ONLINE NON-RIGID MOTION AND SCENE LAYER SEGMENTATION

BY ALI E. ELQURSH

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Ahmed Elgammal

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2014

ABSTRACT OF THE DISSERTATION

Online Non-rigid Motion and Scene Layer Segmentation

by Ali E. Elqursh

Dissertation Director: Ahmed Elgammal

In the past, different kinds of methods were devised to detect objects from videos. Based on the assumption of stationary camera, the now ubiquitous background subtraction learns the appearance of the background and then subtracts it to segment the scene. In practice such assumption is highly restrictive, and to handle moving cameras other methods were devised. For instance, motion segmentation targets the segmentation of different rigid motions in the video, while scene layer segmentation attempts to find a segmentation of the scene into layers that are consistent in space and time. Yet, such methods still suffers from other limitations such as the requirement of point trajectories to span the entire frame sequence. On a different aspect, recent years have witnessed a large increase in the proportion of videos coming from streaming sources such as TV Broadcast, Internet video streaming, and streaming from mobile devices. Unfortunately, most methods that process videos are mainly offline and with a high computational complexity. Thus rendering them ineffective for processing videos from streaming sources. This highlights the need for novel techniques that are online and efficient at the same time. In this dissertation, we first generalize motion segmentation by showing that under a general perspective camera trajectories belonging to one moving object form a low-dimensional manifold. Based on this, we devise two methods for online nonrigid motion segmentation. The first method tries to explicitly reconstruct the

low-dimensional manifolds and then cluster them. The second method attempts to directly separate the manifolds. We then show how motion segmentation and scene layer segmentation can be combined in a single online framework that combines the strength of both approaches. Finally, we propose two methods that assign figure- ground labels to layers by combining several cues. Results show that our framework is effective in detecting moving objects from videos captured by a moving camera.

List of Figures

Preface

Portions of this dissertation are based on work previously published or submitted for publication by the author [17, 18, 19].

Acknowledgements

I would like to thank my family for their love and support all along. It was a dream of us all and we can finally feel it. I want to express my deepest gratitude to my advisor, Prof. Ahmed Elgammal, who was guiding me along this path. His advices and encouragements were more than valuable and his support was unlimited. I would like to thank my committee Prof. Dimitris Metaxas, Prof. Vladimir Pavlovic and Dr. Omar Javed of SRI International for their very useful comments to improve the quality of my dissertation.

Dedication

To my dear wife Omayma, our adorable son Sherif, and my loving parents.

Table of Contents

Chapter 1

Introduction

1.1 Overview

Detecting objects of interests is the first step in many computer vision tasks. For example, one way to perform activity recognition is to first identify the actors in the video and then extract the features needed to classify the activity. Yet, even with the importance of such a step, a reliable way to detect objects from general cameras remains elusive.

In this dissertation we address the problem of online motion and scene layer segmentation. By online we mean that the result for the current frame must be available immediately. By motion segmentation we mean that we want to segment the differently-moving object from the scene and the background. By scene layer segmentation we mean that we want to learn a layered representation of the differently-moving objects such that any input frame can be composed from these learned layers.

We start by highlighting the similarities and differences between several related computer vision problems such as motion segmentation, video segmentation, and background subtraction. We then follow with the challenges of detecting objects of interest under realistic condition.

In computer vision there are several related yet different approaches to solve the problem of detecting objects of interest from video. In a sense, all these approaches attempt to achieve the same goal; that of detecting different entities in the scene. However, such approaches vary drastically in their underlying assumptions.

1) *Background Subtraction*: Background subtraction is the gold standard in detecting objects of interest when the camera is stationary. It is based on a simple idea; if

one “learns” the appearance of the background, then for each input frame “subtract” the learned background and what-ever remain are foreground objects. The simplicity and effectiveness of this approach has led to it almost universal use in the surveillance industry. Ironically, the key to its success - the assumption of stationary cameras - is also its main limitation as a general object detection technique.

2) *Motion Segmentation*: Whereas objects can be composed of different parts with different appearances, they typically move as a coherent region. Therefore, one can argue, that by detecting differently moving objects in the scene one can easily detect objects of interest in the scene. However, there exist several challenges. The observed image motion is the result of the perspective projection of the 3D motion. Thus, the straightforward approach of segmenting based on translational motion over two frames typically leads to poor results. On the other hand, as will be seen in the related work section, models using richer models, such as affine camera motion, can only segment a very small subset of pixels; namely those that can be tracked through the entire frame sequence. Furthermore, many methods make the assumption that objects are rigid, and as a consequence over-segment articulated and non-rigid objects. For all those reasons, motion segmentation is still an active area of research.

3) *Video Segmentation*: Rather than attempting to detect entire objects right away one can relegate such a decision to a higher level process and attempt to find a mid-level representation of the video. Video segmentation generalizes the idea of image segmentation to the temporal domain. It attempts to find spatiotemporal regions that have coherent motion and appearance. However, it is unclear how to use such a representation for detecting objects of interest in video.

4) *Object detection*: Learning models for specific objects and using these models in detection is a standard computer vision practice. However, so far such models are object specific and has to be learned from large corpora of pre-segmented/labeled images. Furthermore, it is unclear how to use such detectors to detect and segment objects from video streams. They also suffer in term of accuracy. For example, a recent evaluation [32] on a challenging dataset showed that state of the art face detection algorithms could only achieve 70% accuracy with more than 1 FPPI (false positive per image). Another

recent evaluation [11] on state of the art pedestrian detection algorithms found that such algorithms has a recall of no more than 60% for un-occluded pedestrians with 80 pixel height at a 1 FPPI (false positive per image). The performance degrades significantly if the targets are smaller in size or occluded. This makes such algorithms far from being useful in real-world applications.

Why detecting objects from video streams is challenging :

1) Offline vs Online: Most videos available nowadays come from video streams. For example, TV broadcast, camera phones and cameras mounted on robots or cars. Unfortunately, existing techniques are mostly offline. Attempts to run these techniques online by using a sliding window approach suffers from limitations. First, there is no way to guarantee that the result from one sliding window is consistent with the next one. Second, the evidence for the decision at any instant comes from the information in the current sliding window and any past information is lost.

In addition to the quantity of streamed videos, there exist a computational argument in favor for online techniques. Most, if not all, offline techniques do not scale linearly with the size of the video, and movie-long videos will typically take unfeasibly long time to be processed. Furthermore, if at a future time, more data becomes available, integrating this new data into the result requires re-running the process over the entire video again. Thus, except for cases where videos are available offline and the detection algorithm runs in linear time, an online method is necessary for the task of detecting objects of interest in videos.

2) What is an Object: Another challenging question is to determine what is an object. Although, this question may seem trivial at first, it is a far more challenging. For example, a building may be composed of many smaller objects such as windows, facades and doors. Or is a forest a single object? In many cases, the answer to this question is pre-determined. For example, a car detector can only detect cars and by using it we are implicitly restricting the kind of objects to be detected. Another example is in background subtraction, where detected object must be moving against a stationary background. In reality, however, we would like to segment "interesting" objects since in practice, most videos contain one or two objects of interest at any single

instant and it is much more efficient to identify and segment these objects.

It remains to define what an interesting object is. In existing literature, there are several ways to do this:

a) *Saliency*: Saliency is a measure of how a part of the input contrast with the rest. Saliency detection is considered to be a key attentional mechanism that facilitates learning and survival by enabling organisms to focus their limited perceptual and cognitive resources on the most pertinent subset of the available sensory data. A salient object can be then defined as the one which covers the most salient regions. This idea is exploited in Section 6.1 where saliency is combined with appearance and other cues to segment the object.

b) Figure-Ground labeling of boundaries: The importance of boundaries can be highlighted by realizing that cues such as occlusion can be observed only at boundaries. Thus, several works have taken the path of labeling boundaries instead of regions as figure or ground, e.g [42, 29, 52].

Both approaches suffer from limitations. While, saliency produces a measure at each pixel, it is difficult to "cut-out" whole objects based on it. On the other hand, figure-ground labeling at boundary typically discards cues that are region based such as surroundedness and compactness.

3) *Motion Segmentation*: Early gestalt psychologists have identified common fate as one of the most important cues that can be used for grouping features. This is evident by the fact that due to physical constraints an object typically moves together. In contrast, other cues such as appearance can vary within the same object. Therefore, an effective motion segmentation technique is essential for achieving scene segmentation.

As will be seen in the related work chapter, models for motion segmentation exhibits a natural trade-off between density and expressive power. While simple translational model can be used to densely segment the scene, they are unable to capture the complexity of perspective effects of projecting 3D motion into images or aggregate information over multiple images. On the other hand, more flexible models, such as affine motion models, can be only applied to trajectories, and thus provide segmentation information over a very small subset of the pixels. Solving this challenge is key for achieving scene

segmentation. In our work we propose using a combination of a sparse motion segmentation to aggregate information over multiple frames and a dense scene segmentation framework that propagates the motion information over the entire set of pixels.

In this dissertation we present a method for motion segmentation based on the idea that motion trajectories lies on a low-dimensional manifold. We then show how we can use multiple cues to assign figure/ground labels to the different motion segments. Finally, we show how we can learn a layered representation of the scene and simultaneously segment the input frames into two layers in an online framework.

1.2 Proposed Framework

Our proposed framework is depicted in Figure 1.1. Starting from input frames taken one frame at a time, point features are detected and tracked over multiple frames to form trajectories. Such trajectories implicitly encode the long term motion information. These trajectories are fed to the online motion segmentation module to segment the trajectories into separate motion clusters. A figure/ground labeling step is used to find the optimal assignment of figure/ground labels to each motion cluster. This is achieve by combining several cues such as compactness, and surroundedness. Finally, a scene layer segmentation module uses the input labeled trajectories to simultaneously maintain appearance models for the layers and generate a segmentation of each frame.

The next three sections provide a brief introduction to the three main components in our framework. Namely, online motion segmentation to segment different motions in the scene, figure-ground labeling to label the different motions as belonging to figure or ground, and scene layer segmentation to learn the appearance of the different layers and compute the segmentation of the scene.

1.3 Online Motion Segmentation

Since the early 20th century, gestalt psychologist have identified common fate as one of the most important cues for dynamic scene understanding. In the field of Computer Vision, this is reflected by the vast amount of literature on motion segmentation, video

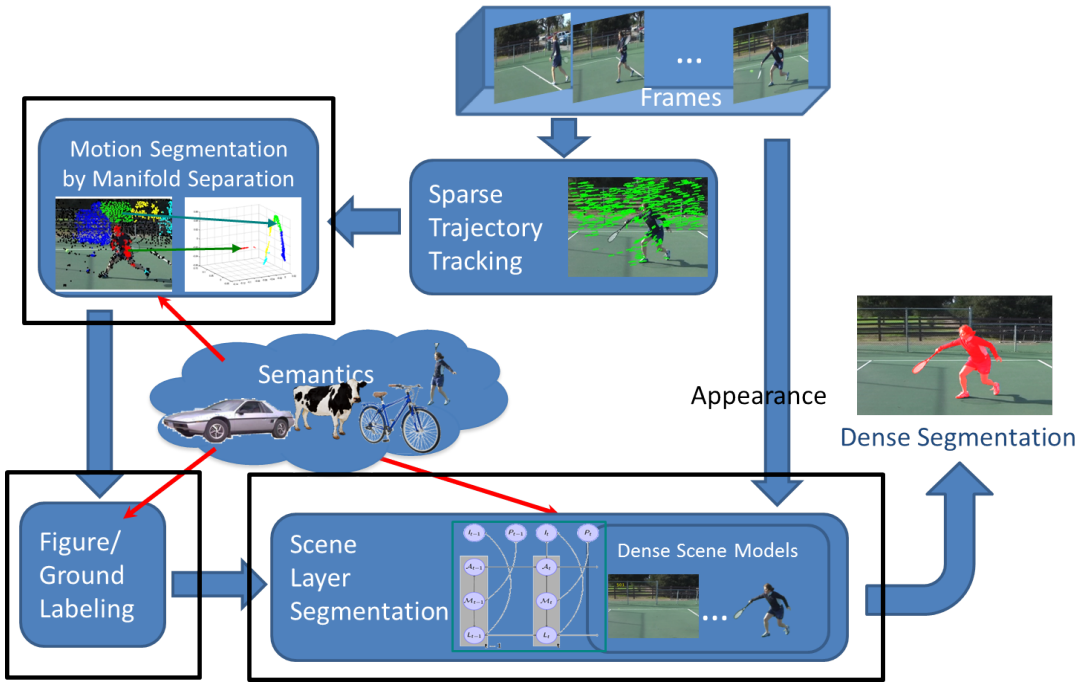


Figure 1.1: Proposed Framework

segmentation, and tracking. Specifically, motion segmentation deals with the problem of segmenting feature trajectories according to different motions in the scene, and is an essential step to achieve object segmentation and scene understanding.

Recent years have witnessed a large increase in the proportion of videos coming from streaming sources such as TV Broadcast, Internet video streaming, and streaming from mobile devices. Unfortunately, most motion segmentation techniques are mainly offline and with a high computational complexity. Thus rendering them ineffective for processing videos from streaming sources. This highlights the need for novel online motion segmentation techniques.

There exist a plethora of applications that would benefit from online motion segmentation. For example, currently activity recognition is either restricted to videos captured from stationary cameras (where existing background subtraction techniques can be used to segment the different actors [58]), or restricted to process videos offline using motion segmentation techniques. This is complicated even further if, after processing, more data becomes available, with the only solution typically being to reprocess the entire video from the beginning.

Another domain that would benefit from online motion segmentation is that of 3D TV processing. A real-time motion segmentation would enable performing 2D-to-3D conversion and video re-targeting on the fly on viewers devices. Other applications include online detection and segmentation of moving targets, and visual surveillance from mobile platforms to name a few.

As will be demonstrated in the related work chapter, most offline motion segmentation methods suffer from two problems. First, by formulating the problem as that of factorizing a trajectory matrix has led many approaches to assume that trajectories span the entire frame sequence. To handle the case where parts of trajectories are missing, such approaches borrow ideas from matrix completion. However, this is only successful up to a limit, since it assumes that at least there exist some trajectories that span the entire frame sequences. Second, the affine camera assumption restricts the applicability of motion segmentation to those videos where the assumption is satisfied. For example, this is typically true when the depth of the scene is small but not otherwise. To overcome the later problem, we assume a general perspective camera instead of an affine camera. On the other hand, to overcome the former problem, we measure the similarity between trajectories using a metric that depends only on the overlapping frames.

1.4 Figure-Ground Labeling

We see the world as a 2D projection of many 3D objects on the retina. Gestalt psychologists observed that we tend to organize this clutter through a process of figure–ground segregation—i.e., by identifying those regions of the retinal images that are object-related (figures) for further processing, and relegating other regions to the background [13]. They identified many factors which play a role in identifying which regions are figure or ground. Examples of such factors include continuity, convexity, symmetry, parallelism, surroundedness, and common fate to name a few.

Although there is evidence that high-level object understanding can influence figure-ground labeling, most studies have concluded that figure-ground labeling precedes high-level processing (e.g[40, 12]). In addition, neurological evidence suggests that motion cues plays an important role in figure-ground labeling[28]. Inspired by this evidence, it seems promising to devise a method that combines low-level and mid-level cues from appearance and motion to achieve figure-ground labeling from video.

Due to the importance of figure-ground labeling in perceiving important aspects of the visual input, there exist a large literature of work on figure-ground organization. On single images, [43] proposes a shapemes descriptor to evaluate the probability of labeling a contour as belonging to figure-ground. Conditional random fields CRF are then used to infer global figure-ground assignments. [42] extends this work to integrate several low-level, mid-level, and high level cues in a single CRF formulation. However, there is no clear way to generalize such methods to video input. [52] proposes using motion cues along with appearance cues to detect occlusion boundaries. Although, figure-ground assignment of boundaries are used, such assignment do not enforce consistency or completion. In contrast, we assign labels to video segments rather than boundaries thus producing results that are always consistent. Recently, [44] proposes figure-ground labeling for egocentric videos which is a special case of our problem.

There also exist a large literature on video segmentation and motion segmentation. The figure-ground labeling we address is different since video segmentation and motion segmentation attempts to segment regions that are homogeneous in color and/or motion with no notion of what is an object, figure, or ground. Similarly, the emphasis of video object segmentation is to segment out all objects without labeling them as figure or ground.

Recently there have been several attempts to extend traditional background subtraction to the moving camera settings. [48] uses orthographic motion segmentation over a sliding window to segment a set of trajectories. This is followed by sparse per frame appearance modeling to densely segment images. In [39], an iterative method is proposed that maintains block based appearance models in a Bayesian filtering framework. Since such methods typically use dominant motion or occlusion cues to determine

Figure 1.2: Automatic figure-ground labeling from video.

what is foreground and background, they do not always match our expectations about what is figure or ground.

1.5 Online Scene Layer Segmentation

One may argue that the ultimate goal of computer vision is to learn and perceive the environment in the same way children learn. Without access to pre-segmented visual input, infants learn how to segment objects from background using low level cues. Inspired by this evidence, significant effort in the computer vision community has focused on bottom up segmentation of images and videos. This has become ever more important with the proliferation of videos captured by moving cameras.

Our goal is to develop an algorithm for foreground/background segmentation from freely moving camera in a online framework that is able to deal with arbitrary long sequences. Traditional video segmentation comes in different flavors depending on the application, but falls short of achieving this goal. In background subtraction, moving foreground objects are segmented by learning a model of the background with the assumption of a static scene and camera. Motion segmentation methods attempt to segment sparse point trajectories based on coherency of motion. However, they lack a model of the appearance of foreground or background. Video object segmentation attempts to segment an object of interest from the video with no model of the scene background. On the other hand, there are several segmentation techniques that attempt to extend traditional image segmentation to the temporal domain. Such techniques are

typically limited to segmenting a short window of time.

It is frequently the case that low-level cues may be ambiguous if one only considers a short window of frames. Existing approaches either ignore this problem or resort to processing the whole video offline. Offline methods can typically produce good results on short sequences but the complexity increases rapidly as more frames need to be processed. The key to solving this problem is to recognize that to handle long sequences in an online way one has to learn and maintain models for the background and foreground regions. Such models serve the purpose of compactly accumulating the evidence over a large number frames and are essential for high level vision tasks.

1.6 Contributions

In this section we highlight the key contributions of the dissertation

1.6.1 Motion Segmentation as Manifold Separation

The first contribution of this dissertation is that we provide a simple proof for why trajectories belonging to a rigid object form a manifold of dimension three. This generalizes the problem of affine motion segmentation from subspace separation (linear manifold segmentation) to that of (general) manifold segmentation. It also explains why previous approaches using spectral clustering methods produced superior results while using simpler models.

