Robust Motion Segmentation Using Markov Thresholds
by
Joshua N. Migdal
Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Science at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY September 2003
© Massachusetts Institute of Technology 2003. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
August 27, 2003

Certified by
W. Eric L. Grimson
Professor
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Students
Robust Motion Segmentation Using Markov Thresholds

by

Joshua N. Migdal

Submitted to the Department of Electrical Engineering and Computer Science on August 27, 2003, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

In this thesis, a new method for performing motion segmentation on image sequences is introduced. This method uses the popular background model / background subtraction paradigm used in other motion segmentation algorithms. However, our approach exploits the spatial and temporal dependencies that objects in motion impose on their images. This is achieved by the construction of a Markov random field of binary motion segmentation variates. Three variations of field structures were tested against two traditional threshold-based background subtraction algorithms. The segmentations produced by the MRF-based techniques were consistently more accurate and less prone to noise and background camouflaging effects than their traditional counterparts. While the speed of this new approach is slower than the two traditional algorithms, its time complexity remains linear. Thus, the difference in algorithm speed will become less significant as processors evolve, while the performance characteristics of this new approach will remain superior.

Thesis Supervisor: W. Eric L. Grimson
Title: Professor
Acknowledgments

I would like to take a moment to thank the people that helped make this thesis possible:

- My advisor, Eric Grimson, who not only provided me with the time and resources necessary to conduct my research, but who also gave me the bright idea to use MRFs in the first place.

- Kristina Ekholm, whose love and constant encouragement got me through the most uncertain of times.

- My parents, Mike and Donna Migdal, who have always supported me through all of my endeavors.

- Tomas Izo, who helped me refine my ideas while taking part in many great discussions.
Contents

1 Introduction ................................................................. 9
   1.1 Applications .......................................................... 10
   1.2 Organization .......................................................... 11

2 Background Processes .................................................. 13
   2.1 Background Modeling ................................................ 13
      2.1.1 Kalman Backgrounds ........................................... 14
      2.1.2 Mixtures of Gaussians ....................................... 15
   2.2 Background Subtraction ............................................ 19
   2.3 Background Model/Subtraction Dichotomy ....................... 21
   2.4 Limitations ............................................................ 22

3 Markov Random Fields .................................................. 25
   3.1 Markov Properties .................................................. 25
   3.2 Gibbs Distributions ............................................... 27
   3.3 Computational Factorization ..................................... 30
   3.4 Gibbs Sampling ..................................................... 31

4 Markov Thresholding .................................................... 33
   4.1 Base Field Structure ............................................... 33
   4.2 Posterior Distribution ............................................. 36
   4.3 Field Structures ................................................... 41
      4.3.1 Markov Model $M_1$ ......................................... 41
### 4.3.2 Markov Model $M_2$ ........................................ 41
### 4.3.3 Markov Model $M_3$ ........................................ 43
### 4.4 Implementation Considerations .................................. 46
  - 4.4.1 Potential Functions ........................................ 46
  - 4.4.2 Annealing Schedule ........................................ 47
  - 4.4.3 Optimization Simplifications .............................. 48
### 4.5 What is in a Name ............................................. 48

### 5 Results .................................................................. 53
  - 5.1 Hysteresis Background Subtraction ............................ 53
  - 5.2 Analysis Overview ............................................... 56
  - 5.3 Algorithm Parameters .......................................... 56
  - 5.4 NIST 1 ............................................................ 57
  - 5.5 NIST 2 ............................................................ 68
  - 5.6 PETS .............................................................. 74
  - 5.7 Near Field ......................................................... 81
  - 5.8 Gait Recognition .................................................. 88
  - 5.9 Algorithm Speed .................................................. 93

### 6 Conclusion ................................................................ 95
  - 6.1 Summary .......................................................... 95
  - 6.2 Future Work ....................................................... 97

### A Computational Factorization ......................................... 99

### B Hausdorff Distance ..................................................... 103

### C MRF C++ Source ....................................................... 105
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Typical Motion Segmentation</td>
<td>11</td>
</tr>
<tr>
<td>2-1</td>
<td>Background Model/Subtraction Separation</td>
<td>21</td>
</tr>
<tr>
<td>3-1</td>
<td>Example of a Simple Undirected Graph</td>
<td>26</td>
</tr>
<tr>
<td>3-2</td>
<td>Bayes Net Expansion of a MRF</td>
<td>27</td>
</tr>
<tr>
<td>4-1</td>
<td>Second-Order Neighborhood System</td>
<td>33</td>
</tr>
<tr>
<td>4-2</td>
<td>2-Dimensional Lattice</td>
<td>34</td>
</tr>
<tr>
<td>4-3</td>
<td>Cliques of a Lattice</td>
<td>35</td>
</tr>
<tr>
<td>4-4</td>
<td>Neighborhood System of the Posterior Distribution</td>
<td>37</td>
</tr>
<tr>
<td>4-5</td>
<td>An Image and its Underlying Segmentation</td>
<td>38</td>
</tr>
<tr>
<td>4-6</td>
<td>Neighborhood Structure of $M_2$</td>
<td>42</td>
</tr>
<tr>
<td>4-7</td>
<td>Neighborhood Structure of $M_3$</td>
<td>44</td>
</tr>
<tr>
<td>4-8</td>
<td>Background Matching Threshold</td>
<td>49</td>
</tr>
<tr>
<td>4-9</td>
<td>MRF Dynamic Background Matching Threshold</td>
<td>50</td>
</tr>
<tr>
<td>5-1</td>
<td>NIST 1, Subsequence 1</td>
<td>60</td>
</tr>
<tr>
<td>5-2</td>
<td>NIST 1, Subsequence 1, Center Column</td>
<td>61</td>
</tr>
<tr>
<td>5-3</td>
<td>NIST 1, Subsequence 2</td>
<td>62</td>
</tr>
<tr>
<td>5-4</td>
<td>NIST 1, Subsequence 2, Center Column</td>
<td>63</td>
</tr>
<tr>
<td>5-5</td>
<td>$R(A, B)$ similarity measure for NIST 1</td>
<td>66</td>
</tr>
<tr>
<td>5-6</td>
<td>$H(A, B)$ similarity measure for NIST 1</td>
<td>67</td>
</tr>
<tr>
<td>5-7</td>
<td>NIST 2, Subsequence 1</td>
<td>70</td>
</tr>
<tr>
<td>5-8</td>
<td>NIST 2, Subsequence 1, Center Column</td>
<td>71</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Detecting motion is so intuitive that most of us will go through our whole lives without ever questioning that ability. But the ease with which we can separate, or segment out, a moving object from everything that is not moving in our field of view belies the complexity of the task. Like so many problems in computer vision, motion segmentation is embarrassingly difficult.

Before we can approach the problem of motion segmentation, it is necessary to have as precise a definition of it as possible. In the broadest sense, motion segmentation involves the segmentation, or partitioning, of the pixels of an image into regions based on the motion, or relative velocity, of the objects in the camera’s field of view. For our purposes, we assume a camera setup that is stationary with respect to the ground. Under this camera setup, we strive to segment all objects that are moving, regardless of their velocities.

While it is necessary to have as precise a definition of motion segmentation as possible, the definition will be ambiguous in some ways. For example, suppose a tree’s leaves were rustling in the wind. Would the tree be segmented in its entirety, or only its leaves and moving branches? Likewise, suppose a woman were walking down the street. Would the leg that is currently on the ground taking the step not be included in the segmentation, as it has no velocity at that time?

The examples stated above arise from the motion of non-rigid bodies. While it may seem appealing to simply remove non-rigid motion from our definition of
segmentation, as it would certainly get rid of many ambiguous cases, it is not a good idea. Many of the most interesting segmentation problems are those involving non-rigid bodies, such as ourselves. To remove them from our working definition would be too restrictive and would yield motion segmentations that would be less consistent with our intuitive comprehension of an object in motion.

That disparity between our intuitive understanding of the world and the mathematical formalisms with which we choose to model it is of fundamental importance to computer vision. The algorithms we implement are our mathematical formulations; their output, our working definitions. We choose those algorithms and base their performance on how well they correlate to our fuzzy, non-mathematical, intuitive understanding of the world. While the disparity and its minimization certainly pose a problem, it is also a challenge, and an opportunity. The elimination of that disparity is our goal.

1.1 Applications

Motion segmentation is a low-level vision process, and as such shares many of the same qualities of other low-level vision processes. For instance, like optical flow and shape from shading techniques, motion segmentation requires only the rawest of input data — a lattice of pixels. And, like other low-level vision processes, the output of a motion segmentation algorithm — a binary map representing the moving objects within the camera’s field of view — can be used directly or, more commonly, as input to a higher-level vision process.

Many higher-level vision applications, such as object tracking, recognition, and classification require moving objects for their input. Unless these algorithms are to remain abstract and work only with simulated data, a motion segmentation algorithm must be employed. It is therefore essential that the binary maps provided by the motion segmentation algorithm be as accurate and precise as possible.

One area of research that requires moving objects for input is gait recognition. Researchers working on gait recognition attempt to recognize an individual based on the
Figure 1-1: Typical output from a motion segmentation algorithm. This is a scene of a highway, taken from an overpass (courtesy [7]).

way they walk. The input to these gait recognition algorithms are sequences of binary silhouettes. These silhouettes may be provided by a low-level motion segmentation algorithm that analyzes frames of an image sequence and determines which pixels correspond to objects in motion. The sequences of silhouettes are then analyzed to identify which individuals correspond to which sequence of silhouettes.

Another popular area of research that requires accurate motion segmentation is behavior analysis and tracking. In this circumstance, the idea is to track moving objects, whether they are cars, bicycles, or pedestrians, and learn their typical routes. Abnormal behavior can then be identified as a tracked object that does not conform to its predicted route. Before any of the analysis can be done, however, the moving objects must first be identified within the sequence of image frames. This is most likely achieved using a motion segmentation algorithm.

1.2 Organization

Chapter 1 briefly described motion segmentation, how we define it, and what are its possible uses. Chapter 2 will introduce background processes, the motivation behind them, and the dichotomy between background maintenance and background subtraction. Chapter 3 will cover the ideas, use, and theory behind Markov Processes,
especially as they relate to motion segmentation. Chapter 4 ties together background processes and Markov theory to introduce this thesis’ unique contribution to the field of motion segmentation. Chapter 5 details the performance characteristics of this new approach, by itself and in comparison to current motion segmentation techniques. Chapter 6 will give an executive summary, a discussion of possible improvements, and a subjective analysis of the limits and limitations of these techniques in defining motion segmentation.
Chapter 2

Background Processes

Background modeling, also referred to as background subtraction, is one of the most successful methods for obtaining silhouettes of moving objects. Although the terms are often used interchangeably, they are different processes. Background modeling involves the creation and maintenance of a specific model of the appearance of the background in the field of view of the camera. Background subtraction is the process by which an image frame is “subtracted” from that background model. Although background modeling could possibly be used to model moving backgrounds, it is primarily used in a camera setup that is stationary with respect to the ground, such as the one we assume.

Whereas background modeling is concerned with the maintenance of the background model used to represent the backdrop of a particular scene, background subtraction is the process by which an image frame is compared to that background model. It is necessary and appropriate to make the distinction between the subtraction and modeling processes because they are separable and interact through a well-defined interface.

2.1 Background Modeling

As previously mentioned, background modeling is the process of creating and maintaining some model of the background of a particular scene. There are several different
ways of doing this, and we will discuss some of the most important ones. For a brief overview of many different background modeling techniques, we refer the reader to [13].

For a background model to be useful in a range of environments, it must be adaptable. A background model must be able to change over time, to account for different lighting conditions and changes in the environment. In outdoor settings, a background model must be able to cope with day-night cycles, sunshine, clouds, rain, and other natural phenomena. It must also deal with incorporating objects into the background, such as recently parked cars, idling construction equipment, etc. Indoors, a background model must be able to cope with light switches being thrown, sunshine through windows, and the dynamics of a typical office day.

2.1.1 Kalman Backgrounds

The Kalman filter, which was first suggested as a background model by [5], is a popular choice [6][7][11]. It is a simple, adaptable, per-pixel based method. Each filter predicts the next background value of a particular pixel in the image. The background is updated based on the equation

$$B_{t+1} = B_t + (\alpha_1(1 - S_t) + \alpha_2 S_t)D_t$$  \hspace{1cm} (2.1)

where $B_{t+1}$ is the next predicted background, $S_t$ is the current binary segmentation (foreground=1, background=0), and $D_t$ is the difference between the current image frame and the currently predicted background, $B_t$. The range of values for each element of $B_t$ are typically $[0, 255]$ for intensity (grayscale) images or $R, G, B$ triples for 24-bit images. The method by which $S_t$ is calculated is part of the subtraction process, not the modeling process, and will be covered in section 2.2. The learning rates, $\alpha_1$ and $\alpha_2$, are chosen (or learned) based on how quickly the background changes in the environment in which the camera is located. Typical values for $\alpha_1$ and $\alpha_2$ are 0.1 and 0.01 respectively [7].

The purpose of the binary segmentation in the maintenance of this background
model is simply to choose which learning rate, $\alpha_1$ or $\alpha_2$, by which the difference image, $D_t$, will be multiplied. Thus, the background will adapt more rapidly when a pixel is labeled 'background' than when it is labeled 'foreground'. A value of $\alpha_2 = 0$ implies that pixels labeled 'foreground' have no effect on the background model. While at first this may seem appropriate, it also implies that moving objects that come to rest will never be incorporated into the background model.

Using Kalman filters as the basic unit in the background model has its limitations. If foreground objects are allowed to be incorporated into the background, then by $\alpha_2 > 0$, transient foreground objects will have an adverse effect on background prediction. Furthermore, the model assumes a noise model that is consistent over all pixels. This model is valid for representing sensor noise, but local variations in pixel intensity and color that occur because of scattering or other lighting effects cannot be individually modeled.

### 2.1.2 Mixtures of Gaussians

Gaussian background models are among the most robust available. Many sources of pixel variances, including sensor noise, lighting variations, and specular highlights, can be amalgamated together and modeled surprisingly well by simple gaussian distributions. Wren, et al. used a single gaussian per pixel to model the background in Pfinder [14]. Stauffer and Grimson generalized this model by allowing for multiple gaussian distributions per pixel [12]. By allowing more than one gaussian per pixel, multimodal backgrounds (e.g. leaves rustling in the wind or flashing construction signs) can be modeled. It is also important to note that allowing more than one gaussian per pixel makes it possible to introduce objects into the background without altering the current background model. It is the mixture of gaussians background model that we use in our own algorithm. For a comprehensive explanation of this background model, we refer the reader to [12]. What follows is an overview.¹

¹Not all of the information provided in this section is consistent with [12]. We implemented a very similar, but not quite identical, mixture of gaussians background process. What follows is consistent with our implementation.
Overview

Consider the simplest gaussian background model, consisting of a single gaussian per pixel. Let us further assume that the incoming frames are grayscale (intensity) images, so that our gaussians are one-dimensional. Each gaussian with mean $\mu$ and standard deviation $\sigma$ has the probability density function

$$
\eta(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}
$$

(2.2)

where $\eta(x, \mu, \sigma)dx$ represents the probability that an intensity in the range $[x-\frac{dx}{2}, x+\frac{dx}{2}]$ is generated by that gaussian. Therefore, the further away from the mean a pixel intensity is, the less likely that it is part of the background.

