Bayesian Visual Surveillance

From Object Detection to Distributed Cameras

Bayesian Visual Surveillance

From Object Detection to Distributed Cameras

ACADEMISH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. mr. P.F. van der Heijden
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit
op dinsdag 17 januari 2006, te 12:00 uur

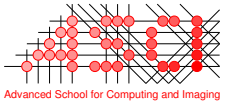
door
Wojciech Piotr Zajdel
geboren te Kraków, Polen

Promotiecommisie:

Promotor:	Prof. dr. ir. F.C.A. Groen
Co-promotor:	Dr. ir. B.J.A. Kröse
Overige leden:	Prof. dr. ir. D.M. Gavrila
	Prof. dr. J. Biemond
	Prof. dr. J.L. Crowley
	Dr. ir. T. Heskes
	Dr. ir. F.P. Voorbraak

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

This work was supported by the technology Foundation STW (project nr. ANN.5312) applied science division of NWO and the technology program of the Dutch Ministry of Economic Affairs.



This work was carried out in graduate school ASCI.
ASCI dissertation series number 119.

This book has been typeset by the author using LaTeX 2$_\varepsilon$.
Cover design by: Krzysztof Radoszek.
Produced by: Nowy Projekt (www.nowyprojekt.pl).

ISBN: 90-9020252-8

CONTENTS

CHAPTER 1

INTRODUCTION

In 1949, when Eric Blair (a.k.a George Orwell) published the "1984" novel, the idea of omnipresent surveillance systems was considered a dangerous utopia. When 50 years later, in 1999, John de Mol introduced his "Big Brother" show on Dutch TV, the idea proved feasible and made a spectacular comeback. Despite occasional controversies, in recent years visual surveillance systems seem to have made steady progress on their way from evil and oppressive to practical and indispensable. Societal developments, like the aging of population, globalization, increased mobility have lead to an increased demand for computer systems for assistance in safety, comfort or communication. These trends, together with steady advances in technology, inspire research on "smart" environments that observe the users in it and make decisions on the basis of the measurements. Examples are safety systems for elderly, traffic control systems, or security systems in public places. In order to be "aware" of what is going on, these systems are equipped with various sensors, such as infrared detectors, radar, cameras, microphones, mounted at many locations.

Visual surveillance Many systems rely exclusively on cameras as the sensing devices. Such vision-based systems attract research attention due to common availability of cameras, for example, embedded in the closed-circuit TV (CCTV) networks. Moreover, video data (in contrast to e.g., radar data) can be directly interpreted by humans and provide relatively high-resolution measurements that give access to many details about the environment. On the downside, high sensor resolution implies large quantities of video data that require tedious analysis in order to find out specific information of interest. While human operators are still indispensable to analyze highly-complicated scenes, computer techniques potentially reduce the effort of an operator, by e.g., processing video fragments with fairly structured scenes, and only notifying the operator in case of "suspicious" events. Besides the above on-line support for security personnel, computer-based analysis of video might also be applied to off-line processing of pre-recorded video material.

Visual surveillance embeds a range of image understanding techniques that automate the analysis of video data in order to extract various emergency events in machine-

interpretable way. These techniques can be studied in a variety of settings and configurations due to the fact that the information of interest is highly domain-specific. In the simplest case, one aims to detect and count some specific objects of interest, like humans or vehicles. More elaborate applications involve analyzing a sequence of video frames and following an object through the sequence. The most sophisticated scenarios entail interpreting behavior of an object to detect theft, violence, unattended luggage or illegal vehicle maneuvers.

An important aspect of visual surveillance is the camera setup. Basic configurations rely on a single camera and assume that the scene of interest can be completely covered by the camera field of view. Alternatively, one can consider multiple cameras that observe the same scene. Such configurations are useful for extracting depth information about the scene (e.g., stereovision) and resolving cases when objects in the scene occlude each other. Finally, in wide areas (eq. airports, highways or shopping centers) the field of view of a single camera cannot fully cover the entire region of interest. Therefore, wide-area surveillance largely relies on multiple, and importantly, sparsely distributed cameras. Sparse distribution implies that the cameras provide information only about selected areas — disjoint scenes spread over the complete area under surveillance.