1.6.2 Manifold Separation

After showing that the motion segmentation can be cast as a manifold separation problem, we need to devise a method for manifold separation. Unlike, subspace separation where one can exploit the geometric constraint for motion segmentation under affine camera assumption, separating manifolds is a much more challenging problem. Our second contribution is to devise two different methods for manifold separation. In the first, we explicitly reconstruct the manifolds in a low-dimensional space and then segment them in that space. In the second we use label propagation to segment the

manifolds without explicitly reconstructing them. Both methods are formulated as an online framework.

1.6.3 Figure Ground Labeling using Multiple Cues

We propose two different methods for figure-ground labeling. In the first, we show how combining saliency with other cues that discriminate between layers can be used to segment a video into figure and ground. In the second method, we combine several cues in a single energy function. In addition to cues that discriminate between layers, cues such as surroundedness and compactness help to find the optimal assignment of figure-ground labels. This method is used within our online scene layer segmentation framework to achieve segmentation of the scene into foreground and background layers.

1.6.4 Dynamic Bayesian Model for Scene Layer Segmentation

Most motion segmentation methods produce a segmentation of the point trajectories. Thus, segmentation information at the pixel level is mostly nonexistent. Furthermore, there exist no model of how the different objects of the scene looks like. In chapter 6.3, we propose a dynamic Bayesian model for scene layer segmentation. At each frame, we maintain appearance models for the individual layers and simultaneously generate a segmentation of each frame.

1.6.5 Online Moving Camera Background Subtraction

We propose a novel online method that learns appearance and motion models of the scene (background and foreground) and produces segmentations of video frames. It uses long term point trajectories, and thus is able to accumulate information over long sequences of frames, while at the same time performs a constant computation per frame. To achieve this, we describe a method to automatically update a low-dimensional representation of these trajectories and incrementally update a set of clusters in a novel coordinate free way. Rather than producing a single segmentation as an output, it uses Bayesian filtering to maintain a belief over different labellings, and appearances of the background and foreground. This enables our approach to recover from errors. By

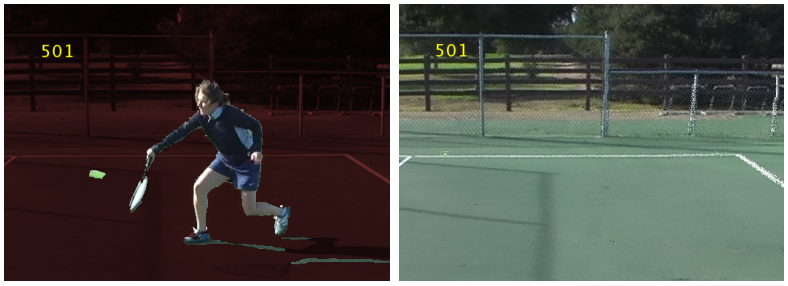


Figure 1.3: Left:(a) Automatic segmentation obtained on long sequence with fast articulated motion. Our method is able to segment the tennis ball even when no trajectories exist on it. Right: (c) One sample from the background pixel based appearance model learned automatically.

combining long term sparse trajectories and dense models of motion and appearance, our method is able to achieve superior results to the state of the art. We also devised an automatic method to determine foreground background labeling based on multiple static and dynamic cues.

1.7 Advantages

Our method has several advantages over existing approaches. It processes frames online and thus can easily handle arbitrary long videos. The use of long term trajectories allows it to accumulate long term motion information and prevents merging of objects that were known to move differently. Unlike affine motion segmentation, we accomplish this without assuming an affine camera model, thus enabling automatic segmentation of articulated and non-rigid motion. Our approach is able to handle multiple moving objects and maintain motion and appearance models. Such models enable our approach to learn the appearance of the foreground and background even if they are partially occluded. For example, Fig 1.3 shows a frame of a tennis sequence and the corresponding background model learned. Notice, that we are able to detect the tennis ball even when there are no trajectories on it. Finally, our appearance models can be used by a higher level reasoning framework to learn richer models of objects.

1.8 Outline

Chapter 2. introduces some background materials that are used in the rest of the thesis. This includes Laplacian Eigenmaps for dimensionality reduction, a semi-supervised learning method called label propagation, and graph-cuts for energy minimization. Chapter 3. covers the related literature in the areas of video segmentation, motion segmentation and layered scene segmentation. The traditional formulation of motion segmentation is introduced and its limitations are discussed. In chapter. 4, we show how the motion segmentation problem can be formulated as a manifold segmentation problem. The challenges are introduced and we show how existing literature address them. Chapter 5 then follows with two methods for online motion segmentation based on manifold separation. Chapter 6, demonstrates how multiple cues can be combined to achieve figure/ground labeling of video segments and our dynamic Bayesian model for scene layer segmentation. Experimental results for motion segmentation, figure-ground labeling and the entire framework is presented next in chapter 7. Finally, conclusions are presented in chapter 8.

Chapter 2

Background

2.1 Manifold Learning

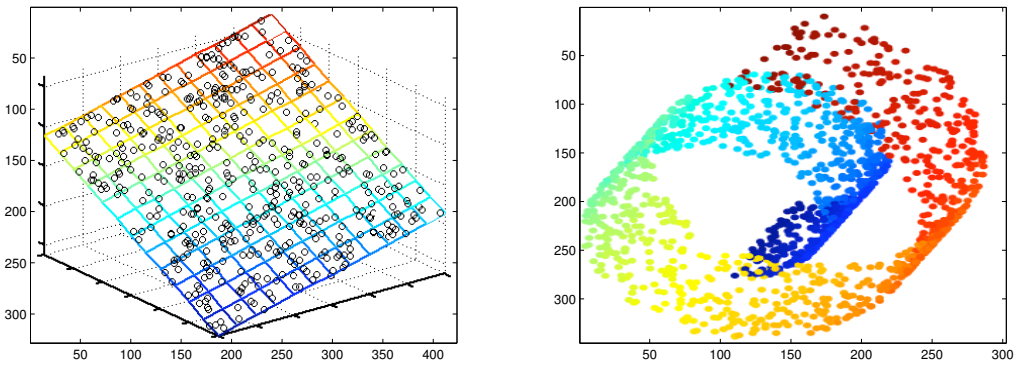
In many areas of artificial intelligence, information retrieval, and data mining, one is often confronted with intrinsically low-dimensional data lying in a very high-dimensional space. Consider, for example, gray-scale images of an object taken under fixed lighting conditions with a moving camera. Each such image would typically be represented by a brightness value at each pixel. If there were n^2 pixels in all (corresponding to an n×n image), then each image yields a data point in $\mathbb{R}^{n^2}$. However, the intrinsic dimensionality of the space of all images of the same object depends on the number of degrees of freedom of the camera. In this case, the space under consideration has the natural structure of a low-dimensional manifold embedded in $\mathbb{R}^{n^2}$.

There are two kinds of low-dimensional structures that can be discovered; linear and non-linear. As can be seen in Figure. 2.1, a linear structure implies that the data lies on a low-dimensional subspace (linear manifold) while a non-linear structure implies that the data lies on a non-linear manifold.

Learning the manifold implies discovering the low-dimensional structure and is therefore known also as dimensionality reduction. More formally, given k points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in \mathbb{R}^R$, find points $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k \in \mathbb{R}^r$, where $r << R$. Under certain geometric constraints that preserves the topology of the data.

2.2 Laplacian Eigenmaps Dimensionality Reduction

Over the past, several non-linear dimensional reduction (NLDR) techniques has been introduced e.g. Generalized Multidimensional Scaling[6], Local Linear Embedding (LLE)



(a) Linear Structure

(b) Non-linear Structure

Figure 2.1: Left: Linear structure where data lies on a low dimensional subspace. Right: Non-Linear structure where the data lies on a low dimensional manifold.

[47], Isometric feature mapping (ISOMAP) [54], and Laplacian Eigenmaps [2].

Given k points $\mathbf{x}^1, \ldots, \mathbf{x}^k$ in $\mathbb{R}^r$,we construct a weighted graph with k nodes, one for each point, and a set of edges connecting neighboring points. The embedding map is now provided by computing the eigenvectors of the graph Laplacian. The algorithmic procedure is formally stated below.

Step 1 (constructing the adjacency graph). We put an edge between nodes i and j if $\mathbf{x}^i$ and $\mathbf{x}^j$ are "close." There are two variations:

1. ϵ-neighborhoods (parameter $\epsilon \in \mathbb{R}$). Nodes i and j are connected by an edge if $\| \mathbf{x}_i - \mathbf{x}_j \|^2 < \epsilon$ where the norm is the usual Euclidean norm in $\mathbb{R}^l$. Advantages: Geometrically motivated, the relationship is naturally symmetric. Disadvantages: Often leads to graphs with several connected components, difficult to choose ϵ.

2. n nearest neighbors (parameter $n \in \mathbb{N}$). Nodes i and j are connected by an edge if i is among n nearest neighbors of j or j is among n nearest neighbors of i. Note that this relation is symmetric. Advantages: Easier to choose; does not tend to lead to disconnected graphs. Disadvantages: Less geometrically intuitive.

In the second step, the weights for the edges are computed. This is typically done by using the heat kernel $W_{ij} = e^{-\frac{1}{\lambda}\|\mathbf{x}_i - \mathbf{x}_j\|^2}$ if $\mathbf{x}_i$ is connected to $\mathbf{x}_j$, and $W_{ij} = 0$ otherwise.

Finally for each connected component of the graph, we compute the eigenvalues and eigenvectors of the generalized eigenvalue problem.

$$\mathbf{Lf} = \lambda \mathbf{Df},$$

where $\mathbf{D}$ is the diagonal matrix with $D_{ii} = \sum_j W_{ij}$, and $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix. Let $\mathbf{m}_0, \ldots \mathbf{m}_r$ be the eigenvectors corresponding to the $r+1$ smallest eigenvalues, where $r << R$. By dropping the smallest eigen vectors, the coordinates of the ith data point in the low-dimensional space $\mathbf{y}_i$ is defined by the ith entries of eigen vectors $\mathbf{m}_1, \ldots \mathbf{m}_d$.

$$\mathbf{y}_i = [m_1^{(i)}\, m_2^{(i)}\, \ldots\, m_r^{(i)}].$$

2.3 Label Propagation

Label propagation is a semi-supervised learning method. In semi-supervised learning we have many data points $x_1, \ldots, x_n$ and a few labeled points. This is in contrast to supervised learning where all the training points have labels. The problem in semi-supervised learning is how to classify unseen points given the labeled examples and unlabeled examples.

The core assumption in label propagation is the smoothness assumption. It implies that if a point x_1, is near x_2 then the label (cluster) y_1 of $x_1$is expected to be similar to the label y_2 of x_2. Given a graph G and a weight matrix $\mathbf{W}$ such that W_{ij} is the weight of the edge between nodes i and j, a simple idea for semi-supervised learning is to propagate labels on the graph. Starting with nodes $1, 2, \ldots, l$ labeled, each node starts to propagate its label to its neighbors, and the process is repeated until convergence. The goal is to find a labeling of the nodes that is consistent with both the initial partial labeling and the geometry of the data induced by the graph structure.

Formally, let $\mathbf{Y}_l$ denotes the labels for the labeled nodes. Furthermore, let $\hat{\mathbf{Y}} = (\hat{\mathbf{Y}}_l, \hat{\mathbf{Y}}_u)$ denotes the estimated node labels, with $\hat{\mathbf{Y}}_l$, and $\hat{\mathbf{Y}}_u$ corresponding to the labeled and unlabeled nodes respectively. $\mathbf{Y}_l$ and $\hat{\mathbf{Y}}$ are encoded using a one-hot encoding such that each row of $\mathbf{Y}$ is a vector with 1 at the location corresponding to the label of the node and zero otherwise. A general labeling cost function that is used in label

propagation is

$$C(\hat{\mathbf{Y}}) = \left\|\hat{\mathbf{Y}}_l - \mathbf{Y}_l\right\|^2 + \mu\hat{\mathbf{Y}}^T\mathbf{L}\hat{\mathbf{Y}} + \mu\epsilon\left\|\hat{\mathbf{Y}}\right\|^2,$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $D_{ii} = \sum_{i,i\neq j} W_{ij}$ denotes the graph Laplacian. The first term in the cost function encourages the consistency with the initial labeling. The second term encourages consistency with the graph structure. It can be shown that the second term can be rewritten as $\hat{\mathbf{Y}}^T\mathbf{L}\hat{\mathbf{Y}} = \frac{1}{2}\sum_{i,j} W_{i,j}(\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j)^2$. The last term is a regularization term to prevent degenerate situations, for instance, when the graph G has a connected component with no labeled sample. μ and ϵ are two constants that control the relative importance of the terms. In the limit case when $\mu \to 0$, matching the old labels is enforced $\hat{\mathbf{Y}}_l = \mathbf{Y}_l$ and $\hat{\mathbf{Y}}^T\mathbf{L}\hat{\mathbf{Y}}$ is minimized.

Several algorithms that minimize variations of the cost function has been proposed [68, 69]. However, we are interested in an algorithm with a probabilistic interpretation. Specifically, [69] presents an algorithm that uses Markov random walks on the graph with transition probabilities from i to j defined by,

$$p_{ij} = \frac{W_{ij}}{\sum_k W_{ik}},$$

in order to estimate probabilities of class labels. In a matrix form $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, where $D_{ii} = \sum_{j,j\neq i} W_{ij}$. Given the partition of the nodes into labeled and unlabeled nodes, the matrix $\mathbf{P}$ can be written in matrix form as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{ll} & \mathbf{P}_{lu} \\ \mathbf{P}_{ul} & \mathbf{P}_{uu} \end{bmatrix}.$$

The algorithm then works as follows: it assigns to a node x_i the probability of arriving to a positively labeled example, when performing a random walk starting from x_i and until a label is found. This probability can be written as

$$P(\mathbf{y}_{end} = 1|i) = \sum_{j=1}^{n} P(\mathbf{y}_{end} = 1|j)p_{ij}.$$

In matrix form this can be written as

$$\mathbf{Y}_u = (\mathbf{P}_{ul}|\mathbf{P}_{uu}) \begin{pmatrix} \mathbf{Y}_l \\ \mathbf{Y}_u \end{pmatrix}$$

which implies

$$\mathbf{Y}_u = (\mathbf{I} - \mathbf{P}_{uu})^{-1}\mathbf{P}_{ul}\mathbf{Y}_l.$$

Here we abuse the notation and let $\mathbf{y_i}$ denote the probability vector instead of the actual inferred label. The label for a node z_i can be then obtained by D

$$z_i = arg\, max_j\, y_{ij}.$$

2.4 Graph cuts for Energy Minimization

Many computer vision problems can be formulated as the minimization of an energy function which has both unary and pairwise terms. Typically, the unary term captures the evidence at different locations. The pairwise smoothness term captures the interaction between neighboring locations. The major difficulty with the minimization of energy functions lies in the enormous computational cost. Typically these energy functions have many local minima (i.e they are non-convex). However, under certain constraints on the energy function Graph cuts is a technique that can be used to minimize them [5, 4, 3].

Suppose $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph with $\mathcal{V}$ denoting the set of vertices and $\mathcal{E}$ the set of edges. Further more, assume that the graph has non-negative edge weights that has two special vertices (terminals), namely the source and the sink which are denoted by s, and t respectively. A $s-t$ cut or simply a cut is a partition of the nodes into two sets S, T. The cost of the cut is the sum of the weights of all the edges (p, q) such that $p \in S$ and $q \in T$.

$$c(S, T) = \sum_{p \in S, q \in T} w(p, q).$$

The min-cut is then defined as the cut that minimizes the cut cost above. A key result of combinatorial optimization is that finding the min-cut is equivalent to finding the maximum flow from the source s to the sink t. The maximum flow is the maximum "amount of water" that can be sent from the source to the sink by interpreting graph edges as directed "pipes" with capacities equal to edge weights. The theorem of Ford and Fulkerson [21] states that a maximum flow from s to t saturates a set of edges in the graph dividing the nodes into two disjoint parts $\mathcal{S}; \mathcal{T}$ corresponding to a minimum

cut. Thus, min-cut and max-flow problems are equivalent. In fact, the maximum flow value is equal to the cost of the minimum cut.

Let us describe the set of functions that can be minimized using graph cuts. Let $\{x^1, \ldots, x^n\}, x^i \in \{0, 1\}$ be a set of binary-valued variables. Define the class $\mathcal{F}^2$ to be functions that can be written as a sum of functions of up to two binary variables at a time,

$$E(x^1, \ldots, x^n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x^i, x^j)$$

Define the class $\mathcal{F}^3$ to be functions that can be written as a sum of functions of up to three binary variables at a time,

$$E(x^1, \ldots, x^n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x^i, x^j) + \sum_{i<j<k} E^{i,j,k}(x_i, x_j, x_k)$$

Kolmogorov et al. [37], shows that a necessary and sufficient condition to be able to minimize function in $\mathcal{F}^2$ is that it satisfies the regularity condition. i.e.

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0).$$

Furthermore, a function E in $\mathcal{F}^3$ is regular if all projections of E of two variables are regular.

In summary, graph cuts gives us a polynomial time algorithm to minimize energy functions when they satisfy the regularity condition. There are many applications of graph cuts in computer vision. In our work, we use graph cuts in Section 6 to find the optimal solution for the figure ground labeling problem in the offline setting. We also use it again in section 6.3to achieve continuous initialization of our online scene layer segmentation.

Chapter 3

Related Work

Due to the importance of video segmentation as a preprocessing step for many vision applications, there exist a large literature on the topic. In the most general setting video segmentation encompasses all methods that produce a segmentation of the video. However, in the Computer Vision literature, many problems can be classified as a special case of video segmentation. For example, background subtraction refers to segmenting the video when the video is captured from a stationary camera. On the other hand, scene layer segmentation attempts to learn a set of layers with an explicit ordering such that each video frame can be re-composed from these layers.

3.1 Motion Segmentation

Motion segmentation refers to the problem of segmenting a video based on motion. It comes in different flavors depending on what is begin segmented and what cues are used to achieve the segmentation. First, there are approaches that segment a set of extracted point trajectories using motion information alone. We refer to these as sparse motion segmentation algorithms since they only use the motion information at sparse locations to segment them (Not to be confused with algorithms that use sparsity). Second, there are algorithms that segment the entire set of pixels based on motion information in addition to static cues such as color or texture to produce the segmentation. Finally, there are hybrid algorithms that either use sparse motion segmentation with static cues or dense motion segmentation using only motion. Table 1. organizes the different groups based on density and cues. In this dissertation, unless otherwise noted, we take the term motion segmentation to refer to sparse motion segmentation.

Many approaches for motion segmentation are based on the fact that trajectories

	Sparse	Dense
Motion cues	Sparse Motion Segmentation	Dense Motion segmentation using motion only
Motion + Static cues	Sparse Motion Segmentation with Static cues	Dense Motion Segmentation

Figure 3.1: Flavors of Motion Segmentation

generated from rigid motion and under affine projection spans a 4-dimensional subspace. First introduced by [55], this geometric constraint has been used extensively especially in the case of independent motion. Most notably in [9], the problem is reduced to sorting of a matrix called shape interaction matrix with entries that represent the likelihood of a pair of trajectories belonging to the same object. In [36] the problem is reformulated as an instance of subspace separation, making the connection explicit.