There is no mechanism within $\eta(\cdot)$ that provides for the required adaptability of its parameters $\mu$ and $\sigma$. We must give it that ability. To do so, we update those parameters by the following equations:

$$
\mu_{t+1} = (1-\rho)\mu_t + \rho X_t 
$$

(2.3)

$$
\sigma^2_{t+1} = (1-\rho)\sigma^2_t + \rho (X_t - \mu_t)^2 
$$

(2.4)

where $\rho$ is the gaussian learning rate, and $X_t$ represents the intensity at time $t$ of the pixel corresponding to the gaussian whose parameters are being updated. A typical value for $\rho = 0.05$. Note that, in the case of a single gaussian per pixel, the distribution’s parameters are updated each image frame, since the background must adapt regardless of segmentation. If this case were actually used, it would probably be wise to use two rate parameters, as in equation 2.1.

Now consider the case where there is not one, but $K$ gaussians per pixel. For the mixture of gaussians to remain a probability density,

$$
\sum_{i=1}^{K} \left( \int \eta(x, \mu_i, \sigma_i)dx \right) = 1
$$

but as it is, they sum to $K$. A term must be multiplied to each gaussian, so that
their weights add to 1:

$$\sum_{i=1}^{K} w_i \cdot \left( \int \eta(x, \mu_i, \sigma_i) dx \right) = 1. \quad (2.5)$$

The weights are not chosen uniformly, so that each is $\frac{1}{K}$. Rather, they are chosen based on how the previous pixel evidence (past frames) correlate with the associated gaussians. The more evidence, the greater the weight. The update equation for the gaussian weights is

$$w_{k,t+1} = (1 - \alpha)w_{k,t} + \alpha(M_{k,t}) \quad (2.6)$$

where $\alpha$ is the weight learning rate and $M_{k,t}$ is a binary match mask (match=1) indicating which of the $K$ gaussians, if any, the current pixel data lends evidence to. A typical value for $\alpha = 0.05$. The weights are normalized following the update, so that their sum remains 1.

The update procedure for the gaussian parameters $\mu$ and $\sigma$ follow the update equations 2.3 and 2.4. However, in the mixture of gaussians case, only the parameters of the matched gaussian ($M_{k,t}=1$) are updated.

Before we can discuss how $M_{k,t}$ is calculated, we must introduce the background threshold $T$. The threshold $T$ dictates which of the $K$ gaussians are to be considered the background model $B$ for that pixel. Assuming that the gaussians are sorted so that the heaviest weighted gaussian is given the first index,

$$B = \arg\min_b \left( \sum_{i=1}^{b} w_i > T \right) \quad (2.7)$$

states that the least number of the heaviest weighted gaussians that together sum to at least $T$ are considered background. Conversely, the remaining $K - B$ gaussians are considered foreground.

A distinction between foreground and background gaussians is necessary because they are updated differently. Whereas a background gaussian is simply updated according to equation 2.6, a foreground gaussian can be either updated, or replaced. If replaced, it is replaced by a gaussian with parameters $\mu = X_t$, an initially high $\sigma$, 17
and low initial weight.

The choice of which gaussian gets updated or replaced is based on the binary segmentation of the current frame, \( S_t \), and the match mask of each gaussian, \( M_{k,t} \). Determining \( S_t \) is the function of the background subtraction process. Here, we assume it is given. For each pixel \( p \), \( M_{k,t} \) is calculated by:

\[
M_{k,t} = \begin{cases} 
1 & \text{if } S_t(p) = 0, \text{ and if } k = \arg\min_i \left( \frac{X_t - \mu_i}{\sigma_i} \right), 1 \leq i \leq B \\
1 & \text{if } S_t(p) = 1, \text{ and if } k = \arg\min_i \left( \frac{X_t - \mu_i}{\sigma_i} \right), B < i \leq K, \text{ and } \left( \frac{X_t - \mu_k}{\sigma_k} \right) \leq \Gamma \\
0 & \text{otherwise}
\end{cases}
\]

(2.8)

For any pixel, at most one gaussian is matched. If a segmentation of a pixel is background \((S_t(p) = 0)\), then the background gaussian \((1 \leq i \leq B)\) that best matches the pixel’s value \(X_t\) is chosen to be updated. If the segmentation is foreground, the foreground gaussian that most closely matches \(X_t\) is chosen to be updated only if \(X_t\) is within \(\Gamma\) standard deviations of the mean. Otherwise, the least-weighted gaussian is chosen for replacement. A typical value for \(\Gamma = 2.5\).

Finally, consider the most general case involving \(K\) multidimensional gaussians per pixel. Multidimensional gaussians are needed when working with color spaces rather than simple one dimensional intensity spaces. The \(n\)-dimensional gaussian probability density is

\[
\eta_n(x, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)}
\]

(2.9)

where \(x\) is the \(n\)-dimensional sample, \(\mu\) is the \(n\)-dimensional mean, \(\Sigma\) is the \((n \times n)\) covariance matrix, and \(|\Sigma|\) is its determinant.

We use the 3-dimensional RGB color space for pixels in our images. To avoid having to compute the covariance matrix and its inverse, we assume that the red, green, and blue pixel values are independent and that their variances are equal. This allows us to apply equation 2.3 to each color’s mean individually, and to apply equation 2.4 to \(\sigma\) in the multidimensional case as well.
Update Procedure

Section 2.1.2 gave an overview of the theory and the practice that comprises a working mixture of gaussians background model. This section will present the update algorithm used to maintain the background model.

*Given the current segmentation $S_t$ and an incoming image frame,*

1. Calculate $B$ for each pixel (equation 2.7). This separates the background gaussians from the foreground gaussians.

2. Calculate $M_{k,t}$ for each pixel’s $K$ gaussians (equation 2.8). This selects at most one gaussian per pixel for updating.

3. Update all gaussian weights (equation 2.6).

4. Update gaussian parameters for those gaussians whose mask $M_{k,t} = 1$ (equations 2.3 and 2.4). For those pixels that have no matching gaussians (i.e. all $K$ gaussians of that pixel have $M_{k,t} = 0$), replace the least weighted gaussian.

5. For each pixel, normalize gaussian weights and resort according to weight, heaviest first.

### 2.2 Background Subtraction

In section 2.1, we discussed various ways that stationary backgrounds can be represented. In this section, we focus on the ways in which backgrounds are subtracted from image frames containing transient, moving objects to produce the segmentation at time $t$, $S_t$. All of the methods involve measuring some distance $d_t(x,y)$ between an incoming image’s pixels and the background model, and then thresholding the differences according to

$$S_t(x,y) = \begin{cases} 1 & \text{if } d_t(x,y) \geq T_t(\cdot) \\ 0 & \text{otherwise} \end{cases}$$

(2.10)
where $T_t(\cdot)$ is some threshold function that is typically constant.

The simplest distance metric is the Euclidean distance with a constant, implicit variance

$$d = |x - \mu|.$$  

(2.11)

This metric is often used in conjunction with Kalman filter backgrounds [5][6][11] since the Kalman filter tracks the mean pixel value, and not necessarily its deviation. Koller, et al. used an interesting variation of this metric. Instead of taking the distance between an incoming pixel and its predicted value, they instead used a set of filtered responses. Since their work was based on traffic detection, the objects they were segmenting were predominantly squarish. Thus, they knew beforehand that there should be significant energy in the horizontal and vertical directions. They therefore took the derivatives of the incoming images in the x- and y- directions, to boost the signal to noise ratio of the image frames. They used these as part of their filtered responses, and measured distances based on these.

A generalized version of the Euclidean distance takes variance into consideration

$$d = \frac{|x - \mu|}{\sigma}.$$  

(2.12)

Rather than finding some absolute distance away from a mean, this standardizes the distances across pixels with different variances. The threshold in this case is in standard deviations, not absolute distances. This metric is the one used by Stauffer [12] in his background algorithms, as well as our own.2

A more general version of the standardized Euclidean distance is the Mahalanobis distance

$$d^2 = (x - \mu)' \Sigma^{-1} (x - \mu).$$  

(2.13)

If we were not to have made the simplifying assumptions about the independence of the three colors and the equality of their variances, then this is the metric we would have used. It is more accurate, as it does not assume that the data is clustered

2 We do not explicitly threshold on this value, however.
2.3 Background Model/Subtraction Dichotomy

It is understandable why the terms background subtraction and background modeling are used interchangeably: neither would make any sense without the other. Yet to understand how they fit together and interact is to more fully understand the approach as a whole. For instance, it may seem that the simplest of all motion segmentation techniques, frame differencing, has nothing to do with multidimensional mixtures of gaussians. Yet, if we choose the parameters carefully, we see that frame differencing is a specific instance of a much more general family of background models.

Understanding the dichotomy and separating the processes also facilitates the analysis of a backgrounding algorithm’s weaknesses by pinpointing the cause of error. For instance, it is easy to see that in figure 2-1, the background is exactly as we would expect it to be. Therefore, it is likely that any error in segmentation is due to the subtraction process and not due to inaccuracies introduced by the background model.

That isn’t to say that all of the blame lies with background subtraction. The complexity of the subtraction process is in some ways inversely related to the complexity of the model. For instance, a background model that could somehow predict

---

3In frame differencing, the background model is simply the previous image frame. With \( K = 2 \), \( \rho = 1.0 \), and an initial variance of 1.0, we ensure that the background will always be the previous frame.
the segmentation would leave the subtraction process with nothing to do. Likewise, if a model is not complex enough to represent the full scope of what it means to be a background, then the subtraction process may not have enough information with which to produce a perfect segmentation. It is probably the case that neither the model nor the subtraction process are as sophisticated as they need to be in order to produce perfect segmentations. This thesis uses the mixture of gaussians background model and introduces a more sophisticated form of background subtraction using Markov random fields. With a more sophisticated subtraction process comes a more accurate motion segmentation. The theory governing the use of MRFs is introduced in the next chapter.

2.4 Limitations

The current background subtraction techniques work on a per-pixel basis. That is, each pixel is processed against its background model independently of all other pixels in the image. But, objects are not wisps of smoke whose substance can be anywhere and everywhere; they tend to be cohesive. The current background subtraction processes do not take advantage of this fact. If silhouettes appear whole, it is because adjacent pixels happened to be classified as foreground together, not because they have any tendency to be classified alike.

Similarly, objects do not tend to jump around randomly in a camera’s field of view. If an object is at location $x$ at time $t$, it is likely that it will be close to location $x$ at time $t + 1$. But background subtraction techniques are independent in time as well. The output of a subtraction process at time $t + 1$ has nothing to do with its output at time $t$. If a sequence of silhouettes appears to be moving, it is because the same pixels (except for the edges) in adjacent frames happened to be classified as foreground, not because they have a tendency to remain classified as they were. Both the space and time dependencies that moving objects impose on an image sequence’s

\footnote{except through the small amount of information about the previous frame’s segmentation assimilated into the background model through its update procedure.}
pixels are exploited in the background subtraction algorithm presented in chapter 4.
Chapter 3

Markov Random Fields

Markov random fields are a convenient and powerful tool that can be used to exploit the spatial and temporal dependence of motion segmentations. A generalization on one-dimensional Markov chains, MRFs have the ability to model probabilistic dependencies in an arbitrary number of dimensions. In this chapter, the theory behind MRFs, and a procedure to evaluate them, will be introduced. For more information on MRFs, we refer the reader to [3][2][15].

3.1 Markov Properties

At its most basic level, a Markov random field can be visualized as a simple undirected graph $G = (S, \mathcal{N})$, where $S$ is the set of site indices (nodes) and $\mathcal{N}$ is a neighborhood system on $S$ defined as

$$\mathcal{N} = \{N_s, s \in S \mid s \notin N_s$$ and $s \in N_r \leftrightarrow r \in N_s\} \quad (3.1)$$

where $N_s$ is the set containing the neighbors of $s$. A little thought should convince you that $\mathcal{N}$ is just a different way of expressing the edge set of an undirected graph. Expressing the edge set in this way is semantically more meaningful to MRFs.

To illustrate the point, we use figure 3-1 as an example MRF $X$. The set $S$ of indices, $\{A, B, C, D\}$, represent the four probabilistic random variables of $X$, 

25
Figure 3-1: An example of a simple undirected graph. Undirected graphs can be used to visualize Markov random fields. The nodes $S$ of the graph correspond to the random variables $X$ of an MRF and the edges illustrate the neighborhood system over $S$. In this example, the neighborhood system of $A$, $\mathcal{N}_A = \{B, D\}$.

$X = \{X_s, \ s \in S\}$. These random variables can take on any of the values in their state space $\Lambda$. To further concretize things, let us assume that each $X_s \in X$ is a motion segmentation variable. Therefore, their state space is $\Lambda = \{\text{background, foreground}\}$, or $\Lambda = \{0, 1\}$. A setting of those random variables — in this case, a segmentation — is a configuration $\omega \in \Omega$ of $X$, where $\Omega$ is the set of all configurations. For a field of random variables to be Markov, they must satisfy the positivity requirement:

$$P(X = \omega) > 0, \ \forall \ \omega \in \Omega$$  \hspace{1cm} (3.2)

The positivity requirement states that every configuration of the random variables, every segmentation, must be possible. While this seems like a trivial requirement, it plays an important part in making MRFs a practical probabilistic model.

Whereas the nodes of the graph represent an MRF’s random variables, the edges represent the dependencies between them. Specifically, the edges represent the conditional independence relations:

$$P(X_s = x_s \mid X_r = x_r, \ r \neq s) = P(X_s = x_s \mid X_r = x_r, \ r \in \mathcal{N}_s)$$  \hspace{1cm} (3.3)

The local Markov property of equation 3.3 states that the probability of a random variable $(X_s \in X, \ s \in S)$ being in any particular state $(x_s \in \Lambda)$ is independent of the configuration of all other variables in the system, given some setting of its neighbors.
(\mathcal{N}_a). Using the MRF in figure 3-1 for example, \( P(X_A = 0 \mid X_B = 0, X_C = 1, X_D = 1) = P(X_A = 0 \mid X_B = 0, X_D = 1) \), since \( \mathcal{N}_A = \{B, D\} \).