Issues in visual surveillance Automated visual surveillance is a complex task, since it involves analyzing raw video data in order to find relatively high-level summaries of the video content. Given this complexity automated surveillance methods typically follow a hierarchical approach, where the analysis occurs at a number of intermediate levels (Collins et al., 2001; Haritaoglu et al., 2000; Wren et al., 1997; Zhao and Nevita, 2004). At the basic level(s), the analysis of raw pixels yields pixel annotations or "events" that summarize of the video content at various levels of abstraction. The higher-level processing steps take events as input and produce higher-level events until the desired information can be extracted in a machine-readable form.

Regardless of the high-level goals (e.g. theft, violence detection), typical visual surveillance applications need basic "bookkeeping" of the objects of interest. From the viewpoint adopted in this thesis, it is convenient to distinguish between the following bookkeeping tasks: (i) detecting an object in a video frame, (ii) keeping track of the object across multiple frames of video stream of a single camera, and (iii) keeping track of the object in video streams of different cameras. Since these tasks constitute a significant part of the surveillance research, we review them briefly.

- **object detection** Object detection problems can be considered as the lowest level video analysis tasks. In essence, object detection is a pixel classification problem: every pixel in the frame of video sequence has to be classified as a "non-object" pixel or a pixel representing (one of the) objects of interest. Examples include face detection, vehicle detection (Huang and Russell, 1998; Kato et al., 2002), pedestrian detection (Gavrila and Giebel, 2001). The classification (or detection) results from a variety of application-specific criteria. One possibility is to train a classifier

using prior examples of the objects (e.g. face detection). Another possibility entails introducing assumptions about the scene (e.g., the detected object are in motion, while the other objects are static (Rittscher et al., 2000; Stauffer and Grimson, 1999; Sullivan et al., 2000)).

- **multi-object tracking with single camera** Once an object is detected in a frame, visual tracking is the process of finding the location of that object in subsequent frames of the video sequence. When tracking multiple objects simultaneously present in the field of view, one has to resolve the correspondence between objects in the current frame with the objects in the previous frame (so called, *data association* problem). Tracking largely relies on the assumption of smooth motion, which allows to predict object's location in new frame on the basis of past locations. Additional cues follow from that fact that appearance (e.g., color, texture) of an object changes slowly between consecutive frames (Cai and Aggarwal, 1999; Dockstader and Tekalp, 2001; Hager and Belhumeur, 1998; Isard and Blake, 1998; Jang et al., 1997; Koller et al., 1994; Wren et al., 1997).
- **multi-object tracking with multiple cameras** Additional bookkeeping tasks arise when surveillance systems apply multiple cameras to collect the video data. Here, one has to associate appearances of an object from different video streams with each other. In case when the cameras observe the same scene, the object will appear simultaneously in all streams (Cai, 1997; Cai and Aggarwal, 1999; Li et al., 2002; Pedersini et al., 2001; Ruiz-Alzola et al., 2000; Ukita and Matsuyama, 2002), and association can be simplified by camera calibration techniques. Alternatively, every camera observes a scene disjoint from the scenes observed by the other cameras. In this case, association problems are usually more difficult, and have not been studied equally extensively. Examples include vehicle monitoring systems (Huang and Russell, 1998; Pasula et al., 1999) and tracking humans (Collins et al., 2001; Javed et al., 2003; Kettnaker and Zabih, 1999).

Objectives This thesis address selected problems that arise specifically in wide-area surveillance applications, that is, applications relying on sparsely distributed cameras. Our key interest are techniques that help bridge the visual gap between the sparsely distributed fields of view of the cameras. The intended contributions of the thesis lie in either novel probabilistic formulations of the discussed problems or novel probabilistic inference techniques applied to known probabilistic formulations.

