ABSTRACT

Title of dissertation: ROBUST TECHNIQUES
FOR
VISUAL SURVEILLANCE

Son D. Tran, Doctor of Philosophy, 2008

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

The work described here aims at improving the performance of three building blocks of visual surveillance systems: foreground detection, object tracking and event detection.

First, a new background subtraction algorithm is presented for foreground detection. The background model is built with a set of codewords for every pixel. The codeword contains the pixel's principle color and a tangent vector that represents the color variation at that pixel. As the scene illumination changes, a pixel's color is predicted using a linear model of the codeword and the codeword, in turn, is updated using the new observation. We carried out a number of experiments on sequences that have extensive lighting change and compare with previously developed algorithms.

Second, we describe a multi-resolution tracking framework developed with efficiency and robustness in mind. Efficiency is achieved by processing low resolution data whenever possible. Robustness results from multiple level coarse-to-fine search-

ing in the tracking state space. We combine sequential filtering both in time and resolution levels int a probabilistic framework. A color blob tracker is implemented and the tracking results are evaluated in a number of experiments.

Third, we present a tracking algorithm based on motion analysis of regional affine invariant image features. The tracked object is represented with a probabilistic occupancy map. Using this map as support, regional features are detected and matched across frames. The motion of pixels is then established based on the feature motion. The object occupancy map is in turn updated according to the pixel motion consistency. We describe experiments to measure the sensitivity of our approach to inaccuracy in initialization, and compare it with other approaches.

Fourth, we address the problem of visual event recognition in surveillance where noise and missing observations are serious problems. Common sense domain knowledge is exploited to overcome them. The knowledge is represented as first-order logic production rules with associated weights to indicate their confidence. These rules are used in combination with a relaxed deduction algorithm to construct a network of grounded atoms, the Markov Logic Network. The network is used to perform probabilistic inference for input queries about events of interest. The system's performance is demonstrated on a number of videos from a parking lot domain that contains complex interactions of people and vehicles.

ROBUST TECHNIQUES FOR VISUAL SURVEILLANCE

by

Son Dinh Tran

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor Larry Davis, Chair/Advisor
Professor Rama Chellappa
Professor Sung Lee
Professor David Jacobs
Professor Ramani Duraiswami

Acknowledgments

The following dissertation could not be completed without the help and supports of a lot of people. I owe my deep gratitude to them.

First and foremost I would like to thank my advisor, Professor Larry Davis, for giving me an invaluable opportunity to work on many interesting problems over the past four years. He has been very available to me when guidance and advices are needed. Pulling me out of stuck and introducing new research challenges are among the great helps that I got from him during my graduate study. It has been a pleasure and honor to work with and learn from a leading researcher like him.

I would like to thank Professor Rama Chellappa, Professor David Jacobs, Professor Ramani Duwaiswami and Dr. David Harwood for seminars, meetings and lectures that they have organized. Interacting with them, either in or out of classes, greatly broadens my perspective and motivates me to examine problems at different view points. I would also like to thank Professor Sung Lee for agreeing to serve on my thesis committee and for sparing his invaluable time reviewing the manuscript.

My colleagues and friends have made my graduate study a memorable experience and deserve special thanks: Dr. Vinay Shet, Dr. Vitaladevuni Shiv Naga Prasad, Dr. Bohuyng Han, Dr. Kyungnam Kim, Aniruddha Kembhavi, Ryan Farrel, Zhe Lin, Abhinav Gupta, Vlad Morariu, William Schwartz and many more. Often, exchanging ideas with them leads to revisions and improvements of my work. Many of them help collect experimental videos, review and proofread my paper drafts. They have been very kind to make their research methods, many of which

mine are based on, available and ready to use.

I owe my deepest thanks to my family - my father, mother, brother and sisters who have always stood by me, giving me the persistent strength through the ups and downs in life. My wife, ThuyDzung has been greatly supporting me with her love and patient. I dedicate this thesis to all of you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Visual Surveillance

Visual surveillance is a major research area in computer vision with many practical applications such as monitoring passenger movement patterns in airports, traffic conditions on highways, or security in subway stations. A surveillance system often consists of three levels: low level component usually deals with object detection and classification, mid-level component tracks the movement of objects through space. The extracted object tracks or scene states are then provided to the high level components for analysis and interpretation. Below is an overview of my contributions to these components.

1.2 Background Subtraction

Using background subtraction to detect foreground objects involves building a background model from a training sequence (often using an empty scene). New frames are then compared and "subtracted" from the background model, so that what remains is the foreground. The background model consists of a description of the color distributions for each pixel in the image. Usually, each distribution is modelled with a Gaussian, mixture of Gaussians ([79]) or a set of codewords as

in VQ approaches ([40]). One of the main challenges in background modelling is the adaptation of these distributions to lighting changes. My work considered the responses of the camera and object surfaces to incident lighting in designing a new background model that adapts better to color variation, and thereby is able to deal with extensive illumination changes.

1.3 Object Tracking

Tracking algorithms determine the state (location) of objects across frames. The typical implicit goal is given a video sequence I_t to estimate the states X_t of the tracked object as accurately as possible for all t. However, there are situations where other criteria might be relevant. For example, when tracking people in surveillance scenarios, we might require more accuracy only when people approach one another (for example, to determine if they make any interaction). In such situations, we could track at high resolution when it is needed, while maintaining track at lower spatial resolution at other times. My work proposed a coherent probabilistic framework that allows a tracker to operate at multiple levels of spatial resolution to achieve efficiency and robustness.

Traditionally, an adaptive tracking system consists of two major components: object representation and a searching algorithm. When the object moves in the scene, its appearance usually changes due to the variation in object pose or lighting. It is important that the object representation adapts to these changes so that object state can be determined accurately. Many adaptive modelling approaches have been

proposed in the literature ranging from modelling object appearance with mixtures of Gaussians ([30], [92]) to learning the complete object appearance manifolds ([21]). Even with the best modelling technique, it is still difficult to accurately determine the state of the object. Therefore, besides deterministic searching procedures such as mean-shift, probabilistic approaches such as Kalman filter or particle filters are also popular in tracking. Despite its long history, robust object tracking is still a challenge to computer vision. In my work, instead of directly modelling the object appearance, I represent the object with an occupancy map, and reduce the tracking problem to establishing object occupancy maps across frames. The occupancy map construction is carried out based on feature and pixel motion analysis.

1.4 Event Modelling and Recognition

At the highest level, we would like our surveillance system to "understand" interesting interactions that occur in a scene. This involves a number of steps including identifying what are the events of interest, how to model them, and how to recognize them from the video observations. Two of the main challenges are, first, to recognize events that are complex and, second, to do so in the presence of noise.

There are numerous frameworks that have been proposed for event recognition. In declarative approaches (e.g. [66]), events are represented with declarative templates. Events are typically organized in a hierarchy, starting with individual primitive events at the bottom and composite events on top. The recognition of

a composite event proceeds in a bottom-up manner. While events with complex structures can be captured using these approaches, uncertainty is often not modelled and so they are generally not robust to noise. In probabilistic frameworks, such as HMMs (e.g. [57]), events are represented with probabilistic models. Event recognition is usually performed using maximum likelihood estimation given observation sequences. Probabilistic approaches are often robust to noise. However, their representations often lack flexibility (e.g. number of states or actors needs to be known in advance) and hence it is difficult to use them in dynamic scenes.

My work used a combination of logical and probabilistic reasoning and utilized commonsense knowledge to overcome noise or gaps in observations. Logic provides us the compositional power to describe many complex relationships or events, while probability provides us methods to deal with uncertainty.

Chapter 2

Adaptive Background Subtraction with Prediction Models

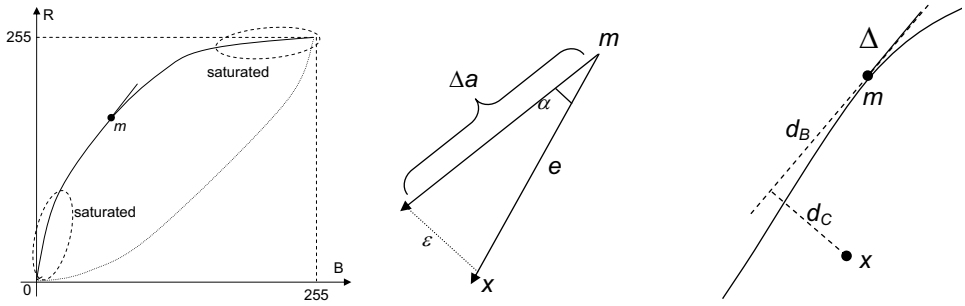
2.1 Introduction and Related Works

Background subtraction is a key component in visual surveillance. It offers a simple and often effective approach to detect foreground objects. In background subtraction, the foreground is extracted from the current video frame through comparison with a previously constructed background model. Common approaches to background modeling construct a probability distribution for a pixel's color values or for some local image features. For simple situations with little noise and change in lighting, a single-mode distribution such as a Gaussian is often sufficient. For more complex settings with high levels of noise or with periodic background motion (e.g. from tree leaves blowing in the wind) a multi-modal distribution is required. Here, a mixture of Gaussian (MOG) ([79]) is typically employed. Kim ([40]) introduced a computationally simpler, but comparably effective, approach based on vector quantization of color values observed during training of the background model. Dealing with illumination changes is a challenge for these methods, especially if fast response to illumination changes is required. For example, in the MOG approach, data is added to the mixture model and old data is removed using a time-varying weight. However, this inevitably leads to lag in accommodating to illumination change during which the system is effectively "blind." Adaptation using

a) frame♯1 b) frame♯394

Figure 2.1: Two frames of an outdoor sequence with large changes in lighting when under direct sunlight and when a cloud passes by.

a Kalman filter approach such as [39] also has similar difficulties, notably the lag problem. Additionally, these methods can also absorb stationary foreground objects into the background model. In the vector quantization approach, the codewords constructed by the VQ algorithm represent regions in color space characterizing the background distribution. [40] used simple pixel-wise autoregressive models to shift these codewords towards or away from the origin of color space as illumination changes; however, as we will see, this is based on a physically incorrect model of how background color changes as illumination varies for real cameras. Here, we employ the VQ approach to background modeling, and motivate the use of a locally linear model to predict and update pixel color codebook values. Our approach is able to deal with large illumination changes using only a small number of codewords per pixel. Training can be done even while foreground objects are moving in the scene. A full range of illumination conditions is not needed during training.



a) b)

Figure 2.2: a) Illustration on R-B plane of the response of the camera sensor to changes in lighting intensity. b) Prediction errors and distances in brightness d_C and d_B in color

2.2 Background Modeling using the Codebook Representation

There are two main observations that motivate our approach. First,in natural outdoor scenes, different surfaces have different irradiant responses when the illumination is changed. To accurately adapt the background model to such changes, a single global set of adaptive parameters, such as suggested by [42], is insufficient. Instead, for each pixel we use a separate set of parameter values that are updated independently as lighting changes. Second, as lighting changes, the color values of individual pixelsdo not lie on a color line passing through the origin of color space, as assumed by [40] and others. Since camera sensors have limited range [0-255], the response is necessarily saturated at the limits so that locally, changes in color as a function of illumination change will not point toward the color space origin. This is illustrated in Figure 2.2. At low light levels all pixels fade to black. At sufficiently high light levels they all tend to white. Between those two extremes they follow a

curved trajectory between black and white depending on the reflectance properties of the corresponding surface point. We point out that reaching saturation is quite common, especially for scenes that are under direct and strong sunlight (Figure 2.1). Models for background adaptation under illumination change have to take a typical camera's non-linear responses into account.

We construct a locally linear model to update codeword colors as illumination changes. So, at each pixel, we use a tangent vector to that pixel's color trajectory to represent the rate and direction of its color variation. As the lighting changes, both are adjusted to better model the local behavior of a pixel's color.

The overview of our background model adaptation is as follows.

- For each pixel, a codeword model is learned during a training period. Each code word contains a principle color values and a tangent vector .
- When scene illumination changes, the change is measured through a gain in a global index value. This gain, in combination with the mean and tangent of a codeword, produces a prediction of the code word color under the changing lighting condition.

With some additional updating based on secondary technical considerations, these predictions constitute our updated background model. Further details are given below.

2.2.1 Background Model Construction

Due to possible variations in the background during training, the color distribution at a background pixel may consist of several clusters. We use a codeword for each cluster. The collection of these code words forms our codebook (CB) background model.

Similar to [40], we represent a codeword w with an 8- tuple $(m, \Delta, \sigma_B, \sigma_C, f, mnrl, t_{first}, t_{las}$ where

- m is the mean color of the cluster,
- Δ is the rate of (color) change at m when there is a unit change in the global index ($\Delta > 0$),
- σ_B and σ_C are the codeword variances in brightness and color respectively,
- f is the occurrence frequency of the current codeword,
- $mnrl$ is the maximum negative running length, which is the maximum period during the training period over which the code word was not observed
- t_{first} and t_{last} respectively are the first access time and the last access time to the code word.

The variable $mnrl$ is used for temporal filtering to eliminate foreground code words during training.

Let $I_t \leftarrow (R_t + G_t + B_t)/3$, and let $CB(x)$ be the set of code words at a pixel. The algorithm for updating the codebook is presented in Figure 2.2.1. Details are explained below.

Initialization $CB \leftarrow \emptyset$,

Repeat Step 1 and 2 till the end of the training sequence, $t = 1 \rightarrow N$

Step 1. Compute the gain a in the global index using I_t and I_{t-1}

Step 2. For every pixel $x = (r, g, b)$,

1. Predict the color of every code word w in $CB(x)$,

$$m = mp + a\Delta \tag{2.1}$$

2. Get the closest codeword to x, $w*$

3. Compute the distance D between x and $w*$

4. If $D < \theta$, update $w*$ using x (eqn. 2.8 and 2.9)

$$w* \leftarrow (m', \Delta', \sigma'_B, \sigma'_C, f + 1, max(t - t_{last}, mnrl), t_{first}, t) \tag{2.2}$$

Else create a new codeword and add to $CB(x)$

$$w \leftarrow (x, \Delta_0, \sigma^0_B, \sigma^0_C, 1, t - 1, t, t) \tag{2.3}$$

5. Update the remaining codewords in $CB(x)$

$$m' = \beta a\Delta + m \tag{2.4}$$

Wrap up the maximum negative running length

$$mnrl = N - t_{last} + t_{first} - 1 \tag{2.5}$$

Perform temporal filtering to eliminate foreground code words.

Figure 2.3: The Codebook Construction Algorithm

In the absence of prior knowledge about the locations of foreground objects, there is inherent uncertainty in how a given background pixel varies in color. However, under the assumptions that the foreground covers only a small fraction of the image and that there is a single source of lighting, such as the sun, we can employ a global gain measurement to infer local lighting variation. Here, we use the difference of medians of brightness energy over time; i.e. the gain, $a = median(I_t) - median(I_{t-1})$. To account for the possibility that the foreground size is significant, pixels with very high frame difference are excluded from the median computation.

Given this global gain, every code word is shifted to a new location using the previously learned mean and tangent, Δ, using linear prediction; i.e. $mp = m + a\Delta$. If a pixel observed color, x, is close to the predicted mean mp of the codeword w, then x is used to update w; otherwise, it is used to initialize a new code word. The distance between a code word mean, m, and a new color measurement, x, is measured as the sum of distances in two directions - one along and the other perpendicular to the local tangent direction Δ,

$$d^2(x, m) = (d_B/\sigma_B)^2 + (d_C/\sigma_C)^2 \tag{2.6}$$

Here $d_B = \Delta/\|\Delta\|(x-m)$ is the projection of $x-m$ on the current tangent direction, and represents the deviation in brightness between x and the codeword. d_C is measured in the direction from x to the tangent - it represents the deviation in color. See Figure 2.2.b for an illustration. σ_B and σ_C are the variances in brightness and color of the current code word.

When a new code word is created, its mean is the current observation, x. The

direction of the tangent vector Δ_0 is initialized toward the origin. Specifically,

$$\Delta_0 = \frac{1}{2}\sigma_B^0 \frac{x}{\|x\|} \tag{2.7}$$

When the distance between a code word, w, and the current observation, x, is less than a threshold θ, w is updated as,

$$\Delta' = \beta_\Delta \cos\alpha(x - m)/a + (1 - \beta_\Delta)\Delta \tag{2.8}$$

$$m' = \beta_m \sin\alpha\varepsilon + mp \tag{2.9}$$

where α is the angle between $mp-m$ and $x-m$; ε is the prediction error, $\varepsilon = x-mp$; β_Δ and β_m are constants controlling the adaptation rates of Δ and m. The mean and the tangent involve six parameters that need to be updated; however, x provides only three observation values. The extra degrees of freedom means that there is flexibility in distributing the adaptation to the mean and the tangent Δ. When the angle, α, is large we mainly update the mean, while when α is small, the adaptation is shifted to changing Δ. This is to prevent the direction of Δ from fluctuating due to small measurement noise. Hence, $\sin\alpha$ and $\cos\alpha$ are selected as the weights in equations 2.8 and 2.9.

σ_B and σ_C are updated as $\sigma'_B = (f\sigma_B+d_B)/(f+1)$ and $\sigma'_C = (f\sigma_C+d_C)/(f+1)$

For all other code words associated with a pixel, the mean is shifted toward the predicted value. The tangents Δ are rotated toward the origin. This approach is ad-hoc, since in this case, the information to update un-accessed codewords (i.e. codewords that don't have observations) is not available. Possible improvements with complete modeling of the color variation curve are discussed in chapter 4.

2.2.2 Foreground Detection

Scene illumination will, of course, also change after training the codebook is completed. Therefore, we keep updating the background model in parallel with foreground detection. The algorithm is the same as in Figure 2.2.1 with the exception that in step 2.2.14, when $D > \theta$, instead of creating a new code word, we mark that pixel as belonging to the foreground. It is important to note that since we keep adjusting the background model up to date, it would be more accurate if the lighting condition at the start of the detection period is close to that at the end of the learning period. This requirement can be relaxed if we experienced all lighting conditions during learning (which we don't assume here), and build complete variation trajectories in the color space for every pixel.

2.3 Results

In this section we compare the performance of our algorithm with the approaches described in [40] and in [39]. [40] also used a vector quantization approach and codebook representation, but did not use the local linear model and index-based prediction as we do here. To deal with illumination change, they used a cache of codewords and a codeword is considered to belong to the background if it is accessed continuously for a certain long period of time. [39] adapted the background model with a Kalman filter style update (first order autoregressive model).

In the three approaches considered here, as well as in many other approaches in background subtraction, a distance threshold, r_D, is used to reject or accept if a

pixel belongs to the background (foreground) or not. The value of this parameter has a direct effect on the detection performance of each method. To make the three methods comparable, we used the same value of r_D for all of them, both during training and detection. Extensive evaluation of the sensitivity of the detection rate to the distance thresholds is an ongoing part of this work.

The first experiment is on an outdoor sequence where the lighting changed extensively during both the training and detection periods. The first 150 frames are used for training. No foreground objects are present during that period. Figure 2.4 shows the results on three frames from the testing sequence. As we can see, with the same distance threshold, both of the approaches in [40] and [39] produce more false alarms than ours. Note that reducing r_D to exclude false alarms also leads to a reduction in detection rate (i.e. producing more holes in the foreground regions). For example, in [40]'s result for frame ♯400, the foreground (true positive) starts to disappear while the background (false alarms) has not been completely eliminated yet. Increasing the adaptation rate also does not necessarily lead to improvements in detection. Figure 2.5 shows the result of such adjustment for [39] and [40] on frame ♯400.

Next, we test our algorithm on an indoor sequence. The lighting from a single white light source is controlled with a dimmer. The variation of lighting intensity is gradual but arbitrary. The first 100 frames are used for training. The scene is empty during the training period. The training frames cover the major range of lighting variation, but not all. Figure 2.6 show the testing results at frame ♯101, ♯200 , and ♯300.