Any collection of random variables that satisfies both the positivity requirement and the local Markov property is a Markov random field. These two properties uniquely define what it means for a set of random variables to be an MRF.

### 3.2 Gibbs Distributions

There is no obvious factorization for the joint probability distribution \( P(X = \omega), \omega \in \Omega \) of a Markov random field. Unlike Bayes nets, whose joint probability factors as:

\[
P(X = \omega) = \prod_{s \in S} P(X_s = s \mid \text{Parents}(X_s))
\]

MRFs have no such factorization. If you were to convert an undirected graphical probabilistic model (MRF) to a directed one (Bayes net), you would replace every undirected edge with two directed edges, as in figure 3-2. However, every undirected edge creates a cycle in the directed graph. Directed cycles break the elegant factorization of Bayes nets since the concept of a node’s parent becomes circular.

![Figure 3-2: A Bayes net-like joint factorization is not possible for an MRF. Each undirected edge of the MRF would need to be converted into two directed edges. But each pair of nodes formerly connected with an undirected edge would then make a (prohibited) directed cycle.](image)

Although no factorization is known for MRFs in general, an equivalence relation between MRFs and Gibbs distributions, first proven by Hammersley and Clifford in
1971 and later published by Besag [1], allows for an efficient computational approximation. The theorem states that \( X \) is an MRF with respect to its neighborhood system \( \mathcal{N} \) if and only if \( \pi(\omega) = P(X = \omega) \) is a Gibbs distribution with respect to \( \mathcal{N} \). Since we know that \( X \) is an MRF with respect to \( \mathcal{N} \),\(^1\) this theorem simply states that \( X \) must also be a Gibbs distribution.

A Gibbs distribution is the probability function \( \pi(\omega), \omega \in \Omega \) with the mathematical form:

\[
\pi(\omega) = \frac{e^{-U(\omega)/T}}{Z}
\]

where \( Z \) is a normalizing constant, \( T \) is a temperature constant, and \( U \) is the energy function. Although the parameters of the Gibbs distribution betray its origins in statistical physics, they are necessary nevertheless. \( Z \) is the partition function:

\[
Z = \sum_{\omega} e^{-U(\omega)/T}
\]

chosen so that \( \pi \) is a proper probability density. The temperature term \( T \) dampens the impact of the energy term on the exponential. Thus, when \( T \) is relatively large, the density \( \pi \) will be more uniform. Conversely, when \( T \) is relatively small, the local minima and maxima of \( \pi \) become more pronounced.\(^2\) The temperature is essential to — and its name somewhat validated by — the simulated annealing process used to estimate the optimal configurations of MRFs. Finally, the energy function \( U \) has the form:

\[
U(\omega) = \sum_{c \in \mathcal{C}} V_c(\omega)
\]

where \( c \) is a clique\(^3\) in \( \mathcal{N} \), \( \mathcal{C} \) is the set of all cliques, and \( V_c \) is a potential function. For \( X \) in figure 3-1, \( \mathcal{C} = \{\{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{B, C\}, \{C, D\}, \{A, D\}\} \). A potential function can be chosen independently for each clique. However, potentials are commonly chosen over repeating patterns in \( \mathcal{N} \). Referring again to \( X \), a single

---

\(^1\)Given that it also satisfies the positivity requirement.

\(^2\)In our experimental setup, a large \( T = 7.0 \), and a small \( T = 0.5 \). The values of \( T \) that make sense to call large and small depend on how \( U \) is defined, however.

\(^3\)A clique is a subset of \( S \) in which all sites are neighbors of each other.
potential function might be chosen for the 1-cliques and another for the 2-cliques. An important property of the potential functions is that each \( V_c \) depends only on those variables \( X_s \in X \) for which \( s \in c \). All variables that are not part of the clique do not play a role in defining that clique’s energy potential.

It is helpful\(^4\) to think about the energy function \( U \) and the potentials \( V_c \) in a physical metaphor. Since there is a negative (-) in the exponential of equation 3.4, the probability of a configuration increases with decreasing energy. Thus, the “stability” of the MRF increases with decreasing energy. Just like with a system of atoms or molecules, an MRF wants to be in the most stable, most probable configuration.

There is some amount of variety in the literature when it comes to defining some of these terms. For example, some choose to define a clique not as we have, but as the largest subset for which all sites are neighbors. This is what we would call a maximal clique. The maximal cliques of \( X \), for instance, are all the 2-cliques. However, the two are functionally equivalent, since a maximal clique over site \( s \) also contains all of the non-maximal cliques over site \( s \) as well.

Another, more common variation is in the way \( \pi \) is defined. Instead of it being defined as a sum of potentials in the exponential, it is sometimes considered a product of potentials:

\[
\pi(\omega) = \prod_{c \in \mathcal{C}} \Psi_c(\omega)
\]

This formulation is simpler to see and perhaps use, since all of the constants in equation 3.4 are implied in the \( \Psi_c \)'s. But in the former, the energy potentials can be isolated, as in equation 3.6. Since the \( V_c \)'s are the only unspecified part, isolating them facilitates their selection. Ultimately both formulations are equivalent, so choose the one that is most convenient for you.

\(^4\)To this author, at least.
3.3 Computational Factorization

Section 3.2 introduced the Gibbs distribution and the equivalence between it and Markov random fields. In this section, we will show how the Gibbs distribution leads to a factorization of the joint probability distribution \( \pi(\omega) = P(X = \omega) \) which can be used to efficiently estimate an optimal configuration. We start with the basic equation for the Gibbs distribution,

\[
\pi(\omega) = \frac{e^{-U(\omega)/T}}{Z}
\]

Consider the site \( s \in S \) and the configuration \( \omega = \{x_1, x_2, \ldots, x_n\}, 1 \leq s \leq n \). Let \( \omega^\alpha \) be the configuration that agrees with \( \omega \) everywhere except possibly at \( s \), where \( X_s = \alpha, \alpha \in \Lambda \). Then the local characteristic of \( X_s \) is:

\[
P(X_s = x_s \mid X_r = x_r, s \neq r) = \frac{\pi(\omega)}{\sum_{\alpha \in \Lambda} \pi(\omega^\alpha)} \tag{3.7}
\]

If we expand the functions in equation 3.7 and make the appropriate cancellations, the equation simplifies to:

\[
P(X_s = x_s \mid X_r = x_r, s \neq r) = \frac{\exp \left[ -\frac{1}{T} \sum_{c:s \in c} V_c(\omega) \right]}{Z_s} \tag{3.8}
\]

\[
Z_s = \sum_{\alpha \in \Lambda} \exp \left[ -\frac{1}{T} \sum_{c:s \in c} V_c(\omega^\alpha) \right] \tag{3.9}
\]

The two equations above are the cornerstone for much of Markov field theory. Notice that these equations require only computations local to \( s \). That is because all of the cliques which do not involve \( s \) cancel out and the ones that are left require only \( s \) and its neighborhood \( N_s \). This is the computational factorization of Markov random fields and will be used extensively to approximate the most probable configuration of the variables.

\[5\text{See appendix A for the complete derivation.}\]
3.4 Gibbs Sampling

Gibbs Sampling is a method used to find the most probable configuration $\omega \in \Omega$ of a Markov random field. It is an approximation technique based on the theory of simulated annealing. The algorithm is remarkably simple, highly parallel, and, theoretically at least, remarkably slow.

The algorithm works by assigning to the MRF an initial configuration. It then perturbs that configuration in discrete time steps $t_1, t_2, t_3, \ldots, t_K$ by sampling from the local characteristics of equations 3.8 and 3.9 for $K$ iterations. With each iteration the temperature term $T$ decreases, according to its annealing schedule. In the simulated annealing metaphor, this slow cooling process “crystalizes” the MRF. If it is done slowly enough, it crystallizes at the point of lowest energy. More formally, the algorithm is presented below:

Given $\omega_0, C, K$

1. $X = \omega_0$, $k \leftarrow 1$, $T \leftarrow \frac{C}{\ln 2}$

2. while $k < K$

   2.1. for each $X_s \in X$, $s \in S$, sample from the local characteristics $P(X_s \mid N_X_s)$ and set $X_s$ to the state sampled.

   2.2. $T \leftarrow \frac{C}{\ln(1+k)}$, $k \leftarrow k + 1$

The reason that this algorithm is intractably slow in theory is because it has only been proven to converge to the MAP estimate when $C = |S|$ [3]. Since the denominator of the annealing schedule is a logarithm in $k$, it takes far too many iterations to bring the temperature down to the point of MRF crystallization, or convergence (i.e. near 0).

In practice, however, a much smaller $C$ can be used. We have used a value $3.0 \leq C \leq 5.0$. Note that the algorithm above iterates through a constant number

\footnote{Recall that the point of lowest energy is the point of highest joint probability.}
of iterations $K$. $K$ is chosen according to the annealing schedule, and the value of $C$ used. The number of iterations should be chosen so that at the $K$th iteration, $T$ is close to 0. A typical low value for $T$ that works well is $0.5$. With this annealing schedule, only a few hundred iterations are needed for the MRF to converge. In fact, in chapter 4, we will show that, with a best guess initial configuration $\omega_0$ and a modified annealing schedule, we do well with $K \approx 10$. 
Chapter 4

Markov Thresholding

The preceding chapters introduced the theory of background modeling and Markov random fields on which our work is based. This chapter describes how the MRFs we formulate are used, the precise structure of those fields, and the bonds — the potential functions $V_c$ — that encode the relationships between neighboring variables within the fields.

4.1 Base Field Structure

All of the field structures used have a common, fundamental structure. That field structure consists of an $m \times n$ lattice of binary segmentation random variables, where $m$ and $n$ are the numbers of rows and columns, respectively, in the corresponding image frames. Thus, there is one segmentation variable for each pixel of an image. The local structure of the field is a second-order, 8-connected neighborhood system,

![Figure 4-1: In a two dimensional lattice, a second-order neighborhood system consists of the first-order horizontal and vertical connections as well as the diagonals.](image)

Figure 4-2: A $6 \times 6$ segment of a larger $m \times n$ lattice. The neighborhood structure of the lattice is a second-order, 8-connected system. This lattice represents the basic MRF configuration used in our experiments.

As shown in figure 4-1. A second-order neighborhood system over a two-dimensional lattice network consists of links from one node to its immediate vertical, horizontal, and diagonal neighbors. The overall lattice structure is shown in figure 4-2.

Since pixels do not have much spatial extent, a lattice structure with the neighborhood system described above may seem to offer only limited spatial correspondence. One may feel it necessary to extend the neighborhood system outward, to encompass a larger area of space. This argument is understandable, as each node’s setting is only dependent on its neighbors’ settings. But, the setting of those neighbors are dependent on their neighbors, and so forth. Thus, each node’s setting is indirectly influenced by the setting of the entire MRF. It is surprising that such a minimal neighborhood structure can embody such global awareness, but that is the beauty of Markov random field theory.

The set of cliques $C$ of a lattice with a second-order neighborhood system consist of 1-, 2-, 3-, and 4- cliques, as in figure 4-3. To distinguish the energy potentials $V_c$ associated with the cliques of $X$ from those that will be introduced in the following sections, we denote the energy potential of such cliques as $\Psi_c$. For example, the energy potential associated with the 2-clique of sites $s$ and $r$ is $V_{\{s,r\}}(\omega) = \Psi_{\{s,r\}}(X_s, X_r)$.

Choosing the energy potentials associated with these cliques is straightforward.
We set the potentials associated with the 1-, 3-, and 4-cliques to be 0. Thus we are only interested in the 2-cliques. We do not hold any preconceptions or assumptions about the importance of any position or orientation within the lattice, so all 2-cliques are treated equally. The energy associated with each 2-clique is as follows:

$$\Psi_{\{s,r\}}(X_s, X_r) = \begin{cases} 
-3, & \text{if } X_s = X_r = 1 \\
-1, & \text{if } X_s = X_r = 0 \\
+3, & \text{if } X_s \neq X_r 
\end{cases} \quad (4.1)$$

The choice of energy potentials is not arbitrary. Since $\Psi_{\{s,r\}} = \Psi_{\{r,s\}}$, the choice of energy potentials must be symmetric with respect to site $s$ and $r$. Therefore, the energy associated with the combination $X_s \neq X_r$ must be the same as that associated with the combination $X_r \neq X_s$. This condition is important since, during Gibbs Sampling, each variable plays the role of $X_s$ and $X_r$. Note, also, that a different energy can be (and is) associated with $X_s$ and $X_r$ being either both foreground (1) or both background (0). More preference is given to the condition where they are both foreground. To personify, two neighboring segmentation variables want to both be foreground more than they want to both be background (but they want to be the same more than they want to be different). To say it another way, there is more spatial dependence for foreground segmentation than for background. We chose these potentials because we would prefer to misclassify pixels as foreground (false positive) than to misclassify them as background (false negative). This helps keep our silhouettes whole and cohesive.
4.2 Posterior Distribution

Up until now, our discussion of MRFs has exclusively involved the prior probability distribution on $X$, $P(X = \omega)$. But the case in which we are most interested is the posterior distribution

$$P(X = \omega \mid D = d)$$

where $D$ is some quantity of data, or evidence. In our case, that data is an incoming frame’s pixels. $D$ is expressed as a field of random variables that can represent every possible $m \times n$ image. Whereas $X$ consists of binary variates, $D$ consists of pixel-value variates. For RGB images, each $D_s \in D$ has three 8-bit color channels, yielding $2^{24}$ possible color settings. We are interested in the posterior distribution because we are interested in finding the most likely segmentation of a given image frame. Without that image, a segmentation would have little meaning.

Although the posterior distribution is the one we need, it is often not in a form suitable for optimization. However, by applying Bayes’ rule to equation 4.2, we obtain

$$P(X = \omega \mid D = d) = \frac{P(D = d \mid X = \omega) \cdot P(X = \omega)}{P(D = d)}$$

(4.3)

Since the denominator, $P(D = d)$, is a constant over the variables in $X$, for optimization purposes, equation 4.3 can be reduced to

$$P(X = \omega \mid D = d) \propto P(D = d \mid X = \omega) \cdot P(X = \omega)$$

(4.4)

For the posterior distribution to represent a valid Markov random field, its probability must be expressible in terms of exponentials. We already know that the prior distribution can be expressed as a Gibbs distribution $P(X = \omega) = \pi(\omega) = e^{-U(\omega)/T}/Z$ simply by a setting of the energy potentials described in the previous section. The likelihood function, $P(D = d \mid X = \omega)$, must also be expressible in terms of exponentials.