The primary objective is reidentification of a person, as it moves between the areas visible to the cameras. Suppose, that we detect and track a person within the field of view of one camera. When the monitored person leaves the visible area tracking stops. Later the person appears at a possibly different camera; he or she is detected again. The question that we attempt to answer is whether the two observed objects are in fact the same individual. This objective is particularly relevant given the fact that current surveillance solutions tend to rely on systems of cameras, rather than just a single sensor. Further,

tracking with sparse cameras has received relatively moderate interest in the surveillance research so far (at least compared to other tracking problems).

The second objective is accurate detection of people in video frames. In the context of applications considered in the thesis, re-identification relies mainly on appearance of the people. Therefore accurate detection of humans in video is necessary for accurate characterization of appearance, which underlies accurate identification.

Probabilistic Approach One of the key issues in the mentioned tasks is the ambiguity of the video data. Factors like camera jitter, illumination changes, variation in object pose imply that the same person will appear differently each time it is observed. Additional ambiguities follow from the fact that various people will appear similar to each other or to other uninteresting objects in the scene.

In this thesis the inherent uncertainty in the video data is approached within a probabilistic framework. For every problem, we represent the input data and the sought output quantities as random variables. The variables representing the input are considered as *observed* variables. The variables representing the output are considered as *hidden* variables. The relation between all variables is expressed in terms of a probability distribution, also referred to as a probabilistic model. Given the model and the data, we apply probabilistic inference methods to estimate the relevant information.

Arguably, for many computer vision problems probabilistic methods turn out more suitable than alternative paradigms for reasoning with uncertainty, like fuzzy logic, default reasoning or rule-based methods (extensions to logical rule-based systems). Firstly, probabilistic methods typically scale better with the size of the problem by using vector-based implementations to manipulate probability distributions and appropriate approximation techniques. More importantly, in our view, probabilistic methods offer more principled approach to computational intractabilities arising in uncertain reasoning problems. In a probabilistic formulation the problem is first encoded by a probabilistic model and then solved by an inference method. When designing a model we focus on possibly exact encoding of all domain-specific knowledge. This approach separates knowledge representation from inference, which boils down to often intractable optimization or integration problems. These problems have been extensively studied in mathematics, physics and computer science communities, and there are many well-motivated efficient approximate algorithms available. A comparison of such algorithms for computer vision problems can be found in (Frey and Jojic, 2005).

Overview of the thesis Chapter 2 provides a comprehensive overview of probabilistic techniques, which form a unifying foundation for studying the problems considered in the subsequent chapters. The probabilistic methods can be conveniently studied in the graphical framework, where, in essence, a probability distribution is represented as a graph, and various computations involving probabilities correspond to manipulation of

the underlying graph. Graphical representation facilitates decomposing often complex computational problems in to a series of simpler subtasks, each defined on a subgraph.

In Chapter 3 we consider the problem of re-identification of multiple people that are being tracked through a wide area with sparsely distributed cameras. In the chapter this task is formulated as a problem of partitioning of a large number of observations (each representing an observed person) into several clusters (each grouping observations of a single person). Our aim is twofold: (i) to design a parametric probabilistic model that evaluates likelihood of hypothetical partitions, and (ii) to design a procedure that combines search for the optimal partition with the search for optimal model parameters. Essentially, the presented solution defines a dynamic Bayes network (DBN) as a probabilistic model for the observations of a single object. The edges of the network define the correspondences between observations of the same object. Accordingly, we derive an approximate EM-like method for selecting the most likely structure of DBN and learning model parameters.