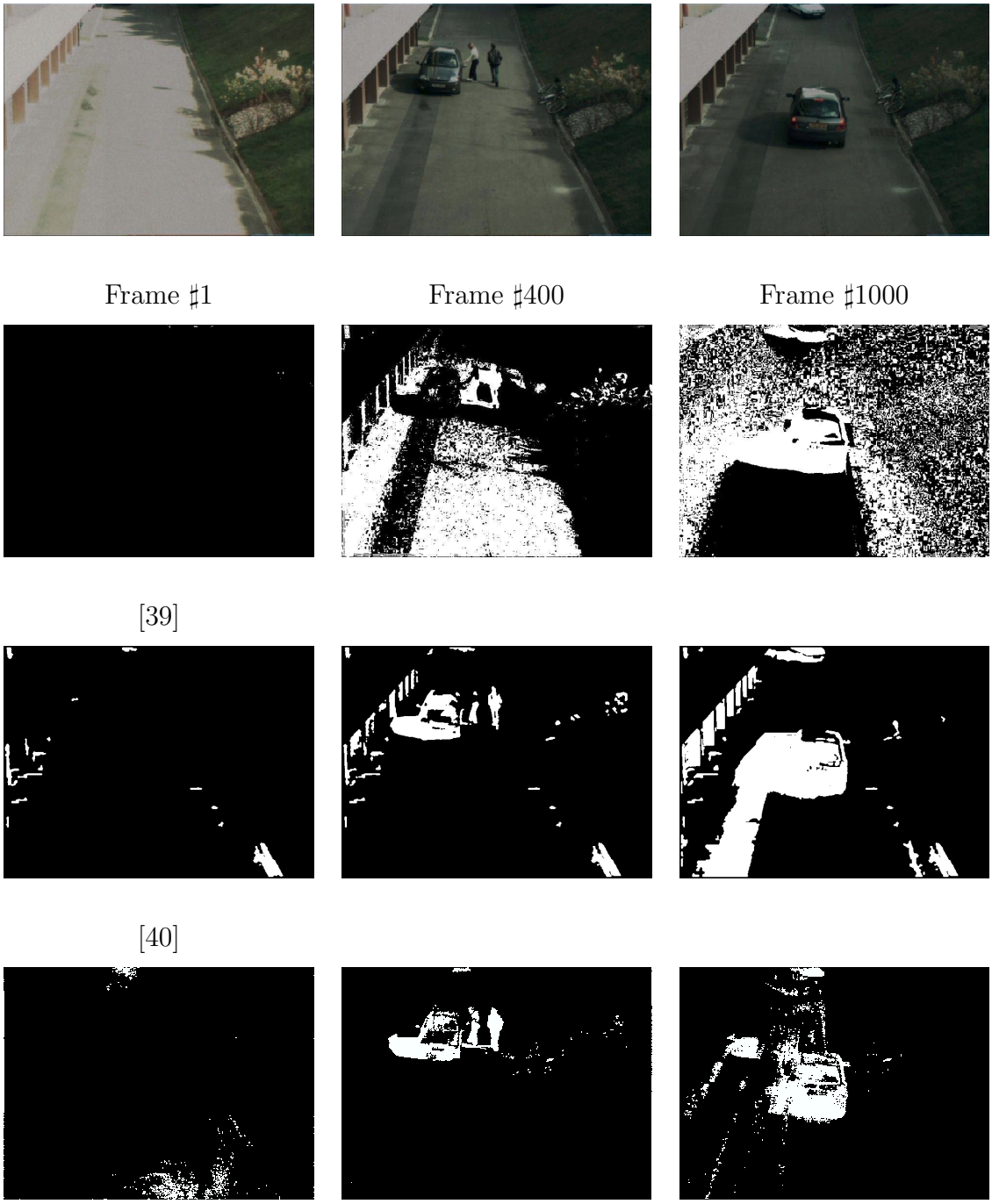


Figure 2.4: The first row shows three frames of the testing sequence. The remaining rows respectively show the result of [39], [40] and our methods on those three frames.

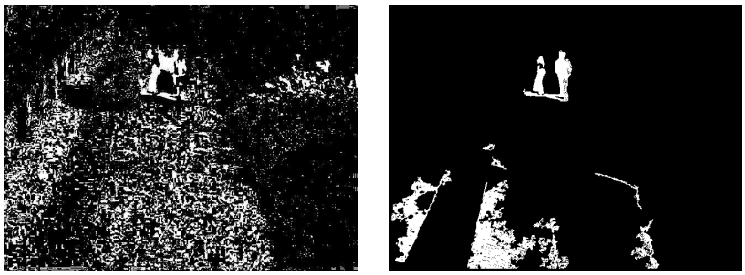


Figure 2.5: Increasing the adaptation rate for [39] (left) and [40] (right) does not necessary improve their results.

Figure 2.6: Results on the indoor sequence. The first row shows three frames of the training sequence. The second row shows three frames of the testing sequence. The remaining rows show the results of [39], [40], and our methods at these three frames.

Chapter 3

Object Tracking at Multiple Levels of Spatial Resolutions

3.1 Introduction

Tracking is a problem that has been extensively studied in computer vision. The typical implicit goal is given a video sequence I_t to estimate the states X_t of the tracked object as accurately as possible for all t. However, there are situations where other criteria might be relevant - for example, when tracking people in surveillance scenarios, we might require more accuracy only when people approach one another (for example to determine if they come close enough to interact). In such situations, we could track at high resolution when it is needed, while maintaining track at lower spatial resolution at other times.

Tracking at low resolution is less expensive computationally and, in some sense, less sensitive to noise. At the same time, high resolution tracking is needed for accurate estimation of object state. Therefore, a method is needed for moving back and forth between resolution levels in both the data and the tracking state space, while maintaining tracking over time. The crucial issue, then, is how to make inferences across different levels of the tracking state space so that the tracking process is continuous. Here, probabilistic modeling and sequential Bayesian filtering are used to solve this problem, providing a means to choose suitable levels of resolution as the tracking progresses in time.

3.1.1 Related Works

We review here work that is most relevant to our approach. Coarse-to-fine searching, or hierarchical searching, is a common method to deal with large search spaces and has been widely applied in computer vision (see e.g. [11], [1]). The exploration of the state space can be attacked from two perspectives: searching and modeling. In [19], at a given time, the state space of the object is searched by propagating particles through multiple layers of simulated annealing. They do not sub-sample the original data and so the amount of computation is nearly the same for all layers. It is well-known that in tracking, especially tracking of large objects, computing particle likelihoods is often the computational bottleneck of the tracking system ([64], [82]). One of our goals is to track efficiently through reducing this burden as much as possible.

With layered sampling in [82], the state space is explored by decomposing the state variable X_t into the sum of several others, X_t^r , $r = 1, 2 \ldots$. Each X_t^r is responsible for sampling at a spatial range r. Coarser pdf's are used to guide the sampling of finer ones through importance sampling. It is implicitly assumed that the decomposed variables are independent, and thus there is no modeling of the dependence between X_t^r's. In our approach, we explicitly model the dependence of an object's state at different resolution levels s and s'.

In addition to stochastic searching, hierarchically pruning of the search space can be done through the use of explicit models. This typically leads to the construction of a hierarchy of state models. In [27], a hierarchy of templates was defined

for the problem of recognition through template matching. The state space was then explored by traversing this tree structure of templates. In a similar vein, but in the context of tracking, [80] defined a fixed tree-based structure whose different levels partition the state space at different levels of granularity. Bayesian filtering (in time) is applied to each level of the tree. The state space is pruned for the low level nodes based on the probability density of their (coarser) parent partitions. In contrast to our approach, the interaction between levels of the tree is not modeled with Bayesian filtering and the observation is not multi-resolution.

In [31] a dynamic Bayesian network with a nonparametric propagation scheme is used for multi-scale object tracking. They also sub-sample the observation to multiple scales, but the number of scales is equal to the number of layers of the network, which is fixed (three). In contrast, the number of scales in our algorithm is dynamic, and in theory can have an arbitrary range.

The chapter is organized as follows. The next section establishes the basis for Bayesian filtering along the spatial and temporal dimensions. Section 3.3 describes different components of the tracker and strategies for controlling it. Section 3.4 shows experimental results of the tracker on a number of video sequences. Section 3.5 concludes the chapter with a discussion.

3.2 Sequential Bayesian Filtering in Spatial and Temporal Dimensions

In this section, we first describe temporal Bayesian sequential filtering and then its extension to include sequential propagation across scales. Some important differences of our approach compared to previous ones are pointed out. The motivation and strategy for adapting the tracker's scale are briefly introduced at the end of the section.

3.2.1 Filtering in Time

Consider a system with the following state and observation models,

$$X_{t+1} = f_t(X_t) + W_t \tag{3.1}$$

$$Z_t = h_t(X_t) + V_t \tag{3.2}$$

where W_t and V_t are respectively the state and measurement noise models at time t (typically, $W_t \sim N(0, Q_t), V_t \sim N(0, R_t)$) and $f_t(X_t)$ and $h_t(X_t)$ are state and observation models. Given the observation sequence $Z_{1:t}$, the state X_t (position, orientation, scale...) of the object at time t is characterized by a pdf $p(X_t|Z_{1:t})$ and is estimated from the prior density distribution, $p(X_{t-1}|Z_{1:t-1})$, using the two steps-prediction and updating (Chapman-Kolmogorov equation, see e.g. [20]),

$$p(X_t|Z_{1:t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Z_{1:t-1})dX_{t-1} \tag{3.3}$$

$$p(X_t|Z_{1:t}) \sim p(Z_t|X_t)p(X_t|Z_{1:t-1}) \tag{3.4}$$

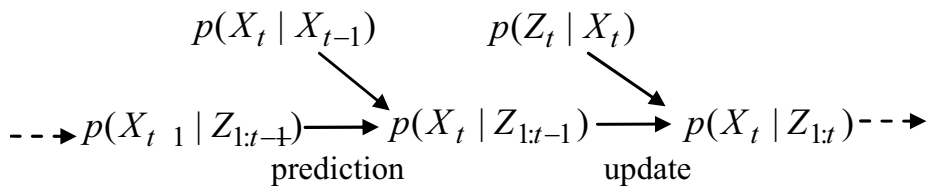


Figure 3.1: Sequential temporal filtering

where $p(X_t|X_{t-1})$ and $p(Z_t|X_t)$ are respectively the assumed state and observation likelihood models. This two-step propagation can be illustrated with the diagram in Figure 3.1

This process is iterative, starting from the assumed prior $p(X_0)$. When the system is linear, an analytic solution can be derived such as in Kalman filtering. In the general case, approximation using Gaussian mixtures ([29]) or a weighted particle set ([33]) is often used.

3.2.2 Filtering across Scales

Now, we extend the above formulation when the state and observation are measured at multiple levels of resolution. Our system now can be written as,

$$X_{t'}^{s'} = f_t^s(X_t^s) + W_t^s \tag{3.5}$$

$$Z_t^s = h_t^s(X_t^s) + V_t^s \tag{3.6}$$

where the superscript s and s' are used to denote resolution levels, typically, $s = s', s' \pm 1$ and$t = t', t' + 1$.. Filtering across scales can be done in a similar manner as filtering in time (3.3), (3.4). For example, to get $p(X_t^s|Z_{1:t}^{1:s})$ from $p(X_t^{s-1}|Z_{1:t}^{1:s-1})$

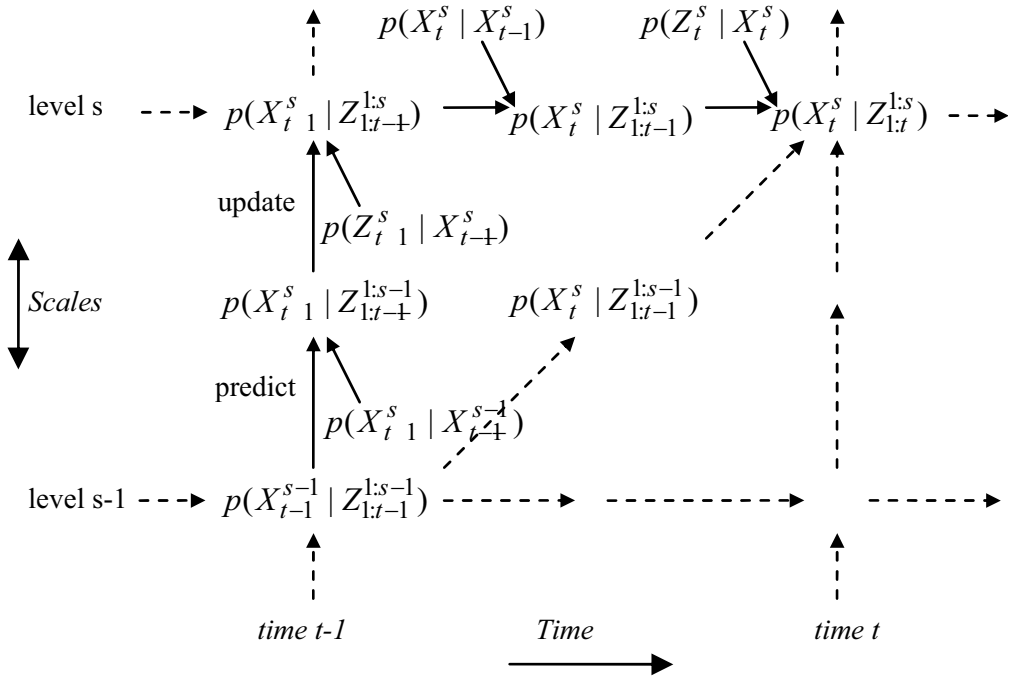


Figure 3.2: Illustration of upward propagations from $(t-s, s-1)$ to (t, s)

given Z_t^s, we proceed as follows

$$p(X_t^s|Z_{1:t}^{1:s-1}) = \int p(X_t^s|X_t^{s-1})p(X_t^{s-1}|Z_{1:t}^{1:s-1})dX_t^{s-1} \tag{3.7}$$

$$p(X_t^s|Z_{1:t}^{1:s}) \sim p(Z_t^s|X_t^s)p(X_t^s|Z_{1:t}^{1:s-1}) \tag{3.8}$$

where $p(X_t^s|X_t^{s-1})$ is modeled based on the transformation of the state variables as the scale is changed - typically a simple re-scaling of the state variables by the relative scales, with the addition of some white noise (i.e. $X_t^{s'} = f(s,s')X_t^s + U$ where $U \sim N(0,\Sigma)$ and $f(s,s')$ measures the relative scaling between two level s and s'.

Other movements along the two dimensions, temporal and spatial resolution, can be performed similarly. The diagram in Figure 3.2 provides an illustration of the propagation of probabilities across resolutions levels. In particular, it shows paths

from $(t-1, s-1)$ to $(t-1, s)$ and from $(t-1, s-1)$ to (t, s). The latter case is a diagonally upward move and can be done when the model $p(X_t^s|X_{t-1}^{s-1})$ is assumed for $X_t^s|X_{t-a}^{s-1}$. In the opposite direction, when we move downward in scales from s to $s' < s$, the state update can be determined not by probabilistic filtering but by deterministic transformations such as $X^{s'} = f(s, s')X^s$. The reason is that since the data at higher resolution s is assumed to be more accurate than the data at a lower level, the state estimation at higher level is often more accurate. Therefore, a probabilistic update step with the less accurate observation at scale $s' < s$ is not needed. A simple projection operation is sufficient.

Here, we try to estimate the state X_t of the object at a specific scale s at a given time t. This is different from other multi-scale attempts such as [31], [80] or [3] where the probability distribution of the state X_t at a time t is estimated for all scales. Several disadvantages for maintaining multiple layers in parallel are as follows. First, the concurrent interaction between layers is often complicated and simplifying assumptions needs to be made. For example, in [31], the dynamic models corresponding to different scales are assumed to be independent. In other words, object motions at different scales are assumed unrelated. Second, the number of scales is often limited and fixed at the onset (up to three, [31] and [3]). This assumption is made principally due to concerns over computational costs. As a results, when there are only few layers, the relative scale between two consecutive layers $s+1$ and s has to be large (e.g. at least two) and therefore it becomes harder to model the relation between X^s ad X^{s+1} accurately. Whereas in our approach, the number of scales can be very large and the relative scale between two consecutive

levels of resolution can be small (e.g. 1.2).

3.2.3 Scale Adaptation

The filtering framework described above supports the ability to control the tracker to move up and down in spatial resolution and in time. We discussed here several principles for controlling scale adaptation. Adaptation strategies for our particular tracker are detailed in Section 3.3.3.

First, our goal here is to track efficiently, therefore we would like to track the object at the lowest scale possible while still maintaining robustness. In [14] robustness is achieved by searching for the most discriminative feature from a pool of features. Here we increase the discriminative power of the tracker by increasing the tracking resolution. The underlying assumption is that when we have a more detailed observation, we are able to make a better estimate of the state of the object and hence can track the object more robustly.

Second, in theory, at a given point (t, s), we can move to any other scale at time t or $t+1$. However, large jumps often lead to poor sequential estimates. Therefore, we change the resolution gradually when we move upward. As mentioned in the previous section, in the downward direction we can make large jumps, e.g. from (t, s) to $(t, s'), s' << s$, without affecting the state estimation quality. However, practically, we also would like to move downward gradually so that the tracking conditions can be monitored continuously.

Third, there are cases when, even at the highest resolution, we still don't

have enough information to distinguish the object from its surrounding cluttered background. Then, we should have a tracker that supports multi-modal tracking so that the tracking can be continued successfully. Here we choose the particle filter for this fail-safe, fall-back case. In particular, if the sample covariance doesn't reduce with increasing scale, our tracker keeps moving up in scale. If, at the highest resolution, the sample covariance is still large, most probably due to multi-modality, the tracker just proceeds with the particle filter tracker at full resolution (Section 3.3). Note that our multi-resolution tracker is not designed to be more accurate than a full resolution tracker, but more efficient.

3.3 The Multi-Resolution Color-Based Tracker

Any probabilistic tracker can be integrated into the framework developed in Section 3.2. We describe one here that tracks color blobs at multiple level of resolutions from s_{min} to s_{max}. It is based on particle filtering ([33], [64]). At any given time instance, t, only one set of N_p particles $(X_t^s(i), w_t^s(i)), i = 1 - N_p$ that samples the object's state space corresponding to the current spatial resolution level s is maintained.

3.3.1 Observation Models

The appearance model we use is similar to [64]. The state variable is $X_t^s = (x_c, y_c, w, l, o)_t^s$, where x_c, y_c, w, l and o respectively denote the blob's center coordinates, its width, length, and orientation. The appearance Z_t^s of the object is

then the N-bin HSV histogram [64] of the pixels in the projected layout $Prj(X_t^s)$, $Z_t^s = Z(I_t^s, Prj(X_t^s))$ where I_t^s is the image at time t and spatial resolution s. We obtain I_t^s by nearest-neighbor sub-sampling of the image at the highest spatial resolution s_{max}. The observation likelihood $p(Z_t^s|X_t^s)$ is then modeled as $p(Z_t^s|X_t^s) \sim \exp(-\lambda D^2(Z_0^{s_{max}}, Z_t^s))$, where λ is a constant, $Z_0^{s_{max}}$ is the (initial) target histogram, and $D(Z_0, Z)$ is the distance based on the Bhattacharyya coefficient ([15]), $D^2(Z_0, Z) = 1 - \sum_{i=1}^{N} \sqrt{p_i(Z)p_i(Z_0)}$, where p_i is the value of the i-th bin in the histograms.

One drawback with the histogram is the loss of spatial information. In [64], the authors divide Z_t^s and $Z_0^{s_{max}}$ into a fixed number,K, of parts and histograms are computed and correspondingly compared for each part. Here, we also divide the tracked object into sub-regions. However, instead of fixing the number of parts, we adapt this number with resolution level, $K = K(s)$, with K being smaller at lower resolution levels. The likelihood is computed as $p(Z_t^s|X_t^s) \sim \exp(-\lambda \sum_{j=1}^{K(s)} D^2(Z_0^{s_{max}}(j), Z_t^s(j)))$, where $Z_0^{s_{max}}(j)$ and $Z_t^s(j)$ are the histograms of the corresponding j-th sub-regions in the target and image region. This likelihood function tends to become more peaked when resolution s increases and is smooth when s decreases. This implies accuracy at high resolution and robustness at low resolution.

3.3.2 State Models

In addition to the state evolution in time, we also need to specify the modeling of state across spatial resolution levels. They are defined as follows.

For $X_t^s|X_{t-1}^s, X_t^s = X_{t-1}^s + W_t$, where $W_t \sim N(0, Q_t)$ is the noise for the state model along the temporal dimension. $Q_t = diag(\sigma t_x^2, \sigma t_y^2, \sigma t_w^2, \sigma t_l^2, \sigma t_o^2)$ and is kept constant for all s and t except when the tracker is resolving a lost track. For $X_t^s|X_t^{s-1}, X_t^s = A^s X_t^{s-1} + W_s$, where is $W_s \sim N(0, Q_s)$ the noise for the state model along the spatial resolution dimension. $Q_s = diag(\sigma s_x^2, \sigma s_y^2, \sigma s_w^2, \sigma s_l^2, \sigma s_o^2)$ is also kept constant for all s and t. $A^s = diag(\beta, \beta, \beta, \beta, 1)$ where $\beta = f(s, s-1)$ is constant. Making β constant has been shown to be a good choice in [82]. Note that we don't change orientation values across resolution levels. For $X_t^s|X_{t-1}^{s-1}, X_t^s = A^s X_{t-1}^{s-1} + W_{s'}$, where $W_{s'} \sim N(0, Q')$. If we consider this diagonally upward movement as a combination of the two previous ones, then $Q' = Q_t + Q_s$. For $X_t^{s-1}|X_t^s, X_t^{s-1} = (A^s)^{-1} X_t^s$, is a deterministic projection. And finally, for $X_t^{s-1}|X_{t-1}^s, X_t^{s-1} = (A^s)^{-1} X_{t-1}^s + W_t^{s-1}$. This diagonally downward move is seen as first a deterministic decrease in spatial resolution and then a stochastic move forward in time. This order may be reversed.

