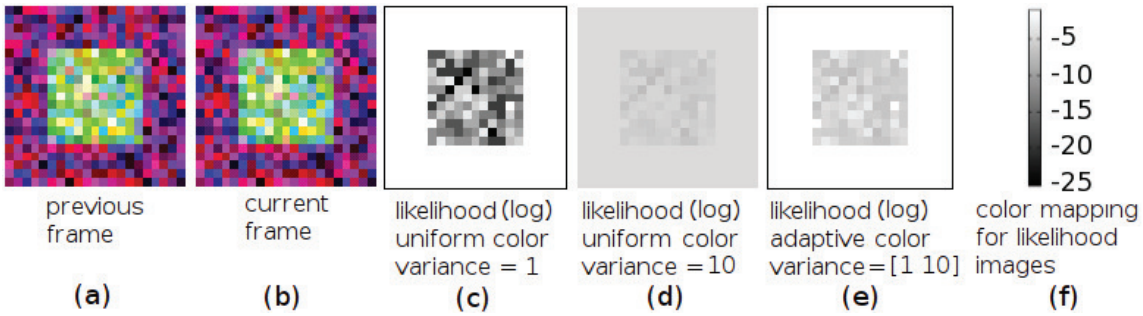

Figure 3.4. Color uncertainty in the central part of the background is best modeled by using adaptive kernel variances. (c) Small uniform variance results in low likelihoods for pixels that have changed color. (d) Large uniform variance results in higher likelihoods of the altered pixels at the expense of lowering the likelihoods of other pixels. (e) Adaptive variance results in high likelihoods for both kinds of pixels.

ground model. Hence, our final implementation uses an adaptive kernel variance procedure for the background model and a fixed kernel variance for the foreground model.

3.2 Results

The proposed method is analyzed on the I2R data set discussed in the previous chapter. The effect of choosing various kernel widths for the background and foreground models is shown in Table 3.1. The table shows the F-measure for each of the videos in the data set for

various choices of the kernel variances. The first 5 columns correspond to using a constant variance for each process at all pixel locations in the video.

The table shows that using a larger spatial variance for the foreground is beneficial (compare columns 2 and 3). The table reports the F-measure averaged over 20 ground truth frames. On average, the setting in column 3 returns the best result among uniform variance settings. The table does not show the differences when in individual frames are considered. When individual frames are observed, variance settings from other columns often outperform the settings in column 3. Using a selection procedure where the best kernel variance is chosen from a set of values gives the best results for most videos (column 6) and frames.

Comparison of our selection procedure to a baseline method of using a standard algorithm for variance selection in KDE (AMISE criterion[1]) shows that the standard algorithm is not as accurate as our method (column 7). Our choice for the variance values for spatial dimension reflects no motion ($\sigma_{\mathbf{S}}^{B} = 1/4$) and very little motion ($\sigma_{\mathbf{S}}^{B} = 3/4$) for the background, and moderate amount of motion ($\sigma_{\mathbf{S}}^{F} = 12/4$) for the foreground. For the color dimension, the choice is between little variation ($\sigma_{\mathbf{C}}^{B} = 5/4$), moderate variation ($\sigma_{\mathbf{C}}^{B} = 15/4$), and high variation ($\sigma_{\mathbf{C}}^{B} = 45/4$) for the background, and moderate variation ($\sigma_{\mathbf{C}}^{F} = 15/4$) for the foreground. These choices are based on our intuition about the processes involved. For videos that differ significantly from the videos we use, it is possible that the baseline AMISE method would perform better.

We would like to point out that ideally the variance value sets should be learned automatically from a separate training data set. In absence of suitable training data for these videos in particular and for background subtraction research in general, we resort to manually choosing these values. This also appears to be the common practice among researchers in this area.

Benchmark comparisons are provided for selected existing methods - MOG [62], the complex foreground model [40] (ACMMM03), the Sheikh and Shah model [58] (jKDE), and SILTP [41]. Figure 3.5 shows qualitative results for the same frames that were reported by Liao et al. [41]. We present results for our kernel method with uniform variances and adaptive variances with RGB features (DFB-rgb and DFBA-rgb respectively), and adaptive variances with a hybrid feature space of LAB color and SILTP features (DFBA-lab+siltp). Except for the Lobby video, the DFBA results are better than other methods. The Lobby

[1]We use the publicly available implementation from http://www.ics.uci.edu/ ihler/code/kde.html.

Column num	(1)	(2)	(3)	(4)	(5)	(6)	(7)
$4*\sigma^B_{\mathbf{S}} \longrightarrow$	3	3	3	1	3	[1 3]	AMISE
$4*\sigma^B_{\mathbf{C}} \longrightarrow$	15	45	45	45	45	[5 15 45]	AMISE
$4*\sigma^F_{\mathbf{S}} \longrightarrow$	3	3	12	1	12	[12]	[12]
$4*\sigma^F_{\mathbf{C}} \longrightarrow$	15	45	45	45	15	[15]	[15]
AirportHall	53.64	66.37	67.95	62.46	56.58	**68.28**	53.01
Bootstrap	58.90	66.96	69.17	61.15	65.56	**71.86**	63.38
Curtain	49.96	71.22	85.66	61.83	38.32	**93.57**	52.00
Escalator	35.32	53.01	54.01	40.84	33.97	**66.37**	32.02
Fountain	56.02	59.00	77.11	52.76	56.82	**77.43**	28.50
ShoppingMall	62.67	70.28	70.95	63.05	62.08	**76.46**	70.14
Lobby	23.27	22.55	21.64	22.78	20.54	13.24	**36.77**
Trees	62.35	78.35	82.61	64.01	80.42	**83.88**	64.30
WaterSurface	46.78	55.63	75.80	51.16	42.81	**93.81**	30.29
Average	49.88	60.37	67.21	53.34	50.79	**70.18**	47.82

Table 3.1. *F-measure* for different kernel variances. Using our selection procedure (Column 6) results in the highest accuracy.

video is an instance where there is a sudden change in illumination in the scene (turning a light switch on and off). Due to use of an explicit foreground model, our kernel methods misclassify most of the pixels as foreground and take a long time to recover from this error. A possible solution for this case is presented later. Compared to the uniform variance kernel estimates, we see that DFBA-rgb has fewer false positive foreground pixels.

Quantitative results in Table 3.3 compare the F-measure likelihoods for our method against MoG, ACMMM03, and SILTP results as reported by Liao et al. [41]. The table shows that methods that share spatial information with RGB features (jKDE, DFB, and DFBA - columns 5, 6, and 7) give significantly better results than methods that use RGB features without spatial sharing (columns 2 and 3). Compared to uniform kernel methods that use RGB feature space along with spatial influence (columns 5 and 6), the variable kernel method (column 7) is more accurate for most videos. Scale-invariant local ternary pattern (SILTP) [41] is a recent texture feature that is robust to soft shadows and lighting changes. We believe SILTP represents the state of the art in background modeling and hence compare our results to this method. Scale-invariant local states [80] is a slight variation in the representation of the SILTP feature. For comparison, we use SILTP re-

	$4*\sigma^B_{\mathbf{S}}$	$4*\sigma^F_{\mathbf{S}}$	$4*\sigma^B_{\mathbf{C}}$	$4*\sigma^F_{\mathbf{C}}$	$4*\sigma^B_{\mathbf{I}}$	$4*\sigma^F_{\mathbf{I}}$	$4*\sigma^B_{\mathbf{a}b}$	$4*\sigma^F_{\mathbf{a}b}$	$4*\sigma^B_{\mathbf{s}iltp}$	$4*\sigma^F_{\mathbf{s}iltp}$
DFBA rgb	[1,3]	12	[5, 15, 45]	15	-	-	-	-	-	-
DFBA lab+siltp	[1,3]	12	-	-	[5, 10, 20]	15	[4,6]	4	3	3

Table 3.2. Parameter values for DFBA implementation.

sults from Liao et al. because in Yuk and Wong [80], human judgement[2] was used to vary a size threshold parameter for each video. We believe results from the latter fall under a different category of human-assisted backgrounding and hence do not compare to our method where no video-specific hand-tuning of parameters was done. Table 3.3 shows that SILTP is very robust to lighting changes and works well across the entire data set. Blue entries in Table 3.3 correspond to videos where our method performs better than SILTP. DFBA with RGB features (DFBA-rgb) performs well in videos that have few shadows and lighting changes. Use of color features that are more robust to illumination change, like LAB features in place of RGB helps in successful classification of the shadow regions as background. Texture features are robust to lighting changes but not effective on large texture-less objects. Color features are effective on large objects, but not very robust to varying illumination. By combining texture features with LAB color features, we expect to benefit from the strengths of both feature spaces. Such a combination has proved useful in earlier work [79]. Augmenting the LAB features with SILTP features (computed at 3 resolutions) in the DFBA framework (DFBA-lab+siltp) results in an improvement in 7 out of 9 videos (last column). The variance values used in our implementation are given in Table 3.2.

We also compare our results (DFBA-lab+siltp) to the 5 videos that were submitted as supplementary material by Liao et al. [41]. Figure 3.6 highlights some key frames that highlight the strengths and weaknesses of our system versus the SILTP results. The common problems with our algorithm are shadows being classified as foreground (row e) and initialization errors (row e shows a scene where the desk was occluded by people when the background model was initialized. Due to the explicit foreground model, DFBA takes some time to recover from the erroneous initialization). A common drawback with SILTP is that large texture-less objects have "holes" in them (row a). Use of color features helps

[2]This was learned via personal communication with the authors.

(1) Video	(2) ACMMM03 rgb	(3) MoG rgb	(4) SILTP [41]	(5) jKDE rgb	(6) DFB rgb	(7) DFBA rgb	(8) DFBA lab+siltp
AirportHall	50.18	57.86	68.02	70.13	67.95	68.28	**70.75**
Bootstrap	60.46	54.07	72.90	71.17	69.17	71.86	**77.64**
Curtain	56.08	50.53	92.40	87.34	85.66	93.57	**94.07**
Escalator	32.95	36.64	**68.66**	53.70	54.01	66.37	49.99
Fountain	56.49	77.85	85.04	57.35	77.11	77.43	**85.88**
ShoppingMall	67.84	66.95	79.65	74.12	70.95	76.46	**82.64**
Lobby	20.35	68.42	**79.21**	27.88	21.64	13.24	62.60
Trees	75.40	55.37	67.83	85.80	82.61	87.64	**87.85**
WaterSurface	63.66	63.52	83.15	78.16	75.80	**93.79**	92.61

Table 3.3. *F-measure* on I2R data. DFBA significantly outperforms other color feature-based methods and improves on SILTP texture features on most videos. Blue color indicates performance better than SILTP.

avoid these errors. The SILTP system also loses objects that stop moving (rows b, c, d, f). Due to the explicit modeling of the foreground, DFBA is able to detect objects that stop moving.

The two videos in the data set where our algorithm performs worse than SILTP are the Escalator video (rows g, h) and the Lobby video (rows i, j). In the Escalator video, our algorithm fails at the escalator steps due to large variation in color in the region.

In the Lobby video, at the time of sudden illumination change, many pixels in the image get classified as foreground. Due to the foreground model, these pixels continue to be misclassified for a long duration (row j). The problem is more serious for RGB features (Figure 3.5 column 2). One method to address the situation is to observe the illumination change from one frame to the next. If more than half the pixels in the image change in illumination by a threshold value of T_I or more, we throw away all the background samples at that instance and begin learning a new model from the subsequent 50 frames. This method allows us to address the poor performance in the Lobby video with resulting F-measure values of 86.77 for uniform-rgb, 78.46 for DFBA-rgb, and 77.76 for DFBA-lab+siltp. T_I of 10 and 2.5 were used for RGB and LAB spaces respectively. The illumination change procedure does not affect the performance of DFBA on any other video in the data set.

3.3 Caching optimal kernel variances from the previous frame

A major drawback with trying multiple variance values at each pixel to select the best variance is that the amount of computation per pixel increases significantly. In order to reduce the complexity of the algorithm, we use a scheme where the current frame's optimal variance values for each pixel location for both the background and foreground processes are stored ($\hat{\sigma}^{B}_{\mathbf{C},\mathbf{x}\ cached}, \hat{\sigma}^{B}_{\mathbf{S},\mathbf{x}\ cached}, \hat{\sigma}^{F}_{\mathbf{C},\mathbf{x}\ cached}, \hat{\sigma}^{F}_{\mathbf{S},\mathbf{x}\ cached}$) for each location $\mathbf{x}$ in the image. When classifying pixels in the next frame, these cached variance values are first tried. If the resulting background and foreground likelihoods are very far apart, then it is very likely that the pixel has not changed its label from the previous frame. The expensive variance selection procedure is performed only at pixels where the resulting likelihoods are close to each other.

Algorithm 1 Efficient variance selection

for each pixel sample $\mathbf{c_x}$ in the current frame **do**
  if $\frac{P(\mathbf{c_x}|\mathsf{bg};\hat{\sigma}^{\mathrm{B}}_{\mathbf{C},\mathbf{x}\ \mathrm{cached}},\hat{\sigma}^{\mathrm{B}}_{\mathbf{S},\mathbf{x}\ \mathrm{cached}})}{P(\mathbf{c_x}|\mathsf{fg};\hat{\sigma}^{\mathrm{F}}_{\mathbf{C},\mathbf{x}\ \mathrm{cached}},\hat{\sigma}^{\mathrm{F}}_{\mathbf{S},\mathbf{x}\ \mathrm{cached}})} > \tau_{BF}$ **then**
    Compute the label likelihoods resulting from use of the cached variance values.
  else
    Search over the values in the variance sets to pick the optimal variances.
    Compute the label likelihoods using the optimal variances.
  end if
end for

Algorithm 1 for efficient computation results in a reduction in computation in about 80% of the pixels in the videos when τ_{BF} is set to 2, with a slight reduction in the F-measure by about 1 to 2% on most videos when compared to the full implementation. The efficient variance selection procedure however still performs significantly better than the uniform variance model by 2 to 10% on most videos.

3.4 Discussion

By applying kernel estimate method to a large data set, we have established, as do Sheikh and Shah [58], that the use of spatial information is extremely helpful. Some of the important issues pertaining to the choice of kernel parameters for data sets with wide variations have been addressed. Having a uniform kernel variance for the entire data set and for all pixels in the image results in a poor overall system. Dynamically adapting the variance for each pixel results in a significant increase in accuracy.

Using color features in the joint domain-range kernel estimation approach can complement complex background model features in settings where the latter are known to be inaccurate. Combining robust color features like LAB with texture features like SILTP in a DFBA framework yields a reasonably accurate background classification system.

We have also shown an example where improvements to the likelihood aspect of the backgrounding system is done without altering any other aspects. Similar changes to earlier systems would require changes to the classification step. For instance, in systems that use a likelihood ratio for classification, the ratio thresholds may need to be readjusted.

Figure 3.5. Qualitative comparison of algorithms on image results reported in Liao et al. [41].

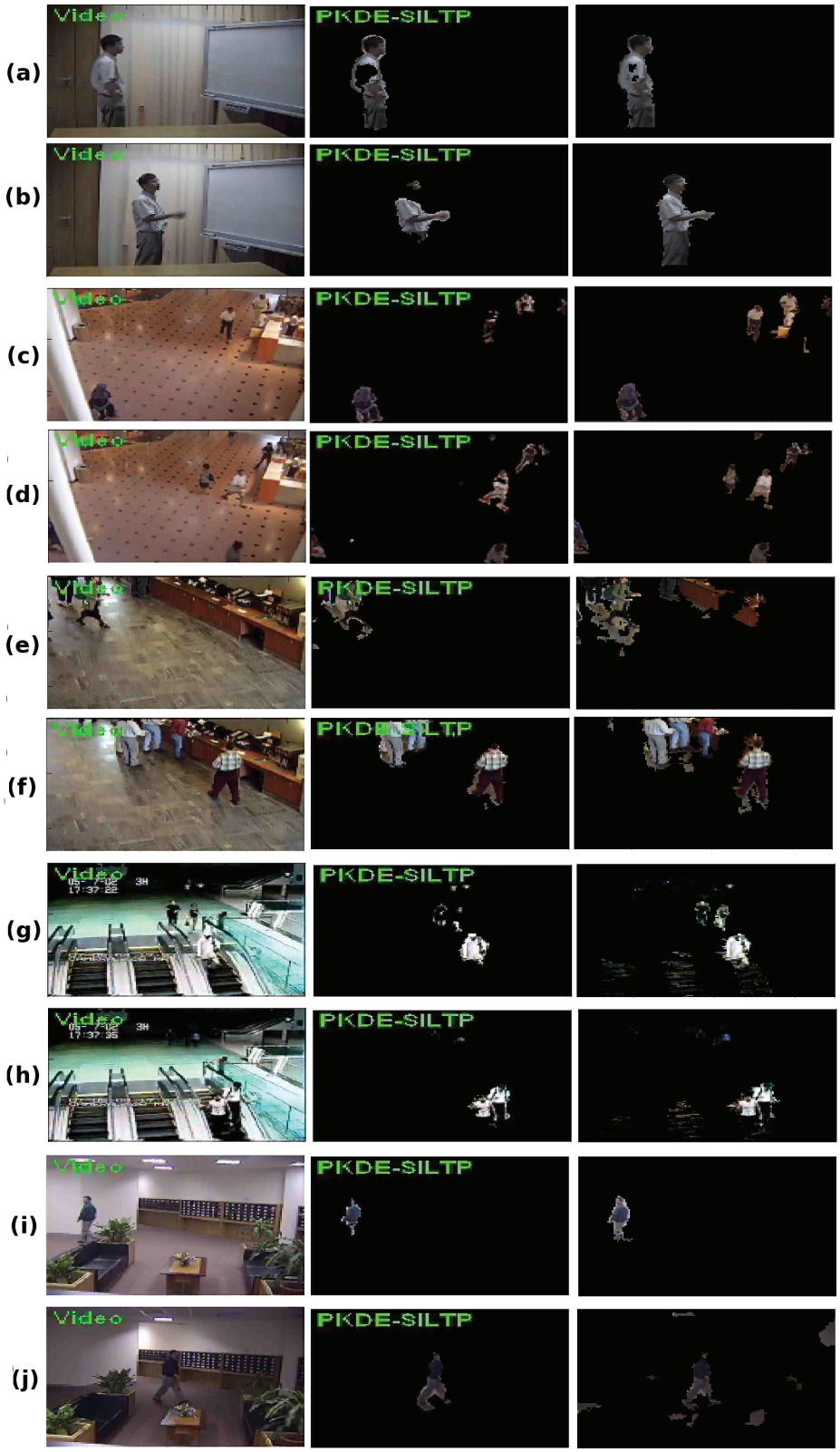

Figure 3.6. Comparison of DFBA and SILTP results. **Column 1:** Original video. **Column 2:** SILTP [41]. **Column 3:** DFBA-lab+siltp.

CHAPTER 4

MOTION SEGMENTATION IN MOVING CAMERA VIDEOS

4.1 Introduction

Motion segmentation in stationary camera videos is relatively straightforward and a pixelwise background model can be used to classify pixels as background or foreground. In the previous chapters, we described various methods and presented approaches for improving the background modeling and segmentation. While these approaches yield reasonably accurate background segmentation for stationary cameras, separating the non-moving objects from moving ones when the camera is moving is significantly more challenging. With a moving camera, the definition of which pixels are background and which are foreground is not so straightforward. Consider a scene where a pedestrian with a hand-held camera is walking while filming a car that is moving in the opposite direction. Comparing two consecutive frames, we would find that almost all pixels have changed. Pixelwise models described in earlier chapters would incorrectly classify all pixels as foreground. For the moving camera scenario, we define background pixels to be the pixels that have changed location only as a result of the camera's motion. Pixels that have changed location due to independent motion by objects in the scene are considered foreground.

Motion segmentation with moving cameras is an active area of research. Early approaches largely rely on estimating the camera's motion, *egomotion*, and compensating for such motion. After using a homography or a 2-d affine transformation to compensate for the camera's motion, the pixelwise background modeling techniques can be applied [29, 25, 55]. This approach is applicable only when the background is reasonably approximated as a planar surface. Furthermore, accurate computation of egomotion from a pair of consecutive video frames is not easy, especially in the presence of independently moving objects.

A common theme in moving camera motion segmentation is to use image plane motion (*optical flow*) [27] or trajectories as a surrogate for real-world object motion. Segmentation may be performed by clustering pixels based on the image plane motion flow

values [59, 75]. Bugeau and Perez [5] use optical flow to predict the locations of current frame pixels in the next frame. Pixels that exhibit significant difference from the expected appearance at their predicted locations are used as initial samples for the foreground pixels. The remaining pixels are labeled using foreground color and motion distributions built from these samples.

Many optical flow estimation algorithms use a notion of *layers* in the scene both for purposes of improving the optical flow estimates and segmenting image into layers depending on the depth of the objects. Sun *et al.* [64] jointly segment the image into layers and estimate flow using an affine model for object motion. However, because the resulting layers in these methods correspond roughly to the depth in the scene, the result is that the background pixels get separated into various layers. In motion segmentation, we are interested in segmenting the background objects into one layer irrespective of differences in their depths.

Simultaneous segmentation and optical flow estimation has been performed in different ways. Cremers [11] formulates the problem as a Bayesian inference problem and solves it using level set methods. A hierarchic coarse-to-fine model for improved accuracy is described by Memin and Perez [43]. Farneback [19] uses simultaneous segmentation and estimation to segment *coherent motion* regions that are consistent with a given affine motion model. Another approach for segmentation is to use optical flow as a first step to identify initial segments. Using an optical flow-based initial segmentation, Kwak *et al.* [38] describe a graphical model to perform subsequent classification by combining pixel positions, appearance (color) information, optical flow estimates, and labels from the previous frames.