The likelihood function encapsulates the capability of a particular model to generate, or explain, the observed data. In our case, it encapsulates the capability of our segmentations to generate, justify, or explain, the image frames that we observe. To
Figure 4-4: The graph representation of the basic segmentation model unit corresponding to the posterior distribution. In the posterior distribution, each $X_s$ has an additional link to its corresponding pixel random variable $D_s$. Note the lack of links between variates in $D$. Each pixel is independent of the value of its neighbors, given a segmentation.

better illustrate how our segmentation model can generate the observed image data, we introduce the three dimensional, complete Markov random field, which directly corresponds to the posterior distribution of our segmentation-only MRF $X$. Figure 4-4 shows the basic model unit of the posterior distribution. Each binary variate $X_s$ has an additional link to its corresponding image pixel, $D_s$. This link encodes the relationship between the image field and the segmentation field.

Notice that each pixel of the image is independent of all other pixels, given a particular segmentation. Not only does this greatly facilitate the computation of the posterior distribution, but it makes great sense as well. It is not the actual image and its corresponding pixels that we are concerned with. Rather, it is the underlying model, or representation, of reality that is of actual significance. Therefore, given that we have a particular model of the world (a segmentation), the image can be generated, pixel by pixel, from that underlying model. Figure 4-5 provides an example of how the image and its underlying segmentation fit together. Obviously, a model for motion segmentation cannot capture all of the essence of reality. The image we see cannot be created based solely on motion segmentation. But, all other things being equal, the correct motion segmentation for a given image frame will explain the image, will
Figure 4-5: A particular segmentation is inferred by the pixel data $D$ that is given. The most accurate segmentation is the one that can best explain, or predict, the image that we see. Given that segmentation, the pixels of the image can then be generated individually, one by one, from the underlying model.

generate that image, more accurately than all other possible segmentations. This is known as the maximum likelihood estimate, and it assumes that the prior over $X$ is uniform (i.e. that all segmentations are equally likely). The maximum a posteriori (MAP) estimate, which we are after, tempers the likelihood function by assuming that some segmentations are intrinsically more valid, more probable, than others.

Our assumptions about the posterior probability distribution ensure that

$$P(D = d \mid X = \omega) = \prod_{s} P(D_s = d_s \mid X_s = x_s)$$

(4.5)

which is in a form perfectly suited for local computation. Whereas the cliques involving only variates in $X$ are denoted by $\Psi_c$, the 2-cliques involving $X_s \in X$ and $D_s \in D$ are denoted by the functions $\Phi_c$, or more specifically $\Phi_{(s)}(X_s, D_s)$.

We must now show that equation 4.5 can be expressed as a product of exponentials.
We begin by determining \( P(D_s = d_s \mid X_s = 0) \). This is the probability of seeing a pixel \( D_s \) with color \( d_s \), given that the segmentation of that pixel has been determined to be background. Recall that the probability densities of our gaussian background models are:

\[
\eta_n(x, \mu, \sigma) = \frac{1}{\sigma^3 \sqrt{(2\pi)^3}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.6}
\]

This is a specific instance of equation 2.9, where there are three dimensions and the variances in each dimension are assumed to be independent and equal. We are only interested in the portion of this equation within the exponential. Coefficients such as \( \frac{1}{\sigma^3 \sqrt{(2\pi)^3}} \) are incorporated into the partition function \( Z \). Thus, \( \Phi_{(s)}(X_s, D_s) = \frac{||D_s - \mu||^2}{2\sigma^2} \) when \( X_s = 0 \). Since there are \( B \) gaussian background models for a given pixel, the one that is used in the calculation above is the one that best matches \( D_s \) (i.e. the one that will minimize \( ||D_s - \mu||^2 \)).

In the case where \( X_s = 0 \), we are fortunate to have a model of the background and have that model come in an exponential form. For the foreground case \( X_s = 1 \), we are not so fortunate. Since we have no model for foreground objects, and since foreground objects can be any color whatsoever, it is best to assume a uniform distribution over the colorspace of \( D \). Since there are \( 2^{24} \) possible colors, the probability of seeing any one of them in a foreground object is \( \frac{1}{2^{24}} \). Thus, \( P(D_s = d_s \mid X_s = 1) = 2^{-24} \), which is not an exponential form. But notice that any number \( x \) can be represented as a negative exponential: \( x = e^{-\ln x - 1} \). Since \( \ln(2^{-24})^{-1} = \ln 2^{24} = 16.6355 \), \( \Phi_{(s)}(X_s, D_s) = 16.6355 \) when \( X_s = 1 \). Therefore, the energy potential \( \Phi_{(s)} \) is:

\[
\Phi_{(s)}(X_s, D_s) = \begin{cases} 
\frac{||D_s - \mu||^2}{2\sigma^2}, & \text{when } X_s = 0 \\
16.6355, & \text{when } X_s = 1
\end{cases} \tag{4.7}
\]

For mathematical completeness, it bears mentioning that any constant can be added to or subtracted from \( \Phi_{(s)} \). Just as any number \( x \) can be represented as \( x = e^{-\ln x - 1} \), so can any number \( ax \) be represented as \( ax = ae^{-\ln x - 1} \). Instead of \( a \) being incorporated into \( Z \), it can just as easily be incorporated into the exponential:
\[ ae^{-ln x^{-1}} = e^{-ln a^{-1}} e^{-ln x^{-1}} = e^{-(ln x^{-1} + ln a^{-1})}. \] Therefore, our choice of \( \Phi_s \) when \( X_s = 1 \) remains somewhat arbitrary, as we have implicitly chosen the coefficient \( a = 1 \).

Another way to think about the posterior distribution of \( X \) is to simply consider the prior distribution over the mixed field containing \( X \) and \( D \). Notice that equation 4.4 can be rewritten as:

\[ P(X = \omega \mid D = d) \propto P(D = d, X = \omega) \] (4.8)

So, instead of thinking in terms of likelihoods, priors, and their products, we need only be concerned with the joint distribution, as before. To optimize this joint probability requires only the optimization over \( X \), as the configuration of the variates in \( D \) is still given. Of course, you cannot simply choose any field structure containing \( X \) and \( D \). The field structure must match the structure implied by the posterior distribution. But once you have that, it may be easier to think of that field on its own.¹

To summarize, the base field structure of the posterior distribution of \( X \) has the same 8-connected neighborhood system over the variates in \( X \), but each \( X_s \in X \) has an additional link its corresponding pixel-valued variate \( D_s \in D \). The energy potentials \( V_c \) associated with the cliques \( c \in C \) that involve only variates in \( X \) have the special designation \( \Psi_c \). Of those cliques, only those involving two variates have nonzero energy potentials. The energy potentials associated with the two-cliques is governed by equation 4.1. The energy potentials \( V_c \) associated with the two-cliques \( c \in C \) involving \( X_s \) and its corresponding image pixel \( D_s \) have the special designation \( \Phi_c \). Since each \( D_s \) is only connected to its corresponding segmentation variate \( X_s \), \( D_s \) is only associated with two-cliques.² The energy potentials of those cliques are governed by equation 4.7. The potential families \( \Psi \) and \( \Phi \) are all that are required to compute equations 3.8 and 3.9 needed to find the MAP estimate of the posterior distribution of the base field structure through Gibbs Sampling.

¹Many in the literature state the posterior over \( X \) in this manner.
²Of course, each \( D_s \) is associated with the one-clique involving only itself. But, since we are optimizing over \( X \) and not \( D \), those potentials are irrelevant.
4.3 Field Structures

In the previous sections, we introduced the field structure common to the three different field structures used. That field structure was the $m \times n$ lattice of binary segmentation random variables with an 8-connected, second-order neighborhood structure over the variates. We also derived the posterior distribution of that field and proved that it, too, was a valid Markov random field with an augmented neighborhood structure involving connections from each $X_s$ to its corresponding pixel variate $D_s$. In the following sections, we will describe the three field structures used in our experiments.

4.3.1 Markov Model $M_1$

The first Markov random field $M_1$ used was the base field structure presented in section 4.1. Recall that the field structure is an $m \times n$ lattice of binary segmentation random variables with a second-order neighborhood structure. The energy potentials associated with the posterior distribution of $M_1$ are those $\Psi_e$'s and $\Phi_e$'s presented in sections 4.1 and 4.2.

This field structure assumes an online computation model. In an online model, the incoming image frames are handled as they arrive, one by one, and a segmentation is provided after each frame. As such, time dependencies are not encoded within $M_1$. Spatial dependencies, however, are encoded within this field structure.

4.3.2 Markov Model $M_2$

The field structure $M_2$ is similar to that of $M_1$ but is rich enough to encode time dependencies as well. $M_2$ begins with the base field structure, but each $X_s \in X$ also has a link to itself in the previous frame. The fundamental model unit for each $X_s$ is shown in figure 4-6. Like $M_1$, $M_2$ also assumes an online computation model.

Since we have introduced a new dimension to our field structure (the time dimension) a revision to our notation is necessary. Whereas before we have always assumed that $X_s$ referred to a particular variable in the MRF $X$ at the current time, we now must allow for variation in time. Therefore, let $s_t$ refer to a position at time $t$, and
let $X_{s,t}$ refer to the variable at position $s$ at time $t$ within $X_t$, the $m \times n$ lattice of binary segmentation random variables at time $t$. Also, recall that the segmentation at time $t$ is $S_t$.\(^3\)

The prior probability distribution for $X_t$ is most likely going to differ for $M_2$ than for $M_1$. Since we assume an online computation model, the previous segmentation $S_{t-1}$ of the variables in $X_{t-1}$ to which each $X_{s,t}$ is linked is already given. Therefore, it is still only necessary to optimize the posterior probability over the variables in $X_t$. The potentials $\Theta_c$ associated with the temporal 2-cliques is then:

$$
\Theta_{\{s,t,s_{t-1}\}}(X_{s,t}, X_{s,t-1}) = \begin{cases} 
-5, & \text{if } X_{s,t} = X_{s,t-1} = 1 \\
0, & \text{if } X_{s,t} = X_{s,t-1} = 0 \\
0, & \text{if } X_{s,t} \neq X_{s,t-1}
\end{cases}
$$

(4.9)

We chose the values of $\Theta_c$ based on how we wanted a pixel’s previous and current segmentations to interact. Obviously, the current segmentation has no effect on the previous one, but the previous segmentation can help decide what the current segmentation should be. Essentially, we wanted the behavior of each segmentation variable to be “sticky on”, meaning that if it was foreground in the past, it would want to remain foreground. Aside from that, we did not want to impose any other relationship on a pixel’s segmentation. That is why it is just as easy for a current

\(^3\) $S_t$ is just a convenient alias for the optimal configuration, or MAP estimate, $\omega_t$ of the variables $X_{s,t} \in X_t$.\(^3\)

Figure 4-6: The basic MRF model unit for $M_2$. Each model unit of $M_2$ is connected to its 8 spatial neighbors and itself from the previous frame. Since the previous frame will always have a segmentation, the $m \times n$ lattice over which the optimization process takes place remains unchanged from that of $M_1$. 
pixel to be labeled foreground when it was background as it is to be labeled background when it was background. The value of \(-5\) was chosen because we wanted the temporal link to be more significant than any individual spatial link, since the temporal link is evaluated only once and the spatial links together eight times.

Also note that the current segmentation \(S_t\) is only dependent on its previous segmentation \(S_{t-1}\). This follows directly from the local Markov property on which our MRFs are based. However, unlike the spatial relationship between variates in \(X_t\), where each \(X_{s,t}\) is indirectly affected by the setting of the entire lattice, the setting of each \(X_{s,t}\) has no time dependencies, indirect or otherwise, beyond its dependency on the setting of \(X_{s,t-1}\). The reason for this is because \(S_{t-1}\) is given to us, and the variates in \(X_{t-1}\) are not further alterable or subject to optimization by us at time \(t\), or any other time.

Since the role of \(S_{t-1}\) is to alter the prior distribution of \(X_t\), we could just as easily have incorporated that knowledge into the one-cliques over each \(X_{s,t}\). Whereas in \(M_1\), \(\Psi_c = 0\) whenever \(c\) was not a two-clique, it could be the case for \(M_2\) that \(\Psi_c = \Theta_{c'}\) whenever \(c\) is a one-clique in \(X_t\) and \(c'\) is its corresponding temporal two-clique. We posed the formulation of \(M_2\) in such a manner because, while not completely necessary for \(M_2\), it lays the notational foundation for the field structure \(M_3\) introduced in the next section.

Although this field structure has a different prior over \(X_t\), the likelihood function (see section 4.2 for details) of the posterior distribution remains the same. This is because \(X_t\) is given for the likelihood function, and each \(D_{s,t}\) is independent of everything else given its corresponding \(X_{s,t}\). Therefore, the energy potentials \(\Phi_c\) which encode the relationship between each \(X_{s,t}\) and \(D_{s,t}\) remain unchanged by the new neighborhood structure over \(X_t\).

**4.3.3 Markov Model \(M_3\)**

The last field structure \(M_3\) that we used is similar to \(M_2\) in that it incorporates temporal links into the base field structure. But, unlike \(M_2\), each \(X_{s,t}\) is connected to itself in the past \((X_{s,t-1})\) and in the future \((X_{s,t+1})\). The basic model unit for
Figure 4-7: The basic MRF model unit for $M_3$. Each model unit of $M_3$ is connected to its 8 spatial neighbors and itself in the previous and subsequent frames. $M_3$ assumes a batch model of computation. As such, the optimization process takes place over all frames at once.

$M_3$ is shown in figure 4-7. It also differs from $M_2$ in that the posterior optimization does not take place over just the variates in $X_t$ but rather over the entire field $X = \bigcup_t X_t$, $1 \leq t \leq T$, where $T$ is the index of the last image frame to be segmented. The configurations of the fields $X_{t-1}$ and $X_{t+1}$ are not given to us, as $S_{t-1}$ was for $M_2$.

Obviously, $M_3$'s field structure precludes the use of an online model of computation. Therefore, a batch model of computation must be employed. During a batch process, all frames are segmented together, rather than one after the other. Once the process has completed, segmentations for every frame in the image sequence will have been computed. The tradeoffs between batch processing and online processing is that, while in batch processing you have more information at hand with which to base decisions, you cannot perform batch processing in real time. Even with a semi-online model, in which you perform a batch process over a sliding window of the last $i$ frames, your output will always be time-delayed by some function of $i$.

In theory, a batch process will always perform at least as well as any online model, since all of the information available in an online model is also available in a batch model. Therefore, the batch approach to a particular problem could simply reduce to the best online approach. In reality, however, batch processing has its own paradigm that does not usually involve the rigid sequential considerations inherent in online approaches. That being the case, it is not impossible for a batch process to produce less-satisfactory results than an online process. Although it may seem counter-productive to test methods that produce worse results, it is sometimes
necessary when investigating optimal solutions to informal problems.\textsuperscript{4}

In some ways, online processes imply a greedy solution to a problem. In an online process, you are forced to make the best decision you can about the current element in the data set given only what you have seen previously. Once a decision has been made, it is set and cannot be changed based on something seen in the future. In a batch process, where all of the data is available, you can always make the optimal decision. If it turns out that a greedy strategy is optimal, then the optimal batch process will reduce to an online process. Then an online process can, and will, produce the best results. Cases like this are truly optimal, since you can achieve the real-time benefits of an online approach and the optimal results guaranteed by batch processing.