Approaches to motion segmentation (and similarly subspace separation) can be roughly divided into four categories: statistical, factorization-based, algebraic, and spectral clustering. Statistical methods alternate between assigning points to subspaces and re-estimating the subspaces. For example, in [26] the Expectation-Maximization (EM) algorithm was used to tackle the clustering problem. Robust statistical methods, such as RANSAC [20], repeatedly fits an affine subspace to randomly sampled trajectories and measures the consensus with the remaining trajectories. The trajectories belonging to the subspace with the largest number of inliers are then removed and the procedure is repeated.

Factorization-based methods such as [55, 36, 30] attempt to directly factorize a matrix of trajectories. These methods work well when the motions are independent. However, it is frequently the case that multiple rigid motions are dependent, such as in articulated motion. This has motivated the development of algorithms that handle dependent motion. Algebraic methods, such as GPCA [59] are generic subspace separation algorithms. They do not put assumptions on the relative orientation and dimensionality of motion subspaces. However, their complexity grows exponentially

with the number of motions and the dimensionality of the ambient space.

Spectral clustering-based methods [67, 7, 41], use local information around the trajectories to compute a similarity matrix. It then use spectral clustering to cluster the trajectories into different subspaces. One such example is the approach by Yan et al. [67], where neighbors around each trajectory are used to fit a subspace. An affinity matrix is then built by measuring the angles between subspaces. Spectral clustering is then used to cluster the trajectories. Similarly, sparse subspace clustering [16] builds an affinity matrix by representing each trajectory as a sparse combination of all other trajectories and then applies spectral clustering on the resulting affinity matrix. Spectral clustering methods represent the state-of-the-art in motion segmentation. We believe this can be explained because the trajectories do not exactly form a linear subspace. Instead, such trajectories fall on a non-linear manifold.

With the realization of accurate trackers for dense long term trajectories such as [46, 53] there have been great interest in exploiting dense long term trajectories in motion segmentation. In particular, Brox et al. [7] achieves motion segmentation by creating an affinity matrix capturing similarity in translational motion across all pairs of trajectories. Spectral clustering is then used to over-segment the set of trajectories. A final grouping step then achieves motion segmentation. More recently, Fragkiadaki et al. [23] proposes a two step process that first uses trajectory saliency to segment foreground trajectories. This is followed by a two-stage spectral clustering of an affinity matrix computed over figure trajectories. The success of such approaches can be attributed in part to the large number of trajectories available. Such trajectories help capture the manifold structure empirically in the spectral clustering framework.

Our approach is also based on building an affinity matrix between all pairs of trajectories, however we process frames online and do not rely on spectral clustering. Deviating from the spectral clustering, is the idea of using non-linear dimensionality reduction (NLDR) techniques followed by clustering to achieve motion segmentation [24].

3.1.1 Layered Motion Segmentation

Layered Motion Segmentation refers to approaches that model the scene as a set of moving layers [60]. Layers have a depth ordering, which together with mattes and motion parameters models how an image is generated. Let n_l denote the number of layers, the ith layer can be characterized by an appearance A_i and optionally a matte L_i. To generate a frame at time t, the appearance of the ith layer is first transformed according to the transformation parameters M_i^t using $f(A_i, L_i, M_i^t)$. The transformed appearances are then overlaid in the order specified by the depth of each layer d_i. Different variations are possible using different parameterizations of the variables or by including additional variables that describe, for example, lighting effects at each frame. maps

It is obvious that if one knowns the assignment of pixels to different segments, it is trivial to estimate the appearance and transformation parameters. Similarly, if one knows the appearance and transformation parameters it is easy to assign pixels to segments. Since initially we know neither of them, this is an instance of a chicken-and-egg problem.

Wang et al. [60] used an iterative method to achieve layered motion segmentation. For every pair of frames they initialize using a set of motion models computed from square patches from the optical flow. Next, they iterate between assigning pixels to motion models and refining the motion models. This process is repeated until the number of pixel reassignments is less than a threshold or a maximum number of iterations is reached. Using the support maps for each segment, all pixels belonging to one segment are wrapped into a reference frame and combined using median filter. A final counting step is used to establish depth ordering based on the assumption that occluded pixels appear in fewer frames.

Originally used for optical flow [33], mixture models were also used to probabilistically model the image generation process. Weiss et al. [63] used Expectation Maximization (EM) algorithm to update the model parameters. In the E-step, the system updates the conditional expectation of L_i given the fixed parameters. This is done by

computing the residual of observed measurement and the predicted appearance given the motion parameters M_i^t. In the M-step, the model parameters A_i, M_i^t are updated based on these "soft" assignments. To incorporate spatial constraints in the mixture model, two methods were proposed. In the first, the images are pre-segmented based on static cues, and assignment in the E-step is based on the residual of all the pixels in each segment. In the second, a MRF prior is imposed on the labels L_i. The motion model of ith layer is represented by the six parameters of an affine transformation. Although the approach reasons about occlusions by assigning pixels to the correct model, it does not infer the depth ordering of different layers. Torr et al.[56] extend the approach by modeling the layers as planes in 3D and integrating priors in a Bayesian framework. Both approaches rely on a key frame for estimation, and therefore can handle a few number of frames. Flexible sprites [34] are another variation of layered motion segmentation where transformations are restricted to a discrete set of predetermined transformations. A Gaussian model for pixel-wise appearance and mattes is used. A variational optimization is then used to learn the parameters of the probabilistic model. Most of these approaches either uses expectation-maximization or variational inference to learn the model parameters.

Wills et al. [65], noted the importance of spatial continuity prior for learning layered models. Given an initial estimate, they learn the shape of the regions using the α-expansion algorithm [5], which guarantees a strong local minima. However, their method does not deal with multiple frames. Kumar et al. [38] proposes a method that models spatial continuity, while representing each layer as composed of a set of segments. Each segment is allowed to transform separately thus enabling the method to handle nonrigid objects by segmenting them into multiple segments distributed over multiple layers. This flexibility comes at the expense of a less semantically meaningful representation.

A common theme of most layered models is the assumption that the video is available before hand. Such assumption prevents the use of such approaches for processing videos from streaming sources. In addition, typically the computational complexity increases exponentially with the length of the videos. In contrast, in our work we propose a

method that learns layered models in an online framework.

3.2 Background Subtraction

The problem of segmenting videos captured by a stationary camera into a static background and moving foreground objects became known as background subtraction. The term evolved from the traditional way to solve the problem; after learning the appearance of the background, the background is subtracted from the input image to highlight moving objects.

Traditionally, background subtraction methods [66, 51, 14] attempt to achieve foreground segmentation by assuming that any motion in the video data is due to moving objects. In [50], a Gaussian mixture model is used to learn the appearance of each background pixel. Given enough samples for each pixel, the parameters for the mixture model can be learned using a variant of Expectation-Maximization. To be able to update the model online, a variation of online K-means is used. Elgammal et al. [15] uses kernel density estimation (KDE) to model the appearance of the background.

The success of these algorithms has led to their ubiquitous use in surveillance systems, where the assumption of a stationary camera is always satisfied. Due to the assumption of a stationary camera, this severely limits their use in videos shot by a moving camera. Several attempts to relax this assumption have led to methods that compensate for the motion of the camera [31, 45]. These methods use a homography or a 2D affine transform to compensate for the motion. This allows them to handle scenes that can be approximated by a plane or when the camera rotates but does not translate.

3.3 Moving Camera Background Subtraction

More recently, there has been several attempts to extend background subtraction techniques to the moving camera case. This setting is more challenging for several reasons. First, as the camera is moving, the background is no longer stationary in the image

frame and per-pixel appearance models are not applicable any more. Second, the traditional definition of foreground as moving objects no longer apply. Finally, background regions with large depth variation may be self occluding and regions of the background may appear and disappear as the camera moves.

Sheikh et al. [48] use orthographic motion segmentation over a sliding window to segment a set of trajectories. This is followed by sparse per frame appearance modeling to densely segment images. The use of orthographic motion segmentation means that motion information outside the sliding window is lost and the sparse appearance modeling fails to capture object boundaries when appearance information is ambiguous. Due to the dependence on long term trajectories only, regions with no trajectories may be disregarded as background altogether. Finally, the method fails if the assumption of orthographic projection is not satisfied. In [39], a method is proposed that maintains block based appearance models in a Bayesian framework. To update the appearance models, the method iterates between estimating the motion of the blocks and inferring the labels of the pixels. Once converged, the appearance models are used as priors for the next frame and the process continues. Due to the iterative nature of the approach, the method is susceptible to reach a local minimum. Although motion information between successive frames is estimated, it is only used to estimate the labels in the current frame and does not carry on to future frames. In contrast, we maintain motion information via long term trajectory and maintain a belief over different labellings in a Bayesian filtering framework.

3.4 Video Segmentation

Image segmentation aims to group perceptually similar pixels into regions. Video segmentation generalizes this concept, and attempts to group pixels into spatiotemporal regions that exhibit coherence in both appearance and motion.

It is important to note the distinction between object segmentation and video segmentation. In practice, one can identify objects that moves coherently but are composed of multiple regions with different appearances. Similarly, articulated objects may be

formed of multiple regions of different motions with common appearance. Typically, a video segmentation technique produces an over-segmentation of such objects.

Several existing approaches for video segmentation are formulated as clustering of pixels from all frames. First, multidimensional features are extracted representing photometric and motion properties. Then clustering is used to gather evidence of pixel groupings in the feature space. Such clustering can be done using Gaussian mixture models [25], mean-shift [61, 10], and spectral clustering [22].

Chapter 4

Motion Segmentation via Manifold Separation

4.1 Introduction

Existing work in Motion Segmentation has largely assumed that cameras are affine. This assumption leads to the following formulation for the motion segmentation problem. Let $\mathbf{X} = [X_1 X_2 \dots X_N]$ be a $4 \times N$ matrix representing a set of N points in 3D using a homogeneous coordinate system. Similarly, let $\mathbf{M} = [M_1 \dots M_F]^T$ be the $2F \times 4$ matrix formed by vertically concatenating F camera matrices. Such model is only valid if the object on which the 3D points lie is rigid, where the motion of the object can be represented as motion of the camera with respect to a stationary object. Under the assumption of affine cameras, such camera matrices can be specified by the first two rows of the projection matrix. Taking the product of the two matrices $\mathbf{X}$ and $\mathbf{M}$, we get a matrix $\mathbf{W}$ where each column represent the projected locations of a single 3D point over F frames. The columns of $\mathbf{W}$ represent the trajectories in 2D image space.

$$\mathbf{W} = \mathbf{MX}$$

$$\begin{bmatrix} x_1^1 & \dots & x_1^N \\ y_1^1 & & y_1^N \\ \vdots & & \vdots \\ x_F^1 & & x_F^N \\ y_F^1 & & y_F^N \end{bmatrix} = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_F \end{bmatrix} \begin{bmatrix} X_1 X_2 \dots X_N \end{bmatrix}$$

Since the rank of a product of two matrices is bounded by the smallest dimension, the rank of the matrix of trajectories $\mathbf{W}$ is at most 4. Trajectories generated from a single rigidly moving object lies on a subspace of dimension 4.

In general when there are multiple moving objects, the columns of the trajectory matrix $\mathbf{W}$ will be sampled from multiple subspaces. The motion segmentation problem can be formulated as that of subspace separation, i.e. assign each trajectory to one of the subspaces and estimate the subspaces.

4.2 Motion Segmentation as Manifold Separation

In this section we show how the problem of motion segmentation can be cast as a manifold segmentation problem. This is demonstrated in two steps. First, we show how trajectories in the three-dimensional space form a three-dimensional manifold. Next, we show how the projection of these trajectories to 2D image coordinates also form a three-dimensional manifold.

Let X be an open set of points in 3D comprising a single rigid object. Together with the Euclidean metric it forms a three-dimensional manifold. The motion of the object at times $t = 2, \ldots, F$ can be represented by rigid transformations $f_2(\mathbf{x}), \ldots f_F(\mathbf{x})$ respectively with $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is a 3D point in the camera coordinate system. The space of trajectories can be therefore defined by the set

$$\Gamma(f) = \{(\mathbf{x}_1, \ldots, \mathbf{x}_F) \in \mathbb{R}^{3F} : \mathbf{x}_i = f_i(\mathbf{x}_1)\ i \neq 1\},$$

with subspace topology. Let $\pi_1 : \mathbb{R}^{3F} \to \mathbb{R}^3$ denote the projection of a point $(\mathbf{x_1}, \ldots, \mathbf{x}_F) \in \mathbb{R}^{3F}$ onto the first factor. Let $\phi : \Gamma(f) \to X$ be the restriction of $\pi_1$to $\Gamma(f)$. Since $f_2, \ldots, f_F$ are continuous maps and ϕ is a restriction of a continuous map, ϕ is also continuous. It is also a homeomorphism because it has a continuous inverse. This implies that the space of trajectories is a manifold of dimension three.

Furthermore, we can show that projecting the 3D trajectories into the image coordinates also induces a smooth manifold. Let $g(\mathbf{x}) = \dfrac{\mathbf{f}}{z}[x\, y]^T$ be the camera projection function that projects a point in the camera coordinate system to image coordinates, where $\mathbf{f}$ is the camera focal length. $g(\mathbf{x})$ is continuous at all points except at points with $z = 0$. Let $\Omega(f) = \Gamma(f) \backslash \{\mathbf{x}_1 \ldots \mathbf{x}_F : z_i \neq 0\}$ be the subset of $\Gamma(f)$ where all points satisfy this constraint, it follows that $G(\mathbf{x}_1, \ldots, \mathbf{x}_F) = (g(\mathbf{x}_1), \ldots, g(\mathbf{x}_F))$ is also a smooth continuous map over $\Omega(f)$. It is therefore easy to show that $G(\Omega)$ is also a

smooth manifold of dimension three.

Note that even though we know that trajectories in image space form a smooth manifold, we do not have an analytical manifold. However, under the assumption that the manifold is densely sampled, empirical methods can be used to model the manifold. In addition, note that each distinct motion in the scene will generate one manifold. In this dissertation we rely on label propagation and dense trajectory tracking to solve the manifold separation problem.

Chapter 5

Online Motion Segmentation

For motion segmentation to be truly online, one have to satisfy two requirements. First, the amount of computation time per frame should not depend on the length of trajectories. This is to prevent the algorithm taking longer and longer with each additional frame. Second, motion information from an arbitrary number of frames needs to be aggregated.

In this chapter we devise two methods for online motion segmentation. Both methods are based on the idea that constructing a similarity (affinity) matrix between trajectories can be done in constant time given a distance metric that can be incrementally computed. In Section 5.1, we provide a metric that satisfies this constraint and use it to construct the affinity matrix.

Both methods then use the computed affinity matrix together with the segmentation from the previous frame to update the motion segmentation result and thus achieving online motion segmentation segmentation. The first method uses dimensionality reduction to compute a low-dimensional representation of the input trajectories. The segmentation result from the previous frame is used to initialize the clustering of trajectories in that space. The details of this method is presented in Section 5.2. The second method, formulates the online motion segmentation as a label propagation problem. The label propagation takes into account the graph structure in the current frame as well as the labels of the trajectories in the previous frame. The details of the approach is presented in Section 5.3.

5.1 Online Affinity Computation

As identified in Chapter 4, trajectories belonging to a single object lie on a three-dimensional manifold. However, such manifolds are not static as they are a function of the motion of the object, which changes over time. To model such dynamic manifolds without resorting to resolving for each frame, we design a distance metric that can be computed incrementally. Such computation has to be done in time independent of the length of the trajectories. In addition, the metric must capture the similarity in spatial location and motion. The intuition is that if two trajectories are relatively close to each other and move similarly, then they are likely to belong to the same object. In this subsection we show how one such metric can be computed incrementally.

We start by introducing some notation. A trajectory $T_a = \{\mathbf{p}_a^i = (x_a^i, y_a^i) : i \in A\}$ is represented as a sequence of points $\mathbf{p}_a^i$ that spans frames in the set A. For simplicity we reserve superscripts for frame references and subscripts for trajectory identification. The motion of a trajectory between frames i and j in the x and y direction is denoted by $u_a^{i:j} = x_a^j - x_a^i$ and $v_a^{i:j} = y_a^j - y_a^i$.

Given two trajectories T_a and T_b we define two distance metrics $d_M^{1:t}(T_a, T_b)$ and $d_S^{1:t}(T_a, T_b)$ representing the difference in motion and spatial location up to time t respectively. In a way similar to [7], the distances are defined as the supremum of distances over pairs of frames. The max function helps "remember" large differences in motion and spatial location. Formally,

$$d_M^{1:t}(T_a, T_b) = max_{\{i-\Delta, i\} \subset X} d_M^i(T_a, T_b), \tag{5.1}$$

$$d_S^{1:t}(T_a, T_b) = max_{i \in X} d_S^i(T_a, T_b), \tag{5.2}$$

where $X = \{x : x < t,\ x \in A \cap B\}$ is the set of overlapping frames up to time t and Δ is a user defined parameter, which controls the amount of smoothing used in computing motion difference. Distances over frames are defined as

$$d_M^i(T_a, T_b) = \frac{(u_a^{i-\Delta:i} - u_b^{i-\Delta:i})^2}{(\sigma_{Mu}^i)^2} + \frac{(v_a^{i-\Delta:i} - v_b^{i-\Delta:i})^2}{(\sigma_{Mv}^i)^2},$$

and

$$d_S^i(T_A, T_B) = \parallel \mathbf{p}_a^i - \mathbf{p}_b^i \parallel / \sigma_S^2.$$

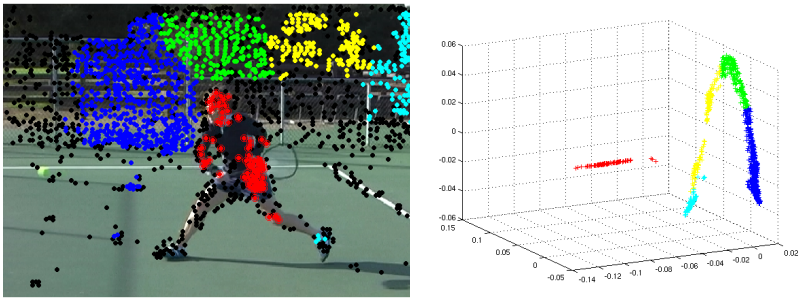


Figure 5.1: Trajectories in the image and their corresponding embedding coordinates. Each cluster is marked with a different color. Black dots are short trajectories not yet added to the embedding. The foreground cluster is marked with two red concentric circles around its trajectories. (Best viewed in color).