An alternative to using optical flow values is the use of tracking information. Sheikh *et al.* [57] observe the trajectories of tracked salient points in the image sequence and use a factorization method to identify the bases for the background trajectories. Outlier trajectories are considered foreground. KDE appearance models are built from the pixels belonging to these trajectories to determine a dense labeling of pixels. Their method however assumes orthographic projections, which may not be a valid assumption in many videos. Brox and Malik [4] segment trajectories by computing the pairwise distances between all trajectories and finding a low-dimensional embedding using spectral clustering. Their method is not online and works on the video by considering all or a subset of frames at once.

Ochs and Brox [50] improved the spectral clustering by using higher order interactions that consider triplets of trajectories. Elqursh and Elgammal [18] proposed an online exten-

sion of spectral clustering by considering trajectories from 5 frames at a time. Because they rely on distance between optical flow vectors, these spectral methods are not guaranteed to group all the background pixels into one cluster. To obtain the complete background as one segment, a post-processing merging step is required where segments with similar motions are merged [4, 50]. The merging step assumes an affine motion model and hence may not work for complex backgrounds, as we show in Section 4.3. Elqursh and Elgammal learn a mixture of 5 Gaussians in the embedded space to represent the trajectories. Any trajectory that is not well explained by the mixture of Gaussians model is assumed to be a foreground trajectory. The parametric Gaussian mixtures model requires the number of mixtures, which can vary from scene to scene.

A significant improvement over the simple appearance and tracking model in the above papers was proposed by Kwak *et al.* [38]. They use a Bayesian filtering framework that combines block-based color appearance models with separate motion models for the background and foreground to estimate the labels at each pixel. However, they use a special initialization procedure in the first frame for segmenting the foreground objects. Their initialization procedure and the earlier trajectory-based methods use image plane motion. As described earlier, this cue is prone to causing errors.

In comparison to the above methods, we use motion information only from two frames at a time and do not require the use of trajectory information from multiple frames. In contrast to Kwak *et al.* , our system is completely online, with no special initialization step for the first frame. Due to automatic determination of the number of observed motions, our system is able to detect objects that are at rest initially and which begin to move during the video sequence.

Object tracking in a moving camera video is another theme in recent work. Chockalingam *et al.* [8] learn a fragmented model of the scene by breaking the image into smaller fragments which are then assigned foreground/background labels and tracked in subsequent frames. Tsai *et al.* [71] achieve tracking by using a spatio-temporal Markov random field (MRF) and introducing pairwise potentials that represent appearance and motion similarity between neighboring pixels. These tracking systems require an initial human-labeled foreground object while our goal is to build a foreground-background segmentation algorithm without any human intervention. Lee *et al.* [39] detect object-like segments called *key-segments* in the image, hypothesize which segments are more likely to be foreground objects, and finally use a spatio-temporal graph to perform segmentation. Although they avoid the requirement of hand-labeling the object of interest, their method is suited for of-

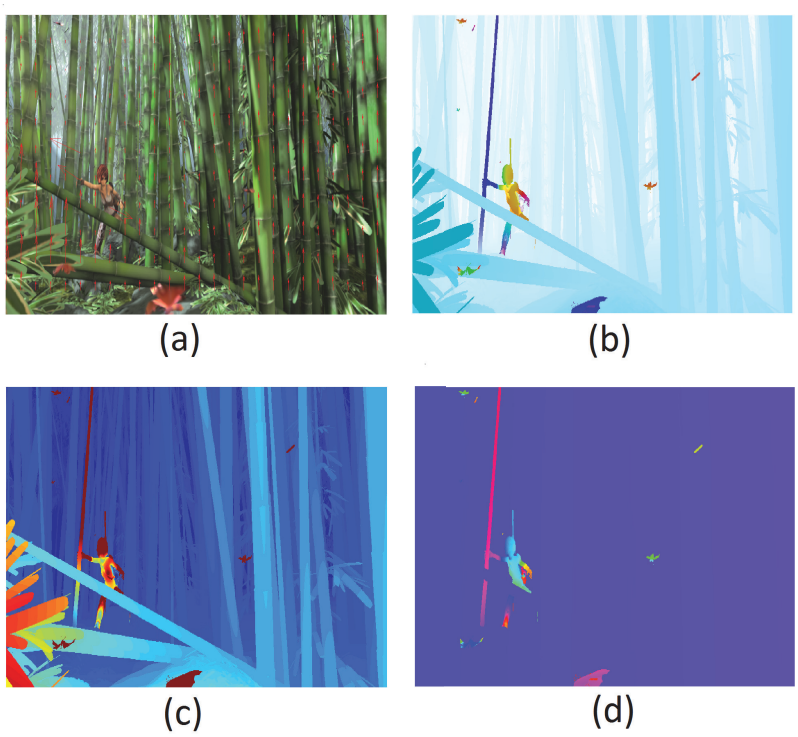

Figure 4.1. (a) A forest scene with a moving person (from the Sintel [6] data set). The person is holding on to a bamboo tree, which moves with the person. There are also a few leaves falling in the scene. (b) Visualization of the ground truth optical flow vectors (using code from [63]). (c) Magnitudes of the optical flow vectors. (d) Orientation of the optical flow vectors. The optical flow vectors and magnitudes on the trees depend on the distance of the trees from the camera. The orientations are not depth-dependent and can much more reliably predict that all the trees are part of the coherently moving background entity.

fline processing of videos because the initial key-segments generation phase requires the processing of all frames of the video. Our goal is to process a video frame-by-frame as they appear in the video stream.

Earlier motion segmentation methods described above rely heavily on image plane motion. Furthermore, the real-world motion is approximated as an affine motion in the image plane. We propose to solve the problem of motion segmentation by considering motion in the 3-d world. The problem with estimating the 3-d motion from images is that the real depth of the objects in the scene is unknown. The depth of the object in the scene greatly influences the optical flow observed in the image. In contrast, it well known that the depth of the object does not affect the orientations of the optical flow vectors in the case of camera translation. More importantly, motion of the camera enforces rigid constraints on the

possible values that the orientations can take at each pixel. For instance, for a camera translating towards a particular location in the world, all optical flow orientations point to the location's projection in the image. Based on these observations, a robust segmentation algorithm has been developed using the depth-invariance property of optical flow orientations in conjunction with the constraints that motion enforces on the orientations.

The use of the optical flow orientations is not entirely new. Recently, Adato *et al.* [1] showed that a polar representation of optical flow is much more effective in analyzing many classes of motions. In a method closely related to ours, Wedel *et al.* [76] improve the process of optical flow estimation by considering the projection of optical flows onto epipolar lines. Yamaguchi *et al.* [78] perform optical flow estimation by explicitly considering epipolar line orientations. However, in both these works, epipolar lines are obtained by first estimating the fundamental matrix. Fundamental matrix estimates are not reliable when there is independent object motion in the scene.

The major drawback of using optical flow is that an object's projected motion on the image plane depends on the object's distance from the camera. Objects that have the same real-world motion can have different optical flows depending on their depth. This can cause a clustering algorithm to label two background objects at different depths as two separate objects although they both have zero motion in the real-world. While this labeling is semantically reasonable because the two segments are likely to correspond to different objects in the world, for the purpose of detecting independently moving objects in the scene, this over-segmentation of the scene is undesirable. For example, in Figure 4.1, the optical flow vectors separate the forest background into many smaller tree segments. A segmentation based on the vectors would have to be followed by a post-processing step to merge smaller segments into one background cluster. Existing algorithms merge segments based on their color, motion, and edge energy [50]. If the number of distinct background layers is known, mixture modeling of the background motion is another solution [4].

An ideal solution would not require the use of such post-processing or prior knowledge about the scene. Our goal is to segment the scene into coherent regions based on the real-world motion of the objects in it. This can be challenging since the information about 3-D motion in the scene is only available in the form of the optical flow field. Our solution is based on the well-known property that in the case of camera translation, while optical flow magnitudes and vectors depend on the depth of the object in the scene, the orientations of the optical flow vectors do not. Figure 4.1 is an example that shows that the optical flow

orientations are reliable indicators of independent motion, much more so than the flow vectors or magnitudes.

Assuming only translational motions in the scene, given the motion parameters of the objects and knowledge about which pixels belong to each object, it is straightforward to predict the orientations at each pixel exactly. Figure 4.2 shows some examples of such predicted orientation *fields* for different motion parameter values. Our problem is the converse: Given the observed optical flow orientations at each pixel, estimate the motion parameters and pixel labels. We solve the problem by starting with a "library" of predicted orientation *fields* which cover a large space of possible translations and then use a probabilistic model to estimate which of these predicted orientation fields are actually being observed in the current image. Since multiple motions (one camera motion and possibly other independent object motions) are possible, we use a mixture model to determine which motions are present in the scene and which pixels belong to each motion. Finally, we favor explanations with fewer 3-D motions. A similar system involving optical flow magnitudes is much more complicated because in addition to estimating the motion parameters, it would be required to determine the object depth at each pixel.

Performing clustering when the number of foreground objects is unknown can be challenging. Techniques such as K-means or expectation maximization (EM) require knowing the number of clusters before-hand. We avoid this problem by instead using a non-parametric Dirichlet process-like mixture model where the number of components is determined automatically. Our system is capable of segmenting background objects at different depths into one segment and identifying the various regions that correspond to coherently moving foreground segments.

The optical flow orientations are not always reliable. Our algorithm is prone to failure when the assumption of pure translation is not satisfied. Also, a foreground object that moves in a direction consistent with the flow orientations due to the camera's motion will go undetected until it changes its motion direction. Some of these errors are handled in our system through the use of the pixelwise color appearance model and priors from Chapter 2. To handle complex camera motions that include rotations, we propose a rotation compensation algorithm in Chapter 5.

Extensive testing is performed on a wide range of videos. Earlier background segmentation methods report results only on 3 or 4 out of 26 videos from the Hopkins segmentation data set [4]. In addition to all 26 videos from this set, we also include results from the SegTrack motion segmentation data set [71]. Although good segmentation results are achieved

on these data sets, these videos have few cases of depth disparity in the background. Consequently, results from other videos with complex backgrounds that can involve many many depth layers, such as in a forest scene, are also presented. To the best of our knowledge, this is the first work to report moving background segmentation results on such a large number of videos spanning different scenarios. The results show the efficacy of the algorithm and its applicability to a wide range of videos. Despite the assumption of translational camera motion, the algorithm is capable of handling many scenarios as exhibited in the data set.

This chapter is organized as follows. Optical flow orientations and the probabilistic segmentation model are explained in Section 4.2. Section 4.3 compares our flow orientation-based segmentations to various other segmentation schemes. Use of color-based appearance modeling and the overall video segmentation workflow is given in Section 4.4. A non-parametric model is described in Section 4.5. Benchmark results are presented in Section 4.6 and qualitative comparisons to some baseline methods in Section 4.7. We conclude with Section 4.8.

4.2 Segmentation using optical flow orientations

Given a camera's translation $t = (t_x, t_y, t_z)$, the resulting optical flows v_x and v_y in the x and y image dimensions are given by:

$$v_x = \frac{t_z \times x - t_x \times f}{Z} \quad \text{and} \quad v_y = \frac{t_z \times y - t_y \times f}{Z}, \tag{4.1}$$

where (x, y) represents a pixel location in the image, Z is the real-world depth of the observed point and f is the camera's focal length [28].

The optical flow orientations,

$$F(t, x, y) = \arctan(t_z \times y - t_y \times f, t_z \times x - t_x \times f), \tag{4.2}$$

are thus independent of the depth Z of the points. Here, $\arctan(y, x)$ returns the arctangent of (y/x) with a range $(-\pi, \pi]$. Figure 4.2 shows the optical flow orientations for a few different camera motions. It may be noted that the orientations are not always constant throughout the entire image. We call the 2-D matrix of optical flow orientations at each pixel the *flow orientation field* (FOF).

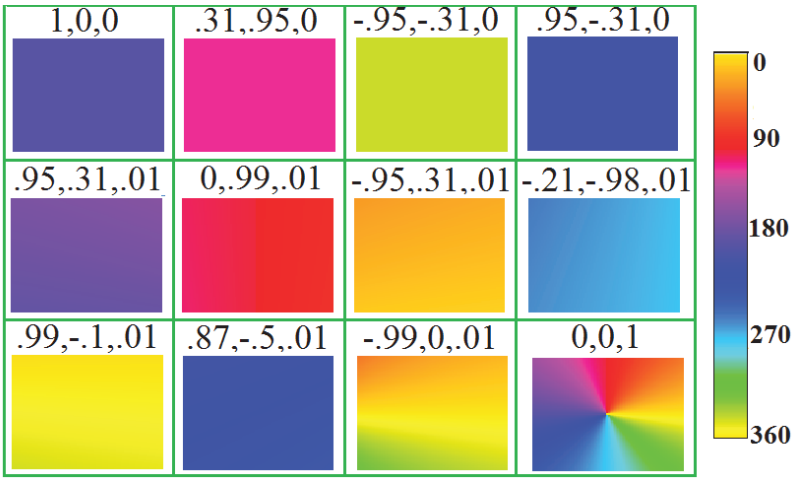

Figure 4.2. A sample set from the orientation fields that are used in our graphical model. Above each field are the motion parameters (t_x, t_y, t_z) that cause it. The colorbar on the right shows the mapping from color values to corresponding angles in degrees.

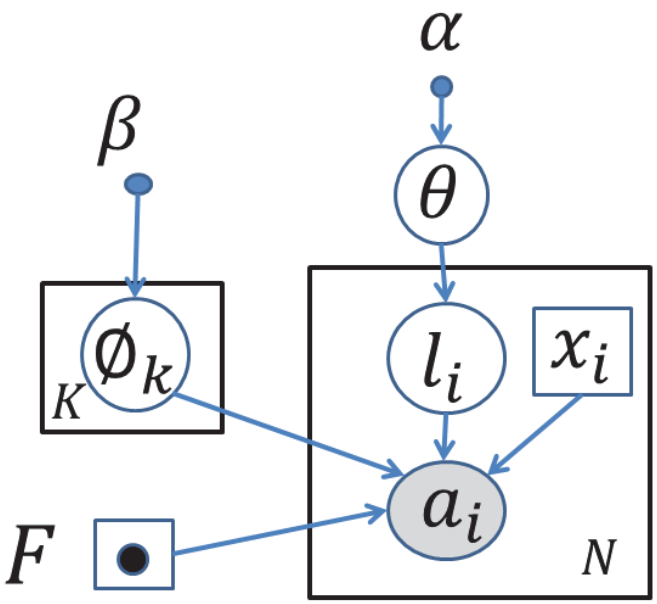

Figure 4.3. A mixture model for segmentation based on optical flow orientations. Notation: Variables inside circles are random variables and variables inside squares are deterministic. The dark colored dot represents a deterministic function, the shaded circle represents an observed variable and small shaded circles represent hyperparameters.

In the probabilistic model given in Figure 4.3, the orientation values returned by an optical flow estimation algorithm [63] are observed variables and the labels for each pixel are latent. At pixel number i, whose location is given by $\mathbf{x}_i = (x_i, y_i)$, we have an observed optical flow orientation a_i and a label l_i that represents which segment the pixel belongs to. Each segment k is associated with a motion parameter tuple $\Phi_k = (t_x^k, t_y^k, t_z^k)$ representing the translation along x, y, and z directions respectively. The continuous velocity space is discretized into a finite number of possible motions: 46 values for translation (t_x, t_y, t_z) are sampled from a unit hemisphere in front of an observer. Φ_k can hence take one of 46 values and the resulting FOFs due to these motion values form a "library" that is used to explain the observed data. Figure 4.2 shows a few of the library FOFs; a complete listing of the FOFs used is provided in the appendix. ϕ_k is used to denote the values that the variables Φ_k take. For a given motion parameter tuple t, denote the resulting flow orientation field at pixel location $\mathbf{x}$ to be $F(t, \mathbf{x})$, which is computed using Equation 4.2.

The graphical model is then defined by the following generative process:

$$\begin{aligned} P(\theta|\alpha) &= \text{Dir}(\theta|\alpha); \\ P(\Phi_k|\beta) &= \text{Uniform}(\beta); \\ P(l_i|\theta) &= \prod_{k=1}^{K} \theta_k^{[l_i=k]}; \\ P(a_i|\Phi = \phi, l_i = k, F, \mathbf{x}_i) &= P(a_i|\Phi_k = \phi_k, F(\phi_k, \mathbf{x}_i)) \\ &= G(a_i; F(\phi_k, \mathbf{x}_i), \sigma_k^2), \end{aligned} \tag{4.3}$$

where $[\cdot]$ represents an indicator function, Dir is a Dirichlet distribution, and $G(\cdot; \mu, \sigma^2)$ is a Gaussian with mean μ and variance σ^2. The last equation means that given the label $l_i = k$ for a pixel at location $\mathbf{x}_i$ and motion parameter $\Phi_k = \phi_k$, the observed orientation a_i is a Gaussian random variable whose mean is $F(\phi_k, \mathbf{x}_i)$. The variance for the Gaussian is the observed variance from $F(\phi_k, \mathbf{x}_i')$ at all pixel locations $\mathbf{x}'$ that were labeled k in the previous iteration. If no pixels were labeled k in the previous iteration, a variance value of $(a_{\mathbf{x}} - F_{\mathbf{x}}(\phi_k))^2$ is used.

We note that the above model is similar to a Dirichlet process mixture model with the exception that we sample Φ_k from a finite set of parameters. Algorithm 2 details the sampling procedure used. The algorithm is similar to the Gibbs sampling procedure for Dirichlet processes with non-conjugate priors as described by Neal (algorithm 8 in [48]).

The algorithm adds additional auxiliary Φ parameters at each iteration and retains the auxiliary parameters that explain any observed data. We begin with $K = 1$ component and add one new auxiliary component ($M = 1$) to the model at each iteration. The model hence adds components as required to explain the data.

Algorithm 2 Sampling procedure in our model

Step 0 : Initialize $\Phi_1 = c_1$ where c_1 is sampled from a uniform distribution over the set of "library" camera motion parameters.

for n iterations **do**

Let K be the current number of motion components. Sample M new motion parameters from β.

for $i = 1$ to N pixels **do**

Sample label l_i with the following probability:

$$P(l_i = c | c_{-i}, a_i, \Phi_1, \Phi_2, ...\Phi_{K+M}) = \begin{cases} b\frac{N_{-i,c}}{N-1+\alpha} P(a_i, \Phi_c) & \text{for } 1 \leq c \leq K \\ b\frac{\frac{\alpha}{M}}{N-1+\alpha} P(a_i, \Phi_c) & \text{for } K < c \leq K+M \end{cases}, \tag{4.4}$$

where c_{-i} represents c_j for all $j \neq i$, $N_{-i,c}$ represents the number of labels l_j that have value c and $j \neq i$, and b is an appropriate normalization constant.

end for

for all $c \in \{c_1, c_2, ...c_N\}$ **do**

Draw a new value $\Phi_c | l_i$ such that $l_i = c$.

end for

end for

4.2.1 Choosing α

The concentration parameter α determines the propensity of the system to add new components. In the absence of suitable training data to learn the concentration parameter, the Gibbs sampler is run with different values for $\alpha_j \in \{.0001, .01, 10\}$ and, from the resulting segmented images, the segmented image that best agrees with the other segmented images is chosen. From each candidate α_j, the segmented result is obtained and an image $b_j(\mathbf{x})$, which has a value 1 at locations that correspond to the largest segment and 0 at all other locations, is created. The sum of these b_j images is then computed: $b_{\text{sum}}(\mathbf{x}) = \sum_{j=1}^{n_\alpha} b_j(\mathbf{x})$, where n_α is the number of different α's being considered. Similarly, f_j and f_{sum} images are computed, where $f_j = 1 - b_j$. The best α corresponds to $\hat{j} = \arg\max_j \sum_{\mathbf{x}} \{b_{\text{sum}}(\mathbf{x}) \times b_j(\mathbf{x})\} + \{f_{\text{sum}}(\mathbf{x}) \times f_j(\mathbf{x})\}$. Intuitively, b_{sum} and f_{sum} are the

pixelwise sum of the votes for the background and foreground from all candidate α's. The best α is the one that best agrees with this voting.

4.2.2 Gradient descent for largest component

Improvements can be made to the results by finding a better fit for the largest segment's motion than provided by the relatively coarse initial sampling of library motion parameters. To achieve this, after $\frac{n}{2}$ iterations, at each iteration, we follow the Gibbs sampling step with a gradient descent step. With the motion parameters corresponding to the largest segment as the starting point, gradient descent is used to find the motion parameters that result in an FOF with the minimum average L1 distance to the observed orientations. Only the pixels that are currently assigned to the largest segment are used in computing the L1 distance. The resulting minimum motion parameter tuple is added as an additional motion parameter to the set of library motions. This process helps in the proper segmentation of observed background orientation patterns that are not well explained by any of the initial set of motions.

4.2.3 Handling pixels with near-zero motion

One of the implications of using the orientations is that the orientation is not defined for pixels that do not move. The orientation values at these pixels can be very noisy. To account for this possibility, pixels that have optical flow component magnitudes less than a threshold T_f (typically 0.5) in both x and y directions are marked as "zero-motion" pixels. They are accounted for by a "zero-motion" FOF and Gibbs sampling is not performed for these pixels.