Returning to the problem of motion segmentation, the approach we have chosen for optimizing a solution over $M_3$ is to optimize the posterior probability of the entire field $X$. Just as optimizing the posterior distribution over the lattice $X_t$ entails complete (albeit indirect) spatial dependence over all variates, optimizing over $X$ entails complete temporal dependence between the variates as well. The potential functions $\Theta_e$ that encode the relationship between each $X_{s,t}$ and its temporal neighbors $X_{s,t-1}$ and $X_{s,t+1}$ are:

$$\Theta_{\{s,s'\}}(X_{s,t}, X_{s,t'}) = \begin{cases} 
-5, & \text{if } X_{s,t} = X_{s,t'} = 1 \\
0, & \text{if } X_{s,t} = X_{s,t'} = 0 \\
0, & \text{if } X_{s,t} \neq X_{s,t'}
\end{cases} \quad (4.10)$$

where $t' = t \pm 1$. The likelihood functions of the posterior distribution for each lattice $X_t$ remain unchanged for $M_3$ because of the same independence relations that left them unchanged for $M_2$.

Recall that the energy potentials $\Phi_c$ which are associated with the likelihood functions require gaussian background models for each frame of the image sequence. Normally, with online segmentation processes, the background model is maintained sequentially, and is only needed for the current image frame. But, since every seg-

\textsuperscript{4}such as motion segmentation.
mentation $S_t$ of $M_3$ is derived from the MAP estimate of the field $X$ simultaneously, the state of the background model at all times $1 \leq t \leq T$ is required. To get the required background model for all $t$, the image sequence was first segmented using $M_1$. During that (online) process, the background model associated with each image frame was stored, so that it could be used during the MAP optimization procedure of $M_3$.

4.4 Implementation Considerations

Although there seem to be many details to be concerned with when implementing a Markov random field optimization algorithm, it is very straightforward in practice. In the following sections, we will discuss the implementation details of the potential functions with respect to the field structures $M_1$, $M_2$, and $M_3$, the simplified annealing schedule used in our experiments, and the simplifying assumptions used to optimize over the various field structures. A sample of the actual code used in the experiments to perform the MRF optimization is given in appendix C.

4.4.1 Potential Functions

Let us first examine the potential functions $V_c$. In the previous sections, we introduced specific names for the potentials depending on which cliques $c \in C$ were being referenced. The cliques involving only segmentation variables at the current time $t$ (a spatial clique) are given the designation $\Psi_c$. The cliques involving a segmentation variable and itself in the past or the future (a temporal clique) are designated $\Theta_c$. Those cliques of the posterior distribution involving a segmentation variable and a pixel variable are $\Phi_c$. These potentials are used in equations 3.8 and 3.9 to determine the probability of each segmentation variable being either foreground or background, given the settings of its neighbors.

The cliques involved in these computations are all two-cliques. As such, the actual implemented potential functions require at most two indices to access the required data. The function that implements $\Psi_c$ requires two points $s$ and $r$, each with an $x$
and y coordinate. Since \( \Psi_c \) is invariant to orientation, the return value is a simple table lookup based on the values of the segmentation variables \( X_s \) and \( X_r \) (the current time \( t \) is implied).

Although the functions \( \Phi_c \) and \( \Theta_c \) involve two-cliques, they are only indexed by \( s \). This is because the variables involved in \( \Phi_c \) are \( X_s \) and \( D_s \). Those variables involved in \( \Theta_c \) are \( X_{s,t} \) and \( X_{s,t-1} \) for \( M_2 \) and \( X_{s,t} \) and \( X_{s,t\pm 1} \) for \( M_3 \). Although \( \Theta_c \) could be implemented using \( t \) as a parameter, it is simpler just to index the variables by \( s \) and in the case for \( M_2 \), look behind one frame, and for \( M_3 \), look both ahead and behind one frame. The implementation of \( \Theta_c \) requires a table lookup over the values of \( X_{s,t} \) and \( X_{s,t-1}/X_{s,t\pm 1} \). The implementation of \( \Phi_c \) requires a table lookup when \( X_s \) is foreground (it is just a number), and a table lookup of a computation done once per \( X_s \) per \( t \) when \( X_s \) is background.

### 4.4.2 Annealing Schedule

The annealing schedule used in our experiments is unlike that which was described in section 3.4. The annealing schedule described previously was a logarithmically decreasing function. The annealing schedule we settled on is a linearly decreasing function:

\[
T = M \cdot k + S
\]

where \( k \) is the current \( (k^{\text{th}}) \) iteration of the Gibbs sampling algorithm, \( S \) is the initial temperature, and \( M \) is the "slope"

\[
M = \frac{E - S}{K - 1}
\]

where \( E \) is the final temperature, and \( K \) is the total number of iterations. Thus, the annealing process begins at the \( (S) \) tart temperature, goes through \( K \) iterations, and stops at the desired \( (E) \) nd temperature. We chose a linear annealing schedule because a) segmentation accuracy was very similar regardless of which schedule we used, and b) a linear schedule requires less computation.
To reduce the number of iterations necessary to achieve good results, the initial configuration $\omega_0$ of the MRF was not randomly chosen. Rather, we threshold the image based on [12] as outlined in section 2.2. This provides a "best guess" initial estimate $\omega_0$ with which to begin the optimization process. With this $\omega_0$, only 5-15 iterations (5 for $M_1$ and, because of the complexity of the field, 15 for $M_3$) are needed for the configurations of the fields to converge and produce good results.

4.4.3 Optimization Simplifications

The segmentation variables along the edges of the lattice do not have the full neighborhood structures described in the previous sections. Likewise, the variables in $X_1$ and $X_T$, where $T$ is the index of the last image frame, are missing temporal neighbors as well. To relieve us of the burden of dealing with these border cases, we chose to have those variables without full neighborhood structures always be segmented as background. Thus, for $M_1$, the edge pixels will always be classified as background. For $M_2$, the edge pixels and the first image frame will be background. For $M_3$, the border pixels, and the first and last image frame will be background. Considering the images are approximately $640 \times 480$ pixels and there are approximately 1,000 frames in a given sequence, this simplification is inconsequential (but very handy).

4.5 What is in a Name

Markov thresholding is the name we chose for the background subtraction processes we have described in this chapter. Although it bears little resemblance to the standard subtraction techniques described in section 2.2, it is still in some ways a thresholding process. To illustrate the point, let us examine field $M_1$ when each $\Psi_c = 0$. This family of potentials leaves $X$ without a neighborhood structure. Each $X_s \in X$ is completely independent of its neighbors, and of the rest of the $m \times n$ lattice. However, for the posterior distribution, each $X_s$ is still connected to its corresponding pixel $D_s$, and the potential functions $\Phi_c$ encoding their relationship remain unchanged.

Now examine figure 4-8. Figure 4-8 overlays the two energy potentials of $\Phi_c$ (one
Figure 4-8: The assumption that foreground objects comes from a uniform distribution over the colorspace of $D$ manifests as a threshold on background match quality for each $D_s$.

when $X_s = 0$ and the other when $X_s = 1$). Since we have ensured that each $X_s$ is independent of all others, this graph shows all there is to making a decision about whether a given pixel should be labeled foreground or background. Indeed, when $\Psi_c$ is always 0, this process becomes identical to [12]. The straight line in figure 4-8, representing the uniform distribution when $X_s = 1$, sets a threshold on the gaussian term. Where the two intersect depends upon the energy value we set for $\Phi_c$ when $X_s = 1$ and the parameters $\mu_k$ and $\sigma_k$ of the background models.

Let us now revert back to the energy potentials for $\Psi_c$ described in equation 4.1. Notice that the only variable element in all of the energy potentials $\Psi_c$, $\Phi_c$, and $\Theta_c$ (for $M_2$ and $M_3$), is the gaussian variance term of $\Phi_c$ when $X_s = 0$. All of the other potentials are fixed and do not change based on incoming image frames or the dynamic background model. They are fixed, just as the energy potential associated with the uniform distribution of $\Phi_c$ is fixed. The energy potentials $\Psi_c$ and $\Theta_c$ associated with
Figure 4-9: The prior distribution of $X$ and the potentials $\Psi_c$ and $\Theta_c$ that correspond to it change the threshold on the gaussian background model set by the uniform distribution of the foreground model. The optimal configuration of $X$ balances the stability of the field with the likelihood of it to generate the observed image field $D$.

the prior distribution of $X$ raise or lower the threshold caused by the intersection of the uniform distribution energy term with the gaussian variance term. Those potentials make it easier or harder for a pixel to match its corresponding background model (see figure 4-9).

Raising and lowering the background matching tolerance for each $X_s$ is an effect of the prior potentials $\Psi_c$ (and $\Theta_c$) during the optimization process. Since those potentials derive their values based on the segmentation value of their neighbors, the choice of which potentials to apply are all interdependent. Thus, if you have a pixel $D_s$ surrounded by pixels which match their background models very well, the segmentation settings of its neighbors will likely lower the threshold on $D_s$ to the point where it, too, will be considered background. The threshold on the background model is dynamic, and reflects the composition and structure of the segmentation, the optimal configuration of the Markov random field. This is why the process is
referred to as Markov thresholding.
Chapter 5

Results

In this chapter we will highlight the strengths and weaknesses of Markov thresholding as a background subtraction process. We will compare the segmentation performance of the three field structures $M_1$, $M_2$, and $M_3$ introduced in the previous chapter against one another. We will also compare the performance of those fields with respect to an implementation of Stauffer’s approach (let us refer to it as $T_s$), and with respect to a hysteresis-based subtraction method derived from it. For the Near Field image sequence, we also compare the five approaches to a ground-truth, manual segmentation $G$.

5.1 Hysteresis Background Subtraction

Before we can discuss the performance of the algorithms, we must first introduce the hysteresis based segmentation process $T_h$ that is derived from $T_s$. It is a simple and effective generalization over the singular threshold method that exploits to some extent the spatial cohesiveness of objects.

Instead of a subtraction process that produces a segmentation by thresholding on a single value $\tau_s$ (e.g. 2.5 standard deviations from the mean), $T_h$ has both a strict background matching threshold $\tau_b$ (typically 1-3 standard deviations) and a strict foreground matching threshold $\tau_f$ (typically 8-12 standard deviations). The segmentation produced by $\tau_b$ will contain many false-positives. That binary map will
likely contain pixels that have been incorrectly labeled as foreground. However, those pixels labeled background must have matched their background models very well to have passed the strict background matching test. As such, those pixels are treated as “ground truth” background pixels.

Conversely, the segmentations produced by $\tau_f$ will contain many false-negatives. That binary map will likely contain pixels that have been incorrectly labeled as background. Likewise, those pixels which are so unlike their background models that they passed the strict foreground test can be confidently labeled foreground. These pixels comprise the set of ground truth foreground pixels.

To combine the information in the segmentations produced by $\tau_b$ and $\tau_f$ into a single binary map, objects must be formed from the individual pixel elements and their binary labels. An object is formed by connected component analysis. As the name suggests, the procedure for forming objects involves assigning a unique tag (i.e. number) to each group of foreground pixels that are spatially connected over a neighborhood structure, such as the second-order, 8-connected neighborhood structure described previously. The effect is that “islands” of foreground pixels become objects.

Objects must be formed from the segmentation produced by $\tau_b$. Since it is difficult for a pixel to match its background model under this threshold, the objectification of this segmentation will produce many foreground objects. To separate the real foreground objects from those caused by noise and other effects, only those objects that contain foreground pixels present in the segmentation produced by $\tau_f$ are kept. All other objects are removed from the final segmentation.

The effect of the two thresholds $\tau_b$ and $\tau_f$ is to separate pixels into three categories: foreground, background, and uncertain. The pixels that are labeled foreground are the set of ground truth foreground pixels and will be classified as foreground in the final segmentation. Likewise, the pixels that are labeled background are the set of ground truth background pixels and will be classified as background in the final segmentation. Those pixels that could not pass either of the threshold tests have an uncertain status, as they are not ground truth pixels. To handle them, those pixels
of uncertain status that are adjacent to, or recursively adjacent to, ground truth foreground pixels are reclassified as foreground in the final segmentation. All others are reclassified as background. Since all uncertain pixels will be labeled as foreground when the image is thresholded against \( \tau_b \), the objectification of that segmentation is a convenient way to determine which uncertain pixels are adjacent (recursively or otherwise) to ground truth foreground pixels. All objects that are not adjacent to ground truth foreground pixels contain only pixels of uncertain status. Therefore, these pixels are reclassified as background in the final segmentation.

\( \tau_f \) will always be greater than or equal to \( \tau_b \). If it were not, then their roles would simply be reversed. When \( \tau_b = \tau_f \), there is effectively only one threshold. This particular setting of the thresholds reduces \( T_h \) to \( T_s \). When \( \tau_b = \tau_s \) then the setting of \( \tau_f > \tau_b \) has the effect of reducing the number of false-positives while increasing the number of false-negatives with respect to \( T_s \). Likewise, when \( \tau_f = \tau_s \), then the setting of \( \tau_b < \tau_f \) has the effect of reducing the number of false-negatives while increasing the number of false-positives with respect to \( T_s \).

Since foreground objects can potentially be of any color, there are many times when the color of a foreground object will match the background model quite well. This situation is an example of the classic motion segmentation problem known as *camouflaging*. It is our experience that direct thresholding techniques that do not incorporate any spatial dependencies between pixels (such as \( T_s \) and \( T_h \)) have difficulty eliminating false-negatives. At the same time, \( T_s \) produces a plethora of false-positives. Thus, to set \( \tau_b < \tau_s \) usually implies a vast increase in false-positives (i.e. noise recognized as foreground objects) without a similar decrease in false-negatives (i.e. silhouette holes and other camouflaging effects). But, to set \( \tau_f > \tau_s \) implies just the opposite. Setting \( \tau_f > \tau_s \) reduces noise in the binary segmentation without wholly undoing the correct segmentation. The settings of \( \tau_b = .8\tau_s = 2 \) and \( \tau_f = 4\tau_s = 10 \) are good in many cases, and are the numbers we used in our own experiments.
5.2 Analysis Overview

Segmentations extracted from four image sequences using five different algorithms ($T_s$, $T_h$, $M_1$, $M_2$, and $M_3$) will be examined and their performance characteristics analyzed. Two similarity metrics were used to compare the silhouettes extracted by each algorithm against one-another. Since there are 5 algorithms, there are $\binom{5}{2} = 10$ distinct pairings.