$(\sigma^i_{Mu})^2$, and $(\sigma^i_{Mu})^2$ are two parameters that control the weighting of motion distances while σ^2_S controls the weighting of the spatial distance. We compute $(\sigma^i_{Mu})^2$, and $(\sigma^i_{Mu})^2$ adaptively for each frame as the variance of $u_a^{i-\Delta:i}$ and $v_a^{i-\Delta:i}$ over all trajectories. For n trajectories, we can collect all pairwise frame distances into two $n \times n$ matrices $\Delta\mathbf{D}^t_M = [d^t_M(T_i, T_j)]$, and $\Delta\mathbf{D}^t_S = [d^t_S(T_i, T_j)]$. It follows that we can compute the total distance between trajectories up to time t, $\mathbf{D}^t$, incrementally from $\mathbf{D}^{t-1}$ and $\Delta\mathbf{D^t}$ by taking the maximum of the two. To convert distances to affinities $\mathbf{W}$ we use

$$\mathbf{W}^t = exp(-(\mathbf{D}^t_M + \mathbf{D}^t_S)). \tag{5.3}$$

5.2 Motion Segmentation via Repeated Dimensionality Reduction

We model each trajectory up to time t as a single point in a low dimensional embedding space. The coordinate of each trajectory in this space is continuously updated. Trajectories belonging to the background are expected to lie on a low-dimensional manifold (background manifold) while trajectories belonging to the foreground objects lie on separate manifolds. This arises due to the similarity in spatial location and motion between neighboring trajectories. Fig 5.1. shows a set of trajectories in the image space and their corresponding manifolds in the embedding space. In the embedding space we model each trajectory manifold using a Gaussian Mixture Model (GMM), where each patch of the manifold is modeled with a multivariate Gaussian. This representation is

continuously updated in a coordinate free manner.

Given the affinity matrix, a lower dimensional representation is computed using Laplacian Eigenmaps [2]. Let $\mathbf{D}$ be the $n \times n$ matrix with elements $D_{ii} = \sum_j W_{ij}$. Laplacian eigenmaps is obtained by an eigen decomposition of the normalized Laplacian $\mathbf{\Gamma}^T \mathbf{\Lambda} \mathbf{\Gamma} = \mathbf{D}^{\frac{-1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{\frac{-1}{2}}$ and keeping the eigenvectors $\mathbf{v}_0, \ldots, \mathbf{v}_m$ corresponding to the $m+1$ smallest eigenvalues and then ignoring $\mathbf{v}_0$. The eigenvectors then defines the coordinates of the trajectories in a lower dimensional space.

This representation of the trajectories has two advantages. First, the distance of non-overlapping trajectories can be measured in the embedding space. Second, if part of an object goes out of view, its trajectories in the embedding space remain valid and can be used to enforce the existence of a cluster at that location.

For the first frame, each trajectory is assigned a cluster number by fitting a GMM of R components. In subsequent frames, after computing the new embedding coordinates, we associate trajectories extending from the previous frame with their old cluster assignment. Given this assignment, the parameters of a new GMM model can be estimated and optimized by running a few iterations of the EM algorithm on the whole set of trajectories. The intuition is that as the distance matrix is updated and the new embedding computed, the spatial relationship of trajectories in the embedding will not change abruptly. Due to the incremental nature of the algorithm, the good initialization prevents the algorithm from being stuck in a local minima. The result of this step is a set of trajectories with their associated cluster labels.

As the embedding space is updated, the number of clusters may also need to be updated. This happens when an object enters or exits the field of view. During the EM iterations if a cluster ends up with zero trajectories, we simply remove the cluster and decrease the number of clusters by 1. To handle increasing the number of clusters, the probability of each trajectory is computed given the GMM parameters. If the number of trajectories with low probability is more than a threshold, we assign these trajectories to a new cluster and perform EM until convergence.

5.3 Motion Segmentation using Label Propagation

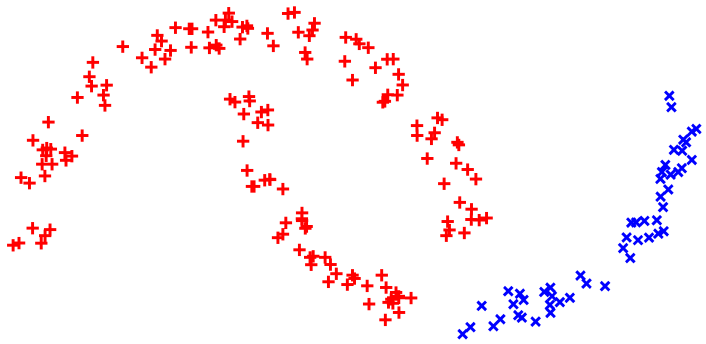
We propose an approach that achieves online motion segmentation by segmenting a set of manifolds through dynamic label propagation and cluster splitting. Starting from an initialization computed over a fixed number of frames, we maintain a graph of pairwise similarity between trajectories in an online fashion. To move to the next frame we propagate the label information from one frame to the next using label propagation. The label propagation respects the computed graph structure while taking into account the previous labeling.

To see why label propagation is well suited for the manifold separation problem, consider the simple two moons example shown at the top of Figure 5.2. Separating the two moons can be cast as a manifold separation problem. However, when applying spectral clustering on this example, due to the proximity, one cluster leaks over the other cluster. On the other hand, with proper initialization, label propagation is able to successfully segment the two moons. As explained in the next sections, the initialization comes from previous frames.

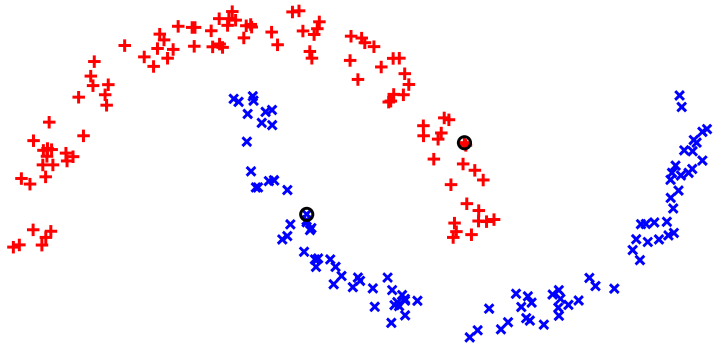
To handle cases where new evidence suggests that one cluster comes from two differently moving objects, we evaluate each cluster and measure a normalized cut cost of splitting the cluster. This process is then repeated for each subsequent frame. Figure 5.3 shows frames 40 and 150 of the sequence *marple7* and the segmentation by our approach. Miss Marple is correctly tracked throughout the sequence even when affected by occlusion as in frame 40.

Given the segmentation of trajectories at frame $t-1$ the goal is to obtain an updated labeling at frame t. To accomplish this we have two address several scenarios. First, new trajectories may be introduced into the distance matrix, and second, motion information from existing trajectories may necessitate splitting or merging existing clusters. In addition, in the former case, new trajectories may belong to existing objects or they may belong to new objects. In this section, we describe how these two cases are handled in our framework.

First we address how new information can be integrated to update our cluster labels



(a) Spectral Clustering



(b) Label Propagation

Figure 5.2: Spectral Clustering vs Label propagation. Colors represent the different clusters and the black circle represent the supervised labels.

Figure 5.3: Frames 40, and 150 from the marple7 sequence and the corresponding segmentation obtained by our online approach.

while assuming that no new objects has entered the scene. At each frame, we assume extended trajectories have a probability distribution over different segment labels. The inferred label is defined as the label with the maximum probability. One way to proceed is to use these as supervised labels and attempt to label new trajectories based on a classifier learned over the labeled samples. There are several problems with this solution. First, existing labels are not revised and thus errors cannot be recovered from. Second, classifier do not take into account the graph structure we have available in hand.

To solve these problems, we use labels for existing trajectories as *prior* knowledge and attempt to label both trajectories extended from previous frames and new trajectories while taking the graph structure into account. We attach to each node (trajectory) in the current frame a dongle node corresponding to the trajectory label in the previous frame. Let $\mathbf{P}_{uu}$ be the transition probabilities obtained from the affinity matrix $\mathbf{W}^t$, the new transition matrix for the augmented graph is

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{ll} & \mathbf{P}_{lu} \\ \mathbf{P}_{ul} & \mathbf{P}'_{uu} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \eta\mathbf{I} \\ \mathbf{I} & (1-\eta)\mathbf{P}_{uu} \end{bmatrix}.$$

Applying Markov random walks on this graph gives $\mathbf{Y}_u = (\mathbf{I} - (1-\eta)\mathbf{P}_{uu})^{-1}\mathbf{Y}_l$. Since the labeled nodes are actually the previous frame labels, this equation becomes

$$\mathbf{Y}^t = (\mathbf{I} - (1-\eta)\mathbf{P}_{uu})^{-1}\mathbf{Y}^{t-1}. \tag{5.4}$$

The parameter $0 \leq \eta \leq 1$ controls how strongly previous labels affect future labels. In the extreme case, when $\eta = 0$, the graph structure is the dominant term. Using label propagation in this manner effectively takes into account the uncertainty in previous labels. Applying the above iteration we get a new probability distribution for each node and the segment labels are obtained by

$$z_l = \text{argmax}_j \; y_{ij}$$

Although, it may seem like an overkill to re-apply label propagation for each incoming frame, there are two reasons why this is not the case. First, by using the previous labels as anchoring points we avoid leaking the labels across clusters such as in spectral clustering (Figure 5.2). Second, in practice equation (5.4) is solved iteratively [8] and

by using the previous frame labels as an initial solution we are able to converge in a few iteration.

Once the new labels are obtained, we need to evaluate if a new object has entered the scene, or if an object that was previously not moving started moving. Note, that label propagation only propagates existing labels and does not introduce new ones. If a new object enters the scene, there must be an associated set of new trajectories. These new trajectories would inevitably receive some label by the label propagation, and introduce high intra-cluster variation within that associated cluster. Similarly, if an stationary object starts moving this will lower the affinities between the object trajectories and the rest of the trajectories in the same cluster. The task is therefore to go over the clusters one at a time and see if any of them requires splitting. This can be accomplished by computing an optimal binary cut using normalized cut and then evaluate the normalized cut cost. If the cost is above a threshold then we keep the cluster intact.

To perform the cut we use the method of [49]. First, we extract the sub-matrix $\mathbf{W}_c$ corresponding to the cluster to be evaluated. We then solve the generalized eigen-value problem $\mathbf{D}_c - \mathbf{W}_c = \lambda \mathbf{D}_c \mathbf{y}$. Next, we extract the eigen-vector corresponding to the second smallest eigen-value, and evaluate the normalized cut cost for different thresholds. The normalized cut cost is expressed as $\frac{\mathbf{y}^T(\mathbf{D}_c - \mathbf{W}_c)\mathbf{y}}{\mathbf{y}^T \mathbf{D}_c \mathbf{y}}$, where $\mathbf{y}$ is the thresholded eigen-vector. The vector y that minimizes the normalized cut cost is then selected as the best cut. The normalized cut cost is a value between 0 and 1. In our approach, if the cut cost is bellow a threshold τ we split the cluster.

5.4 Initialization

Our framework assumes that for a frame at time t we know the labels of the trajectories at time $t-1$. There are two ways to boot strap the system. First, we could use an offline motion segmentation method, which does not depend on the affine camera assumption to generate the initial labels. Another approach is to assign all the trajectories a single label initially and allow our cluster splitting process to discover the number classes

over time. In our experiments we take the later approach and show that even without initialization, our approaches are able to detect the moving objects in the scene. In Figure5.4 we show how starting from an initial assignment of a single cluster, the label propagation process is able to segment the two persons in the scene.

5.5 Conclusion

In the previous chapter, we showed how motion segmentation can be cast as a manifold separation problem. Based on this, we presented two approach that achieves online motion segmentation without sacrificing the accuracy of state of the art methods. Whereas, the first approach is based on the idea of explicitly reconstructing the manifold in a low dimensional space, the second uses label propagation on a dynamically changing graph to implicitly segment the manifold. The approaches are able to maintain labels and recover from errors as more information becomes available. As will be shown in the experiments (Section 7.1), compared to offline approaches, we show competitive results on a benchmark dataset.

We are motivated in our approaches by several applications that require online processing. For example, real-time motion segmentation can be used to perform video re-targeting on-the-fly on viewers devices. Even when the videos are available offline, processing movie-long videos would take in the order of weeks using existing offline approaches.

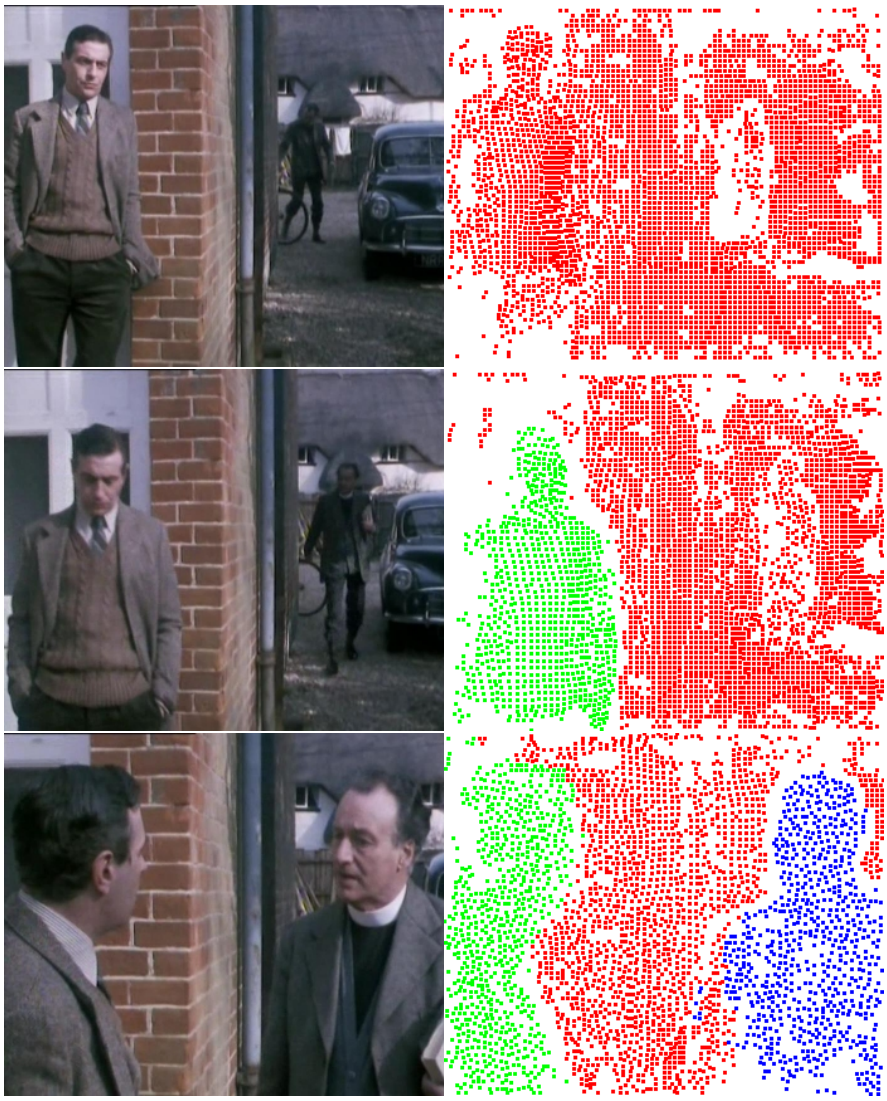


Figure 5.4: Initialization on the marple6 sequence. From top to bottom frames, 100, 160, 260 and their corresponding segmentations. At frame 1, all trajectories are given the same label and there is hardly any motion in the scene. After frame 100, the man leaning on the wall starts to move and is automatically segmented from the background cluster. Similarly, after frame 160 the person coming closer to the camera is also detected and popped out from the background.

Chapter 6

Scene Layer Segmentation

Scene layer segmentation refers to the problem of simultaneously segmenting and learning a layered representation of the input video. The problem of scene layer segmentation is closely related to that of figure-ground labeling. Learning a representation of the different layers implicitly assigns a depth ordering of the different layers. The figure and ground regions can be then inferred from the assigned depth values. On the other hand, by labeling different regions of the image as figure and ground, the task of finding a scene layer representation becomes much easier.

The rest of this Chapter is organized as follows. In Section 6.1, we show how combining multiple cues to find a segmentation of short video sequences into figure and ground regions. This motivates our approach which given motion clusters at each frames, assigns figure-ground labels to them (Section 6.2). We then propose a Bayesian scene layer segmentation approach that, using the labeled motion clusters, computes a two-layered representation of the scene in an online framework (Section 6.3).

6.1 Figure-Ground Labeling via Cue Aggregation

In this section we propose a method that achieves figure-ground labeling from video sequences. We combine saliency, motion, and color features into a single energy function which we optimally optimize. By using saliency, our labeling more closely match what we typically consider figure and at the same time combines many low-level cues in the process. On the other hand, motion and color similarity help disambiguate figure and ground in regions where the saliency cue is least confident.

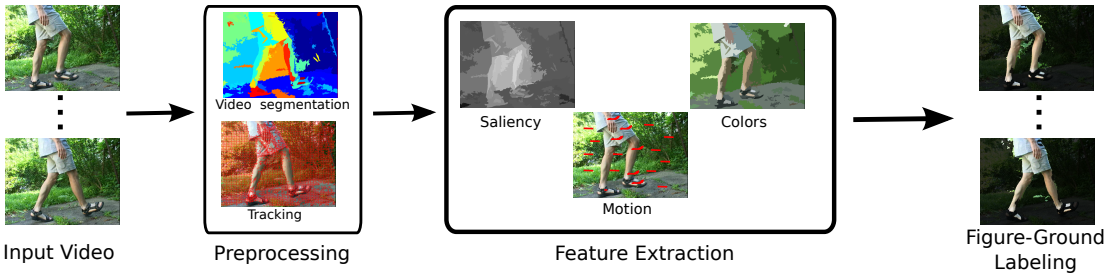


Figure 6.1: Main stages of our approach.

6.1.1 Approach

Our approach consists of several stages; see Figure 6.1. In the first stage, we preprocess the video by applying video segmentation and dense trajectory tracking (Subsection 6.1.1). Next we compute a saliency, motion, and color features for each video segment (Subsection 6.1.1). Finally, we formulate an energy function that integrates saliency, motion and color features and find the optimal assignment of figure-ground labels by minimizing it (Subsection 6.1.1).

Preprocessing

In order to obtain robust features, we aggregate information over a large number of voxels in the video. This is achieved by applying the hierarchical video segmentation approach of [27] to efficiently obtain a video segmentation. The result of this step is a set of segments $\mathcal{V} = \{ V_1, \quad \ldots \quad , V_N \}$ with each segment V_i represented with a set of video voxels $V_i = \{(x, y, t)\}$.