4.3 Segmentation comparisons

The proposed FOF segmentation is compared to existing motion segmentation methods. Spectral clustering of trajectory information [4, 18, 50] has been shown to be useful for motion segmentation. The implementation provided by Ochs and Brox [50] that returns spectral clustering of tracked keypoints is used. Their algorithm is designed to work on trajectories from multiple frames. The number of frames is set to 3 for trajectory tracking (the minimum that their implementation requires). Further, their method uses a merging step that joins segments that have similar motion parameters. Note that FOF segmentation uses only flow information from two consecutive frames and performs no post-processing

Figure 4.4. Comparison of segmentation algorithms. The rows correspond to the original images, spectral clustering [50], and our FOF segmentation. The tracked keypoints used in spectral clustering are shown as squares with their colors representing the cluster memberships. Despite the use of a post-processing merge step in the implementation, in many images, spectral clustering is not certain about some background keypoints (white squares) and in cases with large depth disparity, the background is broken into smaller sections. Our method avoids these errors and also results in a dense labeling of the image. The last column is an example of our method failing because the car happens to move consistently with the FOF due to camera motion. Comparisons to several other methods are presented in Section 4.7.1.

to merge segments. Figure 4.4 shows the segmentations for some example frames. FOF segmentation, despite only using information from two frames and no merging procedure, successfully segments the background in most examples. Images that have large depth disparity show the clear advantage of our method (columns 3 and 4). Here spectral clustering with a subsequent merge step fails and the background is over-segmented depending on depth. The FOF-based clustering is successful in identifying the background objects as one segment.

4.4 Modeling the appearance and the prior

The described FOF-based mixture model returns the number of mixture components, the maximum a posteriori component assignments for each pixel, and the probabilities of each pixel belonging to each component. In order to classify each pixel as background or

foreground, we take the component with the largest number of pixels to be the background component.

In addition to using the FOF-based segmentation, we maintain a color appearance model for the background and foreground at each pixel as described in Chapter 2. A history of pixel data samples from the previous frames is maintained and after classification of pixels in each new frame, new data samples are added to the history. To account for motion, the maintained history at each pixel is motion-compensated and moved to a new location as predicted by the optical flow in the current frame. KDE is used with the data samples to obtain the likelihoods as given in Equations 2.4 and 2.8. The parameter values for the KDE are $\Sigma^B_{\mathbf{C}} = \Sigma^F_{\mathbf{C}} = \frac{15}{4}, \Sigma^B_{\mathbf{S}} = \Sigma^F_{\mathbf{S}} = \frac{5}{4}, T = 5$.

4.4.1 Mixing a uniform distribution component

In cases when the background has been occluded in all the previous T frames, there are no reliable history pixels for the background. To allow the system to recover from such a situation, a uniform color distribution is mixed into the color likelihood:

$$\hat{P}^t_{\mathbf{x}}(\mathbf{c}|\mathtt{bg}) = \gamma^{\mathtt{bg}}_{\mathbf{x}} \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathtt{bg}; \Sigma^{\mathrm{B}}) + (1 - \gamma^{\mathtt{bg}}_{\mathbf{x}}) \times \mathrm{U}, \tag{4.5}$$

where U is a uniform distribution over all possible color values. The mixture proportion is given by $\gamma^{\mathtt{bg}}_{\mathbf{x}} = \frac{\sum_{i \in 1:T} \sum_{\Delta \in \mathcal{N}_{\mathcal{B}}} P^{t-i}_{\mathbf{x}+\Delta}(\mathtt{bg}|\mathbf{b}^{\mathrm{t-i}}_{\mathbf{x}+\Delta})}{\sum_{i \in 1:T} \sum_{\Delta \in \mathcal{N}_{\mathcal{B}}}(1)}$. The implication of this mixture proportion is that if the history pixels are highly confident background pixels, then no uniform distribution is added to the likelihood. When there is unreliable information about background pixels in the history, a larger weight is assigned to the uniform component.

4.4.2 Posterior computation

The classification results from the previous frame contain useful prior information about which pixels are likely to belong to the background. The background posterior probability at each pixel in the previous frame is motion compensated according to optical flow and used as the pixelwise background prior for the current frame. A smoothed(7×7 Gaussian filter with a standard deviation value of 1.75) image of the posterior, $\tilde{P}^{t-1}_{\mathbf{x}}(\mathtt{bg})$, is used for the prior for the background process in the current frame.

The posterior probability of background in the current frame can now be computed by combining the color likelihoods, the segmentation label likelihoods from the graphical model, and the prior:

$$P^t_{\mathbf{x}}(\mathtt{bg}|\mathbf{c}, \mathrm{l}_{\mathbf{x}}) = \frac{\hat{\mathrm{P}}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathtt{bg}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathrm{l}_{\mathbf{x}}|\mathtt{bg}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathtt{bg})}{\sum_{\mathrm{L}=\mathtt{bg},\mathtt{fg}} \hat{\mathrm{P}}^{\mathrm{t}}_{\mathbf{x}}(\mathbf{c}|\mathrm{L}; \Sigma^{\mathrm{l}}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathrm{l}_{\mathbf{x}}|\mathrm{L}) \times \mathrm{P}^{\mathrm{t}}_{\mathbf{x}}(\mathrm{L})}. \tag{4.6}$$

The use of color likelihoods and prior information helps to recover from errors in the FOF-based segmentation as we explain in the results.

4.5 A non-parametric FOF segmentation model

The model discussed in Section 4.2 requires a predefined set of FOFs sampled from the possible space of translation parameters. The set of 46 motions is fixed throughout the inference process. Only the assignment of pixels to each cluster and the variance associated with each cluster is updated in the inference process. Although the results in the next section show that the model works well in practice, the requirement of preselecting a set of motion values is a serious drawback. In this section, instead of relying on a finite number of motion values, we propose a scheme where the motion values are continuously sampled from the space of motion parameters. In this approach, there is no limit on the number of motions inferred by the system. This makes the approach non-parametric and the number of motions is allowed to grow as much as required to explain the observed data.

The model, shown in Figure 4.5, is similar to the earlier model except that Φ_K is sampled thus:

$$\begin{aligned} t_x &\sim \text{Uniform}(-1, 1), \\ t_y &\sim \text{Uniform}(-1, 1), \\ t_z &\sim G(\cdot; 0, .01), \end{aligned} \tag{4.7}$$

where $G(\cdot; \mu, \sigma^2)$ is a Gaussian with mean μ and variance σ^2. The above sampling of Φ_k corresponds to sampling with a high probability in a disk region around a observer's sphere of view.

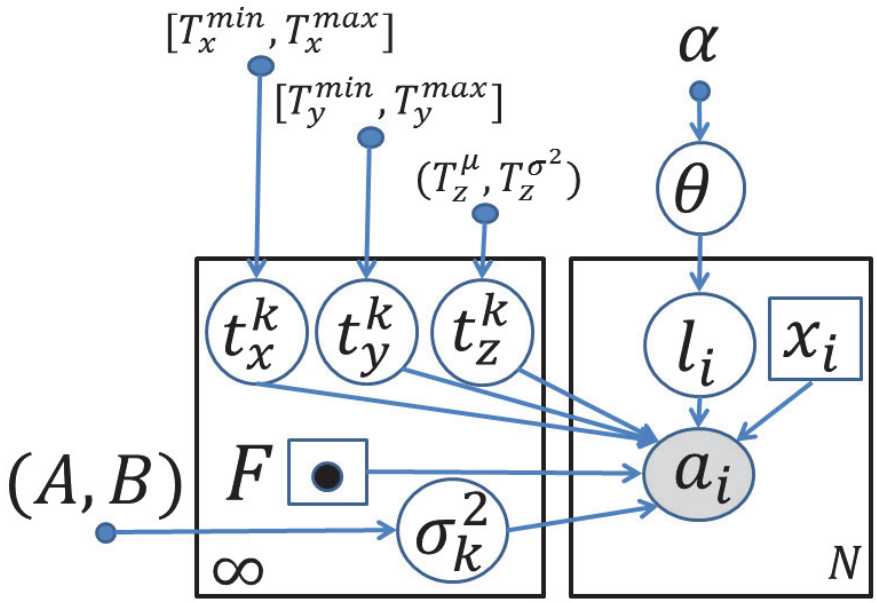

Figure 4.5. Non-parametric mixture model for segmentation based on optical flow orientations.

In addition to sampling for the motion parameters from the above base distributions, we include a inverse gamma prior for the variance parameter for each cluster. The following scheme is used to update the variance for each cluster:

$$\begin{aligned} \hat{A}_k &= A_k + \frac{n_k}{2}, \\ \hat{B}_k &= B_k + \frac{\sum_{i=1:n_k}(a_i - F_{k,i})^2}{2}, \\ \hat{\sigma}_k^2 &= \frac{\hat{B}_k}{\hat{A}_k}, \end{aligned} \tag{4.8}$$

where k represents the cluster whose parameters are being updated, A_k and B_k represent the prior hyper-parameters, and $\hat{A}_k$ and $\hat{B}_k$ are updated parameters after observing data. n_k is the number of pixels assigned to the cluster k, $F_{k,i}$ represents the FOF value for cluster k at pixel i, and a_i is the observed flow orientation at pixel i. We use $A_k = R \times C \times .01$ and $B_k = 15 \times A_k$ where R and C are the number of rows and columns in the image. A_k can be interpreted as the number of pixels that are to be observed before the prior value of variance ($\frac{B_k}{A_k} = 15$) is replaced by the observed variance in the cluster. In practice, it is useful to restrict the updated variance $\hat{\sigma}_k^2$ to lie within reasonable limits. Our implementation

uses a floor of 15 and ceiling of 100 for the variance. This causes any cluster to roughly have a minimum tolerance of 4 degrees and maximum tolerance of 10 degrees between the predicted and observed orientation values at any given pixel.

4.6 Results

The system's performance is evaluated on two existing benchmarks. In addition to these benchmarks, we also present results on a new set of videos that include several with complex background phenomena to highlight the strengths of the system. The first benchmark is a motion segmentation data set [4], derived from the Hopkins data set [70], which consists of 26 moving camera videos. The data set has ground truth segmentation for a few frames sampled throughout the video. The second data set is the SegTrack segmentation data set [71]. The third data set[1] ,which we produced ourselves, is a challenging one with complex backgrounds including trees in a forest and large occluding objects in front of the moving foreground object. This data set is extremely challenging for traditional motion segmentation algorithms.

Tables 4.1 and 4.2 show the average F-measure (Equation 2.13) for each video. We present results of FOF segmentation for both the models described earlier. The first model with predefined motion values we refer to as the finite-K model and the later model with continuously sampled motion values is called the infinite-K model. For both models, pure FOF segmentation results (Table 4.1) as well as the results after adding the color appearance and prior models (Table 4.2) are shown. Three different runs of the finite-K model are shown to understand the effect of changing the number of library motion components. The most accurate method for each video is marked in bold. The tables show that in the finite-K model, for different videos different values of K yield the best accuracy. While the K value of 92 results in the highest accuracy for the most number of videos, K value of 46 yields the highest average accuracy over all videos. Choosing an optimal value of K for all videos is thus not straightforward.

In general, comparing Tables 4.1 and 4.2 shows that the use of color and prior information helps improve the accuracy of FOF segmentation. In the Hopkins set, the videos that are challenging are the ones where the foreground object's FOF matches the camera

[1]The ComplexBackground data set is available for public use at
http://vis-www.cs.umass.edu/motionSegmentation/complexBgVideos.html.

motion's FOF for a long duration (cars4), the foreground object covers a majority of the pixels in the image (marple6, marple9), or where the foreground object is stationary for the first few hundred frames although the ground truth labeling considers them to be moving because they move later on in the sequence (marple6, marple11). In this set, the finite-K model with a large number of components and the infinite-K model perform better than the finite-K model with a small number of components.

Among the SegTrack data set, three videos (marked with *) have multiple moving objects, but the ground truth intended for tracking analysis marks only one primary object as the foreground, causing our system to appear less accurate. We chose to retain the original ground truth labeling and report the numbers as seen. In this set, the performance of the finite-K and the infinite-K models are comparable on most videos. The exception is the parachute video, which actually contains a significant amount of rotation. Because the observed background optical flow orientations are not explainable by one dominant translation, the infinite-K model tends to break the background into smaller segments.

Finally, in the new ComplexBackground videos taken with a hand-held camera, rotation is a major challenge. In general all the methods are less accurate on this set. For the finite-K model a moderate number of motions (46) is most successful in videos where there is rotation (ComplexBackground set and parachute). A small or large number of finite motions or the infinite motion model are both less successful on these videos. Using color information helps in many of these videos for all models. The forest video has the additional challenge that the foreground object moves very slowly in many frames. Despite these challenges in the new videos, our system performs segmentation with reasonable accuracy across multiple data sets. Figure 4.6 shows a few sample segmentation results from four videos.

The infinite-K model is more sound and achieves better segmentation in videos that do not have significant amount of camera rotation. It yields the highest accuracy for the data set which satisfies the assumption of translation motion and returns moderate accuracy on the other data sets. An algorithm to handle camera rotation, discussed in Chapter 5, makes the infinite-K model suitable for a wide range of videos. It has the distinct advantage of not requiring that the number of motions be tuned for each video or data set.

The most relevant papers for foreground-background classification are Kwak *et al.* [38], and Elqursh and Elgammal [18]. Other papers that use the Hopkins data [70, 4, 50] report sparse trajectory classification results for each frame which are not directly comparable to foreground-background classification accuracy measures.

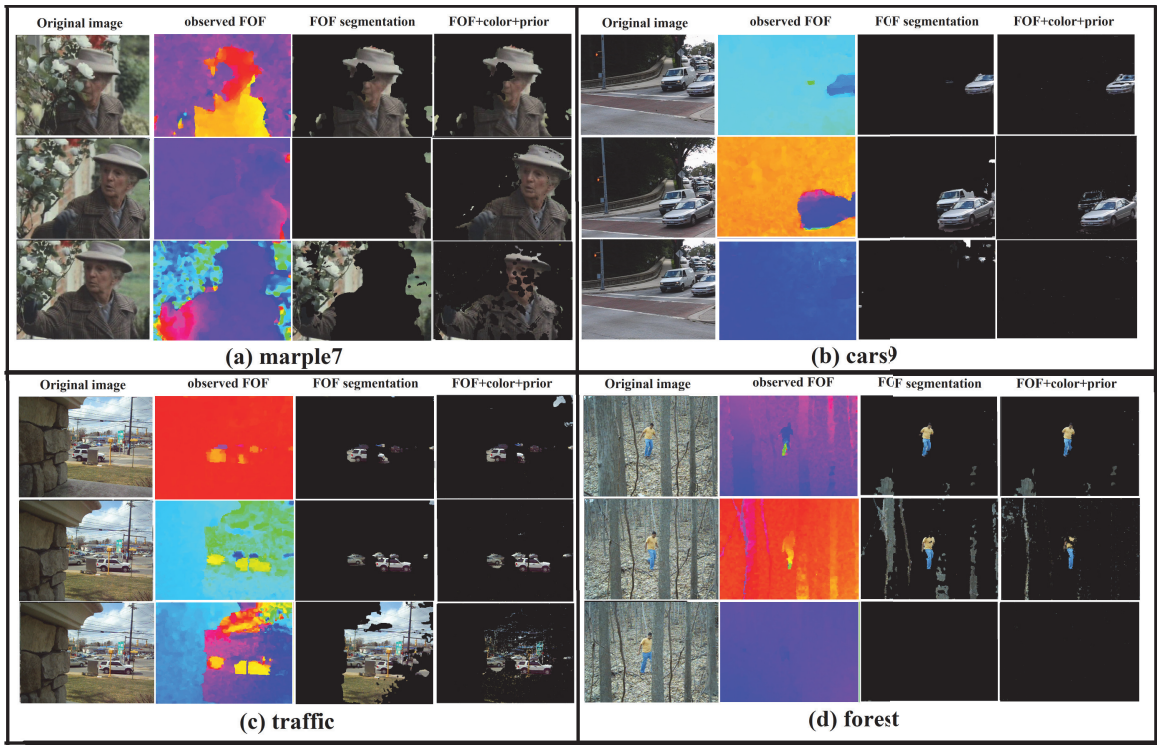

Figure 4.6. Sample results from four videos. The columns are the original image, the observed FOF, FOF segmentation results, and results from combining FOF with color and prior models, respectively. FOF is very accurate when the foreground objects' FOFs are easily distinguishable from the camera motion's FOF. When the observed FOF cannot distinguish between the foreground and the background, FOF segmentation is not accurate. Color and prior information can help in these cases (row 2 in (a)). If the foreground object is not obvious from the FOF for a long duration, the color and prior too are unable to help recover them after some time (row 3 in (b) and (d)). In the new videos(c and d), camera rotation is a challenge (row 3 in (c) and row 2 in (d)). Occasionally, the largest detected segment is the foreground object, which gets labeled as background (row 3 in (c)). Using a prior helps reduce this error as well as errors due to rotation.

Elqursh and Elgammal perform a spectral clustering of trajectories and obtain a dense labeling of pixels. However, segmentation of each frame is performed by considering trajectory information from the current frame as well as four future frames. FOF segmentation is a frame-to-frame segmentation method and hence solving a different problem with the aim of achieving real-time processing of frames.

Kwak *et al.* report results on 3 of the 26 videos in the Hopkins data set, where they use a special initialization procedure to segment the object of interest in the first frame. For the *Cars1*, *People1*, and *People2* videos, they report average F-measure values of .88, .94, and

.87, respectively. FOF segmentation which makes no assumptions about the first frame and does not require an initialization step is not as accurate on the first two videos. In particular, as shown in the *Cars1* video in Figure 4.4 (last column), a heavy penalty is paid when our bootstrapped system fails to detect the object in the first frame. The initialization procedure used by Kwak *et al.* was not reproducible because of lack of detail in the paper. Further, through correspondence with the authors, it was found that in their modified ground truth, they do not consider smaller moving objects other than the primary tracked object to be foreground. Thus, while their reported F-measures are presented here for completeness, a direct comparison of the numbers is not very meaningful.

Videoname	Finite K model K=23	Finite K model K=46	Finite K model K=92	Infinite K model
Hopkins set				
Cars1	47.87	**47.81**	47.61	47.75
Cars2	64.10	46.37	**64.01**	63.16
Cars3	66.86	67.18	67.35	**67.92**
Cars4	28.48	**38.51**	28.65	38.41
Cars5	62.08	**64.85**	63.00	63.54
Cars6	80.11	78.09	**80.47**	79.57
Cars7	55.61	37.63	11.43	**66.83**
Cars8	86.73	**87.13**	86.94	86.78
Cars9	50.03	**68.99**	58.28	60.60
Cars10	56.59	53.98	**64.29**	61.20
Marple1	79.09	65.65	**81.54**	77.59
Marple2	60.50	49.68	**62.84**	59.81
Marple3	78.10	67.83	78.81	**79.05**
Marple4	51.91	61.33	60.75	**63.91**
Marple5	50.36	50.05	**50.36**	**50.36**
Marple6	32.85	26.95	31.81	**33.43**
Marple7	56.03	51.57	**63.74**	59.57
Marple8	69.43	68.89	**80.88**	**80.88**
Marple9	55.70	40.53	**61.63**	58.10
Marple10	32.48	**57.19**	28.60	32.45
Marple11	37.42	37.33	**38.28**	36.75
Marple12	63.90	**65.83**	63.90	63.90
Marple13	69.38	67.09	69.17	**71.53**
People1	53.15	56.76	56.51	**58.77**
People2	**85.45**	85.35	85.08	85.17
Tennis	59.54	61.63	59.67	**63.79**
Segtrack set				
birdfall2	**68.68**	**68.68**	**68.68**	**68.68**
girl	75.92	75.73	**76.07**	75.70
parachute	49.36	**51.49**	46.01	09.54
cheetah*	**13.47**	12.68	10.34	13.26
monkeydog*	10.80	10.79	10.18	**11.94**
penguin*	15.76	14.74	**16.16**	15.39
ComplexBackground set				
drive	31.20	30.13	16.16	**31.51**
forest	18.28	**19.48**	10.61	14.04
parking	37.03	**43.47**	22.92	36.67
store	16.67	**28.46**	16.87	23.13
traffic	56.72	**66.08**	55.94	42.29
Hopkins mean	58.99	57.85	59.45	**61.95**
Segtrack mean	38.99	**39.02**	37.91	32.42
ComplexBg mean	31.98	**37.52**	24.50	29.53
average all videos	52.10	52.05	51.23	**52.78**

Table 4.1. Results. F-measure value for all videos in three data sets using FOF (no color modeling)

Videoname	Finite K model K = 23	Finite K model K = 46	Infinite K model K = 92	Infinite K model
Hopkins set				
Cars1	49.48	**50.84**	49.26	49.09
Cars2	71.14	56.60	72.78	**74.72**
Cars3	74.16	73.57	**75.24**	67.51
Cars4	48.10	47.96	50.32	**51.11**
Cars5	69.11	70.94	**71.17**	67.48
Cars6	83.88	84.34	84.94	**85.24**
Cars7	61.54	42.92	27.25	**83.53**
Cars8	87.27	**87.61**	87.32	87.35
Cars9	50.79	**66.38**	52.59	59.19
Cars10	50.14	50.84	**53.27**	52.29
Marple1	92.39	88.25	**92.54**	90.93
Marple2	66.01	60.88	69.22	**73.18**
Marple3	79.83	70.71	**84.64**	81.15
Marple4	67.64	**69.01**	57.69	67.79
Marple5	**45.25**	45.15	45.24	45.24
Marple6	35.03	23.95	36.06	**36.41**
Marple7	72.45	67.13	74.01	**75.96**
Marple8	**83.66**	80.32	77.08	77.08
Marple9	44.20	36.36	**70.96**	56.37
Marple10	48.00	**58.72**	47.40	46.89
Marple11	**41.66**	41.41	41.09	40.47
Marple12	66.53	**70.01**	64.92	68.71
Marple13	83.71	80.96	**83.80**	82.40
People1	65.82	69.53	66.80	**71.07**
People2	87.97	88.40	**88.72**	88.06
Tennis	66.18	**67.59**	66.26	63.61
Segtrack set				
birdfall2	**75.69**	**75.69**	**68.68**	**75.69**
girl	81.58	**81.95**	81.36	81.71
parachute	20.71	**54.36**	12.20	46.13
cheetah*	**24.15**	22.31	15.45	22.96
monkeydog*	19.47	18.62	19.58	**21.05**
penguin*	23.98	20.71	**24.81**	22.38
ComplexBackground set				
drive	50.04	**61.80**	**37.46**	43.34
forest	25.00	**31.44**	17.07	22.45
parking	56.84	**73.19**	38.58	36.34
store	51.99	**70.74**	24.87	23.20
traffic	67.63	**71.24**	65.75	50.17
Hopkins mean	65.07	63.48	65.02	**67.03**
Segtrack mean	40.93	**45.61**	38.18	44.99
ComplexBg mean	50.30	**61.68**	36.75	35.10
average all videos	59.16	**60.34**	56.85	59.14

Table 4.2. Results. F-measure value for all videos in three data sets using FOF along with color and prior modeling

4.7 More comparisons

In this section, additional comparisons with some other approaches is presented. Comparison of our segmentation results to various other methods is presented in Section 4.7.1. In Section 4.7.2, we compare our model to other models with optical flow orientations as the input.