The first metric used measures the ratio of the intersection of the silhouettes to the union of the silhouettes:

$$R(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where $A$ and $B$ are binary segmentations. The elements of $A$ and $B$ are the coordinate pairs of all foreground pixels in their respective segmentations.

The size of the sets $A \cap B$ and $A \cup B$ are easy to find. Cycling through the binary maps, counting all pixels whose segmentations are foreground in both images gives the size of the intersection. Counting all pixels where either segmentation is foreground gives the size of the union. If the size of both sets is 0 (i.e. there are no foreground pixels present in either segmentation), $R(A, B) = 0$. A perfect segmentation match occurs when $|A \cap B| = |A \cup B|$, which results in the score $R(A, B) = 1$.

The second similarity metric used is the Hausdorff distance measure $H(A, B)$ [4]. The Hausdorff distance is widely used to measure the similarity between finite sets of all types. For a description of this measure, please see appendix B. To measure the Hausdorff distance when one of the segmentations contains only background, the distance in that case is assumed to be $\sqrt{m^2 + n^2}$, where $m$ and $n$ are the numbers of rows and columns respectively in the images. Thus, the distance is assumed to be the largest possible distance.

5.3 Algorithm Parameters

The background model used for all 5 background subtraction algorithms was the multidimensional mixture of gaussians. The parameters of the background model
An implementation consideration that must be addressed is a minimum limit on the value of the standard deviation $\sigma$ allowed for any gaussian. For $T_s$ and $T_h$, that limit is $\sigma = 10$. For $M_i (1 \leq i \leq 3)$ that limit is $\sigma = 5$. The reason for this limit is so that spurious lighting effects that occur too infrequently to be modeled do not interfere with the segmentation process. A lower, lower limit is used for the Markov random fields because a) the MRFs have a better ability to cope with noise and b) this limit effects the false negative (camouflaging) rate of the background subtraction algorithms (so, the lower the better!).

5.4 **NIST 1**

The first sequence to be analyzed comes from the May 2001 USF-NIST data set. The sequence name is 02463C0AL. For more information about this data set, or for information on obtaining it, please see [10]. In this sequence, a person is at the far right side in the camera’s field of view. During this sequence, he walks to the left side of the camera’s field of view, turns toward the camera, and walks back to the right. It ends with the man in the center of the image, walking to the right.

Figure 5-1 shows typical outputs from all 5 algorithms when run on data sequences similar to this one. Algorithm $T_s$ exhibits a fair amount of noise, whereas the other algorithms were able to remove it. $T_h$ removes the noise by virtue of the noise having no ground truth foreground pixels. The fields $M_i$ remove the noise by regularizing the image. Since the foreground noise is surrounded by background, those pixels have an easier time matching their background model. Note that the “noise” in the
upper-middle portion of the image is caused by leaves on the tree that are rustling in the wind. By having those pixels consistently labeled background, the variances associated with those background colors will typically be large. This has the effect of reducing the segmentation sensitivity in that area. If true foreground objects were to pass through that area, they would be more easily camouflaged. Bimodal backgrounds would normally be used to model this area, but since the wind gusts are sporadic, it is doubtful that the rustling leaves could be accurately modeled by such a background.

Figure 5-2 shows the center column of figure 5-1 on a larger scale. This figure shows segmentations characteristic of all the algorithms used. As was noted earlier, the fields $M_i$ have eliminated the noise caused by lighting effects and the slight, repetitive motion of the tree’s leaves. Let us now examine the silhouettes. $M_1$ was able to fill in most of the silhouette holes caused by camouflaging effects seen in $T_s$ and $T_h$. By incorporating time information into the segmentation, $M_2$ and $M_3$ were able to completely fill in the silhouette. It is therefore likely that at some time in the past (for $M_2$ and $M_3$) and/or sometime in the future (for $M_3$), the man was less-camouflaged by the background and thus these algorithms were able to use that information to conclude that the entire area is foreground.

A typical side-effect of the regularization process is that slivers of foreground or background do not survive the segmentation process. The man’s legs in the image are clearly separated, and background should show between his legs in the silhouettes. This is indeed that case for the spatially independent algorithms $T_s$ and $T_h$. But for $M_i$, that small amount of space is surrounded by foreground pixels. In that region, the similarity of each pixel’s color to that of its background model is overpowered by its prior probability of matching the segmentation of its neighbors. The energy potentials $\Psi_e$ and $\Theta_e$ associated with the prior distribution can be changed to alter this behavior. Depending on how sensitive a higher-level application is to this sort of segmentation behavior dictates how the prior potentials should be chosen. We have chosen potentials biased towards false-positive segmentation errors because we regard cohesive silhouettes as more accurate and qualitatively more appealing.

Figure 5-3 shows another subsequence taken from NIST 1. This sequence further
illustrates the performance characteristics of the fields for this camera setup. Figure 5-4 shows the center column of 5-3 in larger detail. In this figure, the ability of the fields to remove noise and to fill in silhouette holes is displayed. Also, it is easy to see that $M_2$ exaggerates the silhouette by filling in the spaces between the man’s legs and by filling in the spaces between his arms and his body. This is more prevalent in $M_2$ than in $M_1$ or $M_3$ because of its unique time dependence.
Figure 5-1: NIST 1, Subsequence 1
Figure 5-2: NIST 1, Subsequence 1, Center Column
Figure 5-3: NIST 1, Subsequence 2
Let us now examine figure 5-5. The first few frames (up to about frame 100), are included for completeness, but should not be scrutinized too heavily. The justification for this is because it takes about that long to completely build up the background
model and get the best possible performance out of the various algorithms. During the first hundred or so frames, each algorithm will provide only partial segmentations, and they are not guaranteed to provide similar partial segmentations. For similarities measured under the $R(A, B)$ metric, higher scores (i.e. those closer to 1) indicate greater similarity. Under the $H(A, B)$ similarity metric, lower scores (i.e. those closer to 0) indicate greater similarity.

The first notable feature of the graphs in figure 5-5 is the repeated dip in similarity around frame 250 present in the first four graphs. The reason for this is that a gust of wind has rustled the leaves and caused a lot of noise to appear in $T$, but not in any of the other algorithms. Thus, the ratios involving $T$ will be reduced, since the size of the union set would grow, but not the size of the intersection.

The other important feature of this graph is that the silhouettes produced by the fields $M_i$ and by $T_h$ are all quite similar. They all share about 90% of the segmented pixels. The silhouettes produced by $M_1$ and $M_3$ are particularly similar. In this case, the segmentations are nearly equivalent.

The Hausdorff measure of similarity presented in figure 5-6 shows relative algorithm performance in a markedly different way. The clearly sinusoidal nature of the graphs involving $T$ is because of the noise prevalent in that segmentation and the path of the man walking through the field of view of the camera. With the noise scattered throughout the segmentation produced by $T$, the Hausdorff distance peaks around frame 275. This corresponds to the point in the man’s path when he is furthest from the middle (he is at the leftmost point of his walking cycle with respect to the field of view of the camera).

There are two prevalent features in those graphs relating each $M_i$ with $T_h$: a broken sinusoid, and a spike around frame 300. The broken sinusoid is due to motion of a person on the steps behind the calibration cube being picked up by each $M_i$ but not by $T_h$. It is sinusoidal because of the walking cycle of the man in the sequence. It is broken because the person on the steps moves erratically, and his motion is only segmented out occasionally. The spike in these graphs is caused by more spurious motion on the far right side in the images being picked up by each $M_i$ but not by $T_h$. 
The three graphs comparing the segmentations produced by each $M_i$ show that they are very similar. Aside for some spurious noise picked up in one algorithm but not the other, the similarity measures are very close to 0.
Figure 5-5: $R(A, B)$ similarity measure for NIST 1
Figure 5-6: $H(A, B)$ similarity measure for NIST 1
5.5 NIST 2

The next sequence that was segmented comes from the May 2001 USF-NIST data set as well. The sequence name is 03512C0AL. This sequence begins with a man with a cane on the left side of the image, facing the camera. He begins turning to the right and walking through the camera's field of view, from left to right. He then turns away from the camera and begins walking back, from right to left. The sequence ends with the man on the far left side, near the point where he began initially.

Figures 5-7 shows a characteristic portion of the NIST 2 image sequence, and figure 5-8 is an enlarged view of the center column of 5-7. These figures again show the noise prevalent in $T_s$ and the ability of $T_h$ to filter that noise out of its final segmentation. The significant features of these silhouettes include the obvious shadow, the man's cane, and the silhouette hole caused by camouflage in the man's right shoulder.

First, let us examine the shadow. Obviously, we would not consider that shadow as part of a motion segmentation, since we casually disregard just that sort of feature. It would be very nice if these algorithms could just as easily distinguish between light-reflectance patterns caused by shadows and those caused by the reflectance off of actual objects. But the fact is, the shadow is a very noticeable feature. The moving shadow changes the color pattern of the background in a very obvious way. Therefore, it could be considered a proper segmentation to include the shadow. After all, a shadow caused by a stenciled mask of a bird attached to a moving flashlight would be rightly associated with motion. To include the shadow of the man on the concrete as a proper moving object to be segmented may or may not be correct, but it is an example of an algorithm's output dictating our working definition.

Next we turn our attention to the various ways the cane was segmented. Due to the size of the images in figure 5-7 and the fact that the cane is parallel to the construction cone, it may be difficult to actually see the cane. The spatially independent algorithms $T_s$ and $T_h$, however, have no significant problems locating it. Because the cane is so thin, however, the fields $M_i$ disregard it for the most part. Again, this is due to the regularization process of MRFs. The cane is such a tiny scrap of foreground that, to
each $M_i$, the prior probability for that particular arrangement is relatively small. A change in the prior potentials $\Psi_c$ and $\Theta_c$ could cause the fields to locate the cane, but would likely sacrifice segmentation accuracy in many other ways.¹

Finally, let us regard the silhouette hole located at the man’s right shoulder. The algorithms $T_s$ and $T_h$ have no mechanism to fill in that region of their segmentations. $M_1$ does have that mechanism, but in this case, regularizing around the hole was obviously the field’s first choice. This is unfortunate, but not unexpected. The silhouette hole is large compared to the neighborhood structure. Thus, the hole in the silhouette appears more as a feature to be regularized around rather than an aberration in a larger pattern of foreground segmentation. Likewise, for $M_3$, the hole is present in several consecutive frames. Thus, when optimizing over time, that hole again appears to be a feature rather than an aberration. Only $M_2$ managed to fill in the hole. It was able to do so because it does not optimize over time. It simply uses the previous segmentation to alter the prior for the current segmentation. Thus, as long as that hole was not present at some frame in the past, that information would propagate all the way through the camouflaging area. This is the principle benefit of $M_2$’s field structure over that of $M_1$ and $M_3$.

¹After all, the fields can be reduced to $T_s$ simply by a setting of 0 for all priors $\Psi_c$ and $\Theta_c$. The cane would then be located, but with a significant impact on overall segmentation.
Figure 5-7: NIST 2, Subsequence 1
Figure 5-8: NIST 2, Subsequence 1, Center Column
Figure 5-9: $R(A, B)$ similarity measure for NIST 2
Figure 5-10: $H(A, B)$ similarity measure for NIST 2
Figure 5-9 shows comparative algorithm performance based on $R(A, B)$ that differs from that of 5-5. Since the background is the same, it can be concluded that the differences must lie in the difficulty of segmenting this particular person. In these graphs, algorithm $T_h$ performed more like $T_s$ than it did like the fields $M_i$. This can be seen by the relatively flat progression of the first, $T_h - T_s$ graph, as opposed to the $M_i - T_h$ graphs. The fields themselves have very similar performance. Again, $M_1$ and $M_3$ have markedly similar segmentations according to this similarity metric.

The Hausdorff distance metric graphs of figure 5-10 shows comparative performance similar to that of figure 5-6. The clear sinusoidal pattern is seen in all graphs where $T_s$ is involved. A broken sinusoid is present in those graphs involving $T_h$ as well. This arises due to a scattering of noise or insignificant background motion picked up in one algorithm but not the other. There is a linear feature present around frame 610 the lasts through until the end in the graphs involving $M_2$ and either $M_i$ or $T_h$. This is due to a small foreground perturbation consistently segmented by $M_2$ but disregarded by the other algorithms. Again, it is clear that $M_1$ and $M_3$ produce the most similar segmentations.

5.6 PETS

This image sequence comes from the 2001 PETS workshop. It is from data set 1, training data, camera 1. For more information about this data set please see http://www.cvg.cs.rdg.ac.uk/VS/PETS2001. To download this data set, please try ftp://pets2001.cs.rdg.ac.uk. This is a long image sequence involving a far-field camera setup overlooking the intersection of a street, parking lot, and building entrance. Cars, bikes, and pedestrians are present in this challenging image sequence.

Figure 5-11 shows a portion of this sequence. In this subsequence, a car is seen entering from the left and another car is backing out of a parking space near the center of the image. The $T_h$ segmentation is predictably similar to $T_s$, sans noise. The fields $M_i$ more accurately segment the car entering from the left by filling in the holes present in $T_s$ and $T_h$. Because the motion of the car backing out of its
parking spot is so slight, the fields do not perform the segmentation so completely. Although this behavior can be viewed as a limitation in this scenario, it is precisely this behavior that makes these fields so tolerant to minute variations picked up by other algorithms.

Figure 5-12 shows another portion of this sequence at a later time. There are many different phenomena present in these frames. First, notice the parking spot where the car in figure 5-11 was leaving. The segmentation algorithms continue (to one degree or another) to segment that area as foreground. This is another classic motion segmentation problem known as \textit{waking}. Waking is the term used to describe an object that is initially part of the background but suddenly moves, or wakes. The portion of the ground that is left looks like foreground when compared to the background model, which assumes that a car is there. Likewise, there is a car in a parking spot that is segmented as foreground. This car is in the process of \textit{sleeping}. Sleeping is the term used to describe an object that was once moving, but is now being incorporated into the background model. Dealing with sleeping and waking objects is essential for modeling real-world situations. It is one of the main reasons for using dynamic, adaptable background models. The time it takes for a sleeping object to be incorporated into the background, or the time it takes for the image patch left after a sleeping object wakes to be incorporated into the background depends on the learning rates $\alpha$ and $\rho$.

There are various ways that sleeping and waking objects can be dealt with. One way is to maintain multiple background models with different learning rates. This has the effect of maintaining background models of the same scene over various time scales. Segmenting from each of those different background models will produce different segmentations, and only the motion present over several time scales will be included in the final segmentation. This is similar in spirit to edge detection techniques that only include edges located within several different scale spaces. Not much has been done regarding this, or other possible techniques, but it is a promising research direction.
Figure 5-11: PETS, Subsequence 1
Figure 5-12: PETS, Subsequence 2
The similarity graphs of figure 5-13 look very different from those of the NIST data sets. These graphs have many downward pointing spikes. While these may seem like prevalent features, they are in actuality simply an indicator of the number of foreground segmentations trailing to 0. These points indicate those frames where nothing is moving. The points to note about these graphs are that the fields $M_i$ are more alike to each other than all other pairings. The $M_i - T_h$ pairings are more similar than the $M_i - T_s$ pairings, but are more volatile than the the $M_i - M_j$ pairings. Thus, while for some frames $T_h$’s segmentation may be quite similar to $M_i$’s, it is not always (or even usually) the case.