Chapter 4 approaches the same problem in a different way. We assume that every person can be associated with an unique, albeit hidden label that identifies the person. We formulate a probabilistic model that ties the hidden labels of people with their measurements (observations) provided by a camera. Given the model and a series of observations, we consider the problem of inferring a corresponding series of labels. Inference is based on the assumed-density filtering algorithm which is applied to compute marginal posterior distributions on labels. From these distributions one can find the most likely labels, which resolve association ambiguities. The presented approach facilitates a principled estimation of the number of objects and various model parameters by Bayesian inference.

Chapter 5 describes a probabilistic approach aimed at detecting humans in video data. Specifically, given a video frame the aim is to accurately characterize the image region which represents a human. This problem can be viewed as the task of assigning to every pixel of an image a binary label that indicates whether the pixel represents a human or a background. Popular approaches for this problem consider the labels as random variables with an appropriate prior distribution, which ensures that the labeled pixels form spatially coherent regions. Typically, the prior takes the form of a Markov Random Field (MRF) model that, due to difficulties with learning, assumes only generic short-range coupling of pixel labels. This chapter presents an alternative model that takes spatial correlations into account in a more flexible way. The model can be easily learned from exemplary data to incorporate correlations characteristic for foreground objects in a given scene (e.g. human silhouettes). For inference in the model we derive an approximate, but computationally efficient Expectation-Propagation algorithm.

Chapter 6 summarizes the conclusions from the previous chapters of the thesis and indicates directions for future research.

CHAPTER 2

PROBABILISTIC GRAPHICAL MODELS

This chapter presents the formalism of probabilistic graphical models, which are the main tool for development of multi-object tracking techniques presented in this thesis.[1] We briefly motivate the suitability of probabilistic modeling for tracking, review the most common types of graphical models and present an overview of inference problems arising in various applications. Further, we focus on a specific class of inference methods — message-passing algorithms, which provide a unified approach for inference in models developed in the thesis. Finally, we discuss methods for probabilistic learning of model parameters.

2.1 Probabilistic Reasoning

Probabilistic reasoning is a formalism to express and process uncertain knowledge that inevitably arises in many complex real-world problems. In essence, probabilistic methods express uncertain statements by means of probabilities and reason under uncertainty by employing probability theory to manipulate the probabilities. These type of methods have gained wide-spread usage in various artificial intelligence applications, including, but not limited to, signal and image processing, machine learning, decision making, bioinformatics or information retrieval.

2.1.1 Motivation

Probabilistic methods derive their applicability from a number of factors that introduce uncertainty to our knowledge relevant to a given problem. First, our knowledge is based on incomplete or imprecise measurements of the physical quantities in question. The measurements might be incomplete due to limited resolution of the sensors, but might be also imprecise due to noise in the measurement process. Various statistical or

[1] Parts of the material in this chapter have appeared as a chapter (Zajdel et al., 2005a) in the *Intelligent Algorithms* book.

systematic artifacts of the process can be often easily captured by a probability distribution. Second, sometimes the physical mechanisms that underlie a given problem are not known exactly. Therefore, due to "ignorance" one approximates these mechanisms with probabilistic formulations. Finally, in some problems the underlying mechanisms are known, but are too complex for exact specification. Due to "laziness" we may deliberately approximate the complicated relations with simplified probabilistic relations.

In particular, the above factors occur in multi-object tracking problems discussed in this thesis. Therefore probabilistic reasoning provides a natural approach for these problems. In multi-object tracking the goal is to infer the identity of an observed object from the measurements provided by a camera. Since cameras cannot directly observe the identity of an object, tracking relies on related, but often ambiguous properties that can be measured, such as location or various appearance features. Moreover, noise sources, like camera jitter or shadows introduce additional uncertainty in the measured quantities. Therefore the relation between the identity of an object and its measurements is conveniently, although only approximately, formulated as a probability distribution.