4.7.1 FOF versus flow vector-based segmentations

In Section 4.3, FOF segmentation results were compared to spectral clustering results of Ochs and Brox [50], which represented the best method among many that were experimented with. K-means, Dirichlet process Gaussian mixture model, and the spectral clustering method of Elqursh and Elgammal [18], were among the other methods used. Sample qualitative results from all methods are given in Figure 4.7. The first four rows are the input image, a visualization of the optical flow vectors [63], the optical flow magnitudes, and optical flow orientations respectively. The optical flow vectors appear to have similar values for all background pixels when the background is relatively simple and at roughly uniform depth from the camera. When background objects are at varying depths (columns 3 and 4), their flow vectors are not uniform. The magnitudes of the vectors (row 3) depend heavily on object depth. Optical flow orientations (row 4), arguably, are the most reliable indicators of independent motion. For orientations, it may be noted that the color blue represents 0 degrees and red represents 360 degrees. Hence they should be considered as equivalent (for instance, in column 3).

As a baseline method, K-means was used with the flow vectors as input. Rows 5, 6, and 7 show the results of K-means clustering for $K = 2$, $K = 3$, and $K = 5$, respectively. The results are highly sensitive to the value of K. For videos with simple backgrounds (columns 1, 2, and 5), small values of K work well. For complex background videos (columns 3 and 4), there is no value of K that yields good results. This can be seen in row 8 where human judgement was used to pick the best results from many different K values.

Since requiring knowledge of K beforehand severely limits the use of K-means to general video settings, a non-parametric mixture model with optical flow vectors as the features is presented next. Results from the accelerated variational Dirichlet process Gaussian mixture model (DPGMM) implementation of Kurihara *et al.* [37] are shown in row 9. Although DPGMM is non-parametric and can adapt to the complexity in the data, the use of optical flow vectors as the features causes the method to over-segment the background.

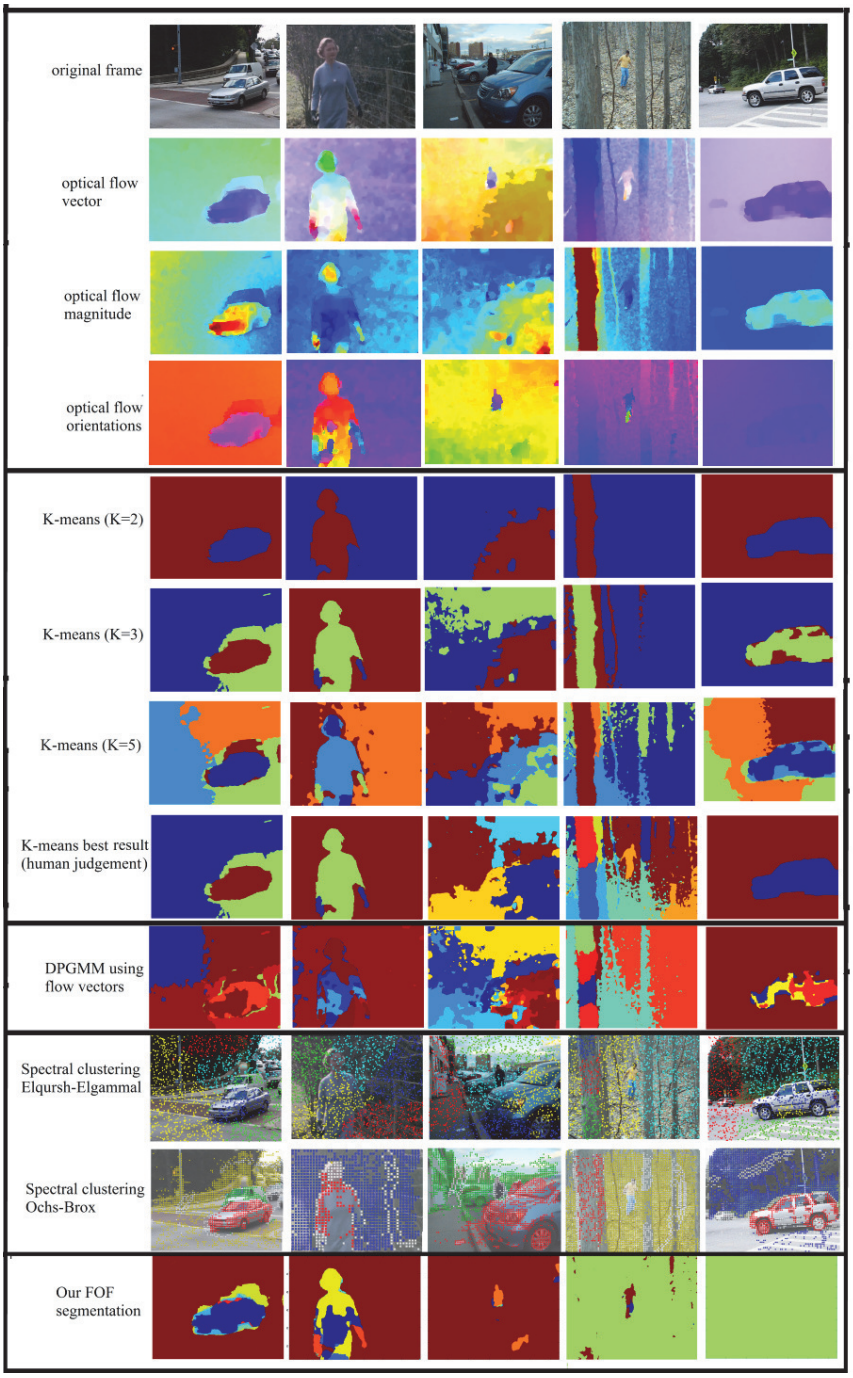

Figure 4.7. Comparison of FOF segmentation to several other optical flow vector-based methods.

Spectral clustering has been shown to be useful for motion segmentation by clustering tracked keypoints based on their long-term trajectories. Elqursh and Elgammal [18]

find a low-dimensional embedding for trajectories from 5 consecutive frames. However, their method tends to separate the background into several clusters. In order to apply their method for background subtraction, they assume that the background is a Gaussian mixture model of 5 components. Row 10 shows keypoints with the colors representing cluster memberships for their algorithm with a mixture model of 5 components[2]. It is not clear whether using a mixture of 5 components would work for across all videos. In some videos, the foreground object keypoints clearly form a separate cluster (column 1 and 5), but in others, they are grouped with the background.

Finally, the spectral clustering of Ochs and Brox [50] which represents the state of the art for segmentation of trajectories is shown. This method considers interaction between triplets of keypoints instead of simply using pairwise distances for the clustering. Further, they use some post-processing to merge regions with similar motion properties. The results from their implementation[3] that includes a merging step is given in row 11. Again, the results show keypoints with the colors representing their clusters. The method works well when the background is fairly at a uniform distance from the camera. Complex backgrounds, however, still suffer from over-segmentation. Both the above spectral methods result in labels for sparse keypoints. For a dense labeling of all pixels, additional processing is required. In contrast, FOF segmentation directly returns a dense labeling of the image.

The results in the last row show the efficacy of FOF-based segmentation across different scenarios. Note that these results are from the orientation-based segmentation alone, without any use of color or prior information. Foreground objects are broken down into smaller segments depending on their motion, but all of the background is correctly identified as one segment. The method is prone to failure when the object's motion happens to be in a direction that is consistent with the orientation field due to the camera's motion (column 5). The object will go undetected until it or the camera change their motion direction. In the above video, the object is detected after 5 frames.

---

[2]Our own implementation of [18]

[3]The number of tracked frames is set to 3 in their method for a fair comparison

4.7.2 Our model versus other models using flow orientations

Section 4.3 shows the advantages of using flow orientations compared to flow vectors. In this section, we compare other models to ours while keeping the feature representation common - flow orientations. Using flow orientations as the common feature representation, our segmentation model is directly compared to K-means and DPGMM in Figure 4.8. Once again, results from K-means (rows 3, 4, and 5) are sensitive to the choice of K. When the correct value of K is provided by human judgement, the resulting segmentations are reasonable (row 6), but still fall short of our FOF results in the last row. For complex backgrounds (columns 3 and 4), K-means with flow orientations results in a better segmentation than K-means with flow vectors in Figure 4.7. DPGMM automatically determines the number of components, but tends to break the background into smaller sections. Our segmentation method, by explicitly modeling the spatial dependency between pixels through the "field" of orientations for a given set of motion parameters, results in a better segmentation compared to both human-assisted K-means and DPGMM.

4.8 Discussion

We have presented a system that performs motion segmentation that uses optical flow orientations. While using optical flow can cause the segmentation of objects with similar real-world motion into separate objects depending on their depth, the use of orientations can avoid such over-segmentation. Our system is able to determine the number of foreground motions automatically. We have shown promising results on a wide range of videos including some that have very complex backgrounds. The main drawback of our method is that we assume that only translational motion is present in the camera. Although it works for a majority of frames, the system is prone to error when the camera rotates. Explicitly modeling the camera rotation could help handle such cases. Incorporating magnitude information in the model can help further improve results, especially in cases where a tracked foreground object suddenly disappears in the FOF observations. Finally, our appearance model update procedure is extremely simple and can be improved by using a sophisticated scheme as described by Kwak *et al.* .

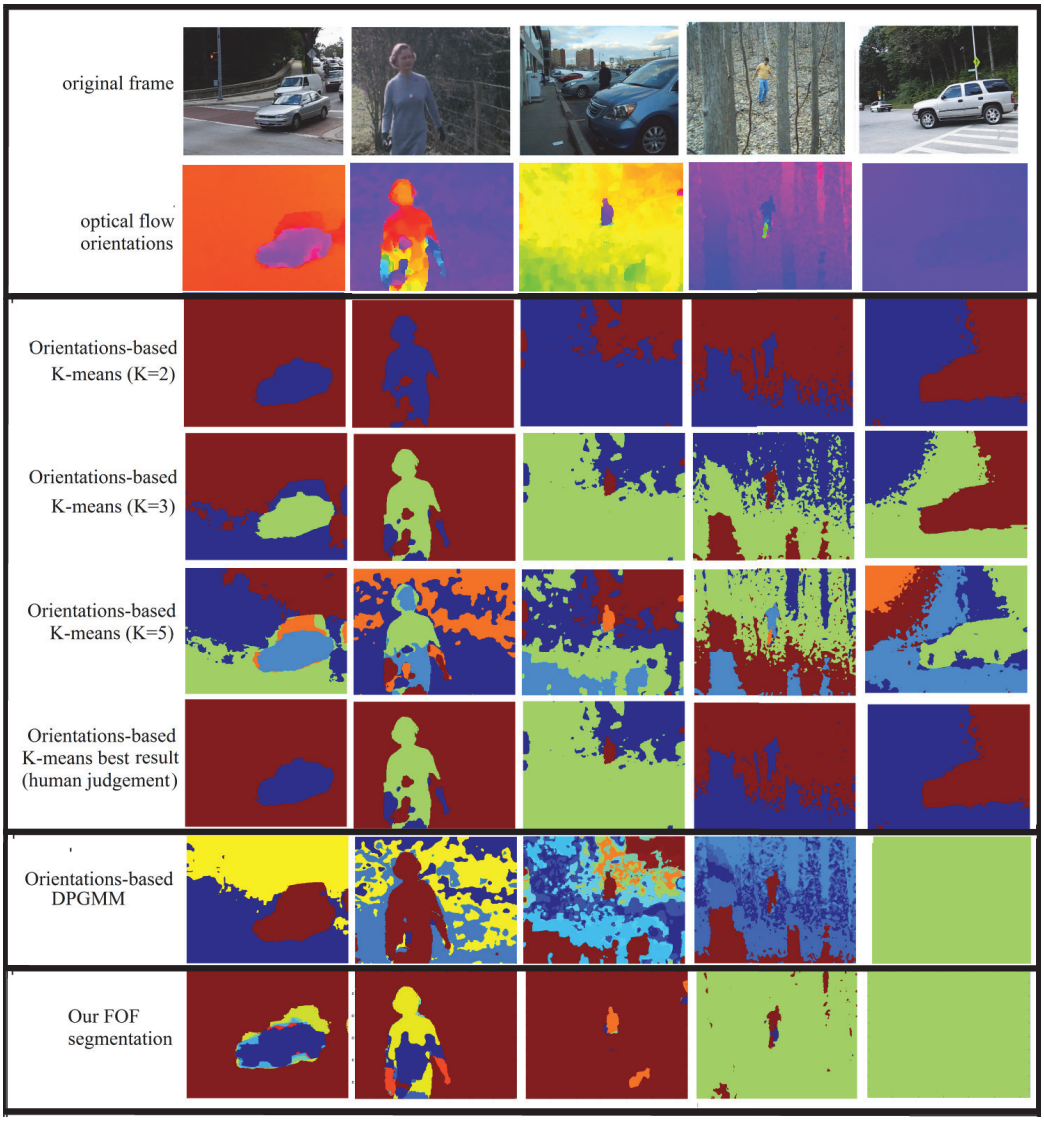

Figure 4.8. Comparison of our FOF model to other methods using orientation fields.

CHAPTER 5

MODELING COMPLEX CAMERA MOTIONS BY ROTATION COMPENSATION

In Chapter 4, we discussed a model to achieve segmentation by predicting the translational motion parameters for the camera and the moving objects in the scene and using the resulting optical flow orientations. One of the assumptions in Chapter 4 was that the camera's motion is only translational. In this chapter, a rotation compensation algorithm is presented that enables the application of the FOF segmentation algorithm to videos in which there is both translation and rotation of the camera.

5.1 Introduction

The assumption of pure translation in the camera allows us to make use of the depth-independence property of the resulting optical flow orientations. A majority of the videos in the Hopkins and Segtrack data sets satisfy the assumption of translation motion and good segmentation results are achieved on these videos. In videos where rotation is a significant component of the camera's motion, the depth-independence property of optical flow orientations is lost. While a small amount of tolerance to rotation is observed in the results, an explicit rotation compensation algorithm would be useful for handling general videos such as those taken using hand-held cameras.

It is interesting to note another depth-independent aspect of optical flow: for pure rotation motion in the camera, the resulting optical flow vectors do not depend on the depth of the objects in the scene. In this chapter, this property is leveraged along with the depth-independence of translational optical flow orientations to design a robust segmentation algorithm. The rotation compensation algorithm helps achieve a much more reliable segmentation when complex camera motion is present.

Estimation of a camera's translation and rotation from the observed optical flow in images is a related and well studied problem. In camera motion estimation, also known as egomotion estimation, the relationship between the camera's translation and rotation to

the observed optical flow and the constraints that the two motions place on the optical flow are exploited in different ways to estimate the motion parameters. Jepson and Heeger [32] constructed a set of constraints for optical flow observations in a manner that effectively negates the rotation component. Prazdny [54] showed that the difference between any two flow vectors yields a constraint that does not depend on rotation. Tomasi and Shi [68] estimated translation by using the property that the change in angular distance between a pair of image points is independent of camera rotation. In the above methods, translation is first estimated using the rotation invariant constraints, followed by rotation estimation. Prazdny [53] estimated the rotation first by using constraints that are independent of camera translation and the depth of the points in the scene. These and other methods for egomotion estimation are described by Tian *et al.* [67]. Reliable estimation of egomotion is difficult in the presence of independently moving objects, which is commonly the case in motion segmentation scenarios. We are not directly interested in computing the egomotion, but rather in transforming the observed optical flow vectors so that the flow vectors due to camera rotation, which cause the flow orientations to become depth-dependent, can be removed.

Recent work by Yamaguchi *et al.* [78] used a rotation compensation procedure in their optical flow estimation algorithm. They estimated the rotation flow at each pixel so that when the rotational flow vectors are subtracted from the observed flow vectors, a pure translational flow field is obtained. Their method estimates the fundamental matrix for consecutive frames and uses epipolar constraints to arrive at the rotational flow. The fundamental matrix basically establishes constraints between matching points in two images from a stereo camera or two consecutive frames in a video sequence. It is a 3x3 matrix that defines epipolar lines in an image pair. Given matching points p_1 and p_2 in homogenous coordinates,[1] the epipolar constraint is that the point p_2 lies on the line Fp_1 in the second image, also called the epipolar line. Yamaguchi *et al.* use the property that application of rotational flow to the points in the first image causes the resulting points to lie on the epipolar line in the second image. The fundamental matrix is typically estimated by first finding corresponding matching points across two images. Although their algorithm is designed for scenes that do not contain independently moving objects, robust methods are available for computing the fundamental matrix in the presence of outliers. This makes the Yamaguchi *et al.* algorithm a potential candidate for use in our task of motion segmenta-

[1] Homogenous coordinates for a point $p = (x, y)$ are $(x, y, 1)$

tion where independently moving objects would be outliers that can affect the fundamental matrix computation. Direct comparison of the methods shows that the Yamaguchi *et al.* algorithm depends heavily on the accuracy of the estimated fundamental matrix and that the proposed orientation fields based rotation compensation is more robust.

In Section 5.2, the nature of optical flow when there is rotation in the camera and an algorithm to compensate for rotation are discussed. Synthetic examples that illustrate the algorithm are in Section 5.3. Some examples from real videos that highlight the algorithm, its strengths, and its drawbacks are presented in Section 5.4. In practice, it is useful to perform rotation compensation only when necessary. In Section 5.5, we discuss an algorithm that determines whether the observed optical flow vectors necessitate the use of the rotation compensation procedure. Results on the benchmark data sets are presented in Section 5.6.

5.2 Modeling and compensating for the flow due to camera rotation

Optical flow due to a camera's motion can be broken down into the flow due to the translational motion and the flow due to the rotational motion of the camera. Assume that a camera's translation and rotation parameters are (t_x, t_y, t_z) and (r_x, r_y, r_z) respectively. With perspective projection, a point at location (X, Y, Z) in the real world projects to the location $(x, y) = (X/Z, Y/Z)$ in the image. The optical flow values due to camera motion along the x (column) and y (row) dimensions are given by

$$\begin{aligned} v_x(\mathbf{x}) &= (\frac{t_z x - t_x f}{Z}) + (-r_y f + r_z y + \frac{r_x xy}{f} - \frac{r_y x^2}{f}) \\ v_y(\mathbf{x}) &= (\frac{t_z y - t_y f}{Z}) + (r_x f - r_z x - \frac{r_y xy}{f} + \frac{r_x y^2}{f}), \end{aligned} \tag{5.1}$$

where $\mathbf{x}$ is the pixel location (x, y) and f is the focal length of the camera.

The flow components clearly separate into flow due to the translational component and flow due to the rotational component of the camera motion. We saw in Equation 4.2 that the orientation of the translational component does not depend on the point depth Z. It may also be noted that the rotational component of the flow does not have a Z term and is thus independent of the point depth. We propose an algorithm that makes use of these two independence properties of optical flow to compensate for rotation, thus enabling a more accurate segmentation of independently moving objects.

We reparameterize the rotational flow components from Equation 5.1 as done by Yamaguchi *et al.* [78]:

$$\begin{aligned} v_x^r &= (r_1 - r_3 y + r_4 x^2 + r_5 xy) \\ v_y^r &= (r_2 + r_3 x + r_4 xy + r_5 y^2), \end{aligned} \tag{5.2}$$

where the superscript r represents the rotational component of optical flow.