For the Hausdorff measure of figure 5-14, the lack of foreground objects manifests as a horizontal line at the maximum distance in the graphs. The results show the same typical patterns observed for the previous image sequences. None of the algorithms are all that similar to $T_s$, due to the scattering of noise in the silhouettes of $T_s$. The $M_i - M_j$ pairings show the most resemblance to each other, and they each show similar results when compared to $T_h$. 

78
Figure 5-13: $R(A, B)$ similarity measure for PETS
Figure 5-14: $H(A, B)$ similarity measure for PETS
5.7 Near Field

This image sequence was taken at the MIT Artificial Intelligence Laboratory. It consists of a person entering the field of view of the camera on the right, walking to the left out of the camera’s view, and then reversing course to end where he started.

As the results in figure 5-15 indicate, this is a particularly challenging sequence. The person seen walking in this sequence is heavily camouflaged: his shirt matches the horizontal depressions in the wall, his shorts match the bag lying on the ground, and his lower legs match the leftmost filing cabinets. Furthermore, he is casting a fairly large shadow on the ground. In other words, this is a typical image sequence taken from within a cluttered office environment.

Because there is so much variation in the background, the batch modeling process $M_3$ achieved the best results. While the person may be camouflaged in one frame, that part of him may contrast nicely with the background in the next. Therefore, optimizing over time provides information beneficial to the segmentation. In a similar way, because of the potentials $\Theta_e$ we chose for $M_2$, the silhouette of the walking person becomes more cohesive as time goes by. This is because once a pixel has been labeled foreground, it will tend to stay foreground, requiring more evidence to the contrary than is usually necessary to make future segmentations of that pixel background.

The similarity measures shown in figures 5-16 and 5-17 indicate that the fields $M_i$ are most similar to each other and that $T_s$ and $T_h$ are more similar to each other than to any $M_i$. This is not surprising, as there is not much noise in the image sequence, so the silhouettes produced by $T_s$ and $T_h$ should be very similar. The graphs also show that $M_1$ and $M_3$ are again very similar. This is because $M_2$ tends to dilate the silhouettes, which causes a small number of border pixels to be consistently mismatched with regard to the segmentations of $M_1$ and $M_3$. The effect is small, however, and figure 5-16 actually shows that $M_1$ and $M_2$ may be the most alike for this sequence.

Although comparing the five algorithms against one another can give a detailed view of the relative performance of each approach, comparing them against the actual
motion segmentation is the only way to gauge ground-truth performance. The problem with acquiring the actual motion segmentation, aside from the issue of defining motion segmentation in the first place, is that it requires laborious hand-segmentation of the image sequence. However, we have provided the ground-truth manual segmentation $G$ of the five-image sequence shown in figure 5-15.

Figure 5-18 shows how well each of the five approaches compares against $G$ using the $R(A,B)$ similarity metric. Under this measure, the three MRF fields $M_i$ match the actual segmentation $G$ better than the two threshold-based approaches. Furthermore, field $M_3$ is the clear winner, with $M_1$ and $M_2$ showing similar performance. Figure 5-19 shows a slightly different picture of algorithm performance. Under the Hausdorff similarity measure, $T_s$ is the least inaccurate, due to the scattering of noise that, while significantly reduced compared to the segmentations of the other image sequences, is not eliminated entirely. Furthermore, under this measure, the field $M_1$ best matches the ground truth segmentation $G$. This indicates that, while $M_3$ has the largest number of correctly segmented pixels, the pixels of $M_1$'s segmentation were all near to $G$'s foreground-segmented pixels. However, $M_2$ and $M_3$ are both so close in score with $M_1$ that this is most likely an artifact of the transient shadow segmentations of each field.
Figure 5-15: Near Field, Subsequence 1, With Ground-Truth Manual Segmentation
Figure 5-16: $R(A, B)$ similarity measure for Near Field
Figure 5-17: $H(A, B)$ similarity measure for Near Field
Figure 5-18: $R(A, B)$ similarity measure v.s. Manual Segmentation
Figure 5-19: $H(A, B)$ similarity measure v.s. Manual Segmentation
Table 5.1: Numbers of sequences present in each of the eight combinations of categories. The categories include surface type (Concrete/Grass), shoe type (A/B), and camera (Left/Right). There are 452 images sequences in total.

<table>
<thead>
<tr>
<th>Surface</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoe Type</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Camera L</td>
<td>70</td>
<td>44</td>
</tr>
<tr>
<td>Camera R</td>
<td>70</td>
<td>44</td>
</tr>
</tbody>
</table>

5.8 Gait Recognition

As mentioned in section 1.1, the binary maps output by motion segmentation algorithms are usually not ends unto themselves. Rather, they are used as input to higher-level vision applications. One such application that uses motion segmentation data is gait recognition. Recall that gait recognition refers to the identification of individuals by the way they walk. To test the effectiveness of this new approach, we used silhouettes computed by it as input to a gait recognition experiment.

The Markov random field segmentation procedure described in the preceding chapters was used to compute the silhouettes of pedestrians from the data set of the well-known gait recognition challenge experiment, HumanID [10]. The data set consists of 452 image sequences of 71 individuals taken from two cameras. Each of the 452 image sequences is categorized based on common characteristics. The categories include whether the sequence was taken from the left or right camera (L/R), whether the pedestrian walks on a concrete or grass surface (C/G), and whether the pedestrian is walking with shoe type A or B (A/B). Table 5.1 shows the numbers of sequences which fall into each of the eight combinations of categories.

The challenge experiments consist of separating the motion segmented data into gallery and probe subsets. The sequences do not consist of full-color images, but rather the silhouettes output from the motion segmentation process. The gallery consists of one sequence for each of the 71 individuals and is used as a representative sample of each person’s distinctive walking style, or gait. The probe consists of a subset of the image sequences that is guaranteed not to intersect with the gallery. The probe does not have to contain a sequence for every individual, and it usually
Table 5.2: The probe sets for each of the challenge experiments in [10]. The number of sequences (and individuals) in each probe subset are in curly braces. The gallery uses the subset \((G,A,R)^{71}\). The third column lists the categories in which the probes and gallery differ.

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Probe</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((G,A,L)^{71})</td>
<td>View</td>
</tr>
<tr>
<td>2</td>
<td>((G,B,R)^{41})</td>
<td>Shoe</td>
</tr>
<tr>
<td>3</td>
<td>((G,B,L)^{41})</td>
<td>View, Shoe</td>
</tr>
<tr>
<td>4</td>
<td>((C,A,R)^{70})</td>
<td>Surface</td>
</tr>
<tr>
<td>5</td>
<td>((C,B,R)^{44})</td>
<td>Surface, Shoe</td>
</tr>
<tr>
<td>6</td>
<td>((C,A,L)^{70})</td>
<td>Surface, View</td>
</tr>
<tr>
<td>7</td>
<td>((C,B,L)^{44})</td>
<td>Surface, View, Shoe</td>
</tr>
</tbody>
</table>

does not. The experiments consist of measuring the similarity of each sequence in the probe with each sequence in the gallery. An identification of an individual present in the probe is successful, at rank \(i\), if that person's probe and gallery sequences are within the top \(i\) for match similarity. For example, at rank 1, the identification of an individual is considered successful if his sequence in the probe best matches — out of all gallery sequences — his own sequence in the gallery.

The gallery is the same across all challenge experiments. It consists of 71 sequences representing each of the 71 individuals. It is the \((G,A,R)\) category combination (see table 5.1). In [10], seven different challenge experiments are presented. The challenge experiments consist of probe sets with varying degrees of similarity to the gallery. With varying degrees of similarity come varying degrees of difficulty for each experiment. The subsets used for the probe in each of these experiments is detailed in table 5.2.

Five different motion segmentation algorithms that have been used to compute the silhouettes for this data set will be compared. The baseline silhouettes were computed using the semi-automatic NIST procedure, \(S_N\), which is described in [10]. Initially, this is the set of silhouettes to outperform. A motion segmentation algorithm similar to the NIST baseline algorithm, but fully automatic, is described in [8]. It is referred to as \(S_r\), and produces better recognition performance than \(S_N\). Also in [8] is a procedure for fitting a model of a pedestrian to the raw silhouettes. The results of this procedure are higher-quality silhouettes, and, usually, higher recognition rates.
This model-fitting procedure was applied to $S_r$ to produce $S_p$ and to $M_2$, the online, time-dependent Markov random field structure described in section 4.3.2, to produce $M_p$. The five sets of silhouette sequences, $S_r$, $S_p$, $S_N$, $M_2$, and $M_p$ were then processed to compute the features used to measure the similarity between probe sequences and the gallery sequences.

The region-based features used to measure similarity between silhouette sequences were first introduced by Lee and Grimson in [9]. These features must be computed on silhouettes in which the person is walking parallel to the image plane (i.e. a side-view). To compute the final, 60-feature vector for each sequence, a number of intermediate features must be computed for each silhouette in the sequence. For each silhouette, the centroid is first computed and an ellipse is fit to the data to obtain an aspect ratio for the individual. Each silhouette is then divided into seven regions. The regions correspond to the areas about the head and shoulders, arms and torso, thighs (two regions), and calves and feet (two regions). For each of the seven regions, an ellipse is fit to the portion of foreground data within the region to describe its centroid, aspect ratio, and the rotation angle of the ellipse.

The centroids, aspect ratios, and rotation angles computed for each silhouette are assumed to be sampled from independent Gaussian distributions. The means and standard deviations for each Gaussian are then computed over the entire silhouette sequence. The 60-feature vector is comprised of the means and standard deviations of: the y-coordinate of the silhouette centroid (2), the aspect ratio of the ellipse fitted to the silhouette (2), the x- and y- coordinates of the centroids for each region (28), the aspect ratios for the ellipses fitted to each region (14), and finally the rotation angles of the ellipses fitted to each region (14). The similarity between any pair of silhouette sequences can then be measured by computing the Euclidean distance between their corresponding feature vectors.

Figures 5-20 and 5-21 show the gait recognition rates for the five motion segmentation algorithms at ranks 1 and 5, respectively. It is clear from these graphs that, overall, recognition accuracy falls sharply for probe sets 4-7 with respect to probe sets 1-3. A quick scan of table 5.2 shows that the last four probe sets have surface con-
ditions that differ from that of the gallery, while the first three probe sets are based on the same (grass) surface. Thus, it appears that the surface on which pedestrians walk significantly influences the manner in which they walk.

Setting aside gait theory, the results also show that SN is the clear loser. In nearly all experiments at either rank, it yields the worst recognition rate. Furthermore, in all experiments at either rank, it fails to achieve a higher recognition rate than either M2 or its pedestrian model-fitted variation, Mp.

Comparisons between M2, Sr, Mp, and Sp are more interesting. As shown in figure 5-20, at rank 1 the recognition rate of Sr bests M2 in all but the last experimental setup. However, applying the pedestrian model to M2 yields much greater recognition rates. Mp bests Sr in every experiment, and it bests Sp using the more challenging probe sets. The recognition rates of Sp and Mp under the first three experimental setups are close (though Sp wins), and it is clear that the model-based silhouettes offer superior recognition performance.

The results at rank 5 illustrated in figure 5-21 shows a different version of recognition performance. At rank 5, M2 loses to Sr under only one probe set, experiment 4. Furthermore, M2 achieves higher recognition rates than the model-based Sp in two of the experiments and ties it in another two. Conversely, at either rank, under no experimental setups does Sp outperform Mp.

The reason that the recognition performance of Sr and M2, and their model-based counterparts Sp and Mp are as close as they are may have to do with their construction. The algorithm that produced Sp is not a general motion segmentation algorithm. It relies heavily upon certain assumptions about the data. For instance, the algorithm assumes that only one moving object will be present within the field of view of the camera. It is also constructed under the assumption that the one moving object is a pedestrian. For instance, the algorithm assumes that the height of the bounding box surrounding the moving object will not change significantly. These limitations, while yielding good silhouettes in this experimental setup, would not lend itself to other, more general, scenarios.

Another reason may be that the potentials chosen for this experiment (i.e. those
Figure 5-20: At rank 1, the probe sequence taken from a particular individual must best match that same person’s gallery sequence to be considered a correct identification. Fitting the pedestrian model to $M_2$ yields a significant improvement in recognition rate and makes it the clear leader for at least the more challenging probe sets.

associated with $M_2$) may not be optimal. In [8], Lee, et al. suggest that recognition performance is reduced when dilated silhouettes are used. Although dilated silhouettes certainly “look” better, clearly there is information pertinent to gait recognition that is lost. Yet, the potentials for $M_2$ were chosen so that segmentation errors were biased towards false-positives. These were chosen precisely because the results “look” better. Furthermore, they look better because they tend to dilate the silhouettes. The dilation is more intelligent than a simple kernel-based graphics operation, however, and one may draw the conclusion that the loss of pertinent gait information would be reduced or minimized. But, it is difficult to say with certainty, and ongoing experiments with different field structures and energy potentials are currently being studied.
Figure 5-21: At rank 5, the similarity of a probe sequence taken from a particular individual and that person's corresponding gallery sequence must be within the top five similarities, when that probe sequence is compared to all gallery sequences. Under this recognition metric, $M_2$ and its model-based counterpart $M_p$ outperform $S_r$ and $S_p$ in several of the experimental setups.

5.9 Algorithm Speed

Perhaps the single largest drawback to the Markov random field approach is its speed. It is markedly slower than either $T_s$ or $T_h$, and $M_3$ is markedly slower than $M_1$ and $M_2$. The reasons why are simple. To segment a particular image frame, $T_s$ requires only a single pass through the pixels of that image. For each pixel, it is compared against its background model and assigned a segmentation label. For $T_h$, a single pass through the pixels of an image are necessary to produce the segmentations based on the thresholds $\tau_b$ and $\tau_f$. Another pass objectifies the segmentation produced by $\tau_b$, and one more produces the final segmentation. Since the last two passes through the segmentation data require only binary comparisons, and since there are typically few foreground objects compared to the total number of pixels, they are very quick.

By contrast, $M_1$ and $M_2$ require up to 10 passes through the pixels of an image. Also, much more work is done at each pixel location than is done for $T_s$ and $T_h$. 93
For each pixel, numerous double-precision floating point operations are performed. These include summing the potentials over the neighborhood structures, calculating exponentials, computing local partition functions \( Z_s \), and computing probabilities. For \( M_3 \), the entire image sequence is first segmented using \( M_1 \), so it is doubly slow. A faster segmentation algorithm (such as \( T_h \)) can be used to obtain the background model that \( M_3 \) uses, but speed is less of an issue for batch processes.