2.1.2 Background

Our description of probabilistic methods assumes the following notation. We represent all quantities relevant to a given problem as random variables x_v, where $v \in V$ is a variable index and V denotes the collection of indices for all variables. A variable x_v takes values in a set denoted as $\mathcal{X}_v$. Thus, a multinomial (i.e. discrete) variable x_v with K possible states takes values in $\mathcal{X}_v = \{1, \ldots, K\}$. An m-dimensional real-valued (i.e. continuous) variable x_v takes values in $\mathcal{X}_v = \mathbb{R}^m$. For any subset $A \subseteq V$ we define as $x_A = \{x_v | v \in A\}$ the collection of the corresponding variables, which take values in a Cartesian product $\mathcal{X}_A = \prod_{v \in A} \mathcal{X}_v$. We identify two specific subsets of variables, denoted as H and E, such that $H \cup E = V$. The subset H contains indices of all unobserved (also known as latent or hidden) variables. The subset E contains indices of the observed variables, i.e. such variables for which we know the value (also known as the evidence).

The basis of any probabilistic method is a probabilistic *model*. A model encodes stochastic relations between all observed and unobserved variables relevant to a given problem. A model is a joint probability distribution (or *joint* for short), denoted as

$$p(x_V) = p(x_H, x_E).$$

Models that involve only discrete variables will be referred to as discrete models; models with exclusively real-valued variables — as continuous models, and models with both types of variables — as hybrid models. Importantly, in many applications defining a model proves complicated due to a large number of involved variables. Section 2.2 presents the formalism of graphical models, which provide a convenient language for design of large-scale probabilistic models.

Given the observed variables x_E, probabilistic methods answer queries about a hidden variable, say x_v, by probabilistic inference, that is, by computing a conditional probability distribution $p(x_v|x_E)$. From this distribution one can derive various characteristics of the variable, like the most likely value, the expected value, the confidence intervals, etc. It is worth noticing however, that probabilistic inference in large-scale model entails integration or maximization over often intractably large domains. Therefore, for many applications the required distributions cannot be computed in a tractable way. In Section 2.3 we review various exact and approximate algorithms for probabilistic inference.

Another important concept is probabilistic learning. Often a model is a parametric function $p(x_V|\theta)$ dependent on unknown parameters θ. Probabilistic learning attempts to estimate the parameters from a "training" data set x_D. There exist two main approaches to probabilistic learning: frequentist and Bayesian. The former finds a single optimal parameter configuration θ^* by maximizing a function that measures how well the parameters fit to the training data (we discuss two most common methods: MAP and ML learning). The later considers an augmented model $p(x_V, \theta, x_D)$ where parameters become additional random variables, which can be estimated by regular inference. Both approaches are discussed in more detail in Section 2.4.

Example Throughout the chapter we illustrate the discussed concepts with the following example. Suppose there is a room with two people. Each person wears uniformly colored clothes. Suppose that we take three pictures, such that at every picture there is only one person visible. For the person in the ith, $i \in \{1, 2, 3\}$, picture we compute a three-dimensional average color denoted as $c_i \in \mathbb{R}^3$. Given the colors c_i, we want tell which pictures present the same person. This information can be encoded by labels s_i, where each label $s_i \in \{a, b\}$ denotes the person in the ith image. In a probabilistic framework, color vectors c_i are considered as continuous evidence variables, $x_E = \{c_1, c_2, c_3\}$, and labels s_i as hidden discrete variables, $x_H = \{s_1, s_2, s_3\}$. We design a model $p(x_H, x_E) = p(s_1, s_2, s_3, c_1, c_2, c_3)$ from which we will be able to compute distributions like $p(s_i|c_1, c_2, c_3)$ and find out the hidden labels from the data.

2.2 Graphical models

Graphical models (Jordan, 1998; Lauritzen, 1996; Pearl, 1988) offer a flexible language for studying large-scale probabilistic models. As explained below, the graphical framework relies on factorial models, that is, models where the joint probability distribution of a large number of variables is as a product of, in general, simple factors each defined on a small subset of variables. In order to achieve factorial representation of a model the graphical formalism exploits various assumptions and simplifications. As a result, graphical models facilitate efficient, albeit in some cases approximate, approach to complicated multi-variate probabilistic problems.