Our rotation compensation algorithm basically tries to find the rotation flow vectors which when subtracted from the observed optical flow vectors results in adjusted orientation values that are very close to some translation motion orientation field. We start with some initial estimate for translation that results in an orientation field that has the minimum absolute sum of errors from the observed orientation field. Next, the 3 translation and 5 rotation parameters are found such that the difference between the orientation field due to the translation parameters and the adjusted observed orientation field, which are the orientations of the observed flow vectors after rotation flow vectors corresponding to the rotation parameters have been subtracted from them, is minimized. Mathematically,

$$(\hat{\mathbf{r}}, \hat{\mathbf{t}}) = \arg\min_{\mathbf{r},\mathbf{t}} \Sigma_i (|\arctan(\frac{o_{i,y} - v_{i,y}^r}{o_{i,x} - v_{i,x}^r}) - F(\mathbf{t}, \mathbf{x}_i)|), \tag{5.3}$$

where F represents the translation orientation field as described in Equation 4.2, $\mathbf{r}$ is the vector $(r_1, r_2, r_3, r_4, r_5)$, $\mathbf{t}$ is the vector (t_x, t_x, t_z), $o_{i,y}$ and $o_{i,x}$ are the observed optical flow vectors in the y and x directions, and $v_{i,y}^r$ and $v_{i,x}^r$ are the predicted rotation flow components along y and x directions for pixel i.

Performing an unconstrained minimization often yields an undesirable result that the returned solution is one where a very large rotation is hypothesized and the resulting rotation compensation makes the translation flow insignificant. To avoid this situation, we constrain the solution such that the returned rotation parameters result in a rotational optical flow whose magnitude is not larger than the observed optical flow magnitude at each pixel.

5.3 Synthetic examples

In this section, we illustrate the algorithm and the effect of rotation compensation on synthetic data. For the interested reader, the thesis appendix (Section A.2) lists some syn-

thetic examples that illustrate that the flow due to a camera's translation and rotation are commutative. Figures 5.1 through 5.4 illustrate the rotation compensation algorithm. In each of the figures, (a) shows the depth map of the objects in the scene. The scene is one of gradually increasing depth from the sides of the image to the center. There are also planar objects placed at different depths from the camera. Dark blue corresponds to a depth of 0 and light blue corresponds to a depth of 50 units from the camera. None of the objects in the scene are moving on their own. The only observed motion is due to the camera's motion. Observed optical flow samples for the scene for a given motion parameter setting (listed above (a)) are shown as arrows in (a). (f) shows the resulting orientation field from the flow vectors in (a). (b) shows the dominant translation flow orientation field (FOF) obtained by performing gradient descent only for the translation parameters with the objective of minimizing the sum of absolute errors between the dominant translation FOF and the observed FOF in (f). (g) shows the difference between the dominant translation FOF and the observed FOF, with dark blue corresponding to 0 and red corresponding to 60 degrees. If no rotation were present in the camera, this image would be all zero. The presence of rotation causes a large difference between the dominant translation FOF and the observed FOF. Using (g) as the basis for segmentation could lead to erroneously inferring that many high difference regions are independently moving objects, when in reality all the objects in the scene are stationary. (c) shows the dominant rotation flow as estimated by our algorithm in Equation 5.3. The corresponding rotation FOF is shown in (h). Subtracting the estimated rotation flow in (c) from the observed flow in (a) results in the rotation compensated flow in (d). The FOF corresponding to the rotation compensated flow is shown in (i). (e) shows the dominant translation FOF obtained from the minima translation parameters returned from Equation 5.3. Note that this differs significantly from the initial dominant translation FOF in (b). The effectiveness of the rotation compensated translation FOF is clearly visible when the difference between it and the rotation compensated flow FOF (i) are observed in (j). The difference image shows very low values and is a significant improvement over (g) for the purposed of identifying that all the objects in the scene are non-moving.

It must be noted that the proposed rotation compensation algorithm is not designed to find the exact rotation parameters of the camera. The algorithm finds the translation and rotation settings that best explain the observed optical flow vectors and orientations. It is possible that many different combinations of translation and rotation yield similar optical flow. The algorithm may find one of these possible settings which is not guaranteed to be the exact set of parameters that actually caused the observed optical flow. For the pur-

pose of motion segmentation, we are not as concerned about computing the exact motion parameters as we are in achieving reliable segmentation.

The rotation compensation algorithm does not always work well. For instance, in Figure 5.5, we see a case where the rotation compensation is not able to return a low difference image (j). This happens when the initial translation FOF (b) is very unreliable.

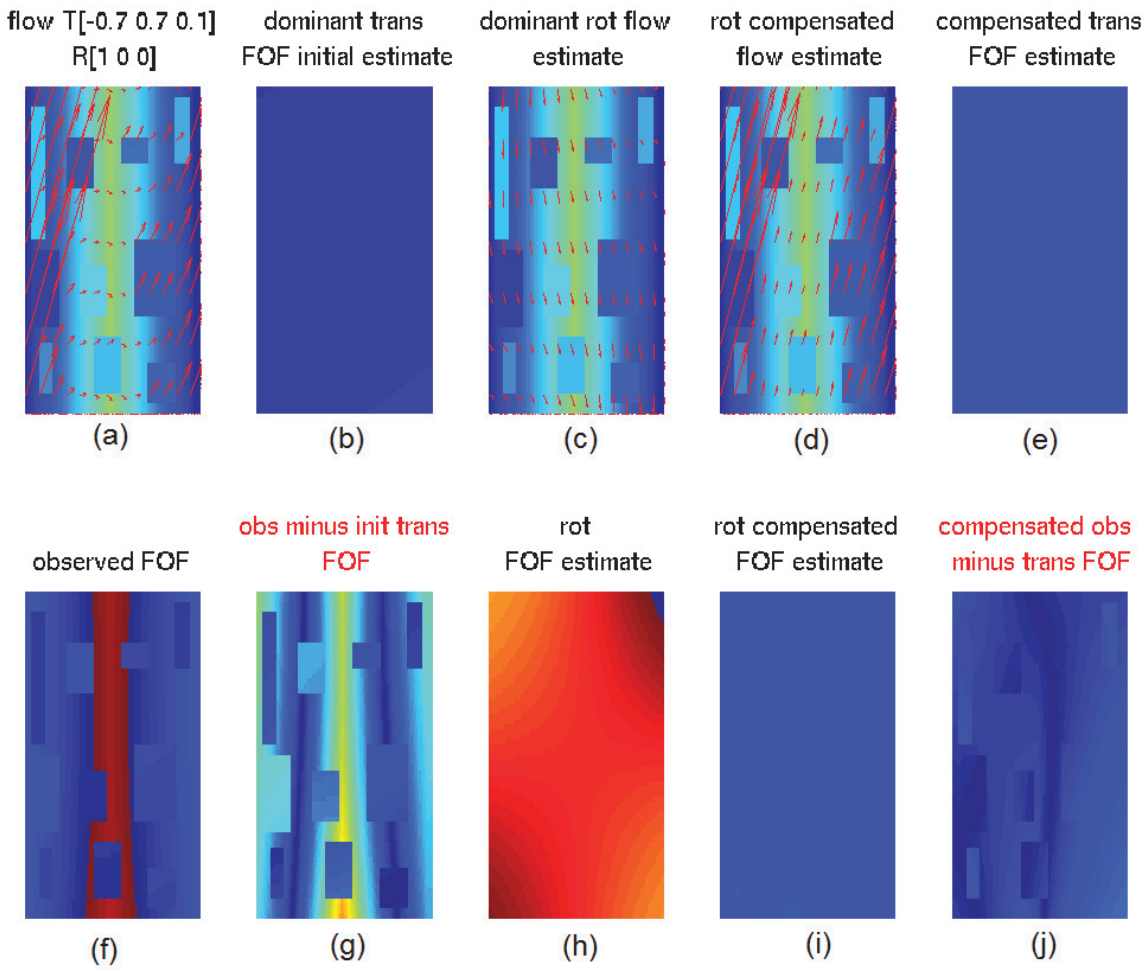

Figure 5.1. Rotation compensation algorithm illustration 1. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50) and arrows showing the optical flow due to camera motion (there are no moving objects in this scene). (f) observed orientation field (FOF) due to flow in (a). The presence of rotation in the camera makes orientations depth dependent. (b) Dominant translation FOF (without considering any rotation compensation). (g) Difference between observed FOF and dominant translation FOF. Depth-dependent nature of flow orientations in the presence of camera rotation makes them less ideal for use as a basis for segmentation. Although all objects are stationary, many regions in the difference image exhibit high difference values (light blue). Dark blue corresponds to a value of 0. (c) Dominant rotation flow estimated using the rotation compensation algorithm in Equation 5.3. (h) FOF corresponding to the estimated rotation flow in (c). (d) Observed flow (a) after subtracting rotation flow (c) from it. (i) FOF corresponding to the rotation compensated flow in (d). (e) Dominant translation FOF considering the rotation compensation using Equation 5.3. This is significantly different from (b). (j) Difference between dominant rotation compensated translation FOF and rotation compensated observed FOF. (j) shows a low difference value for the entire scene, which is a significant improvement over the result in (g). (j) is hence a much more reliable basis for detecting independently moving objects in the scene.

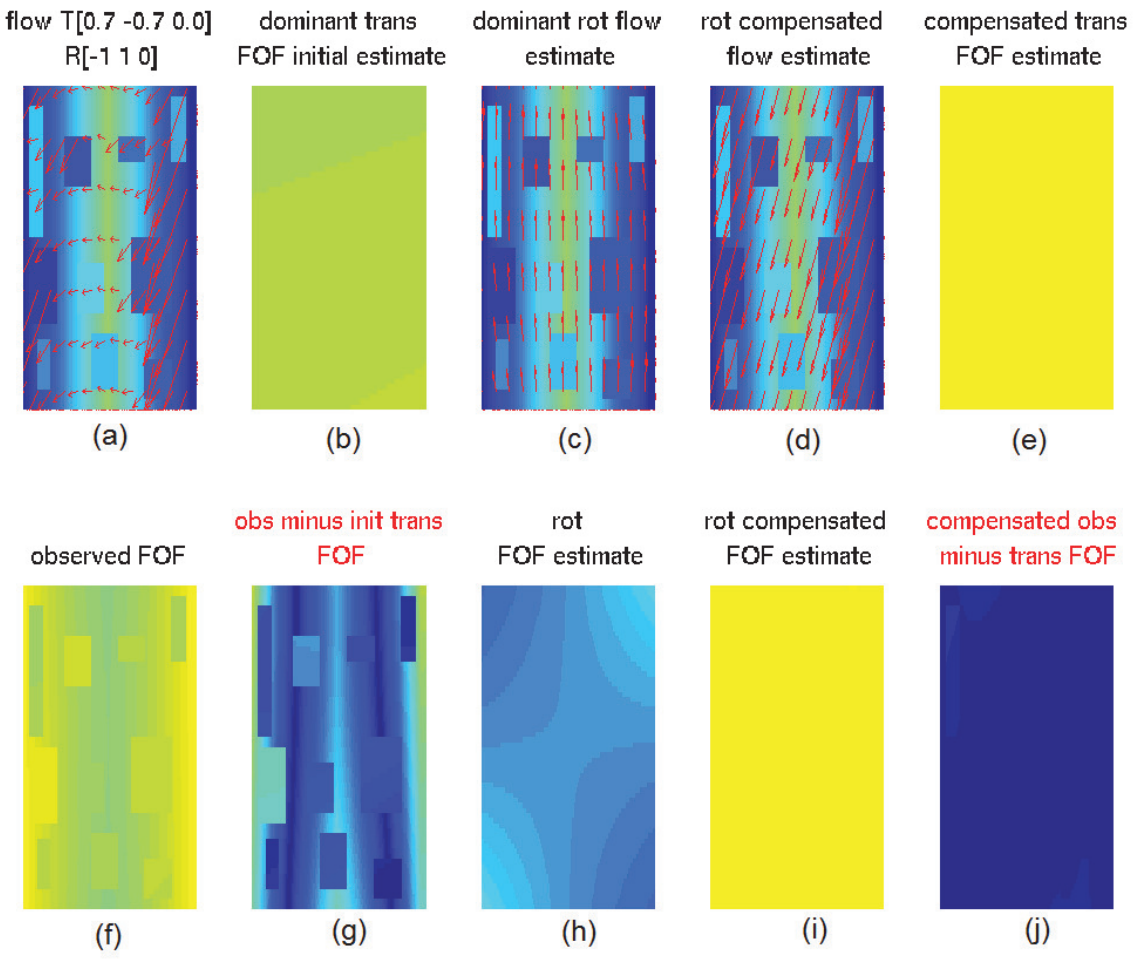

Figure 5.2. Rotation compensation algorithm illustration 2. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50) and arrows showing the optical flow due to camera motion (there are no moving objects in this scene) (f) observed orientation field (FOF) due to flow in (a). The presence of rotation in the camera makes orientations depth dependent. (b) Dominant translation FOF (without considering any rotation compensation) (g) Difference between observed FOF and dominant translation FOF. Depth-dependent nature of flow orientations in the presence of camera rotation makes them less ideal for use as a basis for segmentation. Although all objects are stationary, many regions in the difference image exhibit high difference values (light blue). Dark blue corresponds to a value of 0. (c) Dominant rotation flow estimated using the rotation compensation algorithm in Equation 5.3. (h) FOF corresponding to the estimated rotation flow in (c). (d) Observed flow (a) after subtracting rotation flow (c) from it. (i) FOF corresponding to the rotation compensated flow in (d). (e) Dominant translation FOF considering the rotation compensation using Equation 5.3. This is significantly different from (b). (j) Difference between dominant rotation compensated translation FOF and rotation compensated observed FOF. (j) shows a low difference value for the entire scene, which is a significant improvement over the result in (g). (j) is hence a much more reliable basis for detecting independently moving objects in the scene.

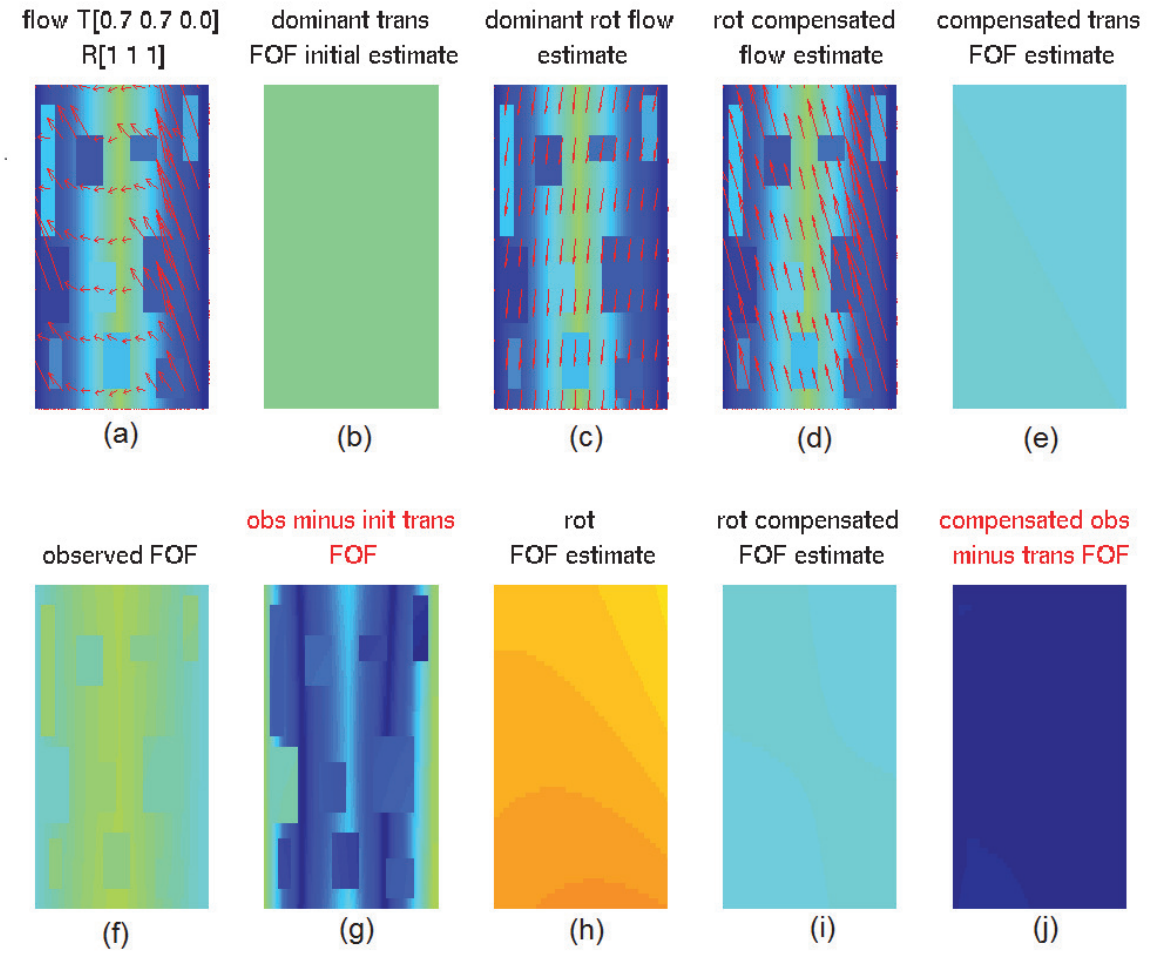

Figure 5.3. Rotation compensation algorithm illustration 3. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50) and arrows showing the optical flow due to camera motion (there are no moving objects in this scene) (f) observed orientation field (FOF) due to flow in (a). The presence of rotation in the camera makes orientations depth dependent. (b) Dominant translation FOF (without considering any rotation compensation) (g) Difference between observed FOF and dominant translation FOF. Depth-dependent nature of flow orientations in the presence of camera rotation makes them less ideal for use as a basis for segmentation. Although all objects are stationary, many regions in the difference image exhibit high difference values (light blue). Dark blue corresponds to a value of 0. (c) Dominant rotation flow estimated using the rotation compensation algorithm in Equation 5.3. (h) FOF corresponding to the estimated rotation flow in (c). (d) Observed flow (a) after subtracting rotation flow (c) from it. (i) FOF corresponding to the rotation compensated flow in (d). (e) Dominant translation FOF considering the rotation compensation using Equation 5.3. This is significantly different from (b). (j) Difference between dominant rotation compensated translation FOF and rotation compensated observed FOF. (j) shows a low difference value for the entire scene, which is a significant improvement over the result in (g). (j) is hence a much more reliable basis for detecting independently moving objects in the scene.

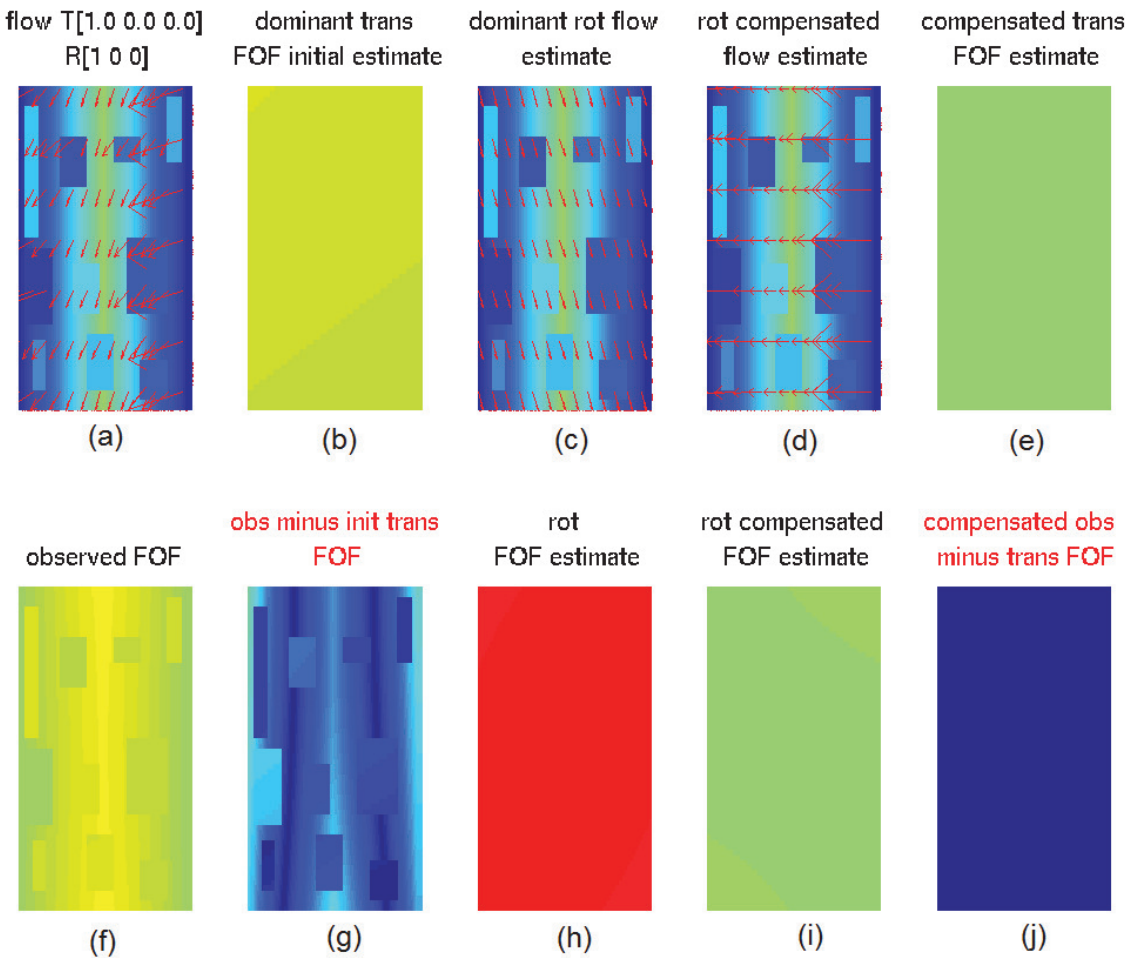

Figure 5.4. Rotation compensation algorithm illustration 4. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50) and arrows showing the optical flow due to camera motion (there are no moving objects in this scene) (f) observed orientation field (FOF) due to flow in (a). The presence of rotation in the camera makes orientations depth dependent. (b) Dominant translation FOF (without considering any rotation compensation) (g) Difference between observed FOF and dominant translation FOF. Depth-dependent nature of flow orientations in the presence of camera rotation makes them less ideal for use as a basis for segmentation. Although all objects are stationary, many regions in the difference image exhibit high difference values (light blue). Dark blue corresponds to a value of 0. (c) Dominant rotation flow estimated using the rotation compensation algorithm in Equation 5.3. (h) FOF corresponding to the estimated rotation flow in (c). (d) Observed flow (a) after subtracting rotation flow (c) from it. (i) FOF corresponding to the rotation compensated flow in (d). (e) Dominant translation FOF considering the rotation compensation using Equation 5.3. This is significantly different from (b). (j) Difference between dominant rotation compensated translation FOF and rotation compensated observed FOF. (j) shows a low difference value for the entire scene, which is a significant improvement over the result in (g). (j) is hence a much more reliable basis for detecting independently moving objects in the scene.