Here the state models are simplified to Gaussian noise processes, i.e. effectively without dynamic or motion models. Modeling the object's motion is only useful when its movement follows the patterns captured in the motion model, and is particularly difficult to model in the case of moving cameras. On the other hand, an obvious disadvantage of the noise model is that, given a fixed number of particles,

the sampling range is usually not large enough for the tracker to follow the tracked object even when its angular velocity is not very large. Fortunately, working at multiple levels of resolution helps in this case. Roughly speaking, if a set of particles at level s_{min} is used to sample a volume $V_{min} = (r_{s_{min}})^d$, where $r_{s_{min}} = 3Q_s^{1/2}$ is the radius and d is the number of dimensions, then through the sequential filtering across multiple scales, we can effectively sample high likelihood portions of a volume $V_{max} = (r_{s_{max}})^d$ where $r_{s_{max}} = f(s_{max}, s_{min})r_{s_{min}}$. In practice, $r_{s_{max}}$ is usually tens of $r_{s_{min}}$. This makes the avoidance of dynamic or motion model a reasonable choice, and the tracker becomes more general with regard to the tracked object's motion patterns.

3.3.3 Scale Adaptation Criteria and Rules

To track the object efficiently, we follow the adaptation schemes laid out in section 3.2.3: track the object at the lowest scale possible, while monitoring to ensure good tracking condition. If the condition starts to deteriorate, the tracker needs to increase resolution to gain more discriminative power. To realize such a scheme, we need a good set of measurements that is indicative of the tracking condition.

Practically, in particle filtering, there are two indicators, the particle likelihood and the sample covariance Σ_t^s , for situations when the observation is not available (due to occlusion or lost track) and when the observation is available but confused (due to distracting objects or background), respectively.

- *Dealing with confusion.* When there is no confusion the likelihood distribution

typically peaks around some mode, and Σ_t^s thus will be small. When there is tracking confusion with another object or features in the background, the likelihood will contain several separated peaks, and Σ_t^s becomes large since samples are spread around these different peaks. Therefore the following stable continuous adjustment rule can be employed to deal with various tracking confusions: *Let* C_{var}^s *be* $||\Sigma_t^s - Q_0^{s_{max}}|| \leq \alpha_{var}$. *When* C_{var}^s *is violated, change the spatial resolution level to* $s' = s+1$*, until it is satisfied.* The threshold α_{var} is established at the initialization of the tracking process with the assumption that there is no confusion at s_{min} at that time, $\alpha_{var} = ||\Sigma_0^{s_{min}} - Q_0^{s_{max}}|| + \epsilon$, where ϵ is a small constant (e.g. $\epsilon \sim 0.001$). Note that when the maximum scale is reached, the tracker simply returns to the full-resolution tracker with all of its strengths (or weaknesses).

- *Dealing with a lost track.* This can be detected using particle likelihoods. When a track is lost, the likelihood values of all particles and hence their maximum $wmax_t^s$ are substantially small. On the other hand, with methods for preventing sample degeneracy applied (see e.g. [20]), when the true state is within the sampling scope, $wmax_t^s$ is quite large. The following control rule can be applied when loss of track occurs: *Let* C_{max}^s *be* $rmax_t^s > \alpha_{max}$. *When* C_{max}^s *is violated, change the spatial resolution to* $s' = s_{min}$ *and proceed with tracking*, where $rmax_t^s$ is the ratio $wmax_t^s/wmax_0^s$, and α_{max} is empirically established (e.g. 0.25). Note that in contrast to the previous rule, this one makes a large downward jump across resolution levels. Such a move is either

for waiting for the object to reappear, in case of occlusion, or for broadening the sampling range, in case of lost track caused by the actual state swiftly moving out of the current sampling volume.

Putting these rules together, and setting C^s_{max}'s priority over C^s_{var}'s, our overall control procedure can be summarized as follows. While moving forward in time, progressively reduce the resolution level s to $s' = s - 1$. If C^s_{max} or C^s_{var} is violated, resolve them using the rules above.

3.4 Experimental Results

For all the experiments in this section, the number of particles is set to 150. The full resolution of images is 640×480. The algorithm is written using C++ and run on a 2.8GHz P4 Windows machine with 1GB RAM.

Accuracy

In this experiment, we test the algorithm's ability to regain the best accuracy at specified times while tracking at lower resolution at other times. We tracked a blob in a medium-cluttered sequence, with known ground truth. Two trackers: a full-resolution tracker tracks the object at highest resolution s_{max} at all times (except for a short initialization period) where as the multi-resolution tracker reduces resolution whenever possible. The lowest value for the scale is 0.086 which is determined by requiring that the size of the tracked blob is at least 5×5. It is required that we move up to s_{max} regularly at an interval of $\triangle T = 30$ frames. This is done by adding some constraint to the control rules in section 3.3.3 so that the tracker is able to

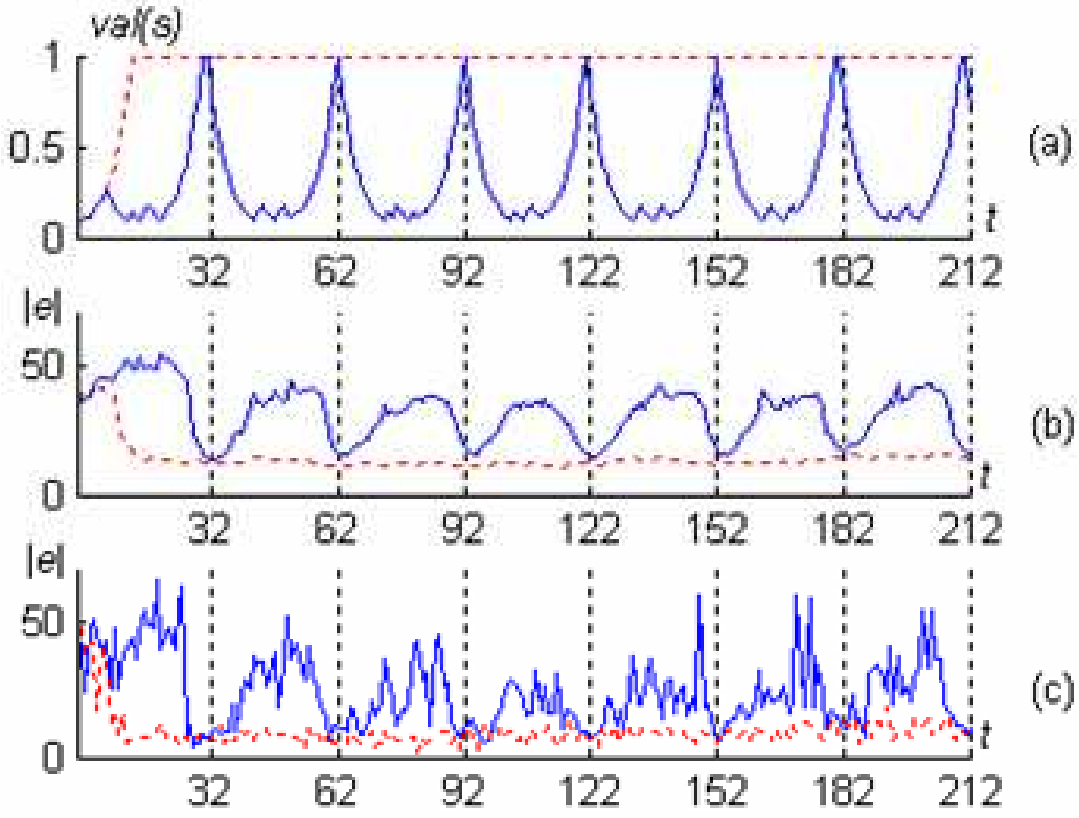


Figure 3.3: Resolutions and errors vs. time. The blue solid line is for the full-resolution tracker, and red dotted one is for the multi-resolution tracker. Where $val(s)$ mean the normalized value of the scale $s, val(s_{max}) = 1$

move up to s_{max} on time.

Figure 3.3.a shows how the resolution s varies with time. Figure 3.3.b shows the mean error of all particles weighted by their weights. Figure 3.3.c shows the error of the particles with maximum likelihood (which is often, but not necessarily, the ones closest to the ground truth). We can see from these results that at the needed time points (i.e. after every $\triangle T$) there is virtually no difference in errors between the full-resolution tracker and the multi-resolution tracker. This is stably true for a long period of time. So, while tracking at low resolution, we can practically move

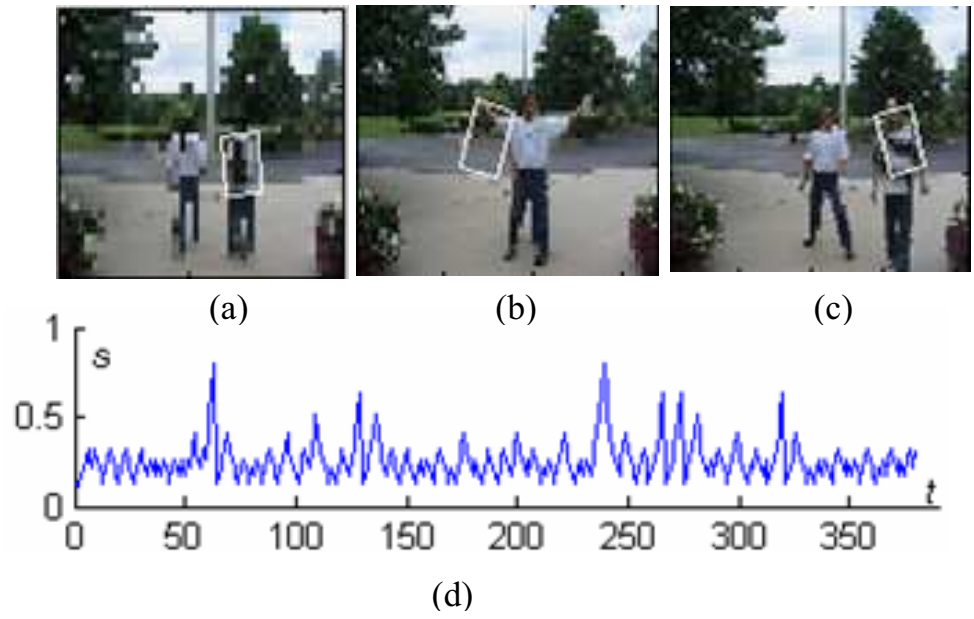


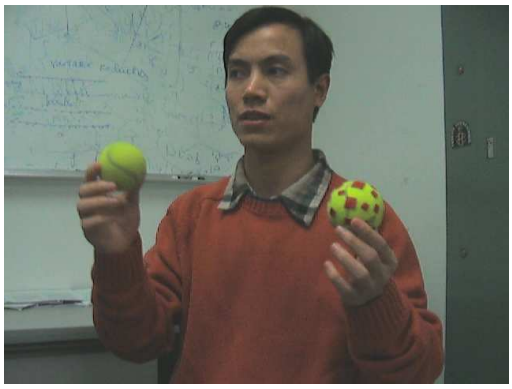
Figure 3.4: A test sequence: a) first frame ($t = 0$, low res.), b) an occluded occasion ($t = 313$, high res.), c) a later frame ($t = 380$, low res.), d) resolution s vs. time t.

up to highest resolution and accurately estimate the state of the object on demand.

Robustness and Efficiency

In the sequence used in this experiment (about 1000 frames of) the tracked object gets occluded many times, for periods as long as 20 frames. The results in Figure 3.4 show that the tracker is able to track through such situations and follow the object closely. For this sequence, the multi-resolution tracker runs at approximately 40-45 frames per second. The full-resolution tracker runs at 7 frames per second. Tests on a variety of different sequences give similar results. The abilities to deal with both lost track and confusion (section 3.3.3) give the tracker good robustness

Figure 3.5 shows several frames from the tracking of the red spotted tennis ball which contains occlusions and confusion. The two tennis balls, which are very similar at low resolution, are clearly distinguishable only at high resolution. The



a) a full res. frame

b) frame♯20

c) frame♯85

d) frame♯109

e) frame♯147

Figure 3.5: The tennis sequence

tracking is also complicated by the fact that the red color on the spotted ball is similar to the reddish color in the background. Without spatial division in the histogram calculation (section 3.3.1), a red region in the sweater can mix with other regions on the other tennis ball and produces a histogram that looks very similar to that of the spotted ball. Figure 3.5.a shows an input frame at full resolution (648×480), i.e. at the normalized scale $s_{max} = 1$. Figure 3.5.b shows frame ♯40 which is at a working resolution $s = 0.05$ (1/20 of the full resolution). When the distracting ball approaches the tracked ball, the tracker increases the resolution to $s = 0.35$ (1/3 of the full resolution), to distinguish them (Figure 3.5.c - frame ♯85). When the tracked ball gets completely occluded, the tracker jumps to the smallest resolution $s_{min} = 0.04$ (Figure 3.5.d - frame ♯109). Figure 3.5.e shows that the tracker resumes tracking of the object at low resolution again when there is no distracting object (frame ♯147). Figure 3.6 shows the distribution of particles when there is confusion at low resolution (left, $s = 0.065$). The confusion is then resolved at higher resolution (right, $s = 0.35$). Darker color means lower likelihood weight and vice versa. As we can see, when the resolution is increased, more information is brought in, and therefore the confusion decreases.

3.5 Conclusion

We proposed a coherent probabilistic framework that allows a tracker to operate at multiple levels of spatial resolution to achieve efficiency and robustness. Probabilities are propagated across spatial resolution using sequential Bayesian fil-

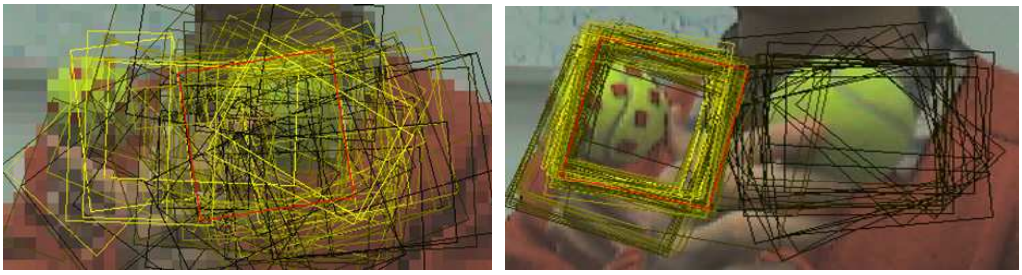


Figure 3.6: Particle distribution at low (left) and high (right) resolution. Darker color means lower likelihood weight and vice versa. The red rectangle is the mean.

tering, similar to temporal filtering. With the added dimension, the multiple resolution filtering framework provides a rich number of opportunities for controlling the tracker to achieve desired system goals. Based on this model, we built a color blob tracker and devised a set of strategies to control the tracker. That the tracker can track object(s) with a substantial saving in computation cost, and that it is able to regain tracking robustly even in a number of difficult situations show the strength and potential of our approach.

Chapter 4

Robust Object Tracking with Regional Affine Invariant Features

4.1 Introduction

Tracking is a fundamental vision problem, with many applications in higher level tasks such as activity analysis, recognition and surveillance. Developing robust tracking algorithms is challenging due to factors such as noisy input, illumination variation, cluttered backgrounds, occlusion, and object appearance change due to 3D motion and articulation.

There are numerous frameworks for tracking. In contour or curve tracking, the object is represented as a (closed) contour or a curve. The object's location and shape evolution over time can be recovered using probabilistic propagation methods ([32]) or local evolution methods such as level sets. Curve tracking typically degrades in clutter. In local feature tracking, the tracked object is represented as a collection of features, often corners or line segments, and tracking is based on establishing feature correspondences between frames. It has been used widely in Shape-from-Motion, or in 2D or 3D tracking with models ([60]). Local feature tracking also degrades in high clutter or noise. Region, blob or template tracking is less sensitive to clutter because more spatial context is used for feature representation (see e.g. [30] [52]). In this chapter, we address the problem of updating the model of an object during tracking when the object is represented by a collection of such regional

features. In the special situation in which the object to be tracked is known a priori, such appearance changes can be learned off-line and used during tracking. This was first done by Waxman and Siebert ([68])using a neural network approach; more recent learning based approaches include Black and Jepson([5]) and Ozuysal et. al. ([60]).

We describe a regional-feature based tracking algorithm designed to cope with large changes in object appearance. The tracking algorithm estimates a time-varying occupancy map of the tracked object which is updated based on local motion models of both the object and the background. The regional features we employ are the MSER features ([51]) that have been used previously for object recognition and wide-baseline stereo matching (see [54]). They are more stable from frame to frame than local features such as corners or lines, making it easier to match them for motion modeling.

We represent the tracked object with a probabilistic occupancy map that represents the changing image shape of the tracked object as it moves through the scene. Figure 4.1 contains examples of regional features and occupancy maps for frames from some sequences that will be used to illustrate and evaluate the algorithm.

The chapter is organized as follows. Related work is reviewed in section 4.2. Section 4.3 provides details of the tracking method including establishing feature motion, expansion for pixel motion and occupancy map updating. Section 4.4 includes experimental comparisons between the proposed tracking algorithm and recently described adaptive appearance based tracking methods.

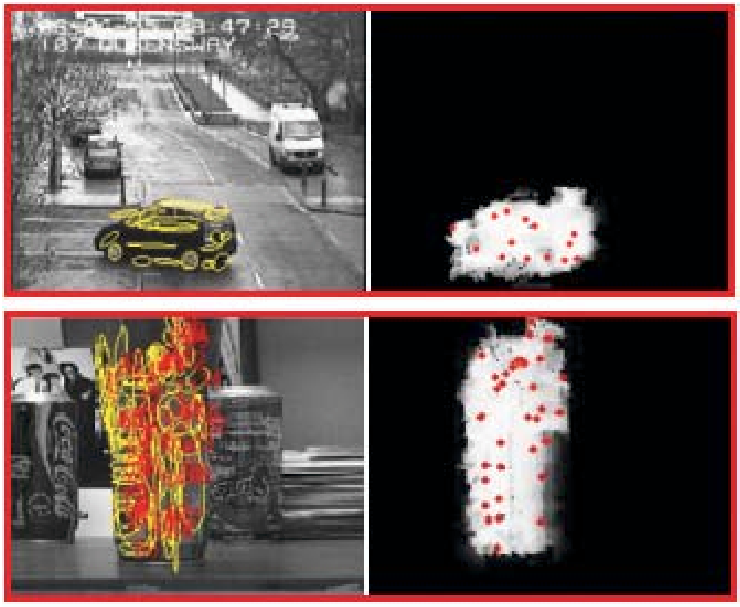


Figure 4.1: Examples of object features (left) and occupancy maps (right) in the car and coke sequences. The red dots are the centers of regional features with high likelihood matches.

4.2 Related Works

There is a vast literature on tracking, and we restrict ourselves to recent approaches related to our research. To address appearance changes, adaptive modeling of object appearance is typically employed. In [89], object appearance is modeled with a mixture of a fixed number of color-spatial Gaussians. Each mixture is considered as a particle in a particle filter system. This representation is quite similar to the object model used earlier in [30], where a variable number of Gaussian kernels is used. The set of kernels is updated between frames using Bayesian sequential filtering. The Gaussian approximation makes these approaches deal very well with gradual appearance changes. However, similar to [35], they have difficulties with

rapid changes or when changes include both occlusion and dis-occlusion, such as when an object rotates (even slowly).

In approaches including [14], [3], ..., background modeling is used to improve the reliability of tracking, essentially by giving more weight to object features that are more prevalent in the object than in the local background. In [14], the appearance of both the object and its local background are updated during tracking. At any given frame, a feature that best discriminates the object from the background is chosen to compute a likelihood map, which predicts the position of the object in the next frame. In [3], the object and background are modeled in the form of an ensemble of weak classifiers. Adaptation is done through the addition of newly trained classifiers and the removal of old ones with high error rates. The approach as formulated does not support scale changes. Our approach, instead of modeling background appearance, models background motion.

Another related tracking paradigm is feature tracking, which constructs and analyzes feature tracks over long periods of time (e.g. [10]). For point features, maintaining tracks over many frames is quite difficult, while more stable features, such as those employed in recent track-by-detection approaches ([60]) need wide-baseline matching or training with the tracked objects. Moreover, they demand feature detection as well as matching to be accurate. In contrast, the feature matching we employ is done on frame-to-frame basis, so no construction of long feature tracks is required. The paper by Donoser et. al. ([**?**]) tracks a single MSER feature, with focus on efficient detector implementation. We employ MSER features, but they could be replaced by other regional features such as IBR, Scale Saliency, etc.([54]).