Next, in preparation to computing motion features, we extract dense trajectories using large displacement optical flow (LDOF) [53]. Denoting trajectory j by T_j, we assign each trajectory to a segment $v(T_j)$ by first finding the segment to which each point along the trajectory belongs to and then selecting the mode of this list.

Feature Extraction

Rather than computing many local feature and determine what features are discriminative for figure-ground labeling, we use state of the art saliency method of [35] to

integrate many low and mid-level image features and produce a saliency map. Such maps are computed for each individual frame and then used to compute a saliency feature for each video segment. Let S represent the saliency volume. We define the saliency of a segment s_i as the mean saliency of all the voxels belonging to the segment

$$s_i = \frac{\sum_{(x,y,t)\in V_i} S(x,y,t)}{|V_i|}.$$

Saliency maps are relatively noisy and without considering the relations between video segments, figure-ground labeling is inaccurate at regions where saliency is least confident. Therefore, we compute color features and motion features for each video segment and use them to measure the similarity between any pair of video segments.

Since video segments represent regions with homogeneous color, we use the mean color $\mathbf{c}_i$ of a segment as a color feature. To compute it we first convert the colors to *Lab* color-space and then compute the mean over all voxels belonging to each segment. The *Lab* color-space is robust to changes in lightness and is able to capture chromatic similarity under a wide range of lighting conditions.

$$\mathbf{c}_i = \frac{\sum_{(x,y,t)\in V_i} c(x,y,t)}{|V_i|}.$$

To compute the motion features $\mathbf{m}_i$ we first transform each trajectory into a vector of relative motions between frames. This effectively removes the dependency on the starting location. Next for each segment we compute the mean relative motion vector $\mathbf{m}_i$. Since for some segments we may have no trajectories occupying a pair of frames we define $\mathbf{o}_i$ as an indicator vector such that element $[\mathbf{o}_i]_j = 1$ if and only if there exist trajectories in segment i overlapping frames j, and $j+1$.

Optimization

To incorporate the pairwise relations between the segments, we first define a graph structure over the segments. With each node representing a segment V_i we define the set of neighbors $\mathcal{N}(V_i)$ as the set of segments which shares any boundary with V_i. However, it is not infrequent that a ground region is surrounded by figure or vice versa. To solve this we augment the set of neighbors with other segments which are similar in

color. Formally, we cluster the color features from all the segments into k clusters using k-means. For each segment in a cluster we augment its neighbors with other segments in the same cluster.

To achieve figure-ground labeling we look for a labeling $L = [l_1 \dots l_N]$ that satisfies the following criteria. First, figure is salient while ground is non-salient. Second, the appearance and motion of neighboring segments with the same label vary smoothly. We formulate an energy function which encapsulates this criteria as

$$E(L) = \sum_{V_i \in \mathcal{V}} |\mathcal{N}(V_i)| f(l_i) + \sum_{V_i} \sum_{V_j \in \mathcal{N}(V_i)} g(l_i, l_j), \tag{6.1}$$

where f is a unary term which measures how well the label satisfies the saliency feature of the segment, and g is a smoothness term that measures compatibility of segments with the same label. By weighting with the number of neighbors we are able to adaptively weight the unary term in such a way that it avoids the bias due to a large number of neighbors. The unary potential f is defined as

$$f(l_i) = (-l_i s_i + (1 - l_i)(1 - s_i)).$$

Here we penalize low saliency for figure segments ($l_i = 1$) and high saliency for ground ($l_i = 0$). The smoothness term g is a combination of two terms g_c, and g_m which measure color and motion compatibility respectively

$$g(l_i, l_j) = \lambda g_c(l_i, l_j) + (1 - \lambda) g_m(l_i, l_j).$$

λ is a parameter that measures the relative importance of each term. The terms g_c, and g_m are defined as

$$g_c(l_i, l_j) = \begin{cases} 0 & l_i \neq l_j \\ exp(-\frac{1}{2}(\mathbf{c}_i - \mathbf{c}_j)^T \mathbf{\Sigma}_c^{-1} (\mathbf{c}_i - \mathbf{c}_j)) & l_i = l_j \end{cases},$$

where Σ_c is a diagonal covariance matrix.

$$g_m(l_i, l_j) = \begin{cases} 0 & l_i \neq l_j \\ exp(-\frac{\|(\mathbf{m}_i - \mathbf{m}_j) \odot \mathbf{o}_i \odot \mathbf{o}_j\|_2}{2(o_i^T o_j)\sigma_m^2}) & l_i = l_j \end{cases},$$

where $\odot$ denotes the element wise multiplication, and σ_m is a parameter that controls how strongly dissimilarity is penalized. Both g_c and g_m penalizes large color or motion differences between neighboring segments with the same label.

The global minimum of the energy function (6.1) can be found using graph cuts [37]. The obtained labels induces a final labeling $F = \cup_{l_i=1} V_i$.

6.1.2 Conclusion

We presented an approach that accurately computes figure-ground labeling from video sequences. By leveraging saliency information together with color and motion cues, it produces labeling that encapsulate salient regions while respecting the natural grouping of objects together. We demonstrated its efficacy by evaluating it on challenging sequences that contain both nonrigid and fast motion. In the future we would like to create a benchmark video dataset for figure-ground labeling of video sequences based on user input.

6.2 Figure-Ground Labeling of Point Trajectories

Given the motion segmentation method presented in 5.2, a motion segmentation cluster represents a part of an object or a part of the background. In this step we proceed by labeling these clusters either foreground or background and thus achieving figure/ground separation.

The problem of figure/ground separation is well studied in psychology of vision. In the case of a single image, the definition of figure/ground can be quite ambiguous. Among the most important factors psychologist studies pointed out to determine figure/ground are: Surroundedness, Size , Orientation, Contrast, Symmetry, Convexity, and Parallelism of the contour. Dynamic figure/ground processing seems to be far less ambiguous compared to single image figure/ground processing. The proposed approach uses motion grouping based on trajectory analysis (common fate), compactness, surroundedness, and spatial closeness. By combining multiple cues, the approach is much more robust than previous approaches which typically assumes that background is the

cluster with largest number of trajectories.

Formally, an energy function is defined over labellings $L = \{l_1, \ldots l_R\}$, where $l_i \in \{0, 1\}$ (0≡foreground, 1≡background) is the label given for each cluster. The energy function encapsulates the evidence from multiple cues.

$$E_l(L) = \alpha_C \sum_i \phi_C(l_i) + \alpha_A \sum_{(i,j)} \phi_A(l_i, l_j) + \alpha_B \sum_{(i,j)} \phi_B(l_i, l_j) + \alpha_S \phi_S(L)$$

The first term of the energy function $\phi_C(l_i) = (1 - l_i) \cdot (max(\frac{var(x)}{var(y)}, \frac{var(y)}{var(x)}) - 1.5)$ is a unary potential that measures the compactness of each cluster in the spatial domain. Foreground objects are more likely to be elongated in the horizontal or vertical direction while background clusters are more spread. The remaining pairwise potentials are only defined for clusters with trajectories that are spatially close in the frame. We define a pairwise potential $\phi_A(l_i, l_j) = -l_i l_j \xi_{\text{Affine}} + (l_i(1 - l_j) + l_j(1 - l_i))\xi_{\text{Affine}}$ which encourages affine motion compatibility between two clusters with background labels and discourages compatibility between foreground and background clusters. The affinity term $\xi_{\text{Affine}} = exp(-var(\text{AffErr}))$ is computed by estimating an affine subspace out of the trajectories in both clusters and then measuring the variance of the error AffErr in projecting these trajectories on the subspace. For this computation we only use trajectories over a small window of frames. $\phi_B(l_i, l_j) = -l_i l_j \xi_{\text{Embed}} + (l_i(1 - l_j) + l_j(1 - l_i))\xi_{\text{Embed}}$ tests for the existence of a clear boundary in the embedding space by penalizing distant clusters labeled as background, and close foreground and background clusters. The affinity ξ_{Embed} is defined as $exp(-min_{\forall x_i, x_j} \|x_i - x_j\|)$, where x_i is an embedding coordinate of a trajectory in cluster i and x_j is an embedding coordinate of a trajectory in cluster j. Finally, $\phi_S(L)$ computes a measure surroundedness of the foreground. Let F, and B denote the set of points belonging to the foreground and background clusters respectively. Then $\phi_S(L) = 1 - \frac{|F \in \text{ConvexHull}(B)|}{|F|}$. $\alpha_C, \alpha_A, \alpha_B, \alpha_S$ are coefficients the determine the relative importance of each term. Since the number of clusters is typically small < 10, the optimal assignment for the labeling can be found by evaluating all possible assignments and finding the minimum.

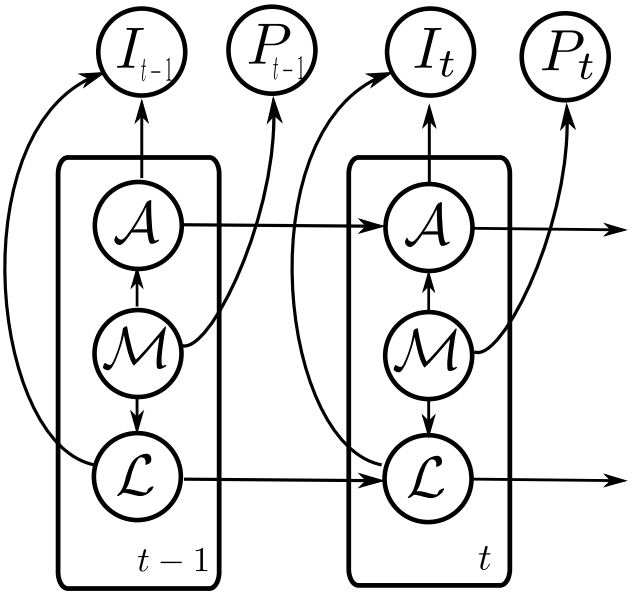


Figure 6.2: Graphical model.

6.3 Bayesian Scene Layer Segmentation

6.3.1 Overview

We begin by introducing some notation. Our Bayesian filtering framework is represented by the graphical model in Fig 6.2(a). At time t our state consists of a tuple $s_t = (\mathcal{A}_t, \mathcal{M}_t, L_t)$. $\mathcal{A}_t = \{A_{b,t}, A_{f,t}\}$ represents the appearance models of the background and foreground respectively. Similarly, $\mathcal{M}_t = \{M_{b,t}, M_{f,t}\}$ are the motion models for the background and foreground. Finally, $L_t = \{l_t^i : l_t^i = \{b, f\}, i = 1 \dots N\}$ where N is the number of pixels, is a pixel wise labeling of the image pixels at time t. For convenience, let $\phi(i) = (x, y)$ be the function that transforms pixel indices to coordinates.

We adopt a camera centric representation for both the appearance and motion models. Let $k = \{b, f\}$ denote which layer the variable belongs to. Let $a_{k,t}^i$ denote a random variable representing the appearance of the ith pixel of layer k at time t. Let $m_{k,t}^i = [u_{k,t}^i \; v_{k,t}^i]^T$ denote a random variable representing the reverse motion of pixel i of the k^{th} layer between frames t and $t-1$. By grouping these random variables by

layer we get $A_{k,t} = \{a^i_{k,t} : i = 1 \dots N\}$ and $M_{k,t} = \{m^i_{k,t} : i = 1 \dots N\}$.

We have two sources of observations, frames I_t and sparse labeled motion vectors P_t. Let I^i_t denote the color of the i^{th} pixel. The labeled sparse motion vectors at frame t is a set $P_t = \{p_{j,t} : j = 1 \dots M\}$ of tuples $p_{j,t} = (q_{j,t}, w_{j,t}, l_{j,t})$, where q is the pixel location, $w_{j,t} = [u \quad v]^T$ denotes the motion vector and $l^p_{j,t} = \{f, b\}$ denotes its layer.

In our model, foreground and background are represented as two layers of pixels. Each layer moves according to the motion vectors in the corresponding motion model. To generate a frame, pixels from the background and foreground are selectively selected based on the labels. Formally, the dynamics of the system can be described by the following equations.

$$L_t = \Omega(M_{f,t}, L_{t-1}),\ \mathcal{A}_t = g(\mathcal{A}_{t-1}, \mathcal{M}_t), \tag{6.2}$$

$$\Omega_i(M_{f,t}, L_{t-1}) = l^{j(i,f)}_{t-1},\ g^i_k(A_{k,t-1}, M_{k,t}) = a^{j(i,k)}_{k,t-1}, \tag{6.3}$$

where $j(i,k) = \phi^{-1}(\phi(i) + m^i_{k,t})$, $i = 1 \dots N$. The function $\Omega = [\Omega_i], i = 1, \dots, N$ takes as input the labels at time $t-1$ and motion model of the foreground and produces the labels at time t. Simply put, the label of i^{th} pixels at time t is the same as the label of a pixel $j(i,f)$ which is $m^i_{f,t}$ pixels away in the previous frame. Similarly the function $g = [g^i_k], i = 1 \dots N,\ k = \{f, b\}$ takes as input the motion model of each layer and moves the appearance models accordingly. Together, these functions describe a process by which the new appearances of the foreground and background are generated given the motion of each pixel of each layer. The observation model can be similarly described by

$$I_t = h(\mathcal{A}_t, L_t),\ P_t = z(\mathcal{M}_t), \tag{6.4}$$

$$h_i(\mathcal{A}_t, L_t) = a^i_{k,t} + \epsilon_I, \quad k = l^i_t,\ \epsilon_I \sim \mathcal{N}(0, \Sigma_I)$$

$$z_j(\mathcal{M}_t) = (j, m^j_k + \epsilon_P, k) \text{ for } j \in 1 \dots N, \quad \epsilon_P \sim \mathcal{N}(0, \Sigma_P)$$

The function $h = [h_i], i = 1 \dots N$ describes how the image is generated given the appearance models and the labels. If the label of pixel i is f, then the observed appearance is the same as the appearance of pixel i in the foreground model $a^i_{f,t}$ plus

some white noise with covariance Σ_I and vice versa. The function $z = [z_j]$, $j = 1 \dots N$ describes how labeled sparse motion vectors P_t are generated given the motion model. The observed sparse motion vector is simply the pixel index, the motion vector from the corresponding motion model with an added white noise with covariance Σ_P, and the label of the vector. Without loss of generality, we observe P_t for a subset of pixels only. Since our Bayesian filtering framework assumes that the observation P_t is available, we describe later how to compute a sparse set of motion trajectories and their associated labels.

Our algorithm can be summarized in the following steps. Using the first few frames in the video sequence we perform initialization (Subsection 6.3.3). For each consecutive frame afterwards, we maintain a low-dimensional representation of the sparse trajectories and cluster them (Section 5.2). Next, we use multiple cues to label each cluster as foreground or background, and thus obtain P_t (Section 6.2). From the clustered sparse trajectories, the motion models $\mathcal{M}_t$ are inferred. Finally, compute the updated appearance models $\mathcal{A}_t$ and labels L_t given the frame I_t, the previous frame appearance models $\mathcal{A}_{t-1}$, labels L_{t-1}, and the inferred motion model $\mathcal{M}_t$ (Section 6.3.2).

6.3.2 Motion Estimation and Bayesian Filtering

Motion Estimation

Given the labeled sparse trajectories, we estimate the motion model for each layer by the marginal posterior probability $p(m^i_{k,t} | p_{j,t} : j = 1 \dots M, l_{j,t} = k)$. However, not all pixels in a layer are associated with a trajectory and, without any assumption, inferring the motion is therefore an ill-posed problem. In reality however, background objects are rigid and foreground objects are articulated. This implies the motion of nearby pixels to be smooth.

For each layer, we construct a pairwise MRF with a set of vertices $\mathcal{V} = \{m^j_{k,t} : j = 1, \dots, N\}$. The set of edges $\mathcal{E} = \{(i, j) : j \in \mathcal{N}(i)\}$ represents pairwise neighborhood relationships on a grid structure defined over the image. The joint posterior probability with respect to $M_{k,t}$ is given by

$$p(M_{k,t}|P_t) \propto \prod_{(i,j)\in\mathcal{E}} \Phi(m^i_{k,t}, m^j_{k,t}) \prod_{i\in\{q_{j,t}:j=1\dots M\}} \Psi(m^i_{k,t}, P_{j,t}), \tag{6.5}$$

where

$$\Phi(m^i_{k,t}, m^j_{k,t}) = \mathcal{N}(m^i_{k,t} - m^j_{k,t}|0, \Sigma_m), \quad \Psi(m^i_{k,t}, P_{j,t}) = \mathcal{N}(m^i_{k,t}|w_{j,t}, \Sigma_p) \tag{6.6}$$

Σ_m, Σ_p are two bandwidth matrix that represent the strength of the relationship between neighboring pixels and covariance of the observed motion vectors respectively. Since both unary and pairwise potentials are Gaussian, this is an instance of Gaussian Belief Propagation (GaBP)[64]. It follows that the joint distribution can be written as $p(M_{k,t}|P_t) \propto e^{-\frac{1}{2}m^TVm+m^Tb}$, where $m = [m^1_u, m^1_v \dots,\ m^N_u, m^N_v]$ is the vector of random variables and $V,\ b$ are the inverse covariance matrix and shift vector respectively. It is straightforward to write down V, b from (6.6). The marginal posterior probability $p(m^i_{k,t}|P_t) = \mathcal{N}(\mu^i_{k,t}, \Sigma^i_{k,t})$ for $i = 1 \dots N$, $k = \{b, f\}$ can be obtained in closed form as $\mu^i_{k,t} = \{V^{-1}b\}_i,\ \Sigma^i_{k,t} = \{V^{-1}\}_{ii}$.

Bayesian Filtering

Our ultimate goal is to estimate L_t given all observations. This can be accomplished by Bayesian filtering. In the first step we predict the appearance models and labels given the motion models. In the second step we update the prediction using the most recent observation. For brevity we will drop the dependence on past observations for the remainder of this section.