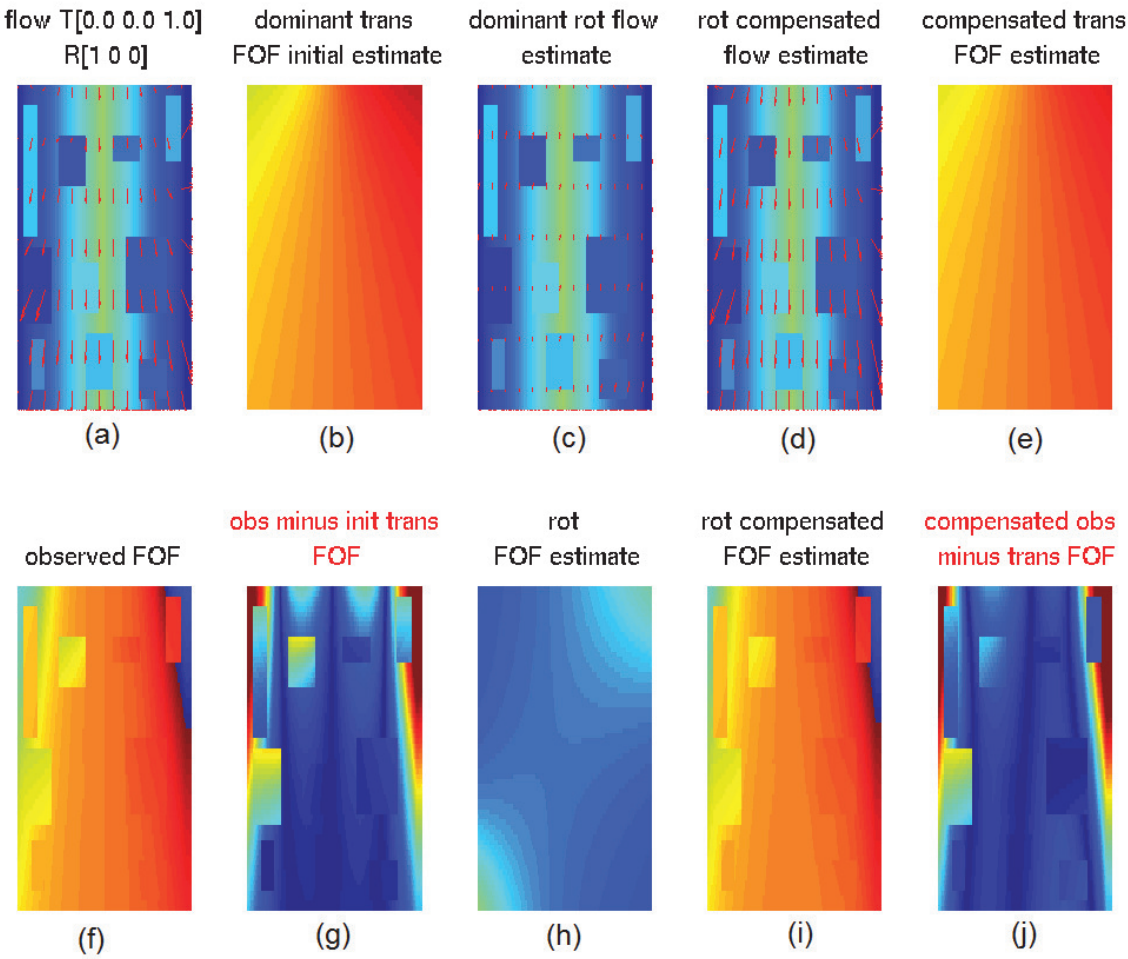

Figure 5.5. Rotation compensation failure illustration 1. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50) and arrows showing the optical flow due to camera motion (there are no moving objects in this scene) (f) observed orientation field (FOF) due to flow in (a). The presence of rotation in the camera makes orientations depth dependent. (b) Dominant translation FOF (without considering any rotation compensation). In this case, the true translation FOF is a radial FOF. Large amount of rotation causes the gradient descent to return an unreliable initial point. (g) Difference between observed FOF and dominant translation FOF. (c) Dominant rotation flow estimated using the rotation compensation algorithm in Equation 5.3. (h) FOF corresponding to the estimated rotation flow in (c). (d) Observed flow (a) after subtracting rotation flow (c) from it. (i) FOF corresponding to the rotation compensated flow in (d). (e) Dominant translation FOF considering the rotation compensation using Equation 5.3. (j) Difference between dominant rotation compensated translation FOF and rotation compensated observed FOF. This is an failure example. Due to a very bad initial estimate for translation FOF, a reliable decomposition into translation and rotation was not possible. The resulting difference image (j) is not any better than the initial difference image (g).

5.4 Real video examples

Figures 5.6 through 5.9 show examples of rotation compensation in real videos. In each of these images, the first row shows the observed flow (a), observed FOF (b), segmentation output (c), and segmentation labels for the image pixels (d). No rotation compensation is applied for the results in (c) and (d). The second row shows the estimated rotation flow (e) and the observed flow after rotation flow has been subtracted from it (f). The FOF of the rotation compensated flow in (f) is shown in (g). Segmentation performed using the rotation compensated FOF results in the output image (h) and segmentation labels shown in (i).

The figures show successful cases where rotation compensation results in a large improvement in the segmentation. It may be noted that in comparison to the original FOF before rotation compensation (b), the visualization of the rotation compensated FOF (g) tends to have a more smooth color distribution for the background pixels. There are fewer discontinuities in the background pixels in the rotation compensated FOF. This is a desirable outcome and demonstrates that the rotation compensation is working as one would expect.

A few failure cases are also presented in Figures 5.10 and 5.11. In the cars10 video, the bus moves very slowly. The FOF values observed on the bus are only slightly different from the background. Without rotation compensation, the FOF segmentation can successfully identify the bus as a moving object. Although there appears to be little rotation in this frame, the rotation compensation algorithm predicts a rotation of a large magnitude. Upon subtracting this large rotation flow from the observed flow, the initial small difference between the bus and background flow orientations gets smoothed out and the bus can no longer be segmented.

Frame numbered 20 in the cars7 video is another example where there does not appear to be rotation in the scene, but rotation compensation returns a rotation of high magnitude. Subtracting the estimated large rotation flow causes the otherwise identifiable car to disappear in the rotation compensated FOF. Note that red color in the FOF image corresponds to 360 degrees and blue corresponds to 0, which means they represent the same orientation and hence indistinguishable. The two failure cases show that performing rotation compensation when there is no rotation in the scene can lead to loss of discriminative power in the FOF observations. This is particularly true if the moving object has flow orientations in roughly the same direction as the camera's FOF, as is the case with the bus in the cars10

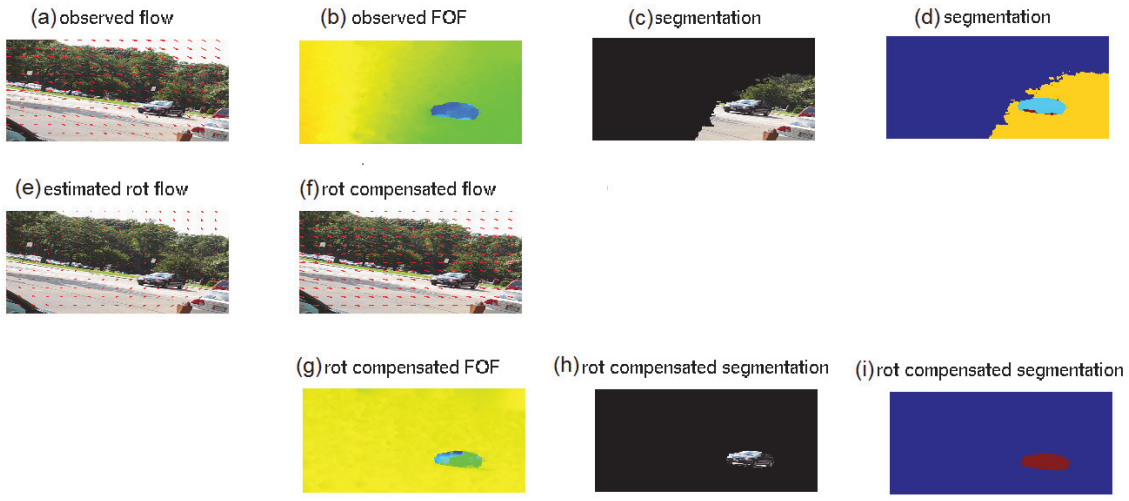

Figure 5.6. Rotation compensation real examples - cars7 video. (a) Observed flow - The camera motion is a counter-clockwise rotation with translation to the left. (b) Observed FOF - camera rotation is evident. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. (i) Segmentation labels returned by the model for the rotation compensated flow orientations.

video and the car in the cars7 video. In the next section, we develop a strategy to avoid using rotation compensation in frames where rotation is clearly not present.

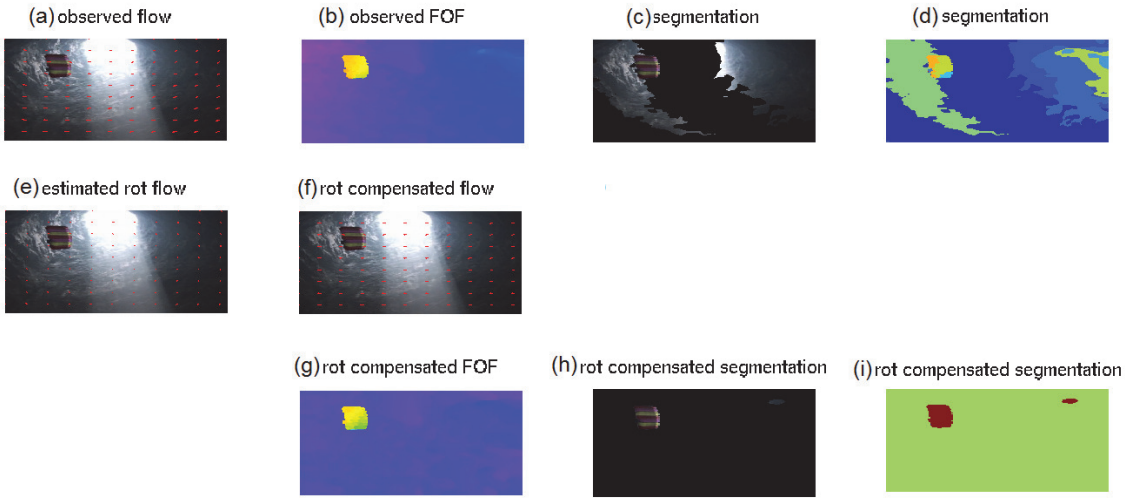

Figure 5.7. Rotation compensation real examples - parachute video. (a) Observed flow - The camera motion is counter-clockwise rotation along with a translation to the right. (b) Observed FOF - camera rotation is evident. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. (i) Segmentation labels returned by the model for the rotation compensated flow orientations.

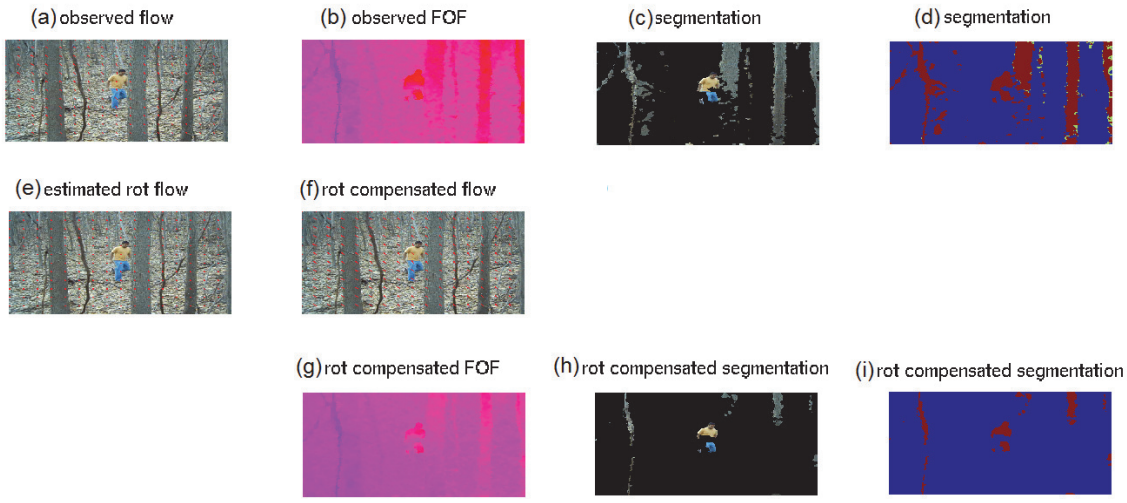

Figure 5.8. Rotation compensation real examples - forest video. (a) Observed flow - The camera motion is mainly translation to the right and downwards along with a slight rotation. (b) Observed FOF - camera rotation is evident. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. (i) Segmentation labels returned by the model for the rotation compensated flow orientations.

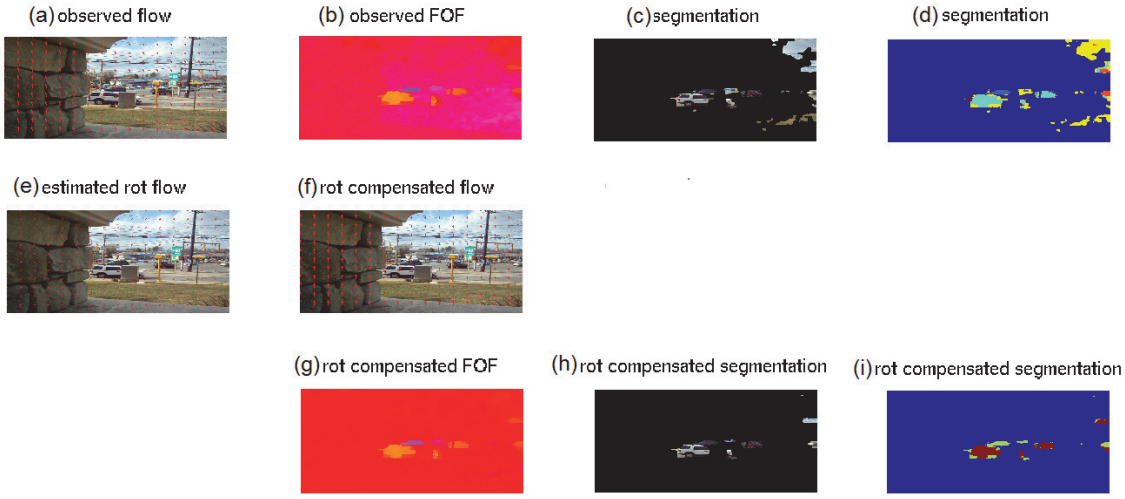

Figure 5.9. Rotation compensation real examples - traffic video. (a) Observed flow - The camera motion is mainly translation downwards with a slight rotation. (b) Observed FOF - camera rotation is evident. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. (i) Segmentation labels returned by the model for the rotation compensated flow orientations.

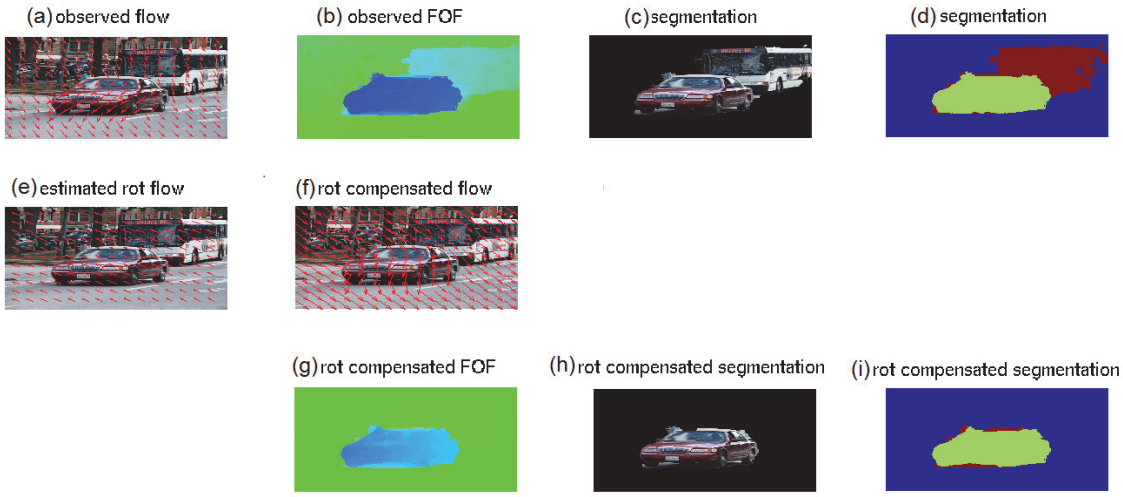

Figure 5.10. Rotation compensation real example (failure case) - cars10 video. (a) Observed flow - The camera motion is pure translation, upwards and to the left. (b) Observed FOF - there is evidently no camera rotation in this frame. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. The flow orientation observations on the slowly moving bus become indistinguishable from the background. (i) Segmentation labels returned by the model for the rotation compensated flow orientations.

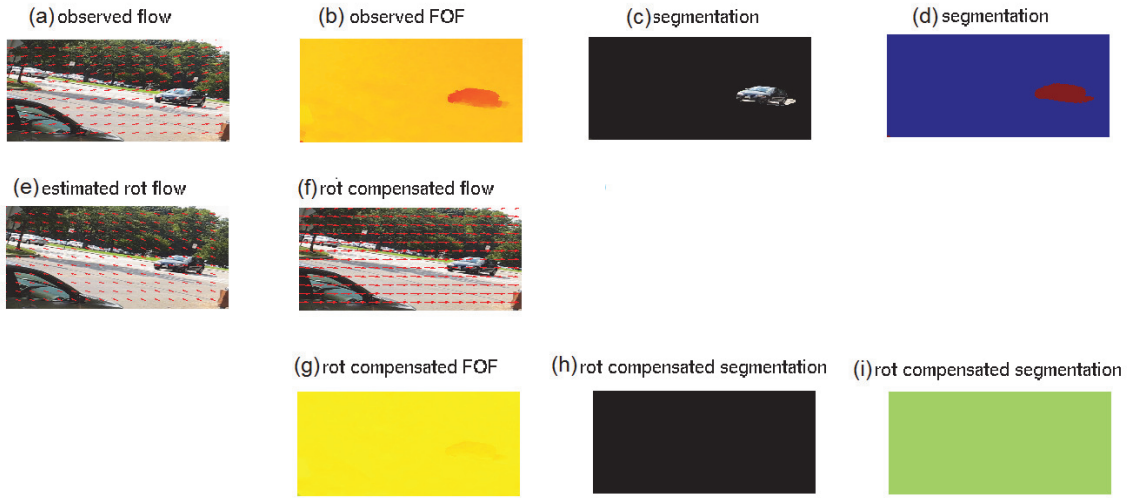

Figure 5.11. Rotation compensation real example (failure case) - cars7 video. (a) Observed flow - the camera motion is pure translation, to the left and slightly downwards. (b) Observed FOF - there is evidently no camera rotation in this frame. (c) Segmentation result based on the observed FOF (no rotation compensation). (d) Segmentation labels returned by the model (no rotation compensation). (e) Estimated rotation flow using our rotation compensation algorithm. (f) Observed flow after rotation flow has been removed from it. (g) FOF corresponding to the rotation compensated flow. (h) Segmentation output resulting from the rotation compensated FOF. The flow orientation observations on the moving car become indistinguishable from the background. (i) Segmentation labels returned by the model for the rotation compensated flow orientations

5.5 Test for the presence of rotation

Application of rotation compensation to a few videos resulted in the observation that in videos where rotation was present, rotation compensation made a significant improvement to the results. In many videos where there was no rotation, using rotation compensation resulted in genuinely moving objects becoming classified as background because the rotation compensation explained away the differences between the background orientations and the foreground orientations by hypothesizing a large amount of rotation for the scene.

In order to avoid the use of rotation compensation in videos where no rotation is present, we test for camera rotation at each frame and apply rotation compensation only if the test concludes that rotation is probable in the current frame. The algorithm to test for rotation is described in Algorithm 3. Intuitively, the algorithm computes what percentage of pixels which are assigned to the largest cluster exhibit orientations that are significantly different from the dominant translation FOF. If in any frame of the video, more than 25% of the pixels assigned to the largest cluster have a large deviation from the dominant translation FOF, rotation is likely to be present and rotation compensation is performed in such frames.