4.3 Technical Approach

Our approach is motivated by the simple assumption that any image element (feature, pixel ...) that is close to the object and moves consistently with the object is, with high probability, part of the object. Our approach to updating the model of a tracked object is, then, based on motion of image elements as opposed to appearance matching. We begin by motivating the use of motion analysis for object model updating. As discussed in the literature review, previous research has focused on appearance-based methods for updating. However, these methods face a difficult trade off between conservatively maintaining an incomplete, but correct, appearance model and aggressively updating the model (but potentially including background elements into the updated model). Conservative updating schemes have difficulty tracking an object whose appearance changes rapidly, even when using a model that allows jumps, such as the WSL model ([35]). On the other hand, a more aggressive appearance updating scheme is more likely to include unwanted elements from the background, which would eventually result in tracking failure, especially in cluttered images. Attempts to resolve this trade-off, such as in [61], employ learning on specific objects or object classes (such as pedestrians). Finally, it is very challenging to update the model correctly if the newly appearing parts bear no appearance similarity to previously seen parts of the object.

Instead, we use motion similarity to update the object model. Consider the idealized situation illustrated in Figure 4.2, where the leftward rotation of object A_t results in occlusion and dis-occlusion on its left and the right sides, respectively. If

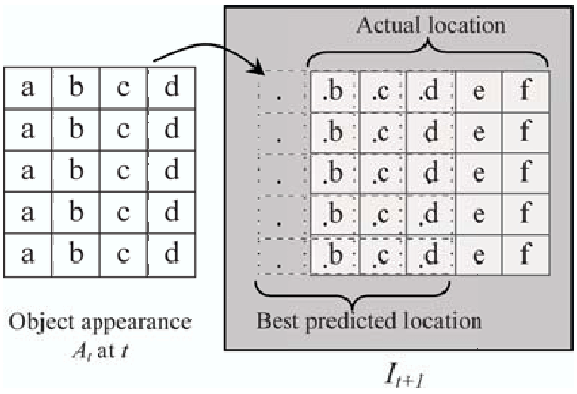


Figure 4.2: The object rotates leftwards. Actual object areas are shown in white.

appearance similarity is employed, the object's location at time $t+1$ is predicted based on the similarity of A_t and the observation at I_{t+1}. Generally[1] , the best estimated location from tracking shifts towards the left as depicted in Figure 4.2; i.e., it is difficult to identify the dis-occlusion of the part of the object labeled *'e'* or *'f'*. Gradually, this will make the estimate of the object's location, drift away from its true location, as has been observed in previous work ([30], [35], [61]). Even keeping a first-time reference template, such as [52] did, would not help in this situation. However, the motion of newly appearing parts such as *'e'* or *'f'* does provide a cue for dis-occlusion identification. In particular, they are considered to be the newly revealed parts of the object if they move consistently with the known parts of the object (e.g. *'d'* or *'c'*) and, at the same time, distinctively with respect to the background. This motivates our approach, which utilizes both the motion of the background and foreground to update the object model during tracking.

[1]Except when the background has a very different appearance from the object or when the similarity measure ignores all spatial information, as in histogram matching

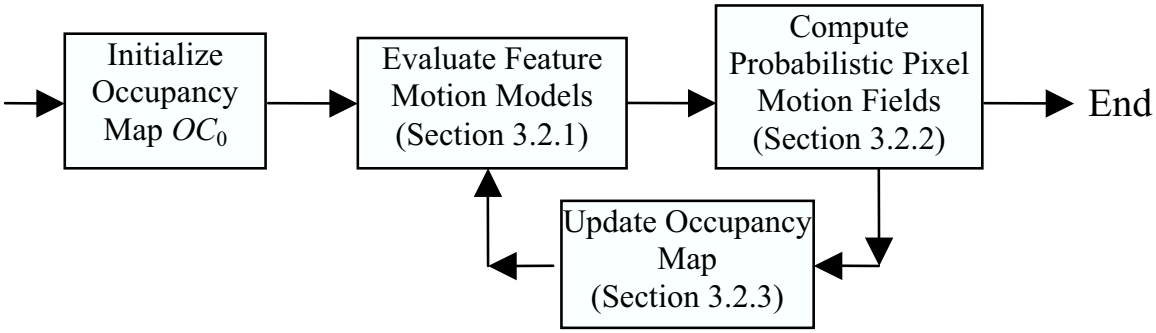


Figure 4.3: Overview of our tracking algorithm

The summary of our approach is as follows. We represent the object with a probabilistic occupancy map and reduce the problem of tracking the object from time t to $t+1$ to that of constructing the object occupancy map at $t+1$ given the occupancy map at time t. We start by computing probabilistic motion models for detected features at time t conditioned on that they belong to the foreground or the background. Feature motion distributions are computed based on the occupancy map at time t and feature similarities across frames (Section 4.3.2.1). From these feature motions, a probabilistic motion model for each pixel is constructed (Section 4.3.2.2). The construction starts from the center of the regional features and expands out to cover the entire image. Finally, pixel motion fields are used to construct the object occupancy at $t+1$ (Section 4.3.2.3). Figure 4.3 shows the flowchart of our tracking algorithm. Details of each step are explained in the following sections.

4.3.1 Object Features

We employ regional features (MSER - Maximum Stable Extreme Region ([51])) for tracking. The spatial extent of such features are approximated using ellipsoids,

which are represented by a vector (xc, a, b, α), specifying the center, major and minor axis length and orientation. Internal feature appearance is represented using the popular SIFT descriptors ([48]).

4.3.2 Feature Motion and Pixel Motion Expansion

4.3.2.1 Feature Motion

We discuss next the computation of the motion distribution for each feature from frame I_t to frame I_{t+1}. Let $F_t = \{f_i\}_1^N$ and $F_{t+1} = \{f_j\}_1^M$ denote the two detected feature sets in I_t and I_{t+1} respectively. A feature $f_i \in F_t$ may or may not be re-detected in I_{t+1}. When it is re-detected, it will ordinarily match with at least one feature $f_j \in F_{t+1}$. Its motion vector, $m(f_i)$, is then determined as $m(f_i) = xc_j - xc_i$, where xc_i and xc_j are the centers of f_i and f_j respectively. Let $M = \{m(f_i)\}$ denote the set of all $N \times M$ possible feature motions. In the sequel, M will be used as the finite (discrete) domain for motion vectors of both features and pixels.

Let $D_{t+1}(f_i)$ denote the event that f_i will be re-detected in frame I_{t+1}. The probability that a feature f_i belongs to the object and has motion $m(f_i)$ is then determined as follows,

$$\begin{aligned} &p(m(f_i), f_i \in Obj | OC_t, I_t, I_{t+1}) = \\ &p(m(f_i), f_i \in Obj, D_{t+1}(f_i) | OC_t, I_t, I_{t+1}) + \\ &\quad p(m(f_i), f_i \in Obj, \overline{D_{t+1}(f_i)} | OC_t, I_t, I_{t+1}) \end{aligned} \tag{4.1}$$

$$= p(m(f_i), f_i \in Obj, D_{t+1}(f_i)|OC_t, I_t, I_{t+1}) =$$

$$p(m(f_i), f_i \in Obj|D_{t+1}(f_i), OC_t, I_t, I_{t+1})p(D_{t+1}(f_i)|I_t)$$

The second term of (4.1) can be dropped since if the feature is not re-detected, it would not take on any motion vector from M. The second term of the last expression is the re-detection probability of a feature f_i. It is modelled with a sigmoid function of the feature's stability score ([51]). Intuitively, the more stable the feature is, the more it is likely to be re-detected in the next frame. For MSER, the stability score can be defined based on feature area variation when the detection threshold are varied ([**?**]). With some independence assumptions, e.g between $f_i \in Obj$ and $D_{t+1}(f_i)$,

$$p(m(f_i), f_i \in Obj|D_{t+1}(f_i), OC_t, I_t, I_{t+1}) =$$

$$p(m(f_i)|f_i \in Obj, D_{t+1}(f_i), OC_t, I_t, I_{t+1}).$$

$$p(f_i \in Obj|D_{t+1}(f_i), OC_t, I_t, I_{t+1}) =$$

$$p(m(f_i)|I_t, I_{t+1}).p(f_i \in Obj|OC_t) \tag{4.2}$$

The second term of (4.2) is the probability that a feature belongs to the object given the occupancy map at time t, OC_t. $p(f_i \in Obj|OC_t) = 1/A(f_i) \sum_{x \in f_i} OC_t(x)$, where $A(f_i)$ is the feature's area - the number of points in the ellipsoid corresponding to f_i. Recall that $m(f_i) = xc_j - xc_i$, where xc_j is the center of the target feature, f_j. The first term measures how likely feature f_i matches feature f_j,

$$p(m(f_i)|I_t, I_{t+1}) \propto exp(-d^2(f_i, f_j)/\sigma_o^2)e_{ij}^H \tag{4.3}$$

where $d(f_i, f_j)$ is the distance between the two SIFT descriptors of f_i and f_j, σ_o

is the empirically determined variance for feature distances, and e_{ij}^H is the error of the match w.r.t. a global motion model H. The distance in the descriptor space accounts for the mismatch in appearance, and the global motion error accounts for the mismatch in location. The latter is introduced based on the observation that object features typically move in a coherent manner. Let $\hat{f}_i = H(f_i)$ be the predicted location of f_i under H, then,

$$\begin{aligned} e_{ij}^H &\equiv e(\hat{f}_i, f_j) \\ &= sqrt(\tfrac{(xc_i - xc_j)^T (xc_i - xc_j)}{(s_i + s_j)} + \triangle(A(\hat{f}_i), A(f_j))) \end{aligned} \tag{4.4}$$

where $\triangle(A(\hat{f}_i), A(f_j)) = |(A(\hat{f}_i) \setminus A(f_j)) \cup (A(f_j) \setminus A(\hat{f}_i))| / |A(\hat{f}_i) \cup A(f_j)|$ measures the difference in area of the two features when aligned and s_i and s_j are feature sizes. Section 4.3.3 describes how the motion model H is constructed.

The motion distribution of a background feature, f_i, is computed similarly, with the exception that in (4.3), there is no error term since there is no reason to establish a global motion for all background features, and $p(f_i \in Bgr|OC_t) = 1 - p(f_i \in Obj|OC_t)$.

4.3.2.2 Expansion for Pixel Motion

Given the set of features F_t, each of which is associated with some motion distribution computed as described in the previous section, we next evaluate the motion distribution for each pixel, x, in frame I_t. The derivation is shown here for the case where $x \in Obj$, similar arguments are used when $x \in Bgr$. Let $m(F_t)$

represent some motion assignment for all features in F_t,

$$p(m(x), x \in Obj|OC_t, I_t, I_{t+1}) =$$

$$\sum_{m(F_t)} p(m(x), x \in Obj, m(F_t), F_t \subset Obj|OC_t, I_t, I_{t+1}) \tag{4.5}$$
$$= \sum_{m(F_t)} p(m(x), x \in Obj|m(F_t), OC_t, I_t, I_{t+1}).$$
$$p(m(F_t), F_t \subset Obj|OC_t, I_t, I_{t+1}) \tag{4.6}$$

The summation in (4.5) omits the term for $F_t \subset Bgr$, since it is then assumed that they would not affect the motion of a foreground pixel. When a global motion H (whose effects are accounted for in (4.3)) is given, motion of individual features can be assumed independent,

$$p(m(F_t), F_t \in Obj|I_t, I_{t+1})$$
$$= \prod_{f_i \in F_t} p(m(f_i), f_i \in Obj|OC_t, I_t, I_{t+1})$$

The first term in (4.6) is the probability that a pixel x takes motion $m(x)$ given that the motion of the features set is $m(F_t)$. It is modelled as follows, up to a normalization factor,

$$p(m(x), x \in Obj|m(F_t), I_t, I_{t+1})$$
$$\propto \sum_{f_i} K(d^2(x, f_i))p(m(x)|m(f_i), I_t, I_{t+1}) \tag{4.7}$$

where $d(x, f_i) = xc_i - x$ is the distance between x and the center of the feature f_i. This is essentially the aggregation of support from multiple features in F_t, where each individual contribution is weighted by a Gaussian kernel, K.

Direct, accurate modelling of the dependency between $m(x)$ and $m(f_i)$ in (4.7) is difficult. Instead, we make use of the motion distribution of the points in

between. In particular, we determine $p(m(x)|m(f), I_t)$ through spatial sequential filtering starting from the center $x_0 \equiv xc_i$ of f_i and ending at the point $x_n \equiv x$.

$$p(m(x_0)|m(f_i), I_t) = p(m(f_i)|I_t) \tag{4.8}$$

The Bayesian filtering process goes through two steps (see e.g. [30]), prediction and update, described below. The entire observation would be I_t and I_{t+1}. Let D_n denote the relevant observations at a step n, $D_n \subset (I_t, I_{t+1})$. The prediction step is then (the Chapman-Kolmogorov equation),

$$\begin{aligned} & p(m(x_n)|D_{0:n-1}) = \\ & \sum_{m(x_{n-1})} p(m(x_n)|m(x_{n-1}), D_{0:n-1}) p(m(x_{n-1})|D_{0:n-1}) \\ & = \sum_{m(x_{n-1})} p(m(x_n)|m(x_{n-1})) p(m(x_{n-1})|D_{0:n-1}) \end{aligned}$$

where the state model $p(m(x_n)|m(x_{n-1})) \propto exp(-(m(x_n) - m(x_{n-1}))^2/\sigma_m^2)$. Next, the updating step to find the posterior is

$$\begin{aligned} & p(m(x_n)|D_{0:n}) \\ & \propto p(D_n|m(x_n), D_{0:n-1}) p(m(x_n)|D_{0:n-1}) \\ & = p(D_n|m(x_n)) p(m(x_n)|D_{0:n-1}) \end{aligned}$$

where $p(D_n|m(x_n))$ is the observation model. Let $NCC(x_n, x_n + m(x_n))$ denotes the normalized correlation score between two image patches: the first one is taken around x_n in I_t and the second one is taken around $x_n + m(x_n)$ in I_{t+1},

$$p(D_n|m(x_n)) \propto exp(-d^2(NCC(x_n, x_n + m(x_n)))/\sigma_{obs}^2) \tag{4.9}$$

Thus from feature motion, we obtain a pixel motion distribution through a filtering process that expands from feature centers. Other alternatives to propagate

information from $m(f)$ to $m(x)$, such as MRF or other graphical models, are possible. Here, we choose sequential filtering mainly for efficiency. Expansions that start from a sparse affine invariant feature matching to pixel-wise correspondence have been used, with different formulations, in [**?**] for image exploration and in [**?**] for growing local features.

4.3.2.3 Occupancy Updating

We describe next the construction of the object occupancy map at time $t+1$, OC_{t+1}, from the motion field computed above and the previous occupancy map. Consider the occupancy at a point x in frame I_{t+1}. Let $S(x)$ denote the set of points x' in I_t that can possibly move to x at time $t+1$; let $FG(x)$ denote the event that x is occupied by at least one of the points $x' \in S(x)$ where $x' \in Obj$ at time t,

$$\begin{aligned} & p(FG(x)) = p(\bigcup_{x' \in S(x)} (x' \rightarrow x, x' \in Obj)) = \\ & 1 - p(\bigcap_{x' \in S(x)} \overline{(x' \rightarrow x, x' \in Obj)}) = \\ & 1 - \prod_{x' \in S(x)} (1 - p(x' \rightarrow x, x' \in Obj)) = \\ & 1 - \prod_{x' \in S(x)} (1 - p(m(x'), x' \in Obj)) \end{aligned} \tag{4.10}$$

where $m(x') = x - x'$. Given the pixel motion fields computed in the previous section, independence in pixel motion can be assumed in the derivation of (4.10). The last expression can be evaluated using (4.6). Similar computation is applied for $p(BG(x))$ which represent the probability that x is occupied by a background pixel.

Finally, for a given pixel x at time $t+1$, there are four possible events corresponding to whether foreground or background pixels move to that pixel or not:

$FG(x) \wedge BG(x)$, $\overline{FG(x)} \wedge BG(x)$, $FG(x) \wedge \overline{BG(x)}$ and $\overline{FG(x)} \wedge \overline{BG(x)}$. Expanding on this set of events, we have

$$OC_{t+1}(x) \equiv p(x \in Obj) =$$

$$p(x \in Obj, FG(x) \wedge BG(x)) + p(x \in Obj, \overline{FG(x)} \wedge BG(x)) +$$

$$p(x \in Obj, FG(x) \wedge \overline{BG(x)}) + p(x \in Obj, \overline{FG(x)} \wedge \overline{BG(x)})$$

Each of these terms can be evaluated as follows, e.g.,

$$p(x \in Obj, FG(x) \wedge BG(x)) =$$

$$p(x \in Obj | FG(x) \wedge BG(x)) p(FG(x) \wedge BG(x)) =$$

$$p(x \in Obj | FG(x) \wedge BG(x)) p(FG(x)) p(BG(x))$$

The second and the third terms are computed using (4.10). The first term is the probability that the pixel x belongs to the object at time $t + 1$ given that both foreground and background can possibly move to that pixel. It is reasonable to set its value to 0.5 to represent this uncertainty. Probabilities associated with the remaining three events can be assigned similarly, (i.e., in order, 0, 1, 0.5).

4.3.3 Implementation Details

4.3.3.1 Motion Model

The global model for object feature motion that is used in equation (4.3) is a 2-D affine (translation, rotation and scaling) motion. An affine model is employed for the following reasons:

1. It adequately reduces the likelihood of gross matching errors. More subtle matching errors are further suppressed during the motion expansion stage discussed previously.

2. More complex motion models, such as a general rigid motion, are difficult to estimate accurately because: (a) Object displacement between consecutive frames is small, and achieving accurate rigid motion estimation in such small baseline cases is difficult. (b) Detection errors, such as center shifting, in regional feature detectors are often significant compared to the feature motions themselves. (c) A rigid model with low error tolerance is sensitive to even slight deformations in the surface of the object.

For these reasons, a simple 2D-affine motion model is used to weigh down 'outliers' in foreground feature matching.

A weighted RANSAC algorithm is used to determine this motion, where the weight for a match is $w_{ij} = P(f_i \in Obj)exp(-d^2(f_i, f_j)/\sigma_o^2)$. In hypothesis generation, at each iteration, we use k feature matches. Each match contributes four constraints on the center location, and major and minor axes' length. Local feature orientation is ignored since its estimation for near-circular features is often not reliable. In hypothesis verification, the matching error between any two features is measured using (4.4). Note that in this formulation, shape information is utilized in both stages of RANSAC. The predicted error of a motion hypothesis H on a feature f_i is then

$$e^H(f_i) = e(H(f_i), f_j^*) \tag{4.11}$$

where $f_j^* = \arg\min_{f_j \in F_{t+1}} e(H(f_i), f_j)$.

A match is considered to be good if $e(f_i, f_j) < \theta$ for some pre-defined θ. The total weight of good matches under H is the goodness score of this hypothesis. The motion model found is used to update the likelihood of every pair of matches in (F_t, F_{t+1}) as in (4.3).

4.3.3.2 Dealing with Low Feature Response

The performance of a robust feature-based tracker is degraded if the links among features between two frames are weak. This may be due to low feature response or poor feature distance measurement. However, as observed in the literature, descriptors such as SIFT work very well on a wide range of image conditions. Therefore, the likely reason that we obtain only few strong feature matches is the number of detected features is small. This happens either when the object is too small or when its appearance is almost homogeneous in texture (a single-color blob). In such (rare) cases, to keep tracking the object, we simply substitute a blob tracking module.

4.4 Experimental Results

We demonstrate the performance of our tracking algorithm through a number of experiments. We first test the sensitivity of the tracker's performance with respect to the degree of accuracy of the initial delineation of the object to be tracked. Trackers are either initialized by hand or by detection methods like independent mo-

tion detection that provide rough bounding box approximations to the objects. For a variety of reasons these initial occupancy maps can either over or underestimate the true spatial extent of the object. A second set of experiments are designed to illustrate how the tracking algorithm adapts to appearance changes due to occlusion and dis-occlusion. For display, all of the occupancy maps are thresholded at 0.5.

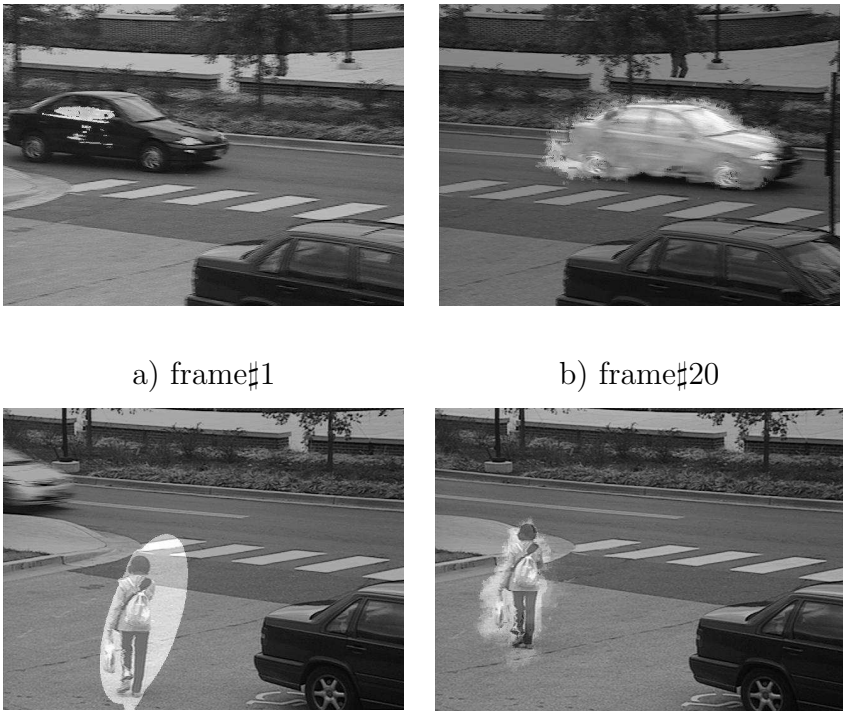
The first experiments illustrate sensitivity of appearance updating trackers to the initial object occupancy map. We initialize the tracker in two ways:

1. First, we assume that the rough location of the object is given and then a disk whose radius is approximately half of the size of the tracked object is centered about this point. This underestimates the true extent of the object. Any detected local feature that intersect significantly or is contained within this disk is assumed to belong to the object. The initial occupancy map is then the union of these initial features' areas and weighted by a Gaussian mask centered at the initial location. Figure 4.4, top, shows the results of our tracker using this initialization method on a video sequence with a moving background. The occupancy maps are overlaid on the images. Since the radius of the disk was small, only a few features are selected to construct the initial occupancy map. Nevertheless, after 20 frames (Figure 4.4.b), the map evolved to be close to the true shape of the moving object.