Prediction We first predict the appearance model given the motion models. Since our model satisfies the Markov assumption, only the marginal appearance model from the previous time step is needed. The prediction step for the appearance models can be expressed as

$$p(a^i_{k,t}|P_t) = \sum_{j=1}^{N}[\sum_{m^i_{k,t}\in\mathbb{N}^2} p(a^i_{k,t}|m^i_{k,t}, a^j_{k,t-1})p(m^i_{k,t}|P_t)]\ p(a^j_{k,t-1}) \tag{6.7}$$

where (6.7)[1] is derived from marginalizing over the motion model. From (6.2),(6.3) we can derive $p(a^i_{k,t}|m^i_{k,t}, a^j_{k,t-1}) = \delta(a^{j(i,k)}_{k,t-1} - a^i_{k,t})$, where $j(i, m^i_{k,t}) = \phi^{-1}(\phi(i) + m^i_{k,t})$. Equation (6.7) thus becomes

[1]Strictly speaking the summation in (6.7),(6.8), and (6.9) contains also an integration over the pixel areas which is neglected here for brevity.

$$p(a^i_{k,t}|P_t) = \sum_{m^i_{k,t} \in \mathbb{N}^2} p(m^i_{k,t}|P_t) p(a^{j(i,m^i_{k,t})}_{k,t-1}), \tag{6.8}$$

The summation in equation (6.8) will lead to an exponential increase in the number of samples needed to represent the appearance. Thus by approximating $p(m^i_{k,t}|P_t) = \mathcal{N}(\mu^i_{k,t}, \Sigma^i_{k,t})$ with $p(m^i_{k,t}|P_t) = \delta(\mu^i_{k,t} - m^i_{k,t})$ in (6.8) we can avoid this problem and obtain $p(a^i_{k,t}|P_t) \approx p(a^{j(i,\mu^i_{k,t})}_{k,t-1})$. Therefore, we use nonparametric density estimation to represent the probability density of the appearance models. Specifically, for each layer, the density of each pixel is represented by a set of N_{KDE} color samples in rgs[2] color space. At any instance, if the color samples for a pixel will exceed N_{KDE}, we discard the oldest color sample. The bandwidth of the KDE is chosen adaptively using the method in [14].

To predict the labels we similarly have

$$p(l^i_t = f|P_t) = \sum_{m^i_{f,t} \in \mathbb{N}^2} p(m^i_{f,t}|P_t) p(l^{j(i,m^i_{f,t})}_{t-1} = f). \tag{6.9}$$

This summation is evaluated exactly and takes into account the uncertainty in the motion vectors.

Update In this step the new observation is incorporated by updating the marginal priors of the appearance model and labels. It turns out that if we want to avoid maintaining a joint belief space of labels and appearance models, updating both the appearance and labels is a chicken and egg problem. Given the label at a pixel one can easily update the corresponding appearance model. On the other hand, if the appearance of a pixel is known, marginal posterior probability of the labels can be computed easily.

We address this by taking the former approach. Given the predicted label prior and appearance models a MAP estimate of the labels is inferred. Next the inferred labels are used to update the corresponding appearance model. In practice we have found that this approach yields excellent results while avoiding the complexity of maintaining a joint belief space. Note that the marginal posterior over the labels is maintained for the next prediction stage.

---

[2]rgs can be computed from RGB by s = R+G+B,r=R/s ,g=G/s

$$p(l_t^i = k|I_t^i, P_t) \propto p(I_t^i|l_t^i = k)p(l_t^i|P_t) = \int_{a_k^i} p(I_t^i|l_t^i, a_{k,t}^i)p(a_{k,t}^i|P_t)\ \mathbf{d}a_{k,t}^i\ p(l_t^i|P_t) \tag{6.10}$$

Since we previously approximated the posterior probability of motion model (by discarding the uncertainty) in the prediction of the appearance model, the correct appearance can be anywhere in a neighborhood around the current pixel. We therefore replace $p(I_t^i|l_t^i = k)$ in (6.10) with $\sum_j p(I_t^i|l_t^j) \cdot \mathcal{N}(\phi(j) - \phi(i)|0, \Sigma_{k,t}^i)$, where $\Sigma_{k,t}^i$is the uncertainty in the motion model at pixel i.

A pairwise MRF, defined over a grid structure over the pixels, is used to enforce smoothness on the labels

$$p(L_t|I_t) \propto \prod_{(i,j)\in\mathcal{E}} \Phi(l_t^i, l_t^j) \prod_i \Psi(l_t^i),$$

where $\Phi(l_t^i, l_t^j) = \mathcal{N}(I_t^i - I_t^j|0, \Sigma_l)(l^i l^j + (1 - l^i)(1 - l^j))$ and $\Psi(l_t^i) = l^i \cdot p(l_t^i|I_t^i, P_t) + (1 - l^i)(1 - p(l_t^i|I_t^i, P_t))$. Σ_l is a bandwidth matrix that represents the strength of the relationship between neighboring pixels. The globally optimal solution l_{MAP}^i can be found efficiently via graph cuts[3].

Finally, updating the corresponding appearance model is done by simply adding the observed frame pixel color I_t^i to the set of samples of the corresponding pixel in layer l_{MAP}^i.

6.3.3 Continuous Initialization

In the first few frames, the appearance models of some pixels will not have the minimum number of samples needed to compute the appearance likelihood. Therefore, a method must be devised to initialize the appearance models of these pixels. From the incremental trajectory clustering we obtain a set of sparse labels l_s^j at a subset of the pixels $j \in S$. To propagate these labels we construct a pairwise MRF with a set of vertices $\mathcal{V} = \{l_t^i : i = 1, \ldots, N\}$. The set of edges $\mathcal{E} = \{(i, j) : j \in \mathcal{N}(i)\}$ represents a grid structure defined over pixel neighborhood in the image. The joint posterior probability with respect to L_t is given by

$$p(L_t|I_t) \propto \prod_{(i,j)\in\mathcal{E}} \Phi(l_t^i, l_t^j) \prod_{i\in S} \Psi(l_t^i, l_s^i), \tag{6.11}$$

where $\Phi(l_t^i, l_t^j) = (l^i(1 - l^j) + (1 - l^i)l^j) \cdot \mathcal{N}(I_t^i - I_t^j|0, \Sigma_l)$ and $\Psi(l_t^i, l_s^i) = (l^i l_s^i + (1 - l^i)(1 - l_s^i))$, Σ_l is a bandwidth matrix that represents the strength of the relation ship between neighboring pixels. The MAP estimate can be computed efficiently by

Graph Cut [3]. This labeling defines a segmentation of the image into foreground and background pixels. For each pixel which does not have the minimum number of KDE samples, depending on the inferred label l^i of each pixel, the color I^i of the pixel is added to the foreground or background appearance model.

6.3.4 Conclusion

We introduced a method that accurately models appearance and motion to achieve robust moving camera background subtraction. Unlike previous approaches, it merges the best of both worlds, long term trajectories to accurately model long term motion dependencies and a Bayesian filtering framework to reason about pixel level appearance models for foreground and background regions. This is achieved in a online framework without sacrificing the accuracy and with a constant processing time per frame. The output is not only the final segmentation per frame but in addition the pixel based background and foreground model. Such models, we believe, are essential for high level reasoning. We evaluated the approach on benchmark sequences as well as on two challenging sequences and demonstrated that the method produces superior results.

Chapter 7

Experiments

Here we provide quantitative and qualitative evaluations of the different approaches provided in this dissertation. In Section 7.1 we present online motion segmentation results for the approach described in Section 5.3. In Section 7.2, we provide experimental results for the figure/ground labeling approach describe in Section 6.1. Although we have presented two approaches for motion segmentation, the approaches that uses repeated dimensionality reduction is intended to be used in conjunction with Bayesian scene layer segmentation. Results for the combined system are presented in Section 7.3.

7.1 Online Motion Segmentation

In this section we evaluate our online motion segmentation algorithm (Chapter 5) on the Berkeley motion segmentation dataset introduced in [7]. The dataset consists of 26 sequences that include rigid and articulated motion. The ground truth for the dataset is provided as frame annotations for 189 frames and the dataset comes with an evaluation tool. However, it is important to note that the evaluation tool was designed for offline algorithms. For instance, it assumes that each trajectory is assigned a single label throughout the sequence and will thus penalize a trajectory which is assigned an incorrect label at the beginning of a video sequence even if the label is corrected at a latter frame. Similarly, if an object is stationary and then moves, the approach is penalized for not segmenting the object while it is stationary. This puts our algorithm at a disadvantage, since it is impossible to detect motion before it occurs. In real applications this can be easily mitigated via a look ahead process, where the decision is delayed by letting the algorithm run several frames ahead. For the sake of consistency we report error measures using the same evaluation tool.

Trajectory tracking is done using LDOF [53] but the approach is not limited to it. As demonstrated by the Middlebury benchmark [1], there now exists several real-time implementations of accurate optical flow that runs on the GPU. For example, [62].

The evaluation tool of [7] yields 5 measures for each sequence, which are then averaged across all sequences. The 5 measures are density, overall error, average error, over-segmentation error and the number of segments with less than 10% error which we abbreviate as *lt10*. The density measures the percentage of labeled trajectories to the total number of pixels. A higher number indicates better coverage of the image. Algorithms that require full trajectories over a sliding window reduces the density. The overall error is the total number of correctly labeled trajectories over the number of labeled trajectories. The tool automatically computes an assignment of clusters to ground truth regions and may assign several clusters to the same region. The average clustering error is the average of the ratio of mislabeled trajectories to the number of trajectories for each region. Since the tool may assign multiple segments to the same ground truth region, the tool also reports an over-segmentation error defined as the number of segments merged to fit the ground truth regions. Additionally the tool reports the number of regions covered with less than 10% error with one region subtracted per sequence to account for the background.

In our experiments we set the parameters to the following values; $\Delta = 3$, $\sigma_S = 300$, $\eta = 0.1$, and $\tau = 1 \times 10^{-3}$. The value for σ_S was set high enough to capture similarity between different parts of the background when a foreground object splits it into two disjoint regions. The remaining parameters were set empirically.

We compare our algorithm to the offline algorithms of [7], RANSAC, GPCA [59], and LSA [67]. The code for RANSAC, GPCA, and LSA was obtained from the Hopkins dataset [57]. It is important to note that [7] is used as a representative of offline spectral clustering algorithms. For the case where we run over more than 10 frames we compare with a baseline online algorithm that uses RANSAC over a sliding window. As noted in [7], other motion segmentation algorithms do not scale efficiently with number of trajectories. For example, over only 10 frames from the people1 sequence, GPCA takes 2963 seconds, and LSA [67] 38614 seconds. It is therefore infeasible to run these

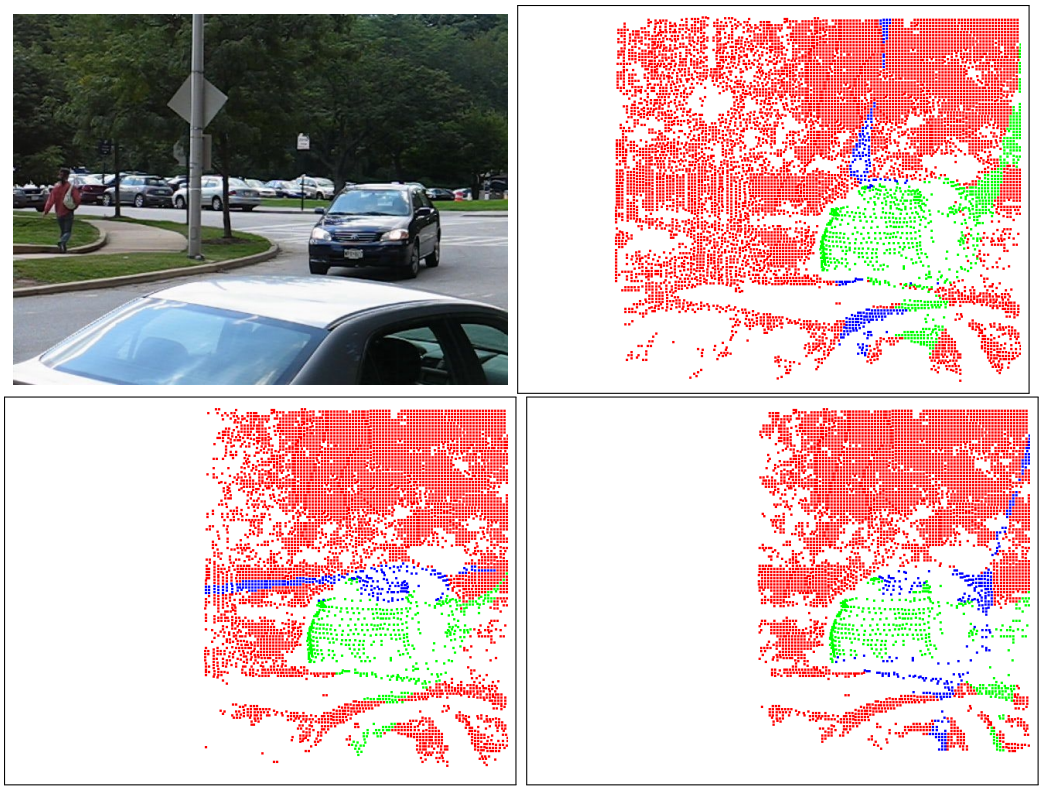


Figure 7.1: Effect on increasing the windows size on the sliding window RANSAC Results. Top Left: frame 40 of the cars4 sequence. Top Right, Bottom Left to Right: the segmentation with sliding window values of 10, 20, and 30 respectively. As the sliding window size increase, less trajectories span the entire window. (Best seen in color).

algorithms on a sliding window.

The RANSAC baseline is always given the total number of ground truth regions in the sequence. Even though increasing the window size may have improved the results, this would have been achieved by reducing the density drastically as it becomes harder to find trajectories that span the entire window. Figure 7.1 shows the effect of increasing the window size on the density and segmentation.

Table 7.1 presents the quantitative results of running our approach on the dataset introduced in [7]. We perform three sets of experiments. In the first, we compare our approach to [7], RANSAC, GPCA, and LSA over the first 10 frames while excluding the first frame. This experiment is designed to quantify the performance of the algorithm with respect to traditional motion segmentation algorithms that require the set

of trajectories to span the entire sequence. In the second, we evaluate the approach over the first 200 frames. To avoid bias in the result due to initialization, we evaluate on ground truth frames starting on or after the 50th frame. This experiment is designed to quantify the performance of the algorithm on long sequences. Such sequences, represent the typical use case of an online algorithm. Finally, in the third set of experiments we evaluate over the entire set of sequences and ground truth annotation images.

Over 10 frames we out-perform GPCA, RANSAC, and LSA while achieving comparable results to results [7]. In fact, if we restrict ourselves to longer sequences we outperform [7] as can be seen in the second experiment. This indicates that our online approach outperforms traditional approaches while maintaining competitive accuracy. Comparing with RANSAC over longer sequences exposes the main problem with any online approach that is based on a sliding window. Information outside the sliding window is not remembered and it is therefore common to merge objects that were known to move differently.

Finally, over the entire dataset, we achieve online performance at the cost of slightly worser performance than [7]. These errors can possibly be further reduced if we employ a look-ahead process where the decision for a trajectory is delayed for several frames.

Figure 7.2 shows the result of applying our method to the *marple2* sequence. At frame 50, the method had already recovered from the bad initialization and is segmenting Miss Marple correctly. Finally as Miss Marple reappears from behind the column, our method re-detects the segment.

Table 7.2. compares the running time of different algorithms over the first 10 frames of the *marple1* sequence. Although the method of [7] uses 19 seconds for the first 10 frames, applying it on a sliding window would require 19 sec. per frame. On the other hand, our non-optimized Matlab implementation takes around 3 seconds per frame. Table 7.3 further shows that the computational time is dominated by updating the $n \times n$ distance matrix and computing the affinity matrix. We believe that real-time performance can be easily achieved since such operations can be easily parallelized on the GPU.

	Density	Overall Error	Average Error	Overseg	lt10
First 10 frames (26 sequences)					
Ours	3.43%	9.69%	29.93%	0.31	21
[7]	3.43%	7.49%	25.92%	0.46	20
RANSAC	3.37%	14.4%	29.87%	0.73	13
GPCA	3.37%	17.86%	28.64%	0.85	7
LSA	3.37%	19.69%	39.76%	0.92	6
Frames 50 - 200 frames (7 sequences)					
Ours	3.26%	6.77%	33.44%	2.57	6
[7]	3.43%	8.32%	37.29%	3.14	6
RANSAC	2.43%	28.3%	45.46%	1.42	0
All frames (26 sequences)					
Ours	3.22%	9%	32.89%	2.30	16
[7]	3.31%	6.82%	27.34%	1.77	27
RANSAC	2.28%	16.04%	42.6%	1.15	9

Table 7.1: Evaluation results on the Berkeley Dataset

Algorithm	Tracks	Time (seconds)
Our method	4625	29
Brox et al. [7]	4699	19
GPCA	4625	1345
LSA	1012	1996

Table 7.2: Computation Time over 10 frames from marple1 sequence

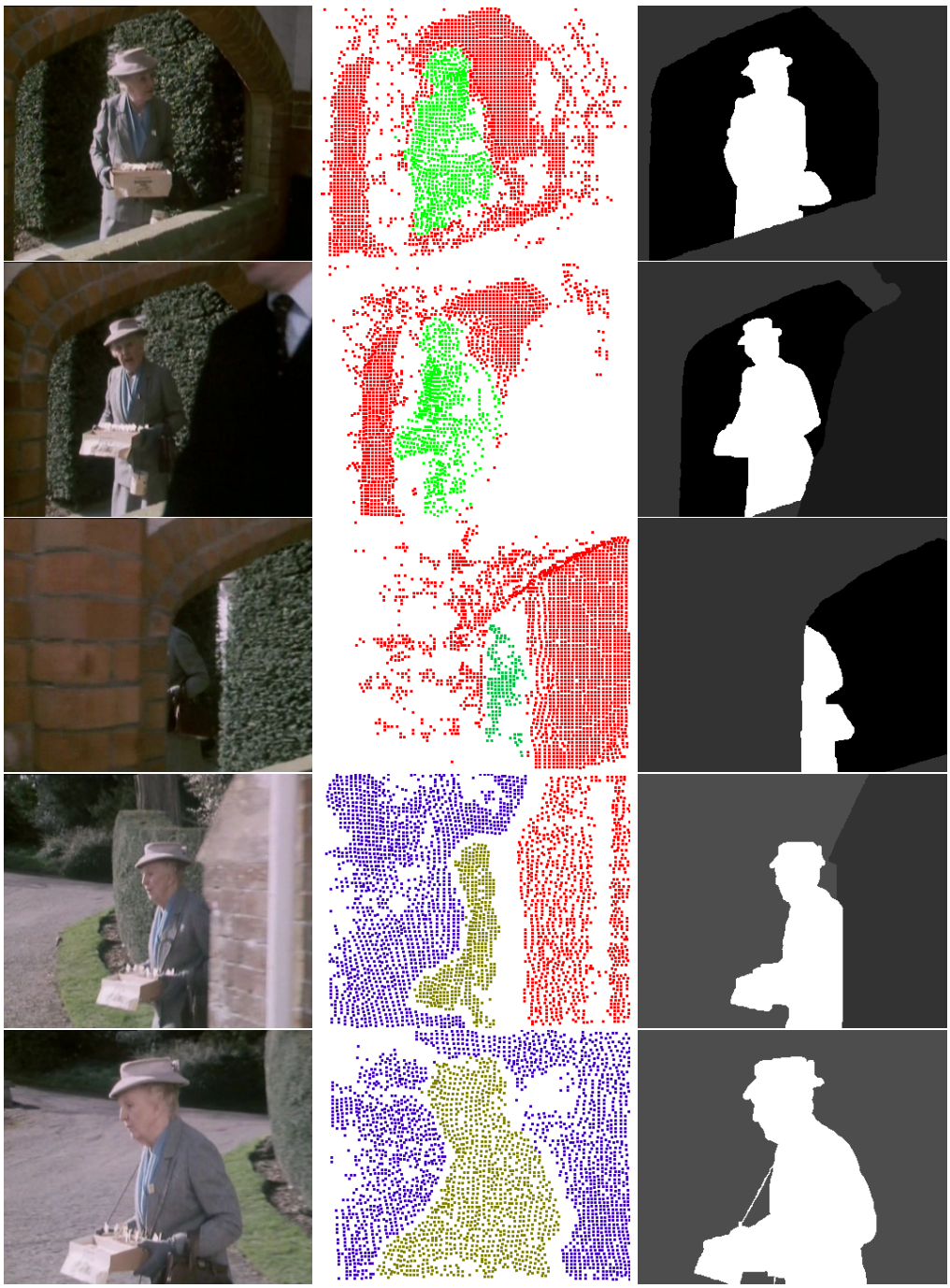


Figure 7.2: Result of our approach on the *marple2* sequence. First Column: frames 50, 110, 135, 170, 200 of the sequence. Second Column: segmentation results. The third column shows ground truth frames associated with the frames. Starting from a incorrect initialization, our approach is able to automatically detect the person in the scene. Clusters maintain their labels under partial occlusion as can be seen with the background segment between frames 110 , 135. However, when a segment is totally occluded its label is lost and is assigned a new label once it is dis-occluded. (Best seen in color.)

track	dist	aff	lblprop	Total
34	1521	1169	403	3027

Table 7.3: Computation time over different stages on a single frame from *marple1*. The stages are: tracking (**track**) **,** distance matrix update (**dist**), affinity matrix (**aff**) and label propagation (**lblprop**). Times are in msec. The number of trajectories is 4427. Optical flow computation used in tracking is not included.