Representation complexity A fundamental problem with models that include a large number of variables is that their joint distribution is difficult to represent. For example, in a model with N multinomial random variables, each defined on K states, there are K^N joint state configurations. Therefore an "exhaustive" or "naive" definition of the joint distribution requires $K^N - 1$ parameters that indicate probability mass assigned to every configuration. In the case of continuous or hybrid models typically the problem does not become simpler. A model with N, m-dimensional real-valued variables requires defining a probability density function (pdf): $\mathbb{R}^{Nm} \rightarrow (0, +\infty)$ on a Nm-dimensional configuration space. Analogously, a hybrid model consists of a joint distribution for the discrete variables and a separate pdf on the continuous variables for each configuration of the discrete states.

Conditional independence Theoretically, an exhaustive definition of model would be necessary if all variables were directly dependent on each other. However, in practice, one can assume that certain variables, which we denote collectively as x_A, are *conditionally independent* of variables x_B given variables x_C. We express this fact as

$$p(x_A|x_B, x_C) = p(x_A|x_C) \propto f(x_A, x_C), \tag{2.1}$$

where the key point is that probability of x_A is not a function of x_B as long as x_C are known.

Conditional independence assumptions (Lauritzen, 1996) allow expressing the joint distribution as a product of terms, which are defined on subsets of variables, thereby simplifying the representation of the model. To see this property we rewrite the joint using the definition of a conditional probability

$$p(x_A, x_B, x_C) = p(x_A|x_B, x_C)p(x_B, x_C)$$

and simplify $p(x_A|x_B, x_C)$ using the conditional independence assumption (2.1)

$$p(x_A, x_B, x_C) = p(x_A|x_C)p(x_B, x_C) \propto f(x_A, x_C)f(x_B, x_C). \tag{2.2}$$

Suppose that x_A and x_B represent a single variable each, and x_C the remaining $N - 2$ variables. The joint distribution expressed as a product requires only $\mathcal{O}(K^{(N-1)})$ parameters. Naturally, one may find or assume further conditional independencies within the factors. Sometimes exploiting such independence properties may reduce the complexity of representation from exponential to nearly linear in the number of variables.

Finding appropriate conditional independencies in a given domain resembles knowledge engineering tasks. A knowledge engineer has to have enough understanding about the domain to describe it with relevant variables and deduct conditional independencies among the variables. Alternatively, for some problems, one may use "training" data that provide observed values of (some) variables in the model, and attempt to automatically estimate conditional indecencies from these data. Such an approach is known as structure learning or model selection (see Section2.4). However, the prevailing approach is to experimentally evaluate a few intuitively designed models based on different independence assumptions and select the optimal one.

Graphical representation Graphical models represent a factorial distribution with a graph, where random variables are nodes and edges (or their lack) indicate the factorization structure. At the first sight, graphical models just effectively visualize a joint probability distribution. However, the formalism proves useful in development of efficient inference algorithms, which can be expressed as recursions operating on a graph. Moreover, the underlying graph brings forward various properties of an inference algorithm. For example, the popular *belief propagation* algorithm is exact only for models that can be represented as tree-structured graphs. The algorithm can be also executed on graphs with cycles, but in this case it is only an approximation. Further, graphical models constitute a unified implementation environment for seemingly different inference algorithms.

In the rest of this section we review two most popular types of graphical models – with directed and undirected arcs. (There is also a class of models with directed and undirected arcs, known as chain graphs.) Finally we present factor graphs, which allow unified treatment of all types of graphical models.

2.2.1 Undirected graphical models

Intuitively, undirected models depict a factorial expression (that is a product of functions or factors) as an undirected graph. In the graph, variables are nodes and all arguments to a single function make a fully-connected subset of nodes, i.e., a clique. For example, the model (2.2) corresponds to a graph with N vertices and two cliques.