2. In the second method, we used motion detection, specifically frame differencing, to initialize the tracking. Frame differences are first binarized, then dilated and eroded. Any local feature that intersects significantly with this

binary map is assumed to belong to the object. By doing this, we obtain a cluster of features which usually include both object and the nearby background. This method tends to overestimate the true extent of the object. As before, the initial occupancy map was the union of these features' areas weighted by the distance transform of the binarized frame difference. Figure 4.4, bottom, shows the tracking result on a video sequence using this second initialization method. For this sequence, the camera was stationary at the time that the frame differencing was performed, but moved after that. After approximately 70 frames (Figure 4.4.b), the occupancy map approaches the true shape of the object. Since the person moved more slowly than the car did in the previous experiment, the occupancy map approaches the object shape at a slower rate. In general, the better the distinction in motion, the quicker the uncertainty in the occupancy map is removed.

To more systematically evaluate the robustness of our approach to inaccuracy in initialization we carried out the following experiments. We compared our tracker with in [30] and [14] - two recent high quality trackers that attempt to update object appearance models during tracking - on a 100-frame video sequence under a number of different initialization conditions. A surveillance video was selected such that when the initialization is correct (i.e., the specified occupancy map coincides with the true spatial extent of the object to be tracked), all three trackers could track the object through the entire sequence without significant errors. There were also no noticeable shadows or artifacts in the video. The tracked object moves in a



a) frame♯1 b) frame♯20

c) frame♯1 d) frame♯70

Figure 4.4: Under-selection (top) and over-selection (bottom) in initialization. Occupancy maps are overlaid (highlighted).

frontal parallel direction since the algorithm in [14] cannot cope with significant scale changes. The background was cluttered with some distracting objects passing by the tracked object during tracking. The ground truth in a frame was an object bounding box O_i that was manually marked. The tracking outputs were also rectangular boxes T_i. The error in the i-th frame is measured as $e = |(T_i \setminus O_i) \cup (O_i \setminus T_i)|/|T_i \cup O_i|$. The total tracking error E is averaged over all e_i in the last two thirds of the frames in the sequence (i.e. the first third is discarded). We converted the occupancy maps to bounding boxes with a binarization step followed by several morphological operations. Two methods for varying initial conditions were used here.

- In the first, the location and size of the initial bounding box T_1 is chosen as the translation of the ground truth bounding box O_1 (Figure 4.5, left, the top picture) and a tracking session was performed for each position of T_1. The average tracking errors for [30], [14] and our tracker as a function of the translation distance, d, are presented in Figure 4.5 (the left chart). The horizontal axis shows the ratio $r = d/w$, where w is the width of O_1. As we can see, our tracker was able to track the object well and consistently. Trackers that make committed appearance modeling at the beginning such as [30] are usually sensitive to initialization inaccuracies.

- In the second method, we re-scale the initial window T_1 while keeping its center fixed (Figure 4.5, left, the bottom picture). The tracking errors for different sizes of T_1 are presented in Figure 4.5 (the right chart). The horizontal axis shows different scales of the width (or height) of T_1 w.r.t. the width (height)

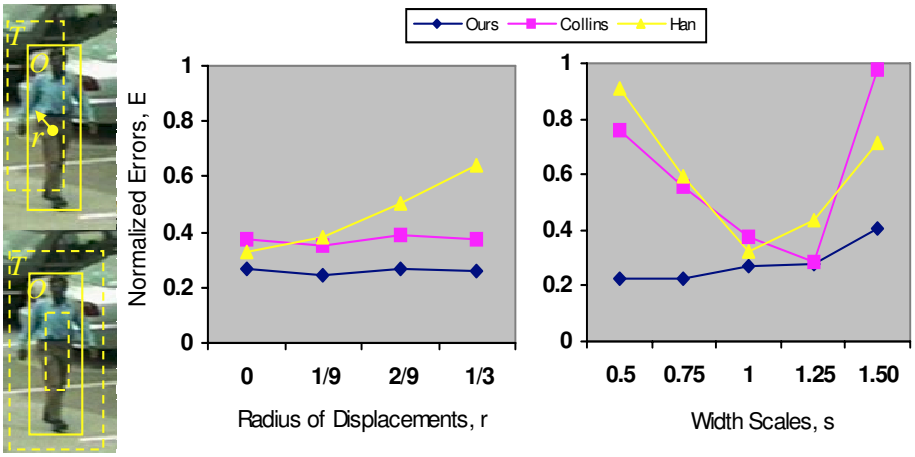


Figure 4.5: Left: the tracked object and some initial windows. Right: normalized tracking errors; the left chart is for translating and the right chart is for scaling initial window. Radius of displacement is measure in term of the ratio $r = d/w$ (see text)

of O_1. Again, our tracker performed quite well in this case and produces stable tracking results for the whole range of initialization variation.

Next, we illustrate our tracker's ability to deal with two challenging situations. As reported in [35] and [61], most trackers have difficulty when the tracked object rotates in front of a cluttered background, such as the coke sequence in [5]. Accommodation for both occlusion and dis-occlusion is required here. Figure 4.6 shows the result of [30], [14] and our tracker on the coke sequence. The trackers from [30] and [14] failed early in the sequence. However, ours was able to incorporate newly appearing parts of the object and tracked it successfully even though the object appearance changed completely during the sequence. Note that since the hand and the shadow moved consistently with the can, they were also included in the occu-

a) frame♯1-Initial b) ♯100-[30]'s c) ♯100-[14]'s

d) ♯100-Ours e) ♯150-Ours f) ♯200-Ours

Figure 4.6: Tracking results on the coke sequence. a) Initial selection for all trackers, b) Results of [30], c) [14] and d) e) f) our tracker. Note the hand and the shadow are also included.

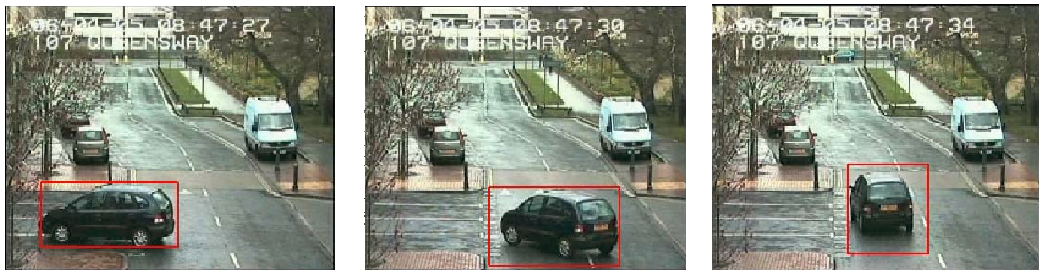
pancy maps constructed by our algorithm. Figure 4.7 shows the tracking results on a surveillance video from published domains[1](rough bounding boxes are shown for better visualization). The challenges in this sequence include cluttered background, partial occlusion, large object shape and size changes as the car turns and moves away.

Rapid illumination changes pose another challenge for robust visual tracking. In general, for appearance-based approaches, when faced with extensive change, illumination modelling has to be added, such as in [38]. Figure 4.8 shows the advantage of using our motion-based occupancy map method instead of an appearance-based

[1]The ETI-SEO and VACE-CLEAR07 tracking evaluation projects

a) ♯1 b) ♯90 c) ♯206

d) ♯1 e) ♯90 f) ♯188



g) ♯1 h) ♯90 f) ♯206

Figure 4.7: Tracking result on the car sequence. Results of [30] (top), [14] (middle) and our tracker (bottom).

technique on a sequence from a published domain[1]. The challenges come from the overall poor lighting condition that creates low contrast between the tracked person and the surrounding and from the abrupt lighting changes as the tracked person walks under different light sources placed along the corridor. Our tracker was able to track the object through all of these changes, while the others failed very early. Note that the other two trackers lost track well before the size of the object changed significantly. A basic mean-shift-based or template correlation-based tracker performs even more poorly on this sequence.

Our tracker is implemented in Matlab and all experiments were run on a 2.8 GHz, 1GB Pentium 4 PC. The tested videos are sequences of 640×480 color images. The average frame rate when the object is small, e.g. 50×50 pixels, is 2Hz, and when the object is larger, e.g. 200×200 pixels, is around 0.2 Hz. The corresponding numbers for [14] (in C) and [30] (in Matlab) are 30, 10 and 0.5, 0.01 Hz respectively.

4.5 Discussion

We described a tracking approach that is based on feature matching and pixel motion expansion. The occupancy map representation makes the tracker robust under a wide variety of tracking conditions. Motion expansion makes the tracker less sensitive to errors in initialization and able to incorporate newly appearing parts of the tracked object in spite of potential visual dissimilarity to the previously visible object surfaces.

Since the expansion step in 4.3.2.2 is based on the motion of image elements,

a) ♯1-[30]'s

b) ♯100

c) ♯130

d) ♯1-[14]'s

e) ♯68

f) ♯118

g) ♯1-Ours

h) ♯100

k) ♯130

l) ♯178

Figure 4.8: Tracking result on the corridor sequence. Results of [30] (first row), [14] (second row) and our tracker (the rest).

there are several points to note here. First, we assume that the tracked object moves with respect to the surrounding background. In practice, for example in surveillance of pedestrians, many times the tracked object remains almost stationary against the background, providing no motion distinction. Fortunately, it is fairly easy to detect such a situation, e.g. using optical flow or, more concretely, using the feature motion established in 4.3.2.1. In such cases, we can avoid expanding when there is only a weak difference between the tracked object's and the background's motion. Second, if there is a large and nearly homogeneous untextured patch nearby the object, its object's occupancy map usually gets combined with this patch, since pixels in that region do not provide strong motion cues. However, this diffusion does not progress far into the uniform region because such uniform regions have few features to support that expansion.

Even though we did not assume a prior for occlusion probability for pixels in Section 4.3.2.2, occupancy values of the occluded parts of the object typically degrade quickly since they do not exhibit motion consistency with any strong feature motion. Therefore, partial occlusion can be accounted for as demonstrated in the experiments in the previous section. However, to obtain a more stable tracking performance in the case of very severe occlusion or motion confusion, it would be better to incorporate occlusion, shape or appearance priors, which our current model does not include. Finally, even though multi-hypothesis tracking is not the point of research here, we believe our approach could be embedded into a multi-hypothesis tracker. One simple extension would be to employ a one-against-all approach, i.e. treating in turn each object as the foreground and the rest as background.

Chapter 5

Event Modeling and Recognition using Markov Logic Networks

5.1 Introduction

We consider the problem of event modeling and recognition in visual surveillance and introduce an approach based on Markov Logic Networks ([65]) that naturally integrates common sense reasoning with uncertain analyses produced by computer vision algorithms for object detection, tracking and movement recognition. We motivate and illustrate our approach in the context of monitoring a parking lot, with the goal of matching people to the vehicles they arrive and depart in.

There are numerous frameworks for event recognition. In declarative approaches (e.g. [57]), events are represented with declarative templates. Events are typically organized in a hierarchy, starting with primitive events at the bottom and composite events on top. The recognition of a composite event proceeds in a bottom-up manner. These approaches have several drawbacks. First, a miss or false detection of a primitive event, which occurs frequently in computer vision, especially in crowded or poorly illuminated conditions, often leads to irrecoverable failures in composite event recognition. Second, uncertainty is often not modeled and so these methods are generally not robust to typical errors in image analysis.

In probabilistic frameworks, such as HMMs (e.g. [91]), or DBNs ([57]), events are represented with probabilistic models. Event recognition is usually performed

using maximum likelihood estimation given observation sequences. While these approaches provide robustness to uncertainty in image analysis, their representations often lack flexibility (e.g. number of states or actors needs to be known in advance) and hence it is difficult to use them in dynamic situations. Furthermore, their performances often degrades significantly when missing observations occur.

In general, the problems of noise or missing observations always exist in real world applications. Our contention is that common sense knowledge, specific to the domain under consideration, can provide useful constraints to reduce uncertainties and ambiguities. Having a good knowledge base (KB) and an effective reasoning scheme helps to improve event recognition performance.

Technically, we address uncertainty in observations and representational richness of event specification by a combination of logical and probabilistic models.

1. Domain common sense knowledge is represented using first order logic statements. Both negation and disjunction are allowed.

2. Uncertainty of primitive event detection is represented using detection probabilities. Uncertainty of logical relations (including event models or logical constraints) is represented with a real-valued weight set based on, for example, domain knowledge.

3. Logical statements and probabilities are combined into a single framework using Markov Logic Networks ([65]).

Our system maintains an undirected network of grounded atoms which correspond to events that have occurred in the video. At any moment, primitive events are

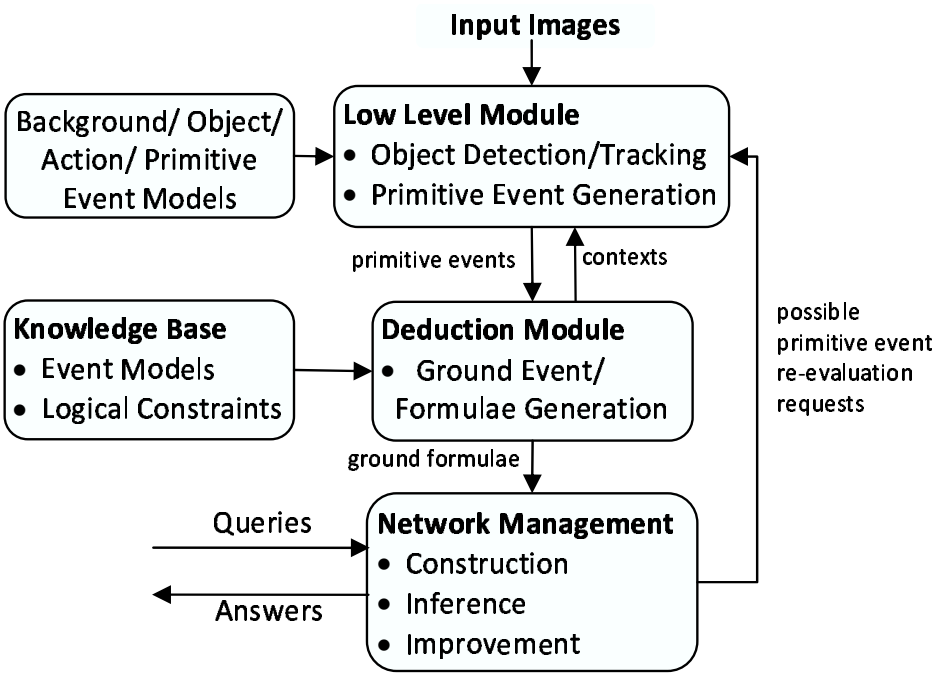


Figure 5.1: Overview of our system

detected with associated detection probabilities. They are then used to ground logical rules in the KB, which generally leads to generating more grounded events. Next, these grounded logical rules are added to the Markov network. The network parameters or structures are revised with these updates. The marginal probability of any (composite) event can be determined using probabilistic inference on this network. Fig. 5.1 shows the overview of our system.

The chapter is organized as follows. Section 5.2 reviews related work; section 5.3 discusses the tasks needed to solve our problem though an example in visual surveillance; section 5.4 reviews the MLN framework; sections 5.5 and 5.6 describe our knowledge representation and network construction; section 5.7 gives implementation details and experiments; section 5.8 concludes the paper with a discussion.

5.2 Related Work

Visual event detection from video has a long history in computer vision. We review here only approaches that are most relevant to ours. Logic has been used for visual event recognition in a number of works. In [66], Rota et al presented an elegant treatment for representing activities using declarative models. Recognition was performed effectively using a constraint-satisfaction algorithm. In [57], Nevatia et al also used a (hierarchical) declarative language but additionally addressed the uncertainty in primitive event detection. Inexact logical inference, which is important for reasoning with commonsense knowledge, was not addressed in these works. In [69], Shet et al used a multi-valued default logic for the problem of identity maintenance. Default reasoning was conducted on a bi-lattice of truth values with prioritized default rules. Identity maintenance rules are prioritized mainly based on domain knowledge. A continuous bi-lattice was used in [70] for human detection. Here, instead of using a multi-valued logic, we use a combination of logic and probability to handle inexact inference including identity maintenance. Each level of prioritization can be mapped to our framework using a rule weight.

A probabilistic network for identity maintenance was used in [59] where Nillius et al successfully linked identities of soccer players from track segments over long periods of time. However, similar to many approaches to multi-hypothesis tracking, the identity maintenance is based on pure matching of object appearances. As pointed out in [69], domain knowledge can be used to improve identity maintenance. For example, ownership of an item can link two identities together over a gap in

observation - two "identities" who appear similar and "possess" some common item are likely to be the same person. This was illustrated with a person exiting and later entering the same vehicle. Such domain-dependent identity maintenance rules can be easily added to our knowledge base and used for inference using the methods discussed in section 5.5.2.4.

The combination of probability and (first order) logic has been pursued extensively in AI and led to the emergence of Statistical Relation Learning (SRL, [28]). SRL representations often involve unrolled or grounded graphical models (directed (Bayesian) as well as undirected (Markov) ones), which are constructed using a frame-based or a logic-based approach. They have been used for human activity recognition (although not in vision-based systems). In [45], Liao et al recognized human activities based on the information about locations they visited provided by GPS sensors worn by users (location-based). Probabilistic inference is performed on an unrolled Markov network formed from a Relational Markov Network, which essentially encodes high-level domain knowledge. Relation weights can be learned using a MAP estimation technique. In [63], Pentney et al recognized human activities based on objects used. The objects were RFID tagged and identified using RFID readers worn by the users (object-use based). Logical rules are grounded and linked to form a probabilistic network within a single time slice. In general, these approaches are intrusive. They require users to wear additional sensors. Therefore their application to general surveillance tasks is limited. Here, we work with visual input and use common sense knowledge to complement limitations in visual perception.

Figure 5.2: A frame from a parking lot sequence and its corresponding foreground regions detected using background subtraction. Here, parked cars introduce significant occlusion and door openings lead to many false alarms.

5.3 Sample Problem

We motivate our approach with the surveillance problem of monitoring a parking lot and determining which people enter or leave in which cars (Fig. 5.2). In a parking lot, cars of various shapes and sizes can park close together. Occlusion is not only unavoidable but sometime severe. This leads to many difficulties in tracking people, since their corresponding foreground blobs may change from complete to fragmented or become totally missing as they move between parked cars. It is often difficult to determine the exact moment a person enters a car, or even which car a person enters. Pure declarative, bottom-up approaches (e.g. [57], [66]) that rely on accurate capture of primitive events will not work well here. Probabilistic action recognition (e.g. HMMs([91])) might fail as well since, locally, an observation may be missing altogether. For more robust event recognition, we propose to use common sense knowledge about the domain under consideration. The knowledge base will contain

rules that range from definite ones such as "if a car leaves, there must exist some person driving it" or "a person can drive only one car at any time" to weaker rules such as "people walking together usually enter the same car" or "if a person puts a bag into the trunk of a car, he or she is likely to enter that car". We represent these rules using first order logic augmented with probabilities to represent the degree of confidence for each rule. Additionally, the recognition of actions or primitive events such as "walking together" or "put a bag into a car" is uncertain. Probability theory provides a convenient method to handle these also. We then need an approach that combines logic and probabilistic elements in a coherent framework.

Briefly, we achieve this by using 1) first order logic formulae to represent domain knowledge, 2) a real-valued weight to represent the confidence in each logic rule, 3) probability to model uncertainty for primitive event and action recognition, 4) a probabilistic logic network, namely the Markov Logic Network, to connect (detected) ground atoms and to perform probabilistic inference (e.g. determine the probability that a person enters some car, given the input sequences). The following sections discuss in detail these aspects for our particular surveillance problem as well as for general surveillance contexts.