7.2 Figure/Ground Labeling

In this section we evaluate the figure/ground labeling approach proposed in (Chapter 6) on videos from the occlusion boundary detection dataset from [52][1]. This dataset consists of 30 short video sequences (approximately 10-20 frames each). Each exhibits very brief camera motion, instantaneous motion of objects in the scene, or a combination of the two. However, since the dataset was created with the emphasis of boundary detection, the ground truth provided is for individual object segmentations and not for figure-ground segmentation. We resort to qualitative evaluation of the method on these sequences. To evaluate the approach on rapid motion we also use the *yunakin* sequence from the Youtube dataset [27]. For all sequences we choose the following values for our parameter settings $\Sigma_c = diag(100, 15, 15)$, $\lambda = \frac{1}{2}$, $\sigma_m = 2$, $k = 5$.

To our knowledge there exist no prior work on figure-ground labeling for videos in the general setting. To demonstrate the efficacy of our approach, we compare our approach with a baseline which does not use motion or color features. Instead, it uses a smoothness prior that encourages spatial smoothness. Figure. 7.3 shows the results on 3 sequences (*rocking horse*, *walking legs*, and *post*) from [52] and the *yunakin* sequence from Youtube dataset [27]. For each video we show one frame from the sequence, the video segmentation result, color features, saliency features, result of the baseline, and the result of the approach.

Comparing the results of the baseline and the approach shows that the baseline

[1]More results are available at
http://www.cs.rutgers.edu/~elqursh/figgrnd/

suffers from missing parts (*rocking horse*), no figure (*walking legs*, *post*), or no ground (*yunakin*). It indicates that saliency information is not enough to determine figure-ground assignment of video segments. This can be explained in part by the fact that saliency gives a rough estimate of where attention is focused in the image and does not define a grouping of regions into figure. In addition, regions close to object boundaries are usually affected by the existence of a salient region besides it. Our approach is able to successfully obtain the correct labeling in all of the sequences. Errors in the results can be attributed to inaccurate video segmentation along the boundary (*walking legs*, *post*), or small segments close to the object that has a high saliency feature (*walking legs*).

7.3 Scene-Layer Segmentation

In this section we evaluate our scene-layer segmentation method (Section 6.3). We evaluate our algorithm qualitatively and quantitatively on five challenging sequences[2]. Our results are compared to state-of-the-art algorithms that use dense point trajectories [48], and belief propagation and Bayesian filtering [39][3]. We also compare the results of our approach with and without the label prior. Parameter settings for all experiments are provided in the appendix.

Feature Tracking

We use LDOF [53] to track dense feature points over pairs of frames. We divide the image into equal blocks and then subsample these trajectories to keep an almost equal number of tracks in each block. This allows us to avoid excessive bias in the clustering step (Subsection 5.2) due to high concentration of trajectories in highly textured areas. New trajectories are automatically incorporated up to a maximum number of trajectories per frame $T^f_{\max}$. In addition, the total number of trajectories maintained is set to not exceed $T^{\text{memory}}_{\max}$ after which we drop trajectories with the oldest ending frame number. Note that at any instant of time we do not store the full trajectories

[2]Project page: http://www.cs.rutgers.edu/~elqursh/projects/bsmc/

[3]We have requested the code and results from the authors but we did not receive a response, results for their approach are reported verbatim from the paper.

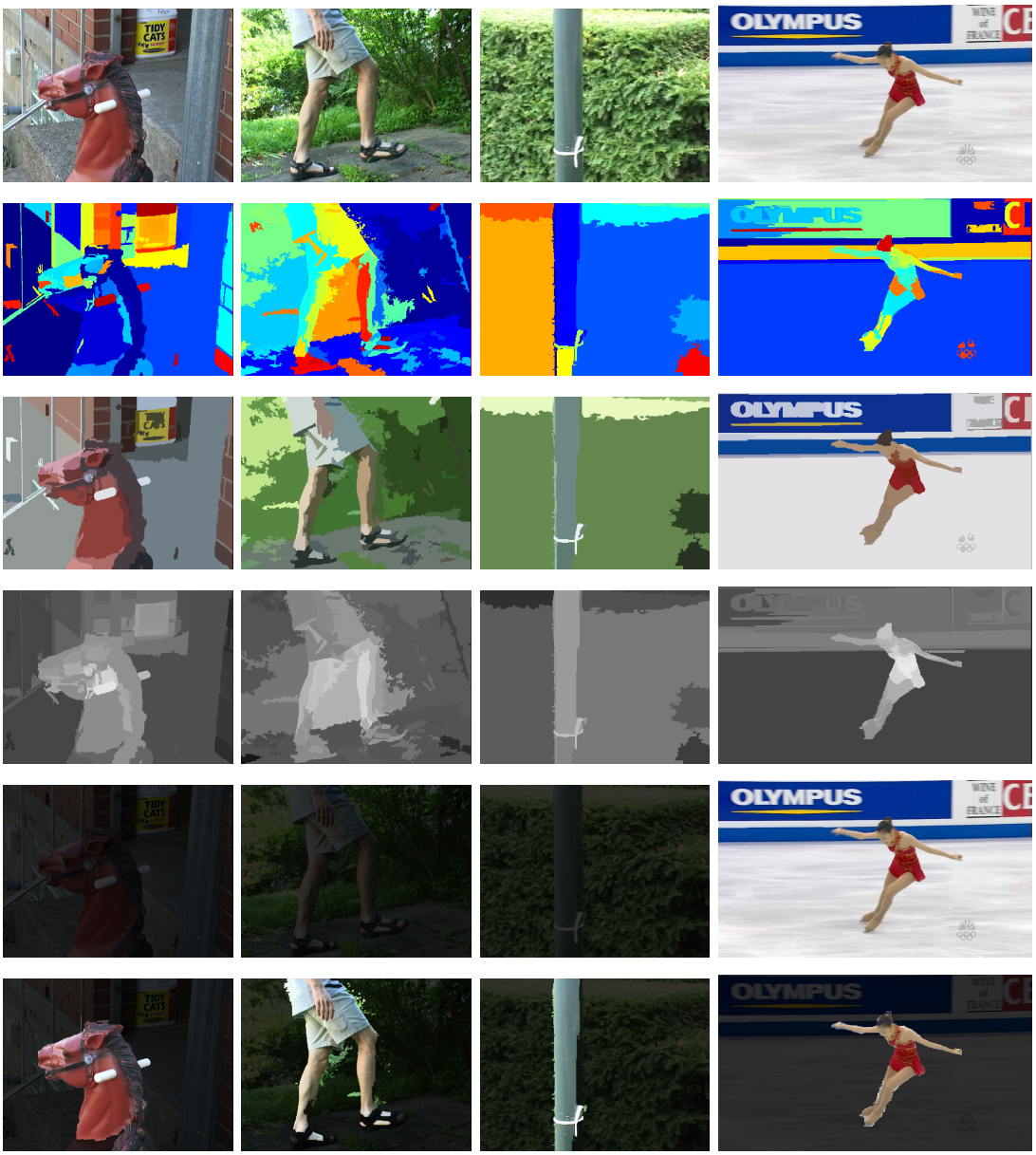


Figure 7.3: Figure-Ground labeling results on 4 sequences. (First row) First frame of the sequence. (2nd row) color coded video segments. (3rd row) Color features. (4th row) Saliency features. (5th row) labeling results using saliency only. (6th row) labeling results using saliency, color, and motion.

but rather maintain a low dimensional representation.

We use the following values for the parameters

- Affinity matrix computation $\lambda_M = 40$, $\lambda_S = 100$.
- Initial number of clusters $R = 5$.
- Figure/Ground Labeling $\alpha_C = 1, \alpha_A = 1, \alpha_B = 1, \alpha_S = 1$.
- Motion Estimation$\Sigma_m = 0.5^2 I_{2\times 2}$, $\Sigma_p = I_{2\times 2}$.
- Bayesian filtering $N_{KDE} = 10$, $\Sigma_l = \frac{10}{255}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 255 \end{bmatrix}$
- Initialization $\Sigma_l = \frac{10}{255}\begin{bmatrix} 1 & & \\ & 1 & \\ & & 255 \end{bmatrix}$
- Trajectory limits $T^f_{max} = 2000$, $T^{memory}_{max} = 2000$.

Results

The first three videos - cars1, people1 and people2 - comes from the Hopkins 155 dataset [57]. Manually annotated ground truth for a subset of frames, around one every 10 frames, is provided by Brox et al [7]. A characteristic of these sequences is that they are short and the objects are always on motion. These sequences are used as a benchmark to compare with other approaches.

Table 7.4. shows quantitative comparison on the first three benchmark sequences[4]. The row labeled ours-1 is our approach with the label prior, while ours-2 is our approach without the label prior. Without the label prior we do not rely on the predicted labels in inferring the new labels. We compare our methods to [48] and [39]. On cars1, people2 our approach ranks 1st in F1-score while in people1 we rank a close second. The reason our approach performs worser on the people1 sequence is that with at

[4]Let TP, FP, and FN denote true positives, false positives, and false negatives. Then $Prec = TP/(TP + FP)$, $Rec = TP/(TP + FN)$, $F1 = 2 \cdot (Prec \cdot Rec)/(Prec + Rec)$.

Table 7.4: Performance comparisons with other methods

	cars1			people1			people2		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Ours-1	0.84	**0.99**	**0.91**	0.94	0.85	0.89	0.69	0.88	0.77
Ours-2	0.85	0.97	0.90	**0.97**	0.88	0.92	**0.87**	0.88	**0.88**
[48]	0.63	**0.99**	0.77	0.78	0.63	0.70	0.73	0.83	0.78
[39]	**0.92**	0.84	0.88	0.95	**0.93**	**0.94**	0.85	**0.89**	0.86

	tennis			drive		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Ours-1	0.86	**0.92**	**0.89**	0.55	**0.96**	**0.70**
Ours-2	**0.90**	0.81	0.85	**0.60**	0.67	0.63
[48]	0.27	0.83	0.40	0.02	0.66	0.04

most 2000 trajectories there are no trajectories from the hips downwards and thus the motion of the legs is not captured by the trajectories. As can be seen from the first row of Figure. 7.5, after initialization the foreground appearance model captures the top portion and it takes a few more frames for it to recover from this error. Figures 7.4, 7.5, 7.6, and 7.7 shows qualitative results on these sequences.

To evaluate the performance of the proposed approach on long sequences with fast motion, two other sequences are used - tennis and drive. The tennis sequence is 466 frames long and also comes with ground truth from [7]. The tennis player pauses at times to wait for the ball, while at others moves fast to intercept it. Due to the fast motion and homogeneous ground color not all objects have trajectory points as seen in Fig.1.3(b). Finally, the drive sequence is 456 frames and was manually annotated with ground truth. This sequence is challenging since cars keep entering and exiting the field of view at different points in the video and due to the forward motion. Figs. 7.4, 7.8 shows qualitative results on the tennis and drive sequences. Thanks to the accurate background model, our method is able to capture the cars on the other side of the road Fig. 7.8.

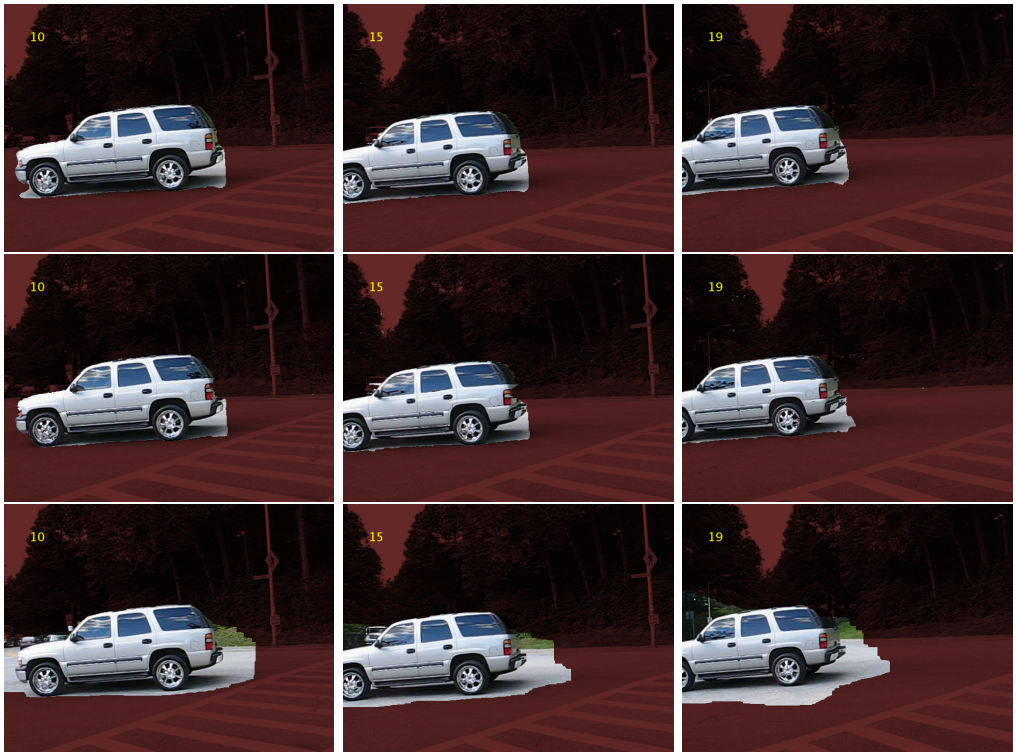


Figure 7.4: Results on cars1 sequence using our method (First row), our method without label prior (Second row), using [48](Third row). For visualization purposes, background regions are darkened while foreground regions maintain their colors. (Best viewed in color).

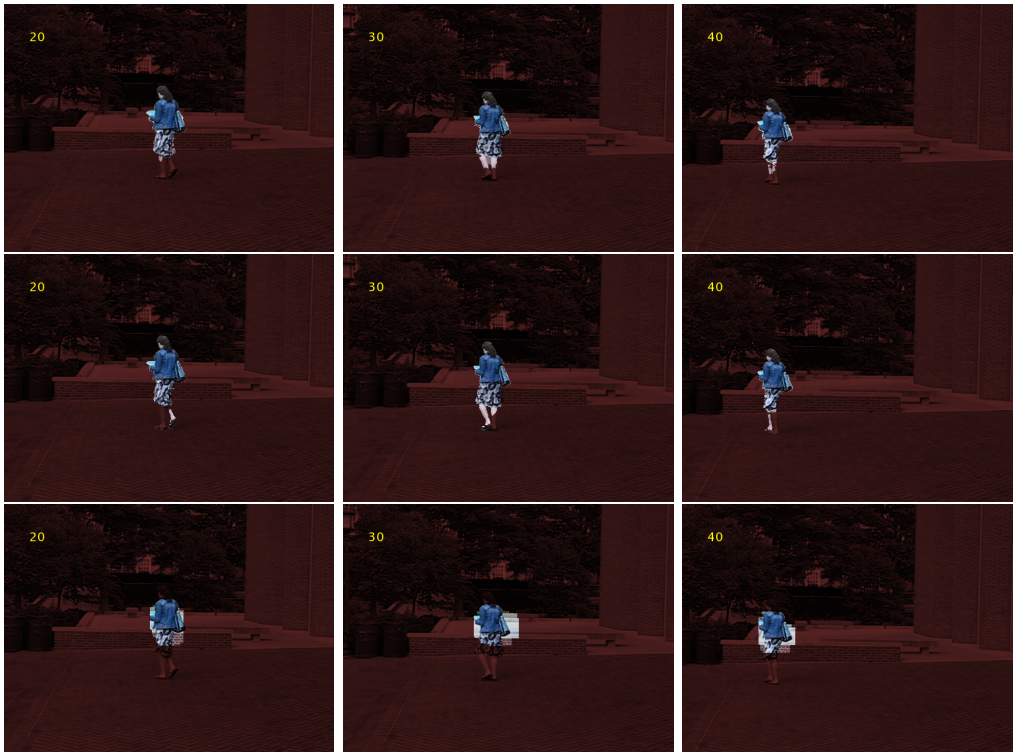


Figure 7.5: Results on people1 sequence using our method (First row), our method without label prior (Second row), using [48](Third row). For visualization purposes, background regions are darkened while foreground regions maintain their colors. (Best viewed in color).