Formally, undirected graphical models are undirected graphs, where every variable $v \in V$ is associated with an unique vertex, and the probability distribution $p(x_V)$ is a product of functions defined on maximal cliques. (A maximal clique is a clique that is not a subset of another clique.) With every clique C there is an associated *compatibility function* $\psi_C(x_C)$ that takes as arguments the vertices from C. The product over all cliques in the graph yields the joint distribution

$$p(x_V) = \frac{1}{Z} \prod_C \psi_C(x_C), \tag{2.3}$$

where Z is a constant ensuring normalization of $p(x_V)$. Note, that the compatibility functions $\psi_C(x_C)$ are *not required* to be proper probability distributions. As long as the product (2.3) is normalizable, the terms $\psi_C(x_C)$ can be non-negative, but otherwise arbitrary, real-valued functions.

The undirected graphical models are also known as Markov networks or Markov random fields (MRFs). These models find applications mainly in statistical physics (e.g. the Ising model, the Boltzmann machine (Chandler, 1987; Hinton and Sejnowski, 1986)) and computer vision (e.g. (Freeman et al., 2000)) communities. The latter application is illustrated below with a typical model.

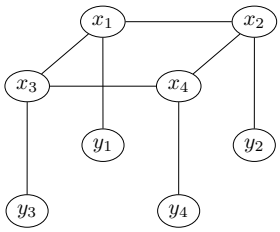


Figure 2.1: An MRF model characteristic for low-level vision problems. The hidden nodes x_i (each associated with the ith pixel) form a 2D lattice. The figure shows a graph that corresponds to four pixels, and includes eight cliques: four of type $\{x_i, x_k\}$ and four of type $\{x_i, y_i\}$.

Low-level vision Figure 2.1 shows an MRF model typical for low-level vision problems (Freeman et al., 2000), where with every pixel i there is an associated hidden variable x_i. For example, in image segmentation problems x_i is a discrete variable that indicates the segment associated with a pixel. Every hidden variable x_i has its observed counterpart y_i, which usually denotes the color of the pixel. In the figure one can identify eight cliques (each consisting of a pair of adjacent nodes). The corresponding joint distribution is

$$p(x_{1:4}, y_{1:4}) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_4)\psi(x_2, x_3)\psi(x_3, x_4)\prod_{i=1}^{4}\psi(x_i, y_i).$$

Typically the functions $\psi(x_i, x_j)$ are chosen to enforce some local consistency constraints (e.g. smooth segment edges), and $\psi(x_i, y_i)$ couple the observed and the hidden variables (e.g. we can interpret $\psi(x_i, y_i)$ as a conditional distribution $p(y_i|x_i)$, and chose it as a Gaussian $\mathcal{N}(\mu_k, V_k)$, where μ_k, V_k are parameters of the kth segment, $k = x_i$). In Chapter 5 we analyze in more detail MRFs and alternative models applied to a binary segmentation problem, where a pixel has to be assigned either to a *background* segment (representing a static scene) or a *foreground* segment (representing an object in motion).

2.2.2 Directed graphical models

In the directed case, a model is represented by a directed acyclic graph with edges leading from a parent node to a child node. Every variable $v \in V$ corresponds to an unique vertex in the graph. Directed models express the joint distribution as

$$p(x_V) = \prod_{v \in V} p(x_v|x_{\pi(v)}), \tag{2.4}$$

where $\pi(v)$ denote the set of parents for vertex v. For a vertex v with a non-empty parent set, $\pi(v) \neq \emptyset$, the factor $p(x_v|x_{\pi(v)})$ is a conditional probability distribution. For a vertex v where $\pi(v) = \emptyset$, the factor $p(x_v)$ is a prior probability distribution.

Directed models are also known under other names, like Bayesian networks (BNs), generative models, causal models or belief networks (Pearl, 1988). This type of models are mainly used in artificial intelligence or machine learning communities, and are especially convenient for problems involving time-series data.