5.4 Background on Markov Logic Networks

Markov Logic Network (MLN, [65]) is one of the unrolled graphical models developed in SRL([28]) to combine logical and probabilistic reasoning. In MLN, every logic formula F_i is associated with a nonnegative real-valued weight w_i. Every

instantiation of F_i is given the same weight. An undirected network, called a Markov Network, is constructed such that,

- Each of its nodes correspond to a ground atom x_k.
- If a subset of ground atoms $x_{\{i\}} = \{x_k\}$ are related to each other by a formula F_i, then a clique C_i over these variables is added to the network. C_i is associated with a weight w_i and a feature f_i defined as follows

$$\begin{aligned} f_i(x_{\{i\}}) &= 1, \text{ if } F_i(x_{\{i\}}) \text{ is true,} \\ &= 0, \text{ otherwise .} \end{aligned} \tag{5.1}$$

Thus first-order logic formulae in our knowledge base serve as templates to construct the Markov Network. This network models the joint distribution of the set of all ground atoms, X, each of which is a binary variable. It provides a means for performing probabilistic inference.

$$P(X = x) = \frac{1}{Z} \exp(\sum_i w_i f_i(x_{\{i\}})). \tag{5.2}$$

where Z is the normalizing factor, $Z = \sum_{X \in \mathbf{X}} \exp(\sum_i w_i f_i(x_{\{i\}}))$. If $\phi_i(x_{\{i\}})$ is the potential function defined over a clique C_i, then $log(\phi_i(x_{\{i\}})) = w_i f_i(x_{\{i\}})$.

5.4.0.3 Inference

Based on the constructed Markov network, the marginal distribution of any event given some evidence (observations) can be computed using probabilistic inference. Since the structure of the network may be very complex (e.g. containing

undirected cycles), exact inference is often intractable. MCMC sampling is a good choice for approximate reasoning ([65]). In MLN, the probability that a ground atom X_i is equal to x_i given its Markov blanket (neighbors) B_i is

$$P(X_i = x_i | B_i = b_i) = \frac{\exp(\sum_{f_j \in F_i} w_j f_j(X_i = x_i, B_i = b_i))}{\exp(\sum_{f_j \in F_i} w_j f_j(X_i = 0, B_i = b_i)) + \exp(\sum_{f_j \in F_i} w_j f_j(X_i = 1, B_i = b_i))} \quad (5.3)$$

where F_i is the set of all cliques that contain X_i and f_j is computed as in Eq. 5.1.

Basic MCMC (Gibb sampling) is known to have difficulty dealing with deterministic relations, which are unavoidable in our case. It has been observed that using simulated tempering ([50], [41]) gives better performance than the basic Gibb sampling ([41]). Simulated tempering is a MC method that is closely related to simulated annealing. However, instead of using some fixed cooling schedule, a random walk is also performed in the temperature space whose structure is predetermined and discrete ([50]). These moves aim at making the sampling better at jumping out of local minima.

5.5 Knowledge Representation

In this section, we will describe our approach to represent knowledge and its associated uncertainty. In our framework, object states and their interactions (including the so-called events, actions or activities as they are interchangeably referred to in other work, e.g. [57], [66]) are all represented with first order logic predicates. A predicate is intensional if its truth value (for a certain grounding of its arguments) can only be inferred (i.e. cannot be directly observed) ([55]). A

predicate is extensional if its truth value can be directly evaluated by a low-level vision module. It is strictly extensional if this is the only means to evaluate it (i.e. it can only be observed and not inferred).

5.5.1 Logical Representation

In [65], the Markov network is constructed using an exhaustive grounding scheme, which can lead to an explosion in the number of ground atoms and network connections. Most of them are irrelevant and create significant difficulties for inference. A more efficient scheme was proposed in [74], which essentially grounded only clauses that can become unsatisfied using a greedy search. It is not clear if this approach could handle dynamic domains that involve, for example, time and location. Here, we represent our knowledge in the form of production rules, $production \rightarrow conclusion$, and use deduction to ground (and add to the Markov network) only literals (including both positive and negative atoms) that are possibly true.

In traditional deductive systems (e.g. [55]), production rules in the form of Horn clauses are used extensively. However, Horn clauses cannot represent negations and disjunctions, which are often required to capture useful commonsense knowledge. To increase our system's representational ability, we allow the following rule forms,

$(\bigwedge_i a_i) \rightarrow b$ Definite (i.e. Horn) clauses are used to define a composite event from sub-events (similar, for example, to multi-thread event definition in [57]),

causal and explanatory relationships between observations and underlying actions (e. g. $use(Bowl) \rightarrow make(Cereal)$ or $at(Resstautrant) \rightarrow have(Dinner)$ in object-use based [63] and location-based frameworks [45])

$(\bigwedge_i a_i)(\bigwedge_j \neg b_j) \rightarrow c$ Many events can only be described with a rule that has negative preconditions, for example, $at(C, S, t) \wedge \neg stopped(C, t) \rightarrow violate(C, S, t)$ where C is a car and S is a stop sign. Identity maintenance ([69], [73]) also often leads to formulae with negative preconditions, for example, $own(H_1, Bag) \wedge take(H_2, Bag) \wedge \neg eq(H_1, H_2) \rightarrow theft(H_2, Bag)$.

$(\bigwedge_i a_i) \rightarrow \neg b$ This form is often used to describe an exclusion relation. For example, the rule "a person P belongs to only one group G" can be written as $belongto(G_1, P) \wedge \neg eq(G_1, G_2) \rightarrow \neg belongto(G_2, P)$. When such a rule fires, negative atoms are added to the ground atom database. As a result, the database will consist of both negative and positive literals.

$(\bigwedge_i a_i) \rightarrow (\bigvee_j b_j)$ Disjunctions are used when a single conclusion cannot be made. For example, $use(Cup) \rightarrow (drink(Coffee) \vee drink(Tea))$. When it fires, all atoms in the conclusion are added to the ground atom database. Disjunctions also arise from existential quantifiers (next section).

These forms, of course, are not the most general ones in First Order Logic. However, practically, they are sufficiently rich to represent a wide range of common sense knowledge and to capture complex events in surveillance domains.

5.5.2 Uncertainty Representation

Uncertainty is unavoidable in practical visual surveillance applications. We consider two classes of uncertainty: logical ambiguity and detection uncertainty. Their sources and ways to represent them are described below.

5.5.2.1 Incomplete or Missing Observations

Occlusion and bad imaging conditions (e.g. dark, shadowed areas of the scene) are two common conditions that prevent us from observing the occurrence of some actions. In some cases, even if a unique conclusion cannot be made, some weaker (disjunctional) assertion might still be possible. Rules with disjunctive effects are often needed then. For example, the statement "if a bag b is missing at some time interval t and location L, then someone must have picked it up" could be formalized as $missing(b, l, t) \rightarrow (\exists p\ passBy(p, l, t) \wedge pickUp(p, b, t))$. Here the action $pickUp(p, b, t)$ can be inferred when its direct detection is missed. This type of formulae involves an existential quantifier and will be expanded to a disjunction of conjunctive clauses when grounded. For example, suppose that $passBy(P_1, L, T)$ and $passBy(P_2, L, T)$ are true for two persons P_1 and P_2 (i.e. two persons P_1 and P_2 passed by when the bag went missing), then the grounding of this rule would be $missing(B, L, T) \rightarrow (passBy(P_1, L, T) \wedge pickUp(P_2, B, T)) \vee (passBy(P_2, L, T) \wedge pickUp(P_2, B$ This expansion obviously is not suitable for infinite domains. However, in practice, most object domains are finite (e.g. number of people or cars is finite) therefore the expansion is feasible for surveillance. As evidence arrives, previously expanded

domains may need to be updated (section 5.6.1).

5.5.2.2 Non-perfect Logical Statements

Common sense statements in the KB are not always true. We use a real-value weight to represent the confidence of each rule in the KB. Rules with absolute certainty, such as "a person can drive only one car at a time", are given an infinite weight. In practice, such a hard clause is "softened" with a maximum weight, $MAXW$, to facilitate the inference process. Rules that are almost always true, such as "a person interacts with only one car", are given strong weights. Weak weights are assigned to rules that describe exceptions (i.e. situations that are possibly true but not common such as "a driver might enter a car from the passenger side").

5.5.2.3 Extensional Evaluation Uncertainty

The evaluation of an extensional predicate, E, by the low-level vision module might return answers with absolute certainty or with some associated (detection) probability, $p_D(E = true)$. For the first case, whether the result is $true$ or $false$, we make E an evidence variable and add it to the Markov network. For the second case, a method to integrate E and its detection probability for high-level logical reasoning are needed.

One approach would be to add this grounded, single-atom clause, $(E, w \propto p_D)$ and its complement, $(E, w \propto 1 - p_D)$ to the Markov network. (Note that using only one of these clauses is not sufficient). This way, the marginal probability,

$p(E = true)$, is fixed to p_D. However, evidence from other sources may change the probability $p(E = true)$, especially when E is not strictly extensional. Therefore, it would be better to add an observation variable O and use these two formulae: $(observe(O) \rightarrow E, w \propto p_D)$ and $(observe(O) \rightarrow E, w \propto 1 - p_D)$. The variable O has a fixed value that represents the corresponding measurement. It is specific to this grounding. The predicate $observe(O)$ will not take part in any logical deduction and is always assumed true. This formulation allows evidence from related sources (beside O) to have their effects on $p(E = true)$.

Extensional predicates can be of various kinds depending on the domain under consideration. Two classes and their associated uncertainty that we consider are object recognition and action detection (see section 5.7.1).

- *Object Detection* Human and car detection are performed using methods described in [85]. Since humans and cars often move, at least for some period of time, we eliminate false detections by keeping only hypotheses that are temporally stable. Hence, we assume that there is no uncertainty with the remaining hypotheses. For other objects such as bag, package, or tools, we consider only specific object detection. (Object class detection is not the focus of our work). Detection probability is calculated based on the matching between the color histogram and shape of the observed foreground and those of targets. (Scale is assumed known approximately based on rough layout of the scene).

- *Action Recognition* We detect two kind of actions: $shakehand(h_1, h_2)$ and $openTrunk(c, h)$. The first action is detected when two persons stand close

but separate and there is a foreground area (located along the torso of each person) that connects them together. The detection probability is measured using the similarity between this area and the stored target templates. The open trunk action is detected by comparing the motion histogram of the trunk area against the stored exemplars when the person stands next to the trunk.

5.5.2.4 Identity Maintenance

Identity maintenance is necessary when there exist multiple identities that actually refer to the same object([69], [73]). In surveillance, it is caused by lack of visual information (appearance, shape...) to make unique identity connections across observation gaps. Solving this problem is critical to recognizing complex activities, which often span over extend periods of time and in different parts of the scene. In [[69], Shet et al addressed it using a default logic. Our approach here is similar to the one proposed in [73] for entity resolution in relational databases ([28]), with a slightly more concise formulation.

Identification of two objects A and B is represented by a predicate $eq(A, B)$. It comes with the following set of axioms (with infinite weights):

- Reflexive: $eq(A, A)$.
- Symmetry: $eq(A, B) \leftrightarrow eq(B, A)$.
- Transitivity: $eq(A, B) \wedge eq(B, C) \rightarrow eq(A, C)$.
- Predicate Equivalence: $P(X_1, Y) \wedge eq(X_1, X_2) \rightarrow P(X_2, Y)$. (For two-ary

predicates but can be similarly stated for n-ary predicates).

The last axiom is important since omitting it may lead to incompleteness. For example, without this axiom, the rule $P_1(X) \wedge P_2(X) \rightarrow P_3(X)$ would not fire to generate $P_3(X_1)$ or $P_3(X_2)$ given the ground atom database $\{eq(X_1, X_2), P_1(X_1), P_2(X_2)\}$, as it should. In practice, only the third and the fourth rules are grounded and entered into the Markov network, while the first two rules are used only for making deductions. This is to avoid overcrowding the network, which affects the inference performance.

The equivalence predicate can be extensionally evaluated or intensionally inferred. Extensional evaluation of $eq(A, B)$ is done using appearance matching. The probability $p(eq(A, B) = true)$ is calculated based on a matching score. Intensional deduction of $eq(A, B)$ can be done using the above axioms and commonsense rules in the KB. Several prioritized rules in [69], such as "possession of some special objects (e.g. car keys) determines owners' identity", can be used here, where each prioritizing level is mapped to a corresponding weight.

5.6 Network Construction

This section describes our deduction algorithm that uses the production rules in the KB (section 5.5.1) to deduce grounded atoms for the Markov network. Due to noise or incompleteness in observations, some events that have not actually occurred might get grounded and added to the ground atom database (ADB). Our procedure is thus a relaxed version of logical deduction and may not be logically consistent.

5.6.1 Deduction Algorithm

Typically, with definite clauses, deduction is performed via forward chaining. In our system, logic rules take richer forms that require us to additionally deal with negative preconditions and disjunctive conclusions. Following are several preliminaries for our algorithm.

- *Close World Assumption(CWA)* Since it is usually not convenient and sometimes impossible to detect (consistently) events that are not happening, such as the $notstopped(C_i, t)$ event (for all cars at all time points), the CWA is used to check for negative preconditions: what is not currently known to be true is assumed false. Then, forward chaining is still used, but is divided into two phases: the first for rules that do not have negative preconditions and the second for the remaining rules. This is to delay, for example, the conclusion that $\neg a$ is true using CWA until all possible ways of deducing a have been tried.

- *Context-dependent Preconditions* Consider the predicate $nearby(P, loc, t)$ in the formula $happenAt(E, loc, t) \wedge nearby(P, loc, t) \rightarrow witness(E, P)$. It would be cumbersome to evaluate $nearby(P, loc, t)$ and add it to the ADB for all people P, all locations loc and all times t. Instead, it should only be evaluated after $happenAt(E, loc, t)$ is true with specific bindings of loc and t. In this case, the satisfaction of the first precondition serves as the context that enables the lazy evaluation of the second one. Generally, we use lazy evaluation for an extensional predicate when it would be expensive to evaluate otherwise due to

the large size of the domain (e.g. ones that involve time or location).

- *Disjunction Domains* Generally, in our system, disjunctions need no special treatment. However, when they are in the scope of an existential quantifier, domain expansion and several bookkeeping steps are required. Consider the statement "if a person H disappears at some location L at time T, that person must enter a car C that is close to L". It can be formalized as $disappears(H, L, T) \rightarrow \exists C\ close(C, L, T) \wedge enters(C, H, T)$. If $close(C, L)$ is evaluated to true, for example, for only two cars C_1 and C_2 given some location L and person H, then this clause can be grounded as $disappears(H, L, T) \rightarrow ((close(C_1, L, T) \wedge enters(C_1, H)) \vee (close(C_2, L, T) \wedge enters(C_2, H)))$. Here, the predicate $close(C, L)$ limits C to the set $\{C_1, C_2\}$ and the existential quantifier is expanded over the entirety of this domain. In general, we eliminate the existential quantifier by considering that the conclusion has two parts, one for defining the object domain ($close(C, L)$) and the other for describing the actual conclusion ($enters(C, H)$). In other words, our general production rule would be $precondition \rightarrow (\exists x\ domaindef_x \wedge conlusion_x)$. An empty clause $domaindef_x$ implies that the domain consists of all instantiations of x.

  During deduction, we may need to expand domains as new objects that satisfy domain predicates are discovered. In such cases, the previous ground formula is replaced with the new one and the Markov network is modified with the new clique.

- *Generation of Extensional Predicates* Usually, extensional predicates are gen-

erated by the low-level module. However, logical constraints might be formulated in a way that leads us to generate extensional predicates using deduction. For example, consider the constraint "if a person h is occluded by a car c, then that person has to appear when the car leaves" which can be formalized as $occluded(c, h, t_1) \wedge carleave(c, t_2) \wedge less(t_1, t_2) \rightarrow appear(h, t_2)$. Here $appear(h, t_2)$ is strictly extensional. It should not be entered into the ADB unless it has been directly observed or an $appear(h', t_2)$ has been observed for some h' and $eq(h', h)$ is true (i.e. h' is actually h, but, due to some breakdown in identity maintenance, it is assigned a different identity). To capture this, generally, for a strictly extensional predicate $P(X_1)$ we add the following domain-independent rule to our KB

$P(X_1) \rightarrow (observe(PX1) \vee (\exists X_2\ eq(X_1, X_2) \wedge P(X_2)))$

where, again, $PX1$ is a "dummy" observation variable and $eq(X_1, X_2)$ defines the domain for the existential quantifier.

The deduction procedure is shown in Fig. 5.3. In step 1(a), when grounding a clause, if context-independent preconditions are satisfied then context-dependent predicates will be extensionally evaluated. Instances that are evaluated to true will be added to the ADB. In step 1(b), all atoms in the conclusion as well as their complemented literals (i.e. E and $\neg E$) are added to the ADB. If an existential quantifier is involved, we need to check and update, if necessary, its previously expanded domain. Step 2 essentially repeats step 1 with the addition of the CWA. For a precondition, $\neg E$, if we are unable to observe or deduce E, $\neg E$ is assumed true. If the related clause

Ground Atom and Formulae Deduction

- *Input* ADB - ground atom database; KB_{pos} - set of definite rules; KB_{neg} - set of rules that have negative preconditions
- *Output* ADB - with new ground atoms added; GS - the set of grounded clauses.

Repeat until no new ground atom is generated

1. Repeat until no new atom

   For $\forall R \in KB_{pos}$, instantiate R w. r. t. ADB and for each instantiation r,

   (a) If all context-independent preconditions are satisfied, then evaluate all context-dependent preconditions and add the newly evaluated atoms to ADB.

   (b) If all succeeded, get effects and add to ADB.

   (c) $GS \leftarrow GS \cup r$.

2. Similar to step 1 for $R \in KB_{neg}$ with CWA added during instantiation of rules.

Figure 5.3: The algorithm for deducing new ground atoms

ends up being grounded (i.e. all other preconditions are evaluated to true) then the literal $\neg E$ is added to the ADB.

All ground clauses are then added to the Markov network. This construction procedure is performed whenever there is a new event generated. It can be done incrementally by deriving only deductions that originate from new events.

5.6.2 An Example

Suppose that we are monitoring a scene with a parked car C_1. At time T_0, person P_1 appears. Then at T_1, P_1 disappears behind C_1. Shortly after that, at time T_2, C_1 leaves and a person appears from behind C_1. This person might be a new person P_2 or could be the person, P_1, previously seen. We would like to know how likely it is that the person P_1 was occluded or that he entered car C_1 (and left with it). Assume that we have a knowledge base that consists of the rules shown in Fig. 5.4.

Here, extensional predicates are $carleave(c,t)$, $less(t_1,t_2)$, $nearby(c,l,t)$, $disappear(h,l,t)$, $observe(o)$, $eq(h_1,h_2)$ and $appear(h,t)$ (all strictly extensional); intensional predicates are $occluded(c,h,t)$ and $enter(c,h,t)$. Given the above scenario, we will now show how our Markov network is constructed according to the algorithms presented in the last section, and how probability distributions are updated as more evidence arrives.

At time T_0 Event $appear(H_1,T_0)$ is constructed by the low-level vision module. The database of ground atoms is updated as $ADB = \{appear(H_1,T_0)\}$. No rule is

F1 If a person h disappears, then either h is occluded by a nearby car or h has entered a nearby car ($nearby(c, l, t)$ defines the domain for the quantifier)

$disappear(h, l, t) \rightarrow (\exists\ c\ (nearby(c, l, t) \wedge occluded(c, h, t)) \vee (nearby(c, l, t) \wedge enter(c, h, t)))$

F2 Entering a car and being occluded by that car are mutually exclusive

$enter(c, h, t) \rightarrow \neg occluded(c, h, t)$

F3 If person h enters a car c at t_1, h will not reappear for all $t_2 > t_1$

$enter(c, h_1, t_1) \wedge appear(h_2, t_2) \wedge less(t_1, t_2) \rightarrow \neg eq(h_1, h_2)$

(if h gets into c temporarily and gets out of it, h is not considered to have entered c, just being occluded)

F4 If person h is occluded by a car, then h should become visible when the car leaves $occluded(c, h_1, t_1) \wedge carleave(c, t_2) \wedge less(t_1, t_2) \rightarrow appear(h_1, t_2)$

F5 (Implicit - domain independent) If a strictly extensional predicate is generated, either we have to observe it or it comes from the predicate equivalence axiom. E.g.

$appear(h, t) \rightarrow (observe(HT) \vee \exists\ h'\ (appear(h', t) \wedge eq(h, h')))$

where HT is the observation associated with this extensional predicate, unique to each grounding

Figure 5.4: The knowledge base for our example. All rules Fi have the maximum weight.