Although the fields \( M_i \) are on the order of one-tenth the speed of \( T_s \) or \( T_h \), faster processors are continually becoming available. When Stauffer’s algorithm was first published, implementations of it ran as fast then as implementations of \( M_i \) run now. Although more complex algorithms demand more from today’s processors, the theoretical time complexity of this approach is still linear. Therefore, this approach will be affected by input length and processor speed in a similar way as its predecessors. For example, doubling the processor speed will halve the running time (or allow twice the frame rate), just as with \( T_s \). Because of this, we do not view speed as a fundamental objection to this approach over \( T_s \) and other explicit thresholding techniques.
Chapter 6

Conclusion

6.1 Summary

Motion segmentation — the partitioning of an image into regions based on the motions of the objects within them — is an important field of computer vision research. Since motion segmentation requires only the rawest of unprocessed input data, a lattice of pixels, it is classified as a low-level vision problem. Like many low-level vision processes, the output of typical motion segmentation algorithms — binary maps indicating which pixels correspond to moving objects and which do not — are used extensively by higher-level vision applications. Some vision applications that require segmentations of all moving objects within a camera’s field of view include tracking, gait recognition, flow analysis, and camera correspondence and calibration.

The approach we take to motion segmentation involves the standard background model / background subtraction paradigm. This involves the creation and subsequent maintenance of a particular model of the appearance of the background in a stationary camera’s view of a scene. Each incoming image frame is then subtracted from, or in some way compared to, that background model to determine which pixels correspond to the background model and which do not. Those pixels that do not correspond to the background model are segmented as foreground and comprise the motion segmentation of a particular image frame.

This thesis introduces a new, more sophisticated technique for performing the com-
parison between image frame and background model. Unlike traditional background subtraction techniques, this approach takes advantage of the spatial and temporal cohesiveness that objects in motion impose on their images.

To exploit the spatial and temporal dependencies, a Markov random field consisting of a lattice of binary random variables was developed. Each random variable of the MRF corresponds to the segmentation of a particular pixel in an image frame. The lattice structure of the MRF is connected by a second-order local neighborhood structure. This neighborhood structure consists of connections from a variate to its eight nearest neighbors.

Three different variations of field structures were used. The first consists of the field structure described above. This structure exploits spatial, but not temporal, dependencies within an image sequence. For the second field structure, each variate is also connected to its segmentation for the previous frame. This arrangement gives more information with which to segment the current image frame. The new information available manifests as a form of temporal dependence between the current segmentation and the previous segmentation of an image sequence. The third and final form used incorporates links between a variate and itself in the past and the future. This is the only field structure that uses a batch processing model rather than an online model. This is also the only field structure to optimize temporally as well as spatially.

The performance of the three fields described above were compared against two traditional motion segmentation algorithms using several image sequences. The results show that all three MRF approaches produced superior segmentations compared to the traditional approaches. For the MRFs, noise (false-positive foreground segmentation) was nearly always eliminated and camouflaging effects (false-negative background segmentation), which manifest as a reduction in the quality of silhouettes, were significantly reduced.

The comparative quality of the three field structures used was varied. Results indicate that, overall, the third field structure yielded the most accurate segmentations. Since this is a batch process, it is not surprising. But, since the difference was slight,
it gives hope that an optimal solution can also be an online solution.

The second field structure tended to overemphasize, or dilate, the foreground segmentations. While this is certainly influenced by the potentials we chose to encode the relationships between variates in space and time (we chose the potentials to bias toward false-positives), it is also a property of the field itself. This manifests most noticeably between the legs of pedestrians, as the leg positions change frequently, and this field structure will attempt to segment according to where the legs were last. However, this field structure is by far the best at handling cases in which an object enters a camouflaging region for a prolonged period of time.

The first field structure’s primary advantages are its simplicity and relative segmentation performance. Since this field is, in effect, a degenerate case of the other two fields, its performance should strictly be worse. But, the comparative difference in segmentation quality between all three structures is small, especially compared to the differences observed between them and the traditional approaches. Also, because this field structure does not take advantage of temporal dependencies, it is easier to parallelize, which may be important from a practical or usability standpoint.

6.2 Future Work

Instantly recognizable as a method to improve on this approach is to allow for the field itself to learn its own structure. We have explicitly chosen the encoding of the relationship between neighboring variates within the MRF. Furthermore, the encodings we chose are constant with respect to orientation and position within the field. This is, of course, suboptimal under most circumstances. In the real world, moving objects are not usually prevalent anywhere and everywhere. Just as objects do not tend to teleport around or disperse in the wind, moving objects do not usually appear anywhere: cars drive on roads, pedestrians walk on sidewalks\(^1\), and delivery trucks back in to loading docks. Within a stationary camera’s field of view, these locations can be learned by adapting the strengths of the bonds within the MRF.

\(^1\)except, perhaps, in Boston
Likewise, the field can also learn orientation. For example, cars tend to be box-like in structure. Therefore, while the field is learning that foreground objects tend to follow roads, it may also learn to regard the first-order (up-down, left-right) neighborhood structure more strongly than the diagonal connections in that same region.

Another area that could be improved upon is the background model. While the multidimensional mixture of gaussians background model is more sophisticated and robust than its peers, it is far from a true representation of the background. While observing its update procedure as moving objects mill about, you instantly realize that this background model has no concept of real objects. Its only objects are pixels, and their only properties are average color and variance. Giving the background model a sense of structure may help reduce the gap between backgrounds and their representations. Allowing the background to model edges as well as pixel processes may provide a more accurate, more robust representation with which to compare incoming image frames. There is much evidence to suggest that edges are a fundamental part of biological vision systems. Including them in a model of the background is a straightforward way to move away from artificial, pixel-based representations to representations that correspond more closely to real objects.
Appendix A

Computational Factorization

Begin with equation 3.7:

\[ P(X_s = x_s \mid X_r = x_r, \ s \neq r) = \frac{\pi(\omega)}{\sum_{\alpha \in \Lambda} \pi(\omega^\alpha)}. \]

This expansion is simply an application of the definition of conditional independence, with the denominator \( P(X_r = x_r, \ r \neq s) \) marginalized over all values of \( X_s = \alpha, \ \alpha \in \Lambda \). Recall that \( \omega^\alpha \) is the configuration that agrees with \( \omega \) everywhere except possibly at \( s \), where \( X_s = \alpha \). It is expressed in this way so that we can represent the probability of a single \( X_s \in X \) using only the Gibbs probability distribution \( \pi \). If we expand out the probability functions, this reduces to:

\[
P(X_s = x_s \mid X_r = x_r, \ s \neq r) = \frac{\frac{e^{-U(\omega)/T}}{Z} \sum_{\alpha \in \Lambda} \frac{e^{-U(\omega^\alpha)/T}}{Z}}{Z \cdot Z^{-1} \sum_{\alpha \in \Lambda} e^{-U(\omega^\alpha)/T}}
\]

\[
= \frac{e^{-U(\omega)/T}}{Z \cdot Z^{-1} \sum_{\alpha \in \Lambda} e^{-U(\omega^\alpha)/T}}
\]

\[
= \frac{e^{-U(\omega)/T}}{\sum_{\alpha \in \Lambda} e^{-U(\omega^\alpha)/T}}
\]

Already \( Z \) is removed from the computation, which is fortuitous, since calculating its value is almost certainly intractable. We can reduce this equation further by expanding out the energy functions and by expressing them as products of exponentials.
rather than as sums within the exponentials:

\[
\frac{e^{-U(\omega)/T}}{\sum_{\alpha \in \Lambda} e^{-U(\omega^\alpha)/T}} = \frac{e^{-T^{-1} \sum_{\alpha \in \Lambda} V_c(\omega)}}{\sum_{\alpha \in \Lambda} e^{-T^{-1} \sum_{\alpha \in \Lambda} V_c(\omega^\alpha)}} = \frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}}
\]

Recall that the energy potential associated with a particular clique \(c\) depends only on those variables \(X_{s_i}\) for which \(s_i \in c\). We now must make a distinction between those cliques which include \(s\) and those which do not:

\[
\frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}} = \frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}}
\]

Notice that, for all potentials \(V_c\) that do not involve \(X_s\), \(V_c(\omega^s) = V_c(\omega), c : s \notin c\). Now we can make the final reduction:

\[
\frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}} = \frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \prod_{\alpha \in \Lambda} e^{V_c(\omega^s)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^\alpha)} \right)^{-1/T}} = \frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^s)} \right)^{-1/T}}.
\]

Restoring the notation to sums within the exponentials yields the computational factorization of equations 3.8 and 3.9:

\[
P(X_s = x_s \mid X_r = x_r, s \neq r) = \frac{\left( \prod_{\alpha \in \Lambda} e^{V_c(\omega)} \right)^{-1/T}}{\sum_{\alpha \in \Lambda} \left( \prod_{\alpha \in \Lambda} e^{V_c(\omega^s)} \right)^{-1/T}} = \frac{e^{-T^{-1} \sum_{\alpha \in \Lambda} V_c(\omega)}}{\sum_{\alpha \in \Lambda} e^{-T^{-1} \sum_{\alpha \in \Lambda} V_c(\omega^\alpha)}}
\]
And, finally

\[
P(X_s = x_s \mid X_r = x_r, s \neq r) = \frac{\exp\left[-\frac{1}{T} \sum_{c \in \mathcal{c}} V_c(\omega)\right]}{Z_s}
\]

\[
Z_s = \sum_{\alpha \in \Lambda} \exp\left[-\frac{1}{T} \sum_{c \in \mathcal{c}} V_c(\omega^{\alpha})\right]
\]
Appendix B

Hausdorff Distance

The Hausdorff distance metric $H$ is a measure of the similarity — or dissimilarity — between two finite sets $A$ and $B$. For a comprehensive overview of this metric, especially of how it is used in computer vision, we refer the reader to [4]. Formally, the Hausdorff distance metric is

$$H(A, B) = \max(h(A, B), h(B, A))$$  \hspace{1cm} (B.1)

where $h(A, B)$ is the directed Hausdorff distance

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|.$$  \hspace{1cm} (B.2)

The function $h$ finds the greatest distance from a point in $A$ to the entire set $B$. The distance from a point $a \in A$ to the set $B$ is the distance from $a$ to the closest point to $a$ in $B$. To compute the value $h(A, B)$, you need to compute the distances from each $a \in A$ to its nearest point $b \in B$, then take the largest of those minimum distances. This is a directed measure because $h(A, B)$ usually does not equal $h(B, A)$.

A simple example will illustrate this point. Let $A = \{6\}$ and let $B = \{10, 5, 8, 1\}$. In this example, $h(A, B) = 1$, since there is only one element in $A$, and it is closest to $5$ in $B$. Conversely, $h(B, A) = 5$, since the element $1$ in $B$ is of distance $5$ away from $6$ in $A$. 

103
Sometimes the directed measure $h$ is convenient to use, but more often the true, undirected distance measure $H$ is appropriate. To make the distance metric undirected, the max of $h(A, B)$ and $h(B, A)$ is taken in equation B.1 to produce $H(A, B)$.

To use this measure on sets of two-dimensional binary segmentations is straightforward. The sets $A$ and $B$ consist of segmentations derived from various subtraction algorithms. A segmentation from one algorithm comprises $A$ and a segmentation from another algorithm comprises $B$. Each set consists of coordinate pairs of foreground segmented pixels. The distance measure $h(A, B)$ uses the Euclidean distance \[ \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \] for the computation of $||a - b||$ in equation B.2.
Appendix C

MRF C++ Source

void MRFModeler::GibbsSampler()
{
    int row, col;
    int k;

    double T;
    int kIterations = 10;
    double C = 3.0;
    double S = 1.0;
    double E = 0.4;
    double M;

    double p;
    Point s;
    int nRows = this->nRows-1;
    int nCols = this->nCols-1;

    // linear temperature scale from (S)tart temp to (E)nd temp
    M = (E-S)/((double)((kIterations-1)));

    // bootstrap MRF annealing with "best guess" segmentation estimate
    for (row=1; row<nRows; row++)
    {
        for (col=1; col<nCols; col++)
        {
            if (variance[row][col] < matchThreshold)
                foregroundMask[row][col] = 0;
else
    foregroundMask[row][col] = 1;
}

// begin annealing
for (k=0; k<kIterations; k++)
{
    //T = C/log(1+30*(k+1));
    //T = C/(k+1);
    T = M*k+S;
    for (row=1; row<nRows; row++)
    {
        s.y = row;
        for (col=1; col<nCols; col++)
        {
            s.x = col;
            p = rand();
            if (p > Ps(s, T)*RAND_MAX) // sampling from the local characteristics
                foregroundMask[row][col] = !foregroundMask[row][col];
        }
    }
}

// probability of segmentation variable Xs P(Xs=xs | Xr=xr, s!=r)
// the binary value xs is stored in val
double MRFModeler::Ps(Point s, double T)
{
    double z[2];
    unsigned char val = foregroundMask[s.y][s.x];

    foregroundMask[s.y][s.x] = 0;
    z[0] = exp(-SumVc(s)/T);
    foregroundMask[s.y][s.x] = 1;
    z[1] = exp(-SumVc(s)/T);

    foregroundMask[s.y][s.x] = val;
    return z[val]/(z[0]+z[1]);
}
double MRFModeler::SumVc(Point s)
{
    double val;
    Point r;
    int x,y;

    val = Vc(s); // potential from site s to pixel connected to s
    // in case of field structure M2,
    // also the potential associated with time

    // potentials from second-order neighborhood of s
    for (x=-1; x<=1; x++)
    {
        r.x = s.x + x;
        for (y=-1; y<=1; y++)
        {
            r.y = s.y + y;
            val += Vc(s, r);
        }
    }
    val -= Vc(s, s); // s-to-s link not part of neighborhood of s!

    return val;
}

double MRFModeler::Vc(Point s, Point r)
{
    if (foregroundMask[s.y][s.x])
    {
        if (foregroundMask[r.y][r.x])
            return -3;
        else
            return 3;
    }
    else
    {
        if (!foregroundMask[r.y][r.x])
            return -1;
        else
            return 3;
    }
}
double MRFModeler::Vc(Point s)
{
    double val=0;
    
    // time clique for field structure M2...comment out for M1
    if (foregroundMask[s.y][s.x])
    {
        if (prevForeground[s.y][s.x])
            val = -5;
    }
    
    if (!foregroundMask[s.y][s.x])
        return val+variance[s.y][s.x];
    else
    {
        // ln 256^3: uniform distribution
        return val+16.63532333438687426013570914996;
    }
}
Bibliography