Algorithm 3 Test to check if rotation is present

Step 1 : Obtain the index of the largest cluster k', its corresponding translation FOF (without rotation compensation) $F_{k'}$, the number of pixels $n_{k'}$ currently assigned to the cluster, and the pixels currently assigned to the cluster $i : l_i == k'$.
Step 2 : Compute the difference D_i between observed orientation values and $F_{k'}$ for all pixels i currently assigned to cluster k'.
Step 3 : Compute median value of D, $\mu = median(D_i)$.
Step 4 : Compute a weight for each pixel $w_i = G(D_i; \mu, \sigma^2 = 15)$, where $G(\cdot; \mu, \sigma^2)$ is a Gaussian function with mean μ and variance σ^2. Note that the variance used here ($\sigma^2 = 15$) corresponds to the floor value used for variance in the infinite-K segmentation model of Section 4.5.
Step 5 : Compute the median of the weights, $\mu_w = median(w_i)$. Determine threshold $T_r = 0.9 \times \mu_w$.
Step 6 : Compute the number of pixels, m, whose weights w_i are lower than T_r.
Step 7 : If ratio $\frac{m}{n_{k'}} > 0.25$, rotation is present in the current frame.

5.6 Results

The proposed FOF rotation compensation algorithm is compared to the algorithm by Yamaguchi *et al.* [78]. Yamaguchi *et al.* compute the fundamental matrix that relates two consecutive frames in the video by matching keypoints across the images. Next, the rotation

flows are computed such that the sum of the errors from the epipolar constraint equation at each pixel is minimized. The rotation flows returned by their algorithm are subtracted from the observed optical flow vectors at each pixel. The rotation compensated optical flow vectors are then used in the infinite-K FOF segmentation model from the previous chapter. The results of this procedure are compared to the FOF rotation compensation algorithm from Section 5.2.

Table 5.1. shows that the FOF rotation compensation algorithm clearly outperforms the Yamaguchi method on a majority of videos. Detailed analysis of the results in individual frames in the videos revealed that the Yamaguchi method returns good results when the estimated fundamental matrix is accurate. In video frames where the fundamental matrix estimate is erroneous, the Yamaguchi method fails. The correct fundamental matrix $\mathbf{F}$ should satisfy the constraint $p_1 \times \mathbf{F} \times p_2^T = 0$, where p_1 and p_2 are the homogenous coordinates of matching points in the two images and T denotes the matrix transpose operation. We verified that in the frames where the Yamaguchi method fails, the average error in the values of $p_1 \times \mathbf{F} \times p_r^T$ was about 25 times the error in the frames where the method works. Thus, it is very likely that the shortcoming is not in the Yamaguchi rotation compensation algorithm itself, but in the fact that the fundamental matrix estimation procedure is unstable and prone to errors. In the Hopkins data set where camera motion is mainly translation and the background is largely planar, estimating the fundamental matrix is less error prone and as a result, the Yamaguchi method returns reasonable results. In the ComplexBackground data set where both the background and camera motion are complex, the Yamaguchi method performs poorly. It may be noted that the videos are taken from uncalibrated cameras and no additional information such as focal length of the camera is known. Obtaining the fundamental matrix in such scenarios can be extremely difficult. The FOF rotation compensation is much more stable and accurate across different data sets and scenarios. It is not adversely affected by the uncalibrated nature of the camera and the presence of outliers (independently moving objects).

The results of performing FOF-based rotation compensation are compared to the earlier translation-only models in Tables 5.2, 5.3, and 5.4. Table 5.2 shows the results of using the FOF segmentation without the color and prior models. Augmenting FOF-based segmentation with the color and prior information yields the results in Table 5.3. To summarize the performance over the three data sets and over all videos, the mean values from the tables are reproduced in Table 5.4.

Videoname	Yamaguchi compensation FOF only	FOF (Infinite K model) compensation FOF only
Hopkins set		
Cars1	40.14	**49.38**
Cars2	39.93	**42.47**
Cars3	51.32	**67.89**
Cars4	**25.61**	25.28
Cars5	59.52	**59.72**
Cars6	**85.70**	83.52
Cars7	34.51	**61.45**
Cars8	**88.79**	87.69
Cars9	47.46	**53.69**
Cars10	24.67	**45.51**
Marple1	56.57	**74.35**
Marple2	36.38	**52.84**
Marple3	61.06	**72.08**
Marple4	33.13	**51.38**
Marple5	**50.36**	**50.36**
Marple6	**31.82**	30.56
Marple7	48.76	**55.13**
Marple8	68.87	**81.30**
Marple9	**57.26**	52.92
Marple10	**34.02**	33.15
Marple11	36.49	**37.34**
Marple12	**66.43**	65.87
Marple13	50.76	**71.31**
People1	45.27	**54.93**
People2	75.11	**84.50**
Tennis	43.36	**61.58**
Segtrack set		
birdfall2	**68.68**	**68.68**
girl	42.04	**73.67**
parachute	39.22	**75.69**
cheetah*	03.64	**11.76**
monkeydog*	08.22	**11.76**
penguin*	**15.61**	15.07
ComplexBackground set		
drive	13.51	**30.93**
forest	04.37	**22.87**
parking	36.63	**51.14**
store	15.56	**53.06**
traffic	43.71	**67.92**
Hopkins mean	49.74	**59.03**
Segtrack mean	29.57	**42.77**
ComplexBg mean	22.76	**45.18**
average all videos	42.82	**54.39**

Table 5.1. Results. Comparison of FOF rotation compensation to Yamaguchi rotation compensation

The first column of numbers corresponds to the finite-K segmentation model with 46 components without the use of rotation compensation. The second column of numbers corresponds to the infinite-K model without the use of rotation compensation. The third column of numbers shows the results for the infinite-K model with rotation compensation. The last column are results for the rotation test followed by rotation compensation in frames that pass the rotation test. The results show that rotation compensation indeed helps in the videos where large amount of rotation is observed. However, in some videos where there is no rotation (Hopkins data set), performing rotation compensation results in less accurate results. Using the algorithm to test for rotation and performing rotation compensation only on videos in which the test passes results in the best performance on a majority of videos. The average over all videos shows that the rotation test followed by the rotation compensation algorithm yields the highest accuracy.

Videoname	Finite K model No rot. comp. FOF only	Infinite K model No rot. comp. FOF only	Infinite K model Rot. comp. FOF only	Infinite K model Rot. test+comp. FOF only
Hopkins set				
Cars1	47.81	47.75	**49.38**	47.75
Cars2	46.37	**63.16**	42.47	**63.16**
Cars3	67.18	**67.92**	67.89	**67.92**
Cars4	**38.51**	38.41	25.28	38.41
Cars5	**64.85**	63.54	59.72	63.54
Cars6	78.09	79.57	**83.52**	80.02
Cars7	37.63	**66.83**	61.45	61.51
Cars8	87.13	86.78	**87.69**	87.15
Cars9	**68.99**	60.60	53.69	60.64
Cars10	53.98	**61.20**	45.51	55.78
Marple1	65.65	77.59	74.35	**79.76**
Marple2	49.68	59.81	52.84	**63.50**
Marple3	67.83	79.05	72.08	**79.09**
Marple4	61.33	**63.91**	51.38	50.99
Marple5	50.05	**50.36**	**50.36**	**50.36**
Marple6	26.95	**33.43**	30.56	30.58
Marple7	51.57	**59.57**	55.13	56.66
Marple8	68.89	80.88	81.30	**81.46**
Marple9	40.53	**58.10**	52.92	56.20
Marple10	**57.19**	32.45	33.15	32.31
Marple11	37.33	36.75	**37.34**	36.75
Marple12	**65.83**	63.90	65.87	65.82
Marple13	67.09	**71.53**	71.31	71.50
People1	56.76	58.77	54.93	**60.59**
People2	85.35	**85.17**	84.50	85.16
Tennis	61.63	**63.79**	61.58	63.67
Segtrack set				
birdfall2	**68.68**	**68.68**	**68.68**	**68.68**
girl	**75.73**	75.70	73.67	75.70
parachute	51.49	09.54	**75.69**	68.31
cheetah*	12.68	**13.26**	11.76	12.21
monkeydog*	10.79	11.94	11.76	**12.56**
penguin*	14.74	**15.39**	15.07	15.32
ComplexBackground set				
drive	30.13	31.51	30.93	**32.71**
forest	19.48	14.04	22.87	**19.70**
parking	43.47	36.67	51.14	**51.05**
store	28.46	23.13	53.06	**52.91**
traffic	66.08	42.29	67.92	**62.04**
Hopkins mean	57.85	**61.95**	59.03	61.16
Segtrack mean	39.02	32.42	**42.77**	42.13
ComplexBg mean	37.52	29.53	**45.18**	43.68
average all videos	52.05	52.78	54.39	**55.72**

Table 5.2. Results. F-measure values for all videos for different models using only FOF segmentation.

Videoname	Finite K model No rot. comp. FOF+color+prior	Infinite K model No rot. comp. FOF+color+prior	Infinite K model Rot. comp. FOF+color+prior	Infinite K model Rot. test+comp. FOF+color+prior
Hopkins set				
Cars1	50.84	49.09	**51.12**	49.07
Cars2	56.60	**74.72**	63.17	74.71
Cars3	73.57	**67.51**	61.50	66.84
Cars4	47.96	**51.11**	22.24	51.11
Cars5	70.94	67.48	**68.52**	67.62
Cars6	84.34	85.24	**87.23**	85.60
Cars7	42.92	**83.53**	60.48	82.87
Cars8	87.61	87.35	**88.02**	87.51
Cars9	66.38	**59.19**	48.06	57.01
Cars10	50.84	**52.29**	45.39	49.02
Marple1	88.25	90.93	92.68	**93.17**
Marple2	60.88	**73.18**	63.05	71.82
Marple3	70.71	**81.15**	80.19	80.27
Marple4	69.01	**67.79**	48.40	52.99
Marple5	45.15	**45.24**	45.24	45.24
Marple6	23.95	**36.41**	34.23	34.18
Marple7	67.13	75.96	**78.60**	75.62
Marple8	80.32	77.08	85.08	**86.92**
Marple9	36.36	56.37	**57.30**	55.83
Marple10	58.72	46.89	**48.25**	47.65
Marple11	**41.41**	40.47	40.42	40.42
Marple12	**70.01**	68.71	67.55	65.67
Marple13	80.96	82.40	**83.27**	82.87
People1	69.53	71.07	65.84	**72.48**
People2	**88.40**	88.06	86.23	87.80
Tennis	67.59	63.61	**70.05**	66.48
Segtrack set				
birdfall2	**75.69**	**75.69**	**75.69**	**75.69**
girl	**81.95**	81.71	81.59	81.71
parachute	54.36	46.13	**91.69**	91.14
cheetah*	22.31	22.96	22.77	**23.53**
monkeydog*	18.62	21.05	**23.64**	21.61
penguin*	20.71	22.38	**23.47**	21.58
ComplexBackground set				
drive	**61.80**	43.34	45.13	52.10
forest	31.44	22.45	38.27	**39.95**
parking	73.19	36.34	**76.02**	75.78
store	**70.74**	23.20	70.07	70.31
traffic	71.24	50.17	**72.04**	68.28
Hopkins mean	63.48	**67.03**	64.32	66.57
Segtrack mean	45.61	44.99	**53.14**	52.54
ComplexBg mean	**61.68**	35.10	60.31	61.28
average all videos	60.34	59.14	61.90	**63.58**

Table 5.3. Results: F-measure values for all videos for the different models using FOF segmentation along with color and prior information.

Videoname	Finite K model No rot. comp.		Infinite K model No rot. comp.		Infinite K model Rot. comp.		Infinite K model Rot. test+comp.	
	FOF only	FOF+color+prior	FOF only	FOF+color+prior	FOF only	FOF+color+prior	FOF only	FOF+color+prior
Hopkins mean	57.85	63.48	**61.95**	**67.03**	59.03	64.32	61.16	66.57
Segtrack mean	39.02	45.61	32.42	44.99	**42.77**	**53.14**	42.13	52.54
ComplexBg mean	37.52	**61.68**	29.53	35.10	**45.18**	60.31	43.68	61.28
average all videos	52.05	60.34	52.78	59.14	54.39	61.90	**55.72**	**63.58**

Table 5.4. Results - summarized . Average F-measure value across videos in each data set for the different models.

5.7 Conclusions

We have presented an algorithm for rotation compensation that leverages the depth-independence properties of translational flow orientations and rotational flow vectors. The use of rotation compensation in conjunction with the flow orientation based motion segmentation results in a system that works well on a wide range of videos. The system is able to handle challenging background phenomena of varying depths in the scene.

A drawback of the current system is that rotation compensation is performed separately from the probabilistic segmentation model. For future work, inclusion of rotation as a variable in the probabilistic model would be an elegant and more complete solution for motion segmentation.

Modeling the presence or absence of rotation by using a suitable prior variable for rotation would be a good improvement over the current test to detect rotation.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Motion segmentation is interesting from a biological as well as computer vision perspective. Higher level vision systems such as object tracking, object recognition, and activity analysis can be built using motion segmentation as the first step. With this motivation, this thesis makes contributions to the research in motion segmentation under different camera conditions.

For the stationary camera case, in Chapter 2 we described a clean probabilistic system that models the various aspects involved as separate components that are brought together using the Bayes rule. The advantage of separating the components is that they can be better understood and interpreted. Suitable learning mechanisms can be designed for the components when they are clearly understood. For instance, we discussed a few examples of how the prior can be learned using the history of pixel labels from previous frames.

In Chapter 3, we presented an adaptive kernel variance method that addressed the important issue of variance adaptation for background modeling. For future extension, the kernel variance values which were hand-picked in this thesis may be automatically learned. Such learning mechanisms for priors, variances, and other aspects of background subtraction can help build adaptable surveillance systems that are more accurate and robust.

Many practical applications of tracking and analysis have been built on the foundation of motion segmentation in stationary cameras. There are fewer established solutions and applications of the same kind for moving cameras because segmentation is a much more challenging problem when the camera is moving. In Chapter 4, we have presented a viable approach for segmentation with a moving camera that is applicable to any scene irrespective of its geometric structure. Existing segmentation algorithms are highly prone to errors when there is a significant depth disparity between the various objects that make up the background. Higher level applications can now be built on top of the proposed motion segmentation algorithm.

Our algorithm makes use of certain properties of optical flow which are not dependent on the depth of the objects, but depend only on the objects' motion. Hence, we are able to

segment a scene based upon the real-world motion of the objects without being affected by their relative depth. This property of our system makes it extremely useful in many practical settings such as hand-held cameras and cameras on autonomous robots. The efficacy of our algorithm is demonstrated on a wide range of videos.

Although the proposed algorithm is very effective, there are many aspects that can be improved. A major shortcoming of the current algorithm is that objects that are moving with the same flow orientation values as the camera are not distinguishable from the background. This is an inherently ambiguous scenario and it is very difficult to detect such objects. One possible way to handle such a scenario is to use information about the flow magnitudes which we currently ignore. Use of region level information instead of pixel level information, such as superpixels, can be helpful in resolving such ambiguities as well as improving the accuracy in general. Edge and texture information can be used to augment the color appearance model that is currently being used.

The rotation compensation algorithm presented in Chapter 5 is early work with scope for many improvements. Currently, the rotation compensation is separate from the flow orientation based segmentation. Rotation compensation could be included elegantly into a probabilistic segmentation model. A prior over rotation parameters instead of the current test for rotation would help in better understanding and modeling of the problem.

Another related line of work is optical flow estimation. While we use optical flow estimates from another algorithm, it could be beneficial to include the flow estimation and motion segmentation within one framework. The constraint that the camera's motion places on possible orientations of the optical flow that we use for segmentation can also be a strong source of information to guide the optical flow estimation process.

Finally, better automatic scene understanding by considering optical flow estimation, motion segmentation, and depth estimation as inter-related problems is an interesting larger research direction that can be pursued based on the ideas laid forth in this thesis.

APPENDIX

ADDITIONAL FIGURES

A.1 Flow orientation fields

In Figure 4.2, a subset of the library FOF's was shown. The complete set of orientations is given here in Figure A.1. The motion parameter tuple $t = (t_x, t_y, t_z)$ responsible for each FOF is listed within each image.

A.2 Translational and rotational flow

It is a well known property that that the optical flow due to a camera's translation and rotation are commutative and when added together result in the same composite flow vectors that would be obtained if the camera translated and rotated at the same time. Figures A.2 and A.3 are two examples to illustrate that the composite flow due to a complex translation and rotation motion of the camera can be obtained by adding the respective pure translation and pure rotation flow. The examples clearly show that if the correct rotation parameters are known and the resulting rotation flow subtracted from the observed composite flow, pure translation flow can be obtained.

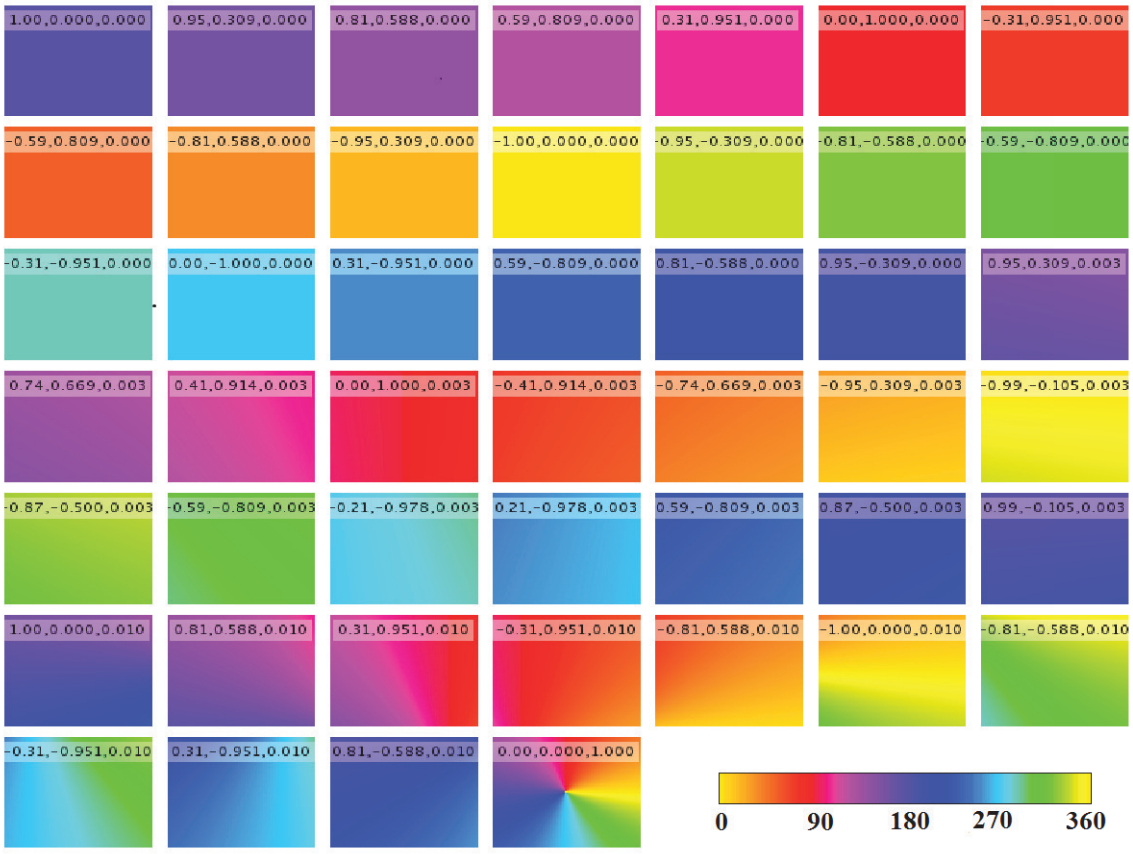

Figure A.1. The complete set of orientation fields used in our model. The motion parameters responsible for each field are given within each image. The color bar on the bottom right of the figure shows the mapping from angles in degrees to color values in the images.