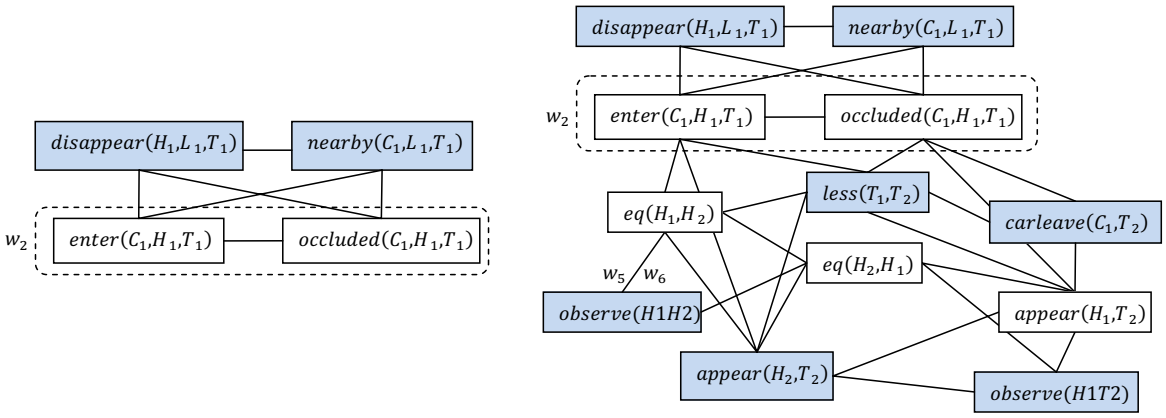


Figure 5.5: Left: the ground network, after T_1, a "clique" formed from rule **F2** is also shown. Evidence nodes are shaded. Right: after T_2. Predicate equivalence atoms and clauses are omitted for clarity.

grounded (i.e. the network is empty).

At time T_1 Event $disappear(H_1, L_1, T_1)$ is constructed by the vision module. Rule **F1** is then fired by the deduction module. The predicate $nearby(c, L_1, T_1)$ is contextually evaluated and a domain for c is established $D_c = \{C_1\}$ (i.e. C_1 is the only car near location L_1). The grounded rule would be $disappear(H_1, L_1, T_1) \rightarrow$ $(nearby(C_1, L_1, T_1) \wedge occluded(C_1, H_1, T_1))$ $\vee$ $(nearby(C_1, L_1, T_1) \wedge enter(C_1, H_1, T_1))$. The ADB is updated, $ADB \leftarrow$ $ADB \cup \{disappear(H_1, L_1, T_1), nearby(C_1, L_1, T_1), \pm enter(C_1, H_1, T_1),$ $\pm occluded(C_1, H_1, T_1)\}$. Rule **F2** is grounded with C_1 and H_1, e.g. $enter(C_1, H_1, T_1) \rightarrow \neg occluded(C_1, H_1, T_1)$ The grounded network is visualized in Fig. 5.5(left).

Running queries for $p_1(occluded(C_1, H_1, T_1) = true)$ and $p_2(enter(C_1, H_1, T_1)$ $= true)$ gives the results $p_1 = 0.5$ and $p_2 = 0.5$. This shows that given no

further evidence, we cannot conclude if H_1 entered C_1 or is occluded by it.

At time T_2 Two extensional events $carleave(C_1, T_2)$ and $appear(H_2, T_2)$ are constructed. $eq(H_1, H_2)$ is evaluated with the appearance of H_2. Suppose the evaluation leads to $p_3(eq(H_1, H_2) = true) = 0.8$, the following rule and its complement(section 5.5.2) are generated, $observe(H1H2) \rightarrow eq(H_1, H_2)$,$w_5 = p_3 MAXW$, where $observe(H1H2) = true$ is the evidence variable. Variables of this type are not added to the ADB since they will never be used for deduction. Equivalence axioms are also grounded with $MAXW$, e.g., $eq(H_1, H_2) \leftrightarrow eq(H_2, H_1)$. Below, for clarity, ground atoms and rules that come from predicate equivalences are omitted from the ADB and figures. Next, the temporal constraint $less(T_1, T_2)$ is satisfied and rules **F3** and **F4** are grounded with $MAXW$. E.g. $occluded(C_1, H_1, T_1) \wedge carleave(C_1, T_2) \wedge less(T_1, T_2) \rightarrow appear(H_1, T_2)$. The database of ground atoms is then $ADB \leftarrow ADB \cup \{\pm eq(H_1, H_2), \pm eq(H_2, H_1),\ appear(H_2, T_2), \pm appear(H_1, T_2),$ $carleave(C_1, T_2)\}$. Note that we do not add the temporal relation $less(T_1, T_2)$ to ADB. Since $appear(H_1, T_2)$ is derived, rule **F5** is fired with its corresponding ground atoms. $observe(H1T2)$ is considered an evidence variable and is set to $false$, since it has not been observed (not in ADB). The ground network now becomes the one shown in Fig. 5.5. Note that evidence nodes do not contribute significantly to network complexity. Now, running queries $p_1(occluded(C_1, H_1, T_1) = true)$ and $p_2(enter(C_1, H_1, T_1) = true)$ gives $p_1 = 0.81$ and $p_2 = 0.21$. Intuitively, since H_2 appears similar to H_1, the sys-

tem inclines toward concluding that H_1 did not enter C_1 but was initially occluded by C_1. Now, suppose that H_1 and H_2 look quite different, e.g. $p_3(eq(H_1, H_2) = true) = 0.3$. Modifying the weights w_5 and w_6 appropriately, and rerunning the queries we obtain $p_1 = 0.28$ and $p_2 = 0.72$. This matches our belief that the H_1 entered C_1 and that H_2 is a new person. (To further explain where H_2 came from requires more rules to be added to our KB. We do not address this here).

By using (relaxed) deduction, numerous irrelevant ground predicates such as $occluded(C_1, H_1, T_2)$ $enter(C_1, H_1, T_2)$ and $disappear(H_1, L_1, T_2)$ were not added to the network. Compared to a complete grounding (28 nodes)([65]), this deduction-based grounding certainly simplifies the network structure (16 nodes, including ones that are generated by predicate equivalence - not shown in Fig. 5.5) and hence reduces the burden on inference significantly.

This example illustrates how our system was constructed and adapts as new information arrives. We can identify a number of issues that would pose challenges to previous approaches. First, uncertainty such as the uncertainty associated with $eq(H_1, H_2)$ is commonplace in visual surveillance. Clearly, a purely logical framework would not be able to deal with this. Second, rules with disjunctive effects such as **F1** are often necessary to capture events. However, they are often not allowed in traditional deductive systems due to strict logical consistency requirements. Third, in a probabilistic approach such as HMMs or dynamic Bayesian networks, it is difficult to model variations in the number of agents (such as the number of cars

or persons in our example) which often arise in surveillance scenarios. Identity maintenance would even present more challenges to them. Our system, on the other hand, possesses mechanisms to deal with these issues, as illustrated in the example.

5.7 Implementation and Experiments

5.7.1 Implementation

We describe here some basic elements needed to address the parking lot application: object set, predicate set, their evaluation and the KB. Three types of objects are considered: cars (denoted as C_i), humans (H_i) and locations (L_i). Time is represented using atomic intervals with granularity of n_I frames (e.g. $n_I = 30$, approximately 2 seconds). Each primitive event or action is assumed to be true within one time interval. Below, time labels are omitted for clarity. Our vocabulary consists of the following predicates:

- Extensional: $inTrunkZone(C, H)$, $inLeftZone(C, H)$, $inRightZone(C, H)$, $disappear(H, L)$, $equal(H_1, H_2)$, $shakeHand(H_1, H_2)$ and $carLeave(C)$ (context-independent), $openTrunk(C, H)$ (context-dependent).

- Intensional: $enter(C, H)$ and $drive(C, H)$

Additionally, we have measurement objects and their corresponding predicates (Sec. 5.5.2.3), $observe(O)$.

Background subtraction, human detection and tracking (see e.g. [18]) techniques were first applied to identify and track object locations. The orientation and

direction of each car were estimated simply using its corresponding foreground blob and parking lot layout. Fig. 5.6.1 shows the estimated layouts of the three detected cars during one experiment.What is critical to inference is locating the left, trunk and right zones of a car when it comes to a halt.

A spatial predicate, for example, $inTrunkZone(C, H)$, is generated when the foot location of person, H, intersects significantly with the trunk zone of the car, C, for a sufficiently long period of time; $disappear(H, L)$ is generated when we lose track of H. Identity maintenance predicates are evaluated using the distance between color histograms of the two participating objects. $shakeHand(H_1, H_2)$ is modeled by analyzing the connecting area between two standing separate persons. $openTrunk(C, H)$ is evaluated base on the motion pattern in the trunk area of car C. The rules that constitute our knowledgebase are listed in the appendix. The maximum weight, $MAXW$, is set to be proportional to the network's size (number of ground atoms [41]). The range $0 - MAXW$ is uniformly discretized to five levels corresponding to very strong, strong, medium, weak and very weak certainties. These values are assigned to rules according to our confidence in them, based on domain knowledge.

5.7.2 Experiments

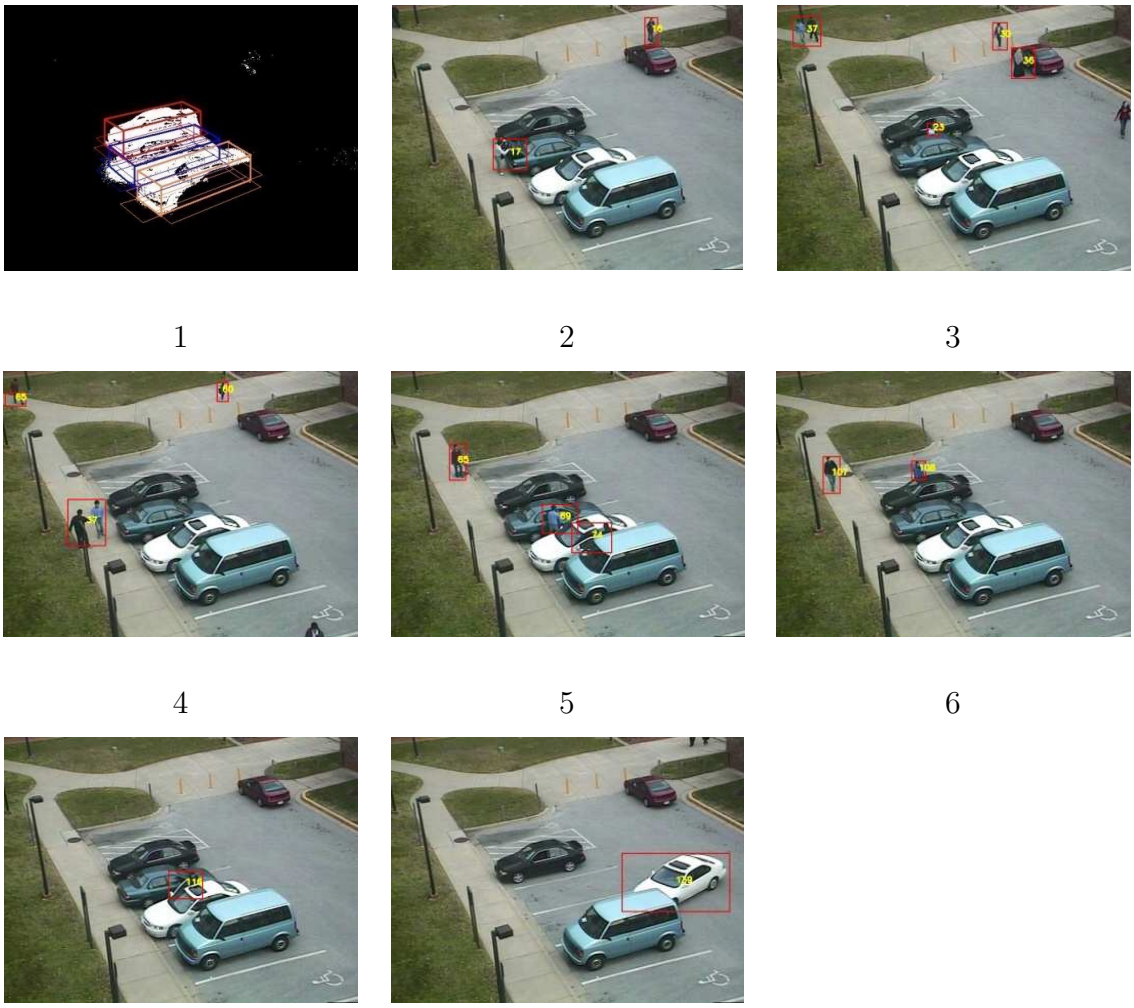
We analyzed a set of parking lot videos that involve a number of people entering different cars as listed in Table 5.1. A typical scenario is as follows. Initially, three cars, C_1, C_2 and C_3, park next to each other. A person H_1 appears, walks up to C_2,

Table 5.1: Four sequences used in our experiments

	♯ of people	♯ of cars	Durations
seq 1	6	3	2 min 10 sec
seq 2	5	3	3 min
seq 3	4	2	1min 30 sec
seq 4	6	3	4 min

opens its trunk (Fig. 5.6.2), puts something in, closes the trunk and then disappears between C_1 and C_2 (Fig. 5.6.3). Then two persons, H_2 and H_3, walk close to each other near the parked cars. They shake hands (Fig. 5.6.4) and disappear between C_2, C_3 and around the left of C_3 respectively (Fig. 5.6.5). Person H_4 walks to C_1 and disappears from the passenger side of C_1 (Fig. 5.6.5). A person H_5 follows a similar path (Fig. 5.6.6). Person H_6 walks to the cars and disappears between C_2 and C_3 (Fig. 5.6.7). Then C_1 pulls out and leaves. Finally, C_2 and C_3 follow in order (Fig. 5.6.8). This scenario consists of a number of interesting interactions and we would like to query the system about who entered or drove which car. The challenges arise mostly due to noise and occlusion, which lead to loss of track well before a person enters a car.

As the scenario unfolds, new events are generated and our ground network evolves accordingly. We can query our system at any instant of time. Here, we ran queries after all cars had departed. Detection probabilities for $openTrunk(C_2, H_1)$ and $shakeHand(H_2, H_3)$ were respectively 0.9 and 0.5. Identity confusion is not



1 2 3

4 5 6

7 8

Figure 5.6: 1) Estimated layouts of the parked cars. 2-8) Key frames from the scenario. Foreground and human detection results are also shown. Irrelevant people and cars are removed.

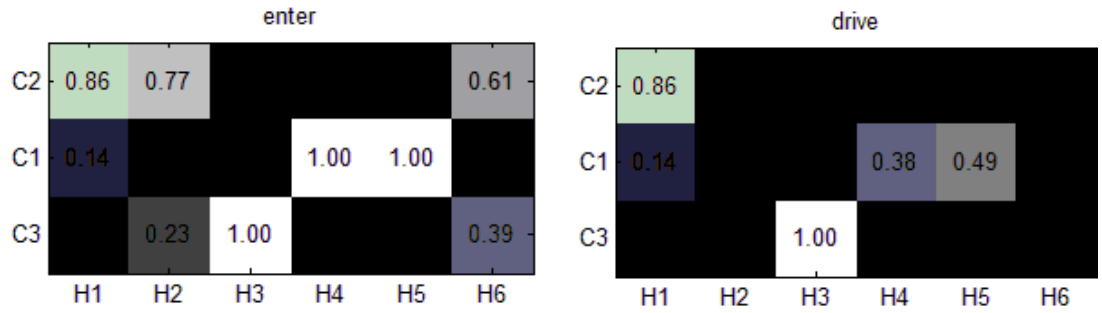


Figure 5.7: Right, probabilities of entering cars for individual persons, $p(enter(c,h) = true)$. Left, probabilities of driving cars, $p(drive(c,h) = true)$. Darker means being lower in value

significant so no related ground atom is generated. Fig. 5.7 shows the results for our queries, the probabilities $p(enter(c,h) = true)$ and $p(drive(c,h) = true)$ for all cars C_i's and human H_i's.

These results can be explained intuitively as follows. Person H_1 disappeared between car C_1 and C_2, so he could have equally entered either of them (with $p = 0.5$). But since he was observed to open C_2's trunk, the probability that he entered C_2 increased to 0.86. Person H_2 could have entered either C_2 or C_3 (each with $p = 0.5$). But he had been detected shaking hands with H_3 (and so probably saying "goodbye"), who entered C_3 with high certainty. Hence, the probability of H_2 entering C_3 was reduced and entering C_2 was increased. However, since the detection probability was not very high ($p = 0.5$), the increase was not as much as for the first person. Persons H_3, H_4 and H_5 entered cars C_3, C_1 and C_1 respectively with high certainties since there was only a single car close to them when they disappeared. Person H_6 disappeared between cars C_1 and C_3. Since there is no further supporting evidence, the probabilities for entering C_1 and C_3 should be the

same. The observed discrepancy is due mainly to sampling approximation.

H_1 and H_3 drove C_2 and C_3, respectively, with high certainty since they had been observed to enter their cars from the driver side and there was no competing alternative. For car C_1, H_4 and H_5 were observed to enter it from the passenger (right) side. The only person who could possibly enter it from the driver side was H_1. But it was strongly believed H_1 entered and drove C_2. Hence, no one entered C_1 from the driver side. This led to the conclusion that either H_4 or H_5 had moved from the passenger side to drive C_1. This reasoning was made possible because of the low-weight rule 5, which states that, although unlikely, drivers can enter a car from the passenger side (the driver side may be blocked). In general, this type of rule is added to increase the system's ability to handle exceptional cases. They bear some of the flavor of default reasoning in the sense that in the absence of supporting evidence for "normal" rules, rules with low weights will be activated to explain the situation under consideration. Note that rule 5 was also applied to other persons, such as H_2 and H_6, but produced negligible effects since competing hypotheses were much stronger. Therefore, the probabilities that H_2 and H_6 drove any car were still close to zero. We examine next how inference results change as we vary rules in our KB. First, suppose that the complement rule (section 5.5.2.3) $obseved(O) \rightarrow E, w = MAXW(1 - p_D)$ is omitted and hence the ground formula $obseved(SHKHAND) \rightarrow shakeHand(H_2, H_3)$ is dropped from our network. A rerun of the queries showed that $p(enter(C_2, H_2) = true) = 0.99935$ (everything else was unchanged) which essentially meant that person H_2 entered car C_2 with very high certainty. The result would also be the same no matter how

the detection probability for the shake-hand action was hypothetically adjusted as long as $w > 0$. This is obviously undesirable for this particular case since we would expect $p(enter(C_2, H_2) = true)$ to increase proportionately to the increase in the confidence of detecting the shake-hand action. Replacing the complement rule makes the system operate as expected.

In the initial querying, our system was able to conclude that either H_4 or H_5 drove car C_1 but was unable to determine which of them did. Consider adding to the KB a very weak rule stating that among the persons entering a car from the passenger side, whoever enters it first is its driver (no new extensional predicate evaluation is needed). When we do this and re-evaluate the queries, $p(drive(C_1, H_4) = true)$ increased to 0.60 while $p(drive(C_1, H_5) = true)$ dropped to 0.23, which matched the observation that H_4 entered C_1 before H_5 and drove it away. This example is intended to illustrate that domain knowledge can be easily added to our system to improve its performance. This would not be so easy in a purely logical system, because a significant amount of consistency checking is required before admitting a new piece of knowledge. Also, it would not be straightforward in a purely probabilistic system, since adding complicated structures involves cumbersome changes to the internal structure of the system.

Our system is implemented using C++ and runs in Windows and Cygwin on an Intel Pentium Dual-Core 1.6 GHz machine with 1GB RAM memory. The average running time for background subtraction, human detection and object tracking, on 640x480 RGB frames, are 5 frames per second.Primitive event detection time is negligible since events that are expensive to evaluate occur sporadically. Average

running time when performing inference for all queries presented here is approximately thirty seconds (maximum number of MCMC step is set to 5000). The number of ground predicates nodes) in the first test is 80, out of which 24 nodes are evidence. The corresponding number of ground clauses (cliques) is 120. Physical memory usage for the ground network was small(less than 2MB).

5.8 Discussion

We described how a combination of a probabilistic graphical model, the Markov Logic network, and first-order logic statements can be used for event recognition in surveillance domains, where unobservable events and uncertainties in detection are common. Logic provides a convenient mechanism to utilize domain knowledge to reason about the unobservable. Probabilistic models give us a coherent framework to deal with uncertainty. The combination brings us the ability to capture interesting interactions in complex domains.

Obtaining a sufficient and efficient knowledge base (KB) is important to our recognition performance. This issue is also receiving increasing attention in the area of human activity understanding([63], [45], [46], [81] ...). Approaches that use efforts from open communities to build common sense KB have been proposed such as Open Common Sense and Open Indoor Initiatives ([46], [81]). Large databases of rules are now available to the public ([46]). Automated rule extraction and weight learning from them have been attempted (e.g. [63]). At its current stage, for visual surveillance, rules and facts collected from such open databases are more often than

not irrelevant and not useful. For example, a query for "parking lot" would output relations such as it is "a place for parked cars", "would be hot in the summer", "may or may not be empty"... ([46]). However, as these knowledge bases grow and, possibly, specialize ([81]), their application to our framework seems promising. Exploiting them is part of our future investigation.