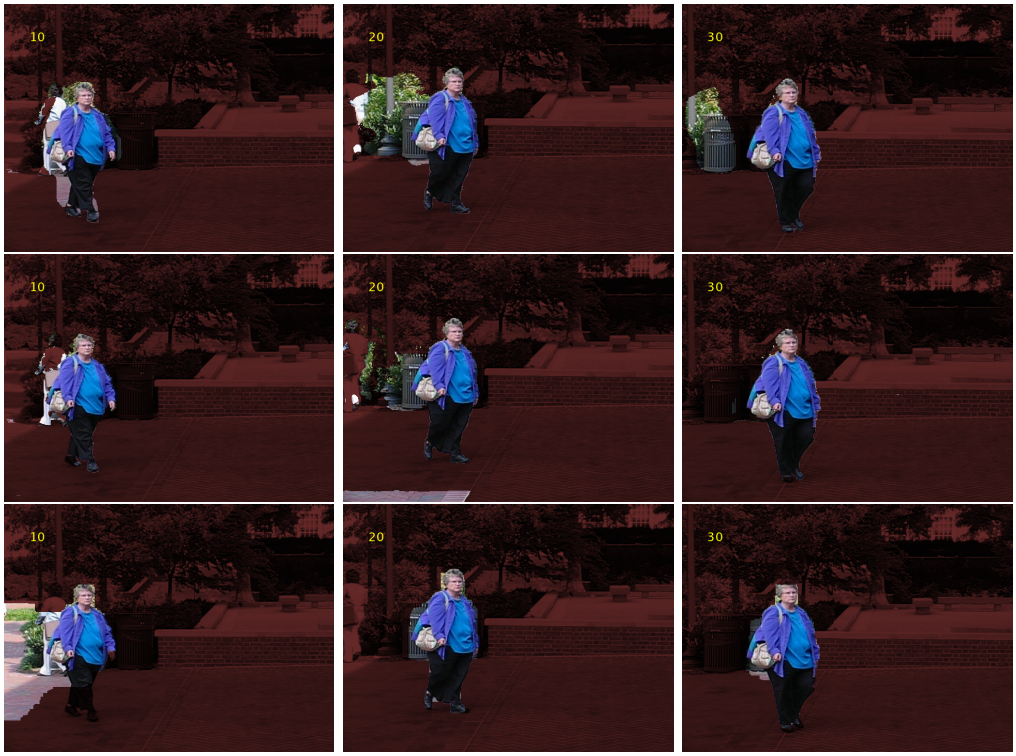


Figure 7.6: Results on people2 sequence using our method (First row), our method without label prior (Second row), using [48](Third row). For visualization purposes, background regions are darkened while foreground regions maintain their colors. (Best viewed in color).

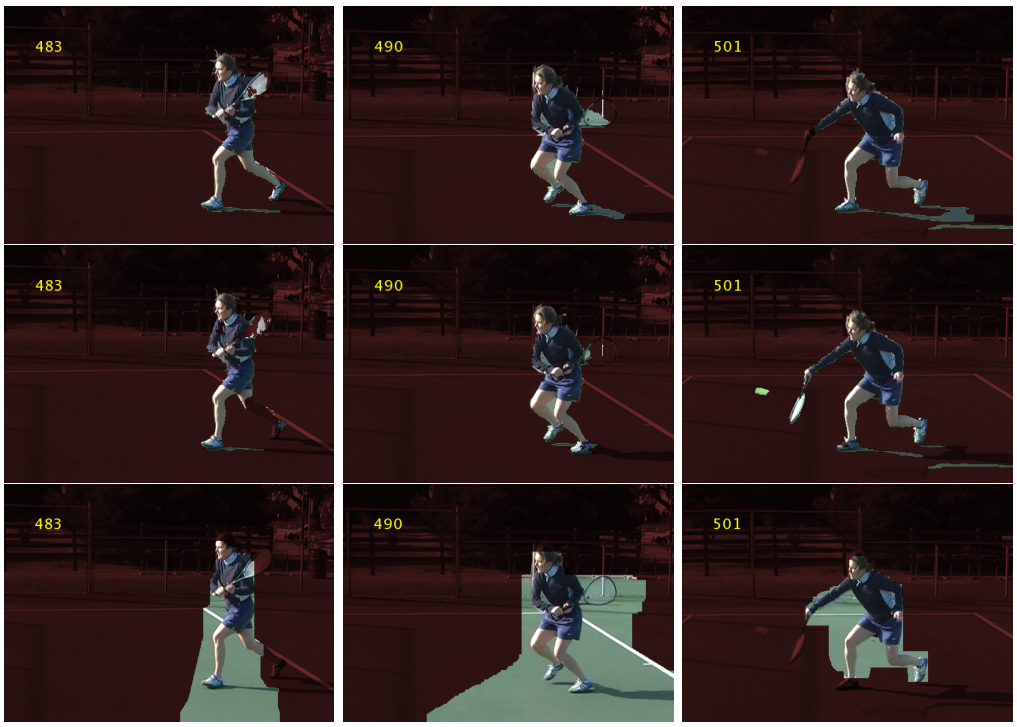


Figure 7.7: Results on tennis sequence using our method (First row), our method without label prior (Second row), using [48](Third row). For visualization purposes, background regions are darkened while foreground regions maintain their colors. (Best viewed in color).

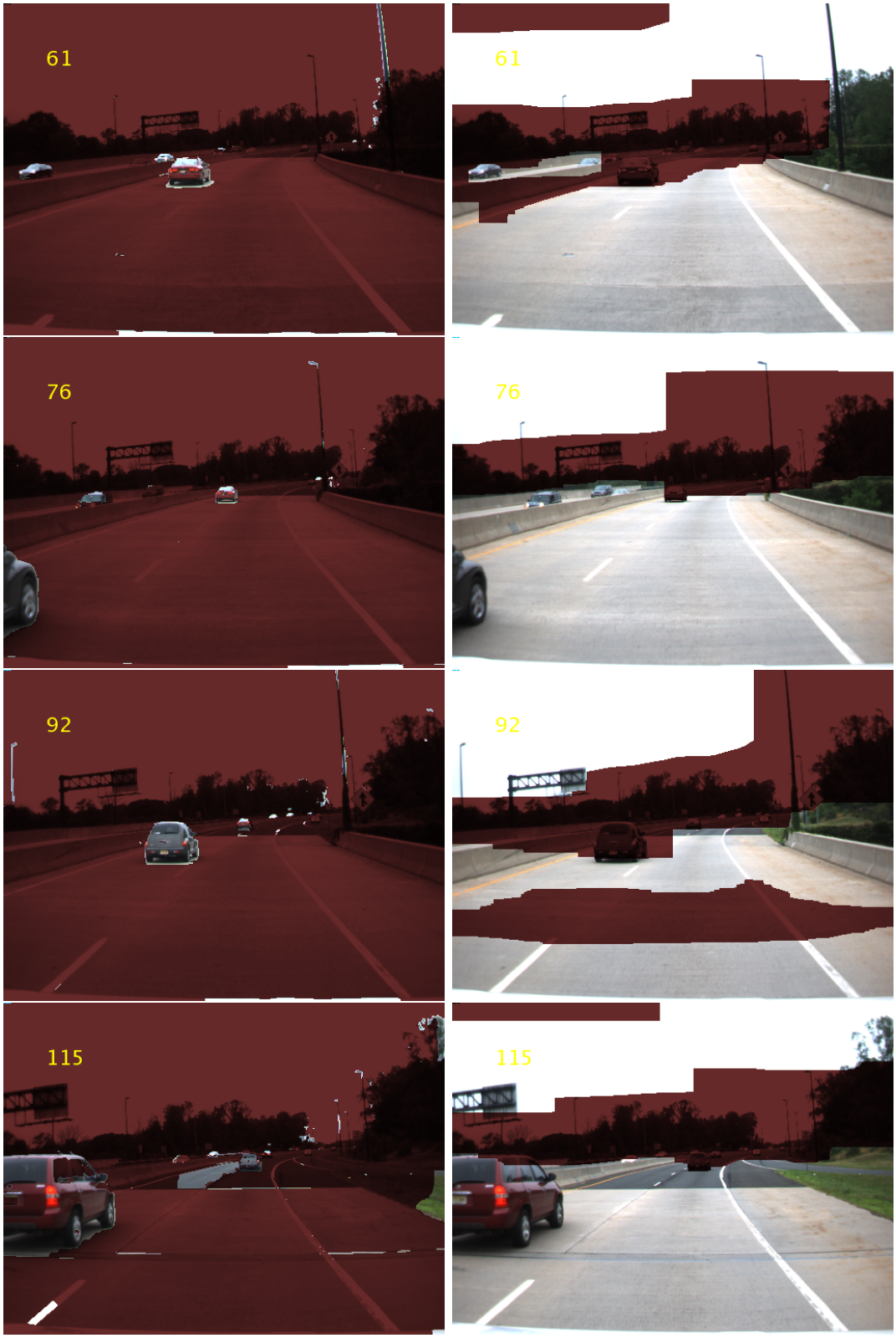


Figure 7.8: Results for drive 1 sequence using our method (Left) and [48](Right). [48] fails on this sequence since it highly deviates from the orthographic projection assumption. Our approach successfully segments the new car as soon as it enters the fields of view at frame 76. Notice also, how cars approaching in the opposite direction are successfully segmented.

Chapter 8

Conclusions

The work presented in this dissertation has provided a solution to the problem of object detection from video. Existing approaches were either not applicable or not effective in solving this problem. For example, while background subtraction is limited to stationary cameras, object detectors required training and suffered from low accuracy. In addition, existing approaches typically only produced a segmentation of the video but no models that can be used for higher level reasoning.

Our framework also addresses several key challenges in object detection from video. By formulating motion segmentation as a manifold separation problem we got rid of the affine camera assumption. On the other hand, our distance metric can be computed online and can handle trajectories of any length. In the context of figure/ground labeling, we showed how multiple cues can be combined to achieve figure/ground labeling. Finally, we solved the trade off between the density of the motion segmentation and the accuracy of the motion models by propagating the sparse motion information using our dynamic Bayesian model. In addition, we maintain dense appearance models of the layers.

There still exist several open areas of research. In the context of motion segmentation, one area for improvement is to use the geometric structure of the manifolds to segment them. For example, the curvature of the manifold can be used to avoid combining separate manifolds in such a way that there is a discontinuity in the manifold. In the context of figure/ground labeling, there exist a big role for semantics in the assignment of figure/ground labels. For example, people and cars are typically foreground.

We are motivated in our approach by several applications that require online processing. For example, real-time motion segmentation can be used to perform video

re-targeting on-the-fly on viewers devices. Even when the videos are available offline, processing movie-long videos would take in the order of weeks using existing offline approaches.

References

[1] Simon Baker, Daniel Scharstein, J P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, November 2010.

[2] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 1396:1373–1396, 2003.

[3] Yuri Boykov and Gareth Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation. *International Journal of Computer Vision*, 70(2):109–131, November 2006.

[4] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 377—-384 vol.1. IEEE, 1999.

[6] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences of the United States of America*, 103(5):1168–1172, January 2006.

[7] Thomas Brox and Jitendra Malik. Object Segmentation by Long Term Analysis of Point Trajectories. In *European Conference on Computer Vision*, pages 282–295, 2010.

[8] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[9] João Costeira and Takeo Kanade. A Multi-body Factorization Method for Motion Analysis. In *International Conference on Computer Vision*, pages 1071–1076. IEEE Comput. Soc. Press, 1995.

[10] Daniel DeMenthon and Remi Megret. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Workshop Statistical Methods in Video Processing ECCV*, 2002.

[11] Piotr Doll, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian Detection : A Benchmark. In *Computer Vision and Pattern Recognition*, 2009.

[12] Jon Driver and Gordon C Baylis. Edge-Assignment and Figure-Ground Segmentation in Short-Term Visual Matching. *Cognitive Psychology*, 306:248–306, 1996.

[13] Rubin Edgar. Visuell wahrgenommene Figuren. *Copenhagen: Gyldendalske*, 192.

[14] Ahmed Elgammal, Ramani Duraiswami, David Harwood, and Larry S Davis. Background and Foreground Modeling using Nonparametric Kernel Density Estimation for Visual Surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, July 2002.

[15] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric Model for Background Subtraction. In *European Conference on Computer Vision*, 2000.

[16] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE, June 2009.

[17] Ali Elqursh and Ahmed Elgammal. Online Moving Camera Background Subtraction. In *European Conference on Computer Vision*, 2012.

[18] Ali Elqursh and Ahmed Elgammal. Video Figure Ground Labeling. In *International Conference on Pattern Recognition*, 2012.

[19] Ali Elqursh and Ahmed Elgammal. Online Motion Segmentation using Dynamic Label Propagation. In *International Conference on Computer Vision*, 2013.

[20] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[21] Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton Univ. Press, 1962.

[22] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nystrom method. *IEEE Transactions on Pattern Analaysis and Machine Intelligence*, 2004.

[23] Katerina Fragkiadaki and Jianbo Shi. Detection Free Tracking: Exploiting Motion and Topology for Segmenting and Tracking under Entanglement. In *Computer Vision and Pattern Recognition*, pages 2073–2080. IEEE, June 2011.

[24] Alvina Goh and René Vidal. Segmenting Motions of Different Types by Unsupervised Manifold Clustering. In *Computer Vision and Pattern Recognition*, pages 1–6. IEEE, June 2007.

[25] Hayit Greenspan, Jacob Goldberger, and Arnaldo Mayer. A Probabilistic Framework for Spatio-Temporal Video Representation & Indexing. In *European Conference on Computer Vision*, 2002.

[26] Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. In *Computer Vision and Pattern Recognition*, volume 1, pages 707–714. IEEE, 2004.

[27] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient Hierarchical Graph-Based Video Segmentation. In *Computer Vision and Pattern Recognition*, pages 2141–2148, 2010.

[28] Rüdiger Heydt, Fangtu T Qiu Krieger, and Zijiang J He. Neural mechanisms in border ownership assignment: motion parallax and gestalt cues. *Journal of Vision*, 3(9):666, 2003.

[29] Derek Hoiem, Andrew N Stein, Alexei A Efros, and Martial Hebert. Recovering Occlusion Boundaries from a Single Image. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[30] Naoyuki Ichimura. Motion Segmentation Based on Factorization Method and Discriminant Criterion. In *International Conference on Computer Vision*, volume 00, 1999.

[31] Michal Irani, Benny Rousso, and Shmuel Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16, February 1994.

[32] Vidit Jain and Erik Learned-Miller. Online Domain Adaptation of a Pre-Trained Cascade of Classifiers. In *Computer Vision and Pattern Recognition*, pages 577–584. IEEE, June 2011.

[33] Allan Jepson and Michael J Black. Mixture Models for Optical Flow Computation. In *Computer Vision and Pattern Recognition*, pages 760–761. IEEE Comput. Soc. Press, 1993.

[34] Nebojsa Jojic and Brendan J Frey. Learning Flexible Sprites in Video Layers. In *Computer Vision and Pattern Recognition*, pages I—-199—-I—-206. IEEE Comput. Soc, 2001.

[35] Tilke Judd, Krista Ehinger, and Antonio Torralba. Learning to Predict Where Humans Look. In *International Conference on Computer Vision*, 2009.

[36] Kenichi Kanatani. Motion Segmentation by Subspace Separation and Model Selection. In *International Conference on Computer Vision*, volume 2, pages 586–591. IEEE, 2001.

[37] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, March 2004.

[38] M Pawan Kumar, P H S Torr, and Andrew Zisserman. Learning Layered Motion Segmentations of Video. *International Conference on Computer Vision*, (iii), 2005.

[39] Suha Kwak, Taegyu Lim, Woonhyun Nam, Bohyung Han, and Joon Hee Han. Generalized Background Subtraction Based on Hybrid Inference by Belief Propagation and Bayesian Filtering. In *International Conference on Computer Vision*, 2011.

[40] Ken Nakayama, Shinsuke Shimojo, and Gerald H Silverman. Stereoscopic depth: its relation to image segmentation, grouping, and the recognition of occluded objects. Perception. *Perception*, 1989.

[41] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition*, pages 614–621. IEEE, June 2012.

[42] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Cue Integration for Figure / Ground Labeling. In *Neural Information Processing*, 2006.

[43] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Figure / Ground Assignment in Natural Images. In *European Conference on Computer Vision*, pages 614–627, 2006.

[44] Xiaofeng Ren and Chunhui Gu. Figure-Ground Segmentation Improves Handled Object Recognition. In *Computer Vision and Pattern Recognition*, number 1, pages 3137–3144, 2010.

[45] Simon Rowe and Andrew Blake. Statistical mosaics for tracking. *Image and Vision Computing*, 14(8):549–564, August 1996.

[46] Peter Sand and Seth Teller. Particle Video: Long-Range Motion Estimation Using Point Trajectories. *International Journal of Computer Vision*, 80(1):72–91, May 2008.

[47] Lawrence K. Saul and Sam T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *JMLR*, 4:119–155, 2003.

[48] Yaser Sheikh, Omar Javed, and Takeo Kanade. Background Subtraction for Freely Moving Cameras. In *International Conference on Computer Vision*, number Iccv, 2009.

[49] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[50] Chris Stauffer and W E L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, pages 246–252. IEEE Comput. Soc, 1999.

[51] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[52] Andrew N Stein and Martial Hebert. Occlusion Boundaries from Motion: Low-Level Detection and Mid-Level Reasoning. *International Journal of Computer Vision*, 82(3):325–357, February 2009.

[53] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. In *European Conference on Computer Vision*, pages 438–451, 2010.

[54] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 2000.

[55] Carlo Tomasi and Takeo Kanade. Shape and Motion from Image Streams under Orthography: a Factorization Method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[56] Philip H S Torr, Richard Szeliski, and P Anandan. An Integrated Bayesian Approach to Layer Extraction from Image Sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):297–303, 2001.

[57] Roberto Tron and René Vidal. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In *Computer Vision and Pattern Recognition*, 2007.

[58] Pavan Turaga, Rama Chellappa, V S Subrahmanian, and Octavian Udrea. Machine Recognition of Human Activities: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[59] René Vidal and Richard I Hartley. Motion segmentation with missing data using powerfactorization and GPCA. In *Computer Vision and Pattern Recognition*, volume 2, pages 310–316, 2004.

[60] John Y A Wang and Edward H Adelson. Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 2(5), 1994.

[61] Jue Wang, Bo Thiesson, Yingqing Xu, and Michael Cohen. Image and Video Segmentation by Anisotropic Kernel Mean Shift. In *European Conference on Computer Vision*, 2004.

[62] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An Improved Algorithm for TV-L Optical Flow. *Visual Motion Analysis*, 1(x):23–45, 2009.

[63] Yair Weiss and Edward H Adelson. A unified mixture framework for motion segmentation : incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition*, 1996.

[64] Yair Weiss and William T Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, October 2001.

[65] Josh Wills, Sameer Agarwal, and Serge Belongie. What Went Where. In *Computer Vision and Pattern Recognition*, 2003.

[66] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analaysis and Machine Intelligence*, 19(7):780–785, July 1997.

[67] Jingyu Yan and Marc Pollefeys. A General Framework for Motion Segmentation: Independent , Articulated , Rigid, Non-rigid ,Degenerate and Non-degenerate. In *European Conference on Computer Vision*, 2006.

[68] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Sch. Learning with Local and Global Consistency. In *NIPS*, volume 1, 2004.

[69] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *International Conference on Computer Vision*, 2003.