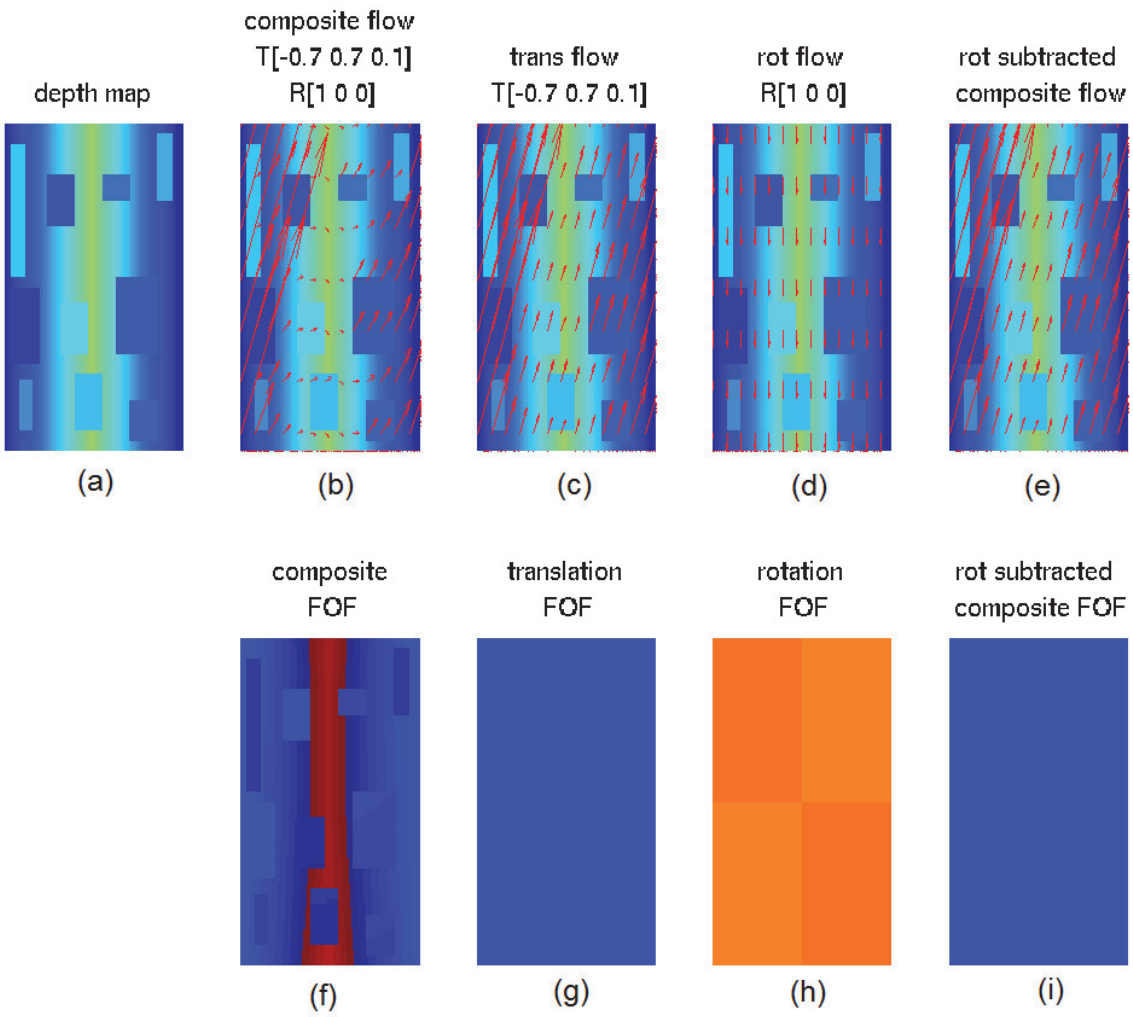

Figure A.2. Flows due to translation and rotation of a camera are commutative- example 1. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50). (b) Optical flow samples due to composite camera motion (parameters listed). (c) Optical flow due to translation alone. (d) Optical flow due to rotation alone. (e) Subtracting rotation flow (d) from composite flow (b) returns translation flow. (f) Orientation field corresponding to the composite flow in (b). (g) Orientation field corresponding to the translation flow in (c). (h) Orientation field corresponding to the rotation flow in (d). (i) Orientation field corresponding to the flow in (e) when rotation flow is subtracted from the composite flow. Clearly, (e) and (i), which are identical to (c) and (g) respectively show that translational and rotational flows are commutative.

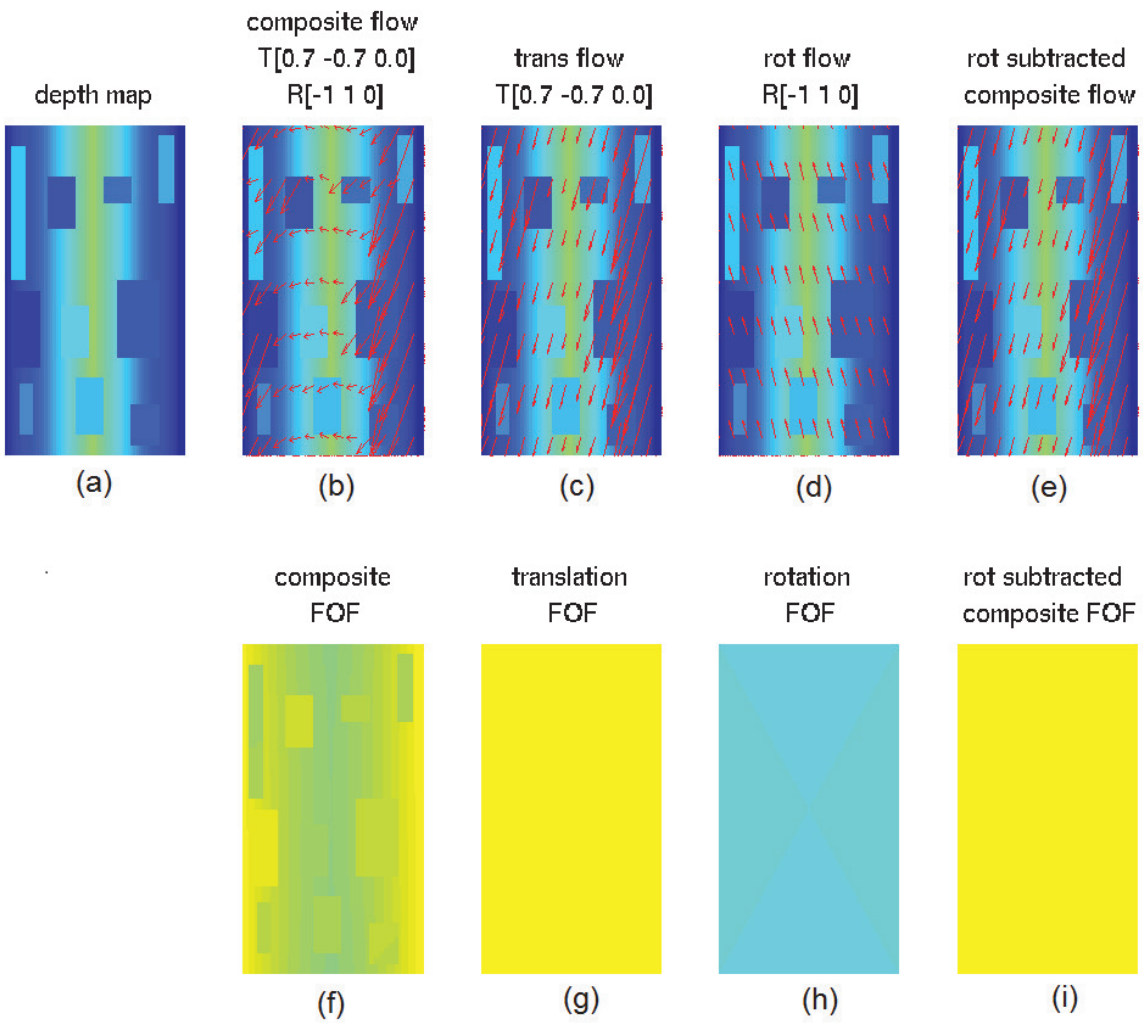

Figure A.3. Flows due to translation and rotation of a camera are commutative- example 2. (a) A synthetic scene with color indicating depth of object from the camera (dark blue = 0, green = 50). (b) Optical flow samples due to composite camera motion (parameters listed). (c) Optical flow due to translation alone. (d) Optical flow due to rotation alone. (e) Subtracting rotation flow (d) from composite flow (b) returns translation flow. (f) Orientation field corresponding to the composite flow in (b). (g) Orientation field corresponding to the translation flow in (c). (h) Orientation field corresponding to the rotation flow in (d). (i) Orientation field corresponding to the flow in (e) when rotation flow is subtracted from the composite flow. Clearly, (e) and (i), which are identical to (c) and (g) respectively show that translational and rotational flows are commutative.

BIBLIOGRAPHY

[1] Adato, Yair, Zickler, Todd, and Ben-Shahar, Ohad. A polar representation of motion and implications for optical flow. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2011), IEEE, pp. 1145–1152.

[2] Aeschliman, C., Park, J., and Kak, A.C. A probabilistic framework for joint segmentation and tracking. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2010), pp. 1371 –1378.

[3] Albright, T D, and Stoner, G R. Visual motion perception. *Proceedings of the National Academy of Sciences 92*, 7 (1995), 2433–2440.

[4] Brox, Thomas, and Malik, Jitendra. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision* (2010), pp. 282–295.

[5] Bugeau, Aurlie, and Prez, Patrick. Detection and segmentation of moving objects in highly dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2007).

[6] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision* (2012).

[7] Caudek, Corrado, and Rubin, Nava. Segmentation in structure from motion: modeling and psychophysics. *Vision Research 41*, 21 (2001), 2715 – 2732.

[8] Chockalingam, Prakash, Pradeep, S. Nalin, and Birchfield, Stan. Adaptive fragments-based tracking of non-rigid objects using level sets. In *Proceedings of the Sixth International Conference on Computer Vision* (2009).

[9] Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., and Wixson, L. A system for video surveillance and monitoring. *Carnegie Mellon University Technical Report, CMU-RI-TR-00-12* (2000).

[10] Collins, Robert T., Lipton, Alan J., Fujiyoshi, Hironobu, and Kanade, Takeo. Algorithms for cooperative multisensor surveillance. In *Surveillance, Proceedings of the IEEE* (2001).

[11] Cremers, Daniel. A multiphase level set framework for motion segmentation. In *4th International Conference on Scale Space Theories in Computer Vision* (2003), Springer, pp. 599–614.

[12] Croner, L. J., and Albright, T. D. Image segmentation enhances discrimination of motion in visual noise. *Vision Res 37*, 11 (1997), 1415–27.

[13] Croner, Lisa J., and Albright, Thomas D. Segmentation by color influences responses of motion-sensitive neurons in the cortical middle temporal visual area. *Journal of Neuroscience 19* (1999), 3935–3951.

[14] Dittrich, W. H. Action categories and the perception of biological motion. *Perception 22*, 1 (1993), 15–22.

[15] Dittrich, W. H., and Lea, S. E. G. Motion discrimination and recognition. *Avian visual cognition [On-line] Available: www.pigeon.psy.tufts.edu/avc/dittrich/* (2001).

[16] Elgammal, Ahmed, Duraiswami, Ramani, and Davis, Larry S. Probabilistic tracking in joint feature-spatial spaces. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (Washington, DC, USA, 2003), CVPR'03, IEEE Computer Society, pp. 781–788.

[17] Elgammal, Ahmed M., Harwood, David, and Davis, Larry S. Non-parametric model for background subtraction. In *European Conference on Computer Vision* (2000), pp. 751–767.

[18] Elqursh, Ali, and Elgammal, Ahmed M. Online moving camera background subtraction. In *European Conference on Computer Vision* (2012), Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, Eds., vol. 7577 of *Lecture Notes in Computer Science*, Springer, pp. 228–241.

[19] Farnebäck, Gunnar. Very high accuracy velocity estimation using orientation tensors parametric motion and simultaneous segmentation of the motion field. In *International Conference on Computer Vision (ICCV)* (2001), pp. 171–177.

[20] Gepshtein, Sergei, and Kubovy, Michael. The emergence of visual objects in space-time. *Proceedings of the National Academy of Sciences 97*, 14 (2000), 8186–8191.

[21] Goyette, Nil, Jodoin, Pierre-Marc, Porikli, Fatih, Konrad, Janusz, and Ishwar, Prakash. Changedetection.net: A new change detection benchmark dataset. In *Change Detection (CDW 12) at CVPR, IEEE Workshop on* (2012).

[22] Han, Bohyung, and Davis, Larry. On-line density-based appearance modeling for object tracking. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2* (Washington, DC, USA, 2005), ICCV '05, IEEE Computer Society, pp. 1492–1499.

[23] Haritaoglu, I., Harwood, D., and Davis, L.S. Hydra: multiple people detection and tracking using silhouettes. In *Image Analysis and Processing, 1999. Proceedings. International Conference on* (1999), pp. 280–285.

[24] Haritaoglu, Ismail, Harwood, David, and Davis, Larry S. W4: Real-time surveillance of people and their activities. *IEEE Transactions Pattern Analysis Machine Intelligence 22*, 8 (2000), 809–830.

[25] Hayman, Eric, and Eklundh, Jan-Olof. Statistical background subtraction for a mobile observer. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2* (Washington, DC, USA, 2003), ICCV '03, IEEE Computer Society, pp. 67–.

[26] Heikkila, M., and Pietikainen, M. A texture-based method for modeling the background and detecting moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 4 (2006), 657 –662.

[27] Horn, Berthold K. P., and Schunck, Brian G. Determining optical flow. *Artificial Intelligence 17* (1981), 185–203.

[28] Irani, M., Rousso, B., and Peleg, S. Recovery of ego-motion using image stabilization. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (1994).

[29] Irani, Michal, Rousso, Benny, and Peleg, Shmuel. Computing occluding and transparent motions. *International Journal of Computer Vision 12* (1994), 5–16.

[30] Javed, O., Shafique, K., and Shah, M. Appearance modeling for tracking in multiple non-overlapping cameras. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 2, pp. 26–33.

[31] Javed, Omar, Rasheed, Zeeshan, Shafique, Khurram, and Shah, Mubarak. Tracking across multiple cameras with disjoint views. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2* (Washington, DC, USA, 2003), ICCV '03, IEEE Computer Society, pp. 952–.

[32] Jepson, Allan D., and Heeger, David J. Linear subspace methods for recovering translational direction. In *Proceedings of the 1991 York conference on Spatial vision in humans and robots* (New York, NY, USA, 1993), Cambridge University Press, pp. 39–62.

[33] Johansson, Gunnar. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics 14*, 2 (1973), 201–211.

[34] Jones, M.C. Variable kernel density estimates. *Australian Journal of Statistics 32*, 3 (1990), 361–371.

[35] Kaewtrakulpong, P., and Bowden, R. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems* (2001), vol. 5308.

[36] Ko, Teresa, Soatto, Stefano, and Estrin, Deborah. Background subtraction on distributions. In *European Conference on Computer Vision* (Berlin, Heidelberg, 2008), ECCV '08, Springer-Verlag, pp. 276–289.

[37] Kurihara, Kenichi, Welling, Max, and Vlassis, Nikos. Accelerated variational Dirichlet process mixtures. In *Neural Information Processing Systems* (2006).

[38] Kwak, Suha, Lim, Taegyu, Nam, Woonhyun, Han, Bohyung, and Han, Joon Hee. Generalized background subtraction based on hybrid inference by belief propagation and Bayesian filtering. In *Proceedings of the 2011 International Conference on Computer Vision* (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 2174–2181.

[39] Lee, Yong Jae, Kim, Jaechul, and Grauman, Kristen. Key-segments for video object segmentation. In *Proceedings of the Sixth International Conference on Computer Vision* (2011).

[40] Li, Liyuan, Huang, Weimin, Gu, Irene Y. H., and Tian, Qi. Foreground object detection from videos containing complex background. In *ACM International Conference on Multimedia* (2003), pp. 2–10.

[41] Liao, Shengcai, Zhao, Guoying, Kellokumpu, Vili, Pietikäinen, Matti, and Li, Stan Z. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2010), pp. 1301 –1306.

[42] Mather, George, and West, Sophie. Recognition of animal locomotion from dynamic point-light displays. *Perception 22* (1993), 759–766.

[43] Mémin, Etienne, and Pérez, Patrick. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision 46*, 2 (Feb. 2002), 129–155.

[44] Micheloni, C., Foresti, G.L., and Snidaro, L. A cooperative multicamera system for video-surveillance of parking lots. In *Intelligence Distributed Surveillance Systems, IEE Symposium on* (2003), pp. 21–24.

[45] Mittal, A., and Paragios, N. Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2004), pp. II–302 – II–309 Vol.2.

[46] Narayana, Manjunath. Automatic segmentation and tracking of moving objects in video for surveillance applications. Master's thesis, University of Kansas, Lawrence, Kansas, USA, 2007.

[47] Narayana, Manjunath, Hanson, Allen, and Learned-Miller, Erik. Improvements in joint domain-range modeling for background subtraction. In *Proceedings of the British Machine Vision Conference* (2012), BMVA Press, pp. 115.1–115.11.

[48] Neal, Radford M. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics 9*, 2 (2000), 249–265.

[49] Nguyen, Nam T., Venkatesh, Svetha, West, Geoff, and Bui, Hung H. Multiple camera coordination in a surveillance system. *ACTA Automatica Sinica 29* (2003), 408–422.

[50] Ochs, Peter, and Brox, Thomas. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2012).

[51] Olivers, C. N. L., and Humphreys, G. W. Spatiotemporal segregation in visual search: Evidence from parietal lesions. *Journal of Experimental Psychology: Human Perception and Performance 30*, 4 (2004), 667–688.

[52] Porikli, Fatih, and Tuzel, Oncel. Bayesian background modeling for foreground detection. In *Proceedings of the third ACM international workshop on Video surveillance & sensor networks* (New York, NY, USA, 2005), VSSN '05, ACM, pp. 55–58.

[53] Prazdny, K. Egomotion and relative depth map from optical flow. *Biological Cybernetics 36*, 2 (1980), 87–102.

[54] Prazdny, K. On the information in optical flows. *Computer Vision, Graphics, and Image Processing 22*, 2 (1983), 239–259.

[55] Ren, Ying, Chua, Chin-Seng, and Ho, Yeong-Khing. Statistical background modeling for non-stationary camera. *Pattern Recognition Letters 24*, 13 (2003), 183 – 196.

[56] Sevilla-Lara, Laura, and Learned-Miller, Erik. Distribution fields for tracking. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2012).

[57] Sheikh, Yaser, Javed, Omar, and Kanade, Takeo. Background subtraction for freely moving cameras. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009* (2009), IEEE, pp. 1219–1225.

[58] Sheikh, Yaser, and Shah, Mubarak. Bayesian modeling of dynamic scenes for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27* (2005), 1778–1792.

[59] Shi, Jianbo, and Malik, Jitendra. Motion segmentation and tracking using normalized cuts. In *Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), ICCV '98, IEEE Computer Society, pp. 1154–.

[60] Shu, Chiao-Fe, Hampapur, Arun, Lu, Max, Brown, Lisa M. G., Connell, Jonathan, Senior, Andrew W., and Tian, Yingli. IBM smart surveillance system (S3): a open and extensible framework for event based surveillance. In *IEEE Conference on Advanced Video and Signal Based Surveillance* (2005), IEEE Computer Society, pp. 318–323.

[61] Siebel, Nils T, and Maybank, Stephen J. The advisor visual surveillance system. In *ECCV 2004 workshop Applications of Computer Vision (ACV)* (2004), pp. 103–111.

[62] Stauffer, Chris, and Grimson, W. Eric L. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (1999), vol. 2, pp. 246–252.

[63] Sun, Deqing, Roth, Stefan, and Black, Michael J. Secrets of optical flow estimation and their principles. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010* (2010), IEEE, pp. 2432–2439.

[64] Sun, Deqing, Sudderth, Erik B., and Black, Michael J. Layered segmentation and optical flow estimation over time. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2012), pp. 1768–1775.

[65] Tavakkoli, Alireza, Nicolescu, Mircea, Bebis, George, and Nicolescu, Monica. Non-parametric statistical background modeling for efficient foreground region detection. *Machine Vision Applications 7* (2009), 1–15.

[66] Thornton, Ian M., Rensink, Ronald A., and Shiffrar, Maggie. Active versus passive processing of biological motion. *Perception 31* (2002), 837–853.

[67] Tian, T.Y., Tomasi, C., and Heeger, D.J. Comparison of approaches to egomotion computation. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on* (1996), pp. 315–320.

[68] Tomasi, C., and Shi, J. Direction of heading from image deformations. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on* (1993), pp. 422–427.

[69] Toyama, Kentaro, Krumm, John, Brumitt, Barry, and Meyers, Brian. Wallflower: principles and practice of background maintenance. In *IEEE International Conference on Computer Vision* (1999), vol. 1, IEEE, pp. 255–261 vol.1.

[70] Tron, Roberto, and Vidal, René. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2007).

[71] Tsai, David, Flagg, Matthew, and Rehg, James M. Motion coherent tracking with multi-label MRF optimization. *Proceedings of the British Machine Vision Conference* (2010).

[72] Turlach, Berwin A. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique* (1993).

[73] Ullman, S. *The interpretation of visual motion*. MIT Press Cambridge, Mass, 1979.

[74] Wallach, Hans, and O'connell, DN. The kinetic depth effect. *Journal of experimental psychology 45*, 4 (1953), 205.

[75] Wang, J. Y.A., and Adelson, E. H. Representing moving images with layers. *IEEE Transactions on Image Processing 3*, 5 (Sept. 1994), 625–638.

[76] Wedel, Andreas, Cremers, Daniel, Pock, Thomas, and Bischof, Horst. Structure- and motion-adaptive regularization for high accuracy optic flow. In *International Conference on Computer Vision* (2009), IEEE, pp. 1663–1668.

[77] Wren, Christopher Richard, Azarbayejani, Ali, Darrell, Trevor, and Pentland, Alexander. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19* (1997), 780–785.

[78] Yamaguchi, Koichiro, McAllester, David, and Urtasun, Raquel. Robust monocular epipolar flow estimation. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2013).

[79] Yao, Jian, and Odobez, J.-M. Multi-layer background subtraction based on color and texture. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on* (2007), pp. 1 –8.

[80] Yuk, J.S.-C., and Wong, K.-Y.K. An efficient pattern-less background modeling based on scale invariant local states. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance* (2011), pp. 285 –290.

[81] Zivkovic, Z. Improved adaptive gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition (ICPR)* (2004), vol. 2, pp. 28 – 31 Vol.2.

[82] Zivkovic, Zoran, and van der Heijden, Ferdinand. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters 27*, 7 (May 2006), 773–780.