Chapter 5

Summary and Potential Research Directions

We have presented in this thesis approaches for robust foreground detection, object tracking and event recognition which are the main components of a visual surveillance system.

- The background model is built with a local linear prediction model. It allows us to accurately capture the variation of background color and hence improves the performance of background subtraction in the presence of extensive lighting change.

- The tracking problem was posed as the construction and updating of object occupancy maps. The motion of robust features is used as the basis for finding the motion of every pixel in the image, which is then used to determine the occupancy map. Experimental results show that the new tracker is quite insensitive to inaccuracy in initialization and robust to large appearance or lighting changes.

- A combination of probabilistic and logic reasoning is proposed for high-level event understanding. Commonsense knowledge is used to reduce uncertainties and ambiguities. They are represented with weighted first order logic formulae. A Markov network of ground atoms is constructed and updated as evidence arrives. Probabilistic reasoning is performed on this undirected

graphical model to determine the probability of interested (queried) events. The combined advantages of both logic and probability is demonstrated in a number of experiments.

Following are a number of potential research directions that based on the current work.

5.1 Background Subtraction

First, both indoor and outdoor scenes have been used to evaluate our method but only with a limited number of settings. A more extensive evaluation for the proposed method are needed. In addition to ROC analysis, PDR ([84]) can be used here.

Second, the global index is an important factor for the accuracy of the prediction models. It is currently computed as the median RGB value taken over all pixels of the input frame. This may lead to inaccuracies if the area of the foreground become significant compared to the area of the background. To deal with such situations, we might need a more selective (sparse) sampling approach. Manual scene masking can be used but automatic selection of representative sample points is more interesting.

Third, if objects come and stay still in the scene for a long time, it would be natural to incorporate them into the background model so that moving objects can be detected more accurately. This is useful in many situations in visual surveillance, for example, in a parking lot or in a street where there are many cars arriving,

parking or leaving. Currently there is no adaptive mechanism for such situations and an approach, possibly with layered background models, is needed. We may need to perform regional analysis and/or object detection instead of staying at the pixel level as we currently do. Approaches to this problem exists in the literature, but there are still open challenges.

5.2 Object Tracking

First, to make the tracker more robust, we need to explicitly handle occlusion, which has not been addressed explicitly in our current tracker. Currently, a partial occlusion of the tracked object would lead to shrinking of its occupancy map. Essentially, this will result in only some part of the object to be tracked; the remaining part would be ignored. An example is shown in Figure 5.1. The tracked object is occluded most at frame ♯17. Its occupancy map is not accurately recovered by the tracker until frame ♯64. We want to restore the actual occupancy map as quickly as possible. This problem is related to the second problem, multi object tracking. Currently, if a set of objects move close to each other, their occupancy maps can be confounded. Coordinating the tracking of multiple objects can help to prevent such merging and maintain the identity of each object.

5.3 Event Modelling and Recognition

Temporal models. In complex domains, interrelated events often occur at different time points with possibly different spans. Currently, time is represented

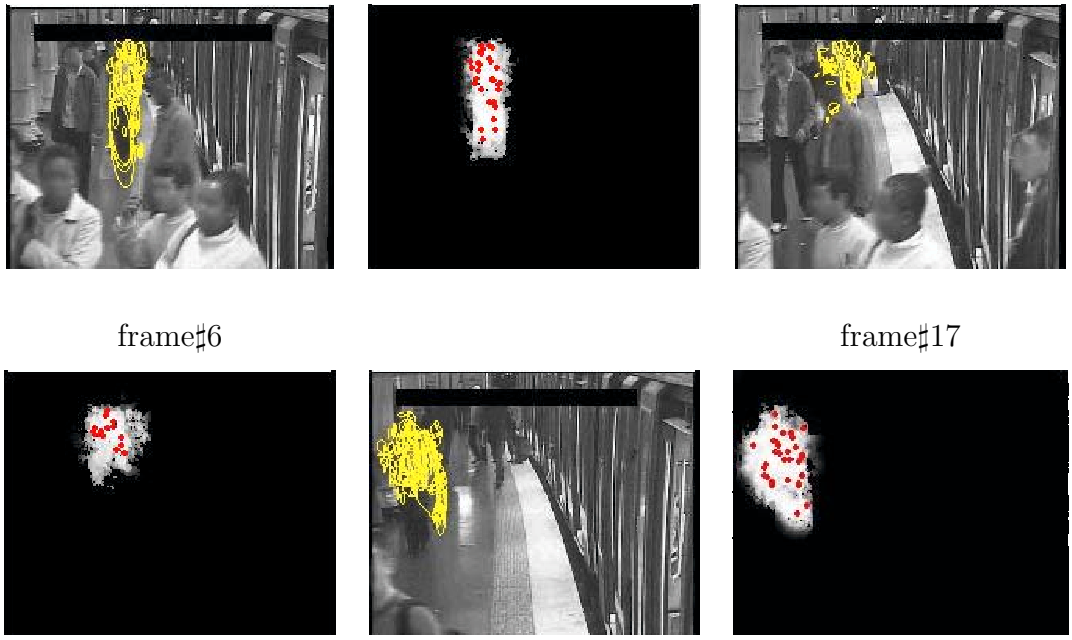


Figure 5.1: Example of tracking with significant partial occlusion at frame ♯17. It takes long time to recover the object's occupancy (till frame ♯64)

with points and intervals and used only for constraint checking. More sophisticated temporal models and reasoning are needed to model activities or events with complex temporal orderings or constraints.

Complex queries. The queries demonstrated in Chapter 4 are on the truth values of individual ground atoms. In general, queries can be more complex (e.g. formed by a conjunction and/or disjunction of the grounded predicates) and can involve quantification, for example "find every person H_i where $P(H_i)$ is true" for some predicate P. These kind of queries cannot be extracted readily from the Markov network. To answer them we may need a conversion procedure or augmented structures for the current Markov network.

Incremental probabilistic inference. Currently, inference for every query is performed from scratch. In particular, if the MCMC or MC-SAT is used, we re-generate the sample set every time a query is run. Even though the inference times on the networks in our experiments were small, it is still desirable to have faster inference, in case intensive querying occurs or when the network size is significantly large. There exist several effective incremental SAT solvers that can be used. Generally, we need to deal with several types of incremental changes: 1) the truth values of some previously evaluated extensional predicates are changed, 2) new rules are grounded, which leads to local changes in the network structure and 3) the knowledge base is revised, which leads to possible changes in the network potentials.

Automatic rule learning. The set of first order logic rules used in Chapter 4 is derived manually based on commonsense knowledge about the parking lot

and related activities. Obviously, this set of rules is quite domain specific and for different domains, different sets of rules are needed. This might be often acceptable since we need to construct a rule set only once for a given domain. Nevertheless, automatic derivation of relevant domain knowledge and automatic weight learning is still desirable. These may be possible based on exploiting large-scale commonsense knowledge bases that are publicly available.

Appendix A

Logic Rules

List of rules and their corresponding weights used for the parking lot problem (see problem description in Sec. 5.3).

1. If a person disappears, he/she enters a nearby car, $w = \frac{4}{5}MAXW$

   $disappear(h) \rightarrow (\exists c\ (inLeftZone(c,h) \vee inRightZone(c,h)) \wedge enter(c,h))$

2. If a person opens the trunk of a car, he/she will (likely) enter that car

   $disappear(h) \wedge openTrunk(c,h) \rightarrow enter(c,h)$, $w = \frac{4}{5}MAXW$

3. A person enters only one car: $enter(c_1,h) \wedge \neg equal(c_1,c_2) \rightarrow \neg enter(c_2,h), w = MAXW$ (if h gets into c temporarily and gets out of it, h is not considered to have entered c, just being temporarily occluded)

4. A person entering a car c from the left (driver) side will (likely) drive c

   $inLeftZone(c,h) \wedge enter(c,h) \rightarrow drive(c,h)$, $w = \frac{4}{5}MAXW$

5. A person entering a car c from the right (passenger) side will (less likely) drive c

   $inRightZone(c,h) \wedge enter(c,h) \rightarrow drive(c,h)$, $w = \frac{1}{5}MAXW$

6. Two persons shaking hand with each other will (likely) not enter the same cars.

   $shakeHand(h_1,h_2) \wedge enter(c_1,h_1) \rightarrow \neg enter(c_1,h_2)$, $w = \frac{4}{5}MAXW$

7. For a car to drive away, it needs a driver: $carLeave(c) \rightarrow (\exists\ h\ enter(c,h) \wedge drive(c,h))$, $w = MAXW$

8. A car has only one driver: $drive(c,h_1) \wedge \neg equal(h_1,h_2) \rightarrow \neg drive(c,h_2), w = MAXW$

9. A person can drive only one car: $drive(c_1,h) \wedge \neg equal(c_1,c_2) \rightarrow \neg drive(c_2,h), w = MAXW$

10. A person has to enter a car to drive it: $drive(c,h) \rightarrow enter(c,h), w = MAXW$

Bibliography

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. In *RCA Engineer*, volume 29, pages 33–41, 1984.

[2] K. Apt and R. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1994.

[3] S. Avidan. Ensemble tracking. In *Proc. CVPR05*, volume 2, pages 494–501. IEEE Computer Society, 2005.

[4] Y. Bar-Shalom and X. RongLi. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[5] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In B. Buxton and R. Cipolla, editors, *Proc. ECCV96*, volume 1 of *LNCS*, pages 329–342, Berlin Heidelberg New York, 1996. Springer.

[6] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. ICCV01*, pages 105–112. IEEE, 2001.

[7] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proc. ICCV03*, pages 26–33. IEEE, 2003.

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI*, 23(11):1222–1239, 2001.

[9] M. Brand, K. Kang, and D. Cooper. Algebraic solution for the visual hull. In *Proc. CVPR04*, volume 1, pages 30–35. IEEE, 2004.

[10] G. Brostow and R. Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *Proc. CVPR06*, volume 1, pages 594–601, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[11] P. Burt and G. van der Wal. An architecture for multiresolution, focal, image analysis. pages II: 305–311, 1990.

[12] G. K. M. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-

silhouette with stereo. In *Proc. CVPR03*, volume 2, pages 375–382. IEEE, 2003.

[13] K. C. Chou, A. S. Willsky, and A. Benveniste. Multiscale recursive estimation, data fusion, and regularization. *CONTROL*, 39(3):464–478, Mar 1994.

[14] R. Collins, Y. Liu, and M. Leordeanu. On-line selection of discriminative tracking features. *IEEE Trans. PAMI*, 27(10):1631–1643, 10 2005.

[15] D. Comaniciu and V. Ramesh. Mean shift and optimal prediction for efficient object tracking. In *ICIP00*, pages 70–73, 2000.

[16] G. Cross and A. Zisserman. Quadric surface reconstruction from dual-space geometry. In *Proc. ICCV98*, volume 1, page 2531. IEEE, 1998.

[17] A. Culotta and A. McCallum. Practical Markov Logic containing first-order quantifiers with application to identity uncertainty. In *HLT/NAACL Workshop*, pages 343–356, 2006.

[18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR05*, volume 2, pages 886–893. IEEE, 2005.

[19] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR00*, pages 2126–2133. IEEE Computer Society, 2000.

[20] A. Doucet, N. de Freitas, and N. G. Edits. Sequential monte carlo methods in practice. *Statistics and Computing*, 10:197–208, 2000.

[21] A. Elgammal. Learning to track: Conceptual manifold map for closed-form tracking. In *Proc. CVPR05*, pages 724–730, Washington, DC, USA, 2005. IEEE Computer Society.

[22] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. pages 751–767, London, UK, 2000. Springer-Verlag.

[23] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. In *Proc. 4th Int'l Conf. on 3D Digital Imaging and Modeling (3DIM 2003)*, pages 46–53, 2003.

[24] Z. Fan, M. Yang, Y. Wu, G. Hua, and T. Yu. Efficient optimal kernel placement for reliable visual tracking. In *Proc. CVPR06*, volume 1, pages 658– 665, Washington, DC, USA, 2006. IEEE Computer Society.

[25] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[26] G. G. Slabaugh, W. Culbertson, T. Malzbender, M. Stevens, and R. Schafer. Methods for volumetric reconstruction of visual scenes. *Intl J. Computer Vision*, 57(3):179–199, 2004.

[27] D. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *ICPR98*, pages 439–444. IEEE Computer Society, 1998.

[28] L. Getoor and B. Taskar. *Intro. to Statistical Relational Learning*. MIT Press, 2007.

[29] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Incremental density approximation and kernel-based bayesian filtering for object tracking. *CVPR01*, 01:638–644, 2004.

[30] B. Han and L. Davis. On-line density-based appearance modeling for object tracking. In *Proc. ICCV05*, pages 1492–1499. IEEE Computer Society, 2005.

[31] G. Hua and Y. Wu. Multi-scale visual tracking by sequential belief propagation. In *CVPR04*, pages 826–833, 2004.

[32] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In B. Buxton and R. Cipolla, editors, *Proc. ECCV96*, volume 1 of *LNCS*, pages 343–356, Berlin Heidelberg New York, 1996. Springer.

[33] M. Isard and A. Blake. Condensation -conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.

[34] J. Isidoro and S. Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Proc. ICCV03*, pages 1335–1342. IEEE, 2003.

[35] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Trans. PAMI*, 25(10):1296–1311, 2003.

[36] S. Joo and R. Chellappa. Attribute grammar-based event recognition and anomaly detection. In *CVPR06 Workshop*, pages 107–205. IEEE, 2006.

[37] K. K. Kutulakos and S. Seitz. A theory of shape by space carving. In *Proc. ICCV99*, pages 307–314. IEEE, 1999.

[38] A. Kale and C. Jaynes. A joint illumination and shape model for visual tracking. In *Proc. CVPR06*, volume 1, pages 602–609, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[39] M. Kilger. A shadow handler in a video-based real-time traffic monitoring system. In *Proc. IEEE Workshop Applications of Computer Vision*, pages 11–18. IEEE Computer Society, 1992.

[40] K. Kim, T. Chalidabhongse, D. Harwood, and L. S. Davis. Background modeling and subtraction by codebook construction. pages 3061–3064, Washington, DC, USA, 2004. IEEE Computer Society.

[41] S. Kok, P. Singla, M. Richardson, and P. Domingos. The Alchemy system for statistical relational AI. *Tech Report - CS Dept., Univ. of Washington*, 2005.

[42] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. pages 189–196, London, UK, 1994. Springer-Verlag.

[43] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. ECCV02*, volume 3, pages 82–96. Springer-Verlag, 2002.

[44] C. Liang and K. Wong. Complex 3d shape recovery using a dual-space approach. In *Proc. CVPR05*, volume 2, pages 878–896, San Diego, USA, 2005. IEEE.

[45] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using Relational Markov Networks. In *Proc. IJCAI05*, pages 773–778. Morgan Kaufmann, Inc., 2005.

[46] H. Liu. The ConceptNet Project - http://web.media.mit.edu/∼hugo/conceptnet/.

[47] D. Lowd and P. Domingos. Recursive random fields. In *Proc. IJCAI*, 2007.

[48] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl J. Computer Vision*, 60(2):91–110, 2004.

[49] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *ICCV99*, pages 572–578, 1999.

[50] E. Marinari and G. Parisi. Simulated tempering: a new Monte Carlo scheme. *Europhysics Letters*, 19(6):451–458, 1992.

[51] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC-2002*, pages 384–393, London, 2002.

[52] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. PAMI*, 26(6):810–815, 2004.

[53] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Siggraph 2000, Computer Graphics Proceedings*, volume 3, pages 369–374. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[54] K. Mikolajczyk, C. S. T. Tuytelaars, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Intl J. Computer Vision*, 65(1-2):43–72, 2005.

[55] J. Minker and D. Seipel. Disjunctive logic programming: A survey and assessment. In *Computational Logic*, pages 472–511, London, UK, 2002. Springer-Verlag.

[56] A. Motilal and L. Davis. A probabilistic framework for surface reconstruction from multiple images. In *Proc. CVPR01*, volume 2, pages 470–476. IEEE, 2001.

[57] R. Nevatia, T. Zhao, and S. Hongeng. Hierarchical language-based representation of events in video stream. In *Proc. of CVPRW on Event Mining*, pages 39–47. IEEE, 2003.

[58] H. Nguyen and A. Smeulders. Tracking aspects of the foreground against the background. In T. Pajdla and J. Matas, editors, *Proc. ECCV04*, LNCS, pages 446–456, Berlin Heidelberg New York, 2004. Springer.

[59] P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking: Linking identities using Bayesian network inference. In *Proc. CVPR06*, volume 2, pages 2187–2194. IEEE, 2006.

[60] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. ECCV06*, volume 3953 of *LNCS*, pages 592–605, Berlin Heidelberg New York, 2006. Springer.

[61] V. Parameswaran, V. Ramesh, and I. Zoghlami. Tunable kernels for tracking. In *Proc. CVPR06*, volume 2, pages 2179–2186, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[62] S. Paris, F. Sillion, and L. Long. A surface reconstruction method using global graph cut optimization. In *Proc. ACCV04*, volume 1. IEEE, 2004.

[63] W. Pentney, A. Popescu, S. Wang, H. Kautz, and M. Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *Proc. AAAI06*.

[64] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV02*, pages 661–675, London, UK, 2002. Springer-Verlag.

[65] M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62:107–136, 2006.

[66] N. Rota and M. Thonnat. Activity recognition from video sequences using declarative models. In *Proc. ECAI02*, pages 673–680, 2002.

[67] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1/2/3):7–42, 2002.

[68] M. Seibert and A. Waxman. Adaptive 3-d object recognition from multiple views. *IEEE Trans. PAMI*, 14(2):107–124, 1992.

[69] V. Shet, D. Harwood, and L. Davis. Multivalued default logic for identity maintenance in visual surveillance. In *Proc. ECCV06*, volume 4 of *LNCS*, pages 119–132, Berlin Heidelberg New York, 2006. Springer.

[70] V. Shet, J. Neumann, V. Ramesh, and L. Davis. Bilattice-based logical reasoning for human detection. In *Proc. CVPR07*, pages 1–8. IEEE, 2007.

[71] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[72] Y. Shi and W. Karl. Real-time tracking using level sets. In *Proc. CVPR05*, pages 34–41, Washington, DC, USA, 2005. IEEE Computer Society.

[73] P. Singla and P. Domingos. Entity resolution with Markov Logic. In *Proc. ICDM*, pages 572–582. IEEE, 2006.

[74] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *Proc. AAAI06*. AAAI Press, 2006.

[75] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *Proc. AAAI06*, pages 488 – 493, Boston, MA, 2006. AAAI Press.

[76] S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Proc. ICCV05*, volume 1, pages I:349–356. IEEE, 2005.

[77] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proc. CVPR00*, volume 1, pages 345–352. IEEE, 2000.

[78] J. Solem, F. Kahl, and A. Heyden. Visibility constrained surface evolution. In *Proc. CVPR05*, volume 2, pages 892–900. IEEE, 2005.

[79] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. pages 245–252, Washington, DC, USA, 1999. IEEE Computer Society.

[80] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *ICCV03*, pages 1063–1070, Washington, DC, USA, 2003. IEEE Computer Society.

[81] D. Stork. The OpenMind Initiative - http://www.openmind.org/.

[82] J. Sullivan, A. Blake, M. Isard, and J. MacCormick. Object localization by bayesian correlation. In *ICCV99*, page 1068, Washington, DC, USA, 1999. IEEE Computer Society.

[83] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 57(1):23–32, 1993.

[84] D. H. T. H. Chalidabhongse, K. Kim and L. Davis. A perturbation method for evaluating background subtraction algorithms. In *Proc. Joint IEEE Int. Workshop on VSPETS03*.

[85] S. D. Tran, Z. Lin, D. Harwood, and L. S. Davis. Umd_vdt, an integration of detection and tracking methods for multiple human tracking. In *Proc. CLEAR07*, pages 1–8. Springer-Verlag, 2007.

[86] T. Tuytelaars and L. V. Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. BMVC-2000*, 2000.

[87] G. Vogiatzis, P. Favaro, and R. Cipolla. Using frontier points to recover shape, reflectance and illumination. In *Proc. ICCV05*, volume 1, pages 228–235. IEEE, 2005.

[88] G. Vogiatzis, P. Torr, and R. Cippola. Multi-view stereo via volumetric graph-cuts. In *Proc. CVPR05*, volume 2, pages 391–399. IEEE, 2005.

[89] H. Wang, D. Suter, and K. Schindler. Effective appearance model and similarity measure for particle filtering and visual tracking. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. ECCV06*, volume 3 of *LNCS*, pages 606–618, Berlin Heidelberg New York, 2006. Springer.

[90] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. Rehg. A scalable approach to activity recognition based on object use. In *Proc. ICCV07*, pages 1–8. IEEE, 2007.

[91] J. Yamato, J. Ohya, and K. Ishi. Recognizing human action in time-sequential images using Hidden Markov models. In *Proc. CVPR92*, pages 379–385. IEEE, 1992.

[92] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *Proc. ICCV05*, pages 212–219, Washington, DC, USA, 2005. IEEE Computer Society.