Generative interpretation Characteristically, one may interpret edges in a directed model as "generative" or "causal" relations, and assume that parent nodes generate or cause the child node. In this thesis we adopt the "generative" semantics since causality is a deeper concept (Pearl, 1988), which may not always provide a suitable interpretation for problems with rather abstract variables as discussed in the following chapters. Accordingly, we will assume that parent nodes generate a child node and refer to a directed graph as a *generative* model.

The semantics of generative relations provide basic intuition for finding suitable conditional independencies relevant to a given problem. Given the expertise about the domain, a knowledge engineer can find out or at least assume which variables generate or cause other variables. We can then *assume* that a variable x_v given its direct causes is conditionally independent of its indirect causes, i.e.,

$$p(x_v|\text{direct-causes}(v), \text{indirect-causes}(v)) = p(x_v|\text{direct-causes}(v)).$$

Therefore by analyzing generative relations for a given domain, we can identify conditional independence relations, which greatly simplify the representation of a model. Additionally, conditional distributions $p(x_v|\text{direct-causes}(v))$ can often be easily quantified by studying to what extent various causes influence the child variable.

Example To illustrate the construction of a generative model we return to the picture-recognition example from Section 2.1.2. We denote the target model as $p(c_{1:3}, s_{1:3})$, where $x_{1:i}$ is a shortcut for $\{x_1, \ldots, x_i\}$. The definition begins with a general factorization

$$p(s_{1:3}, c_{1:3}) = p(c_{1:3}|s_{1:3})p(s_{1:3}), \qquad \text{where} \qquad p(s_{1:3}) = \prod_{i=1}^{3} p(s_i).$$

We refer to $p(s_{1:3})$ as a prior (since it involves only hidden variables). The prior is expressed as a product based on an assumption that the *before receiving any data* the identities of the persons are independent. Next, we attempt to define the conditional model $p(c_{1:3}|s_{1:3})$ where conditioning on $s_{1:3}$ means that we known which person is depicted in every picture. For example assigning $s_{1:3} = \{a, b, a\}$ indicates that c_1 and c_3 represent one person, and c_2 the other. In most cases one can assume a picture of one person does not include any information about the other person. Therefore observations of different people are independent and the conditional distribution $p(c_{1:3}|s_{1:3} = \{a, b, a\}) = p(c_1, c_3|s_{1:3} = \{a, b, a\})p(c_2|s_{1:3} = \{a, b, a\})$ will be factorial. Importantly, when c_1 and

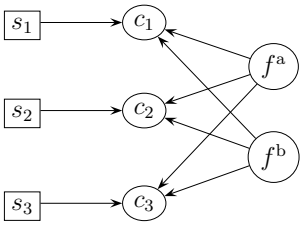


Figure 2.2: A directed graphical model representing factorization of the model in the picture-recognition problem. Rectangular nodes represent discrete variables, ovals — continuous.

c_3 describe the same person, these variables are dependent, and we cannot factorize $p(c_1, c_3|s_{1:3} = \{a, b, a\})$ further.

The problem with the above approach is that we need several types of pdfs for various configurations of $s_{1:3}$. Recall, that $c_i \in \mathbb{R}^3$. If the labels indicate that all pictures show the same person, we need a pdf $p(c_1, c_2, c_3)$ which is a function $\mathbb{R}^9 \to (0, +\infty)$. In general we need pdfs of the form $\mathbb{R}^{3n} \to (0, +\infty)$ where n is the number of pictures hypothetically showing the same person. Therefore generalizations to problems with more pictures are not straightforward. A better approach is to introduce hidden variables f^{a}, f^{b} that represent the actual (but unknown) color of each person. We refer to each f^s as a state of person s. We can assume that the states are independent from each other and the other variables in the model. Next, it is quite reasonable to assume that the observed color c is generated by the actual color of a person, i.e., given the state, the color c does not depend on other observations of the same person. The joint model is

$$p(f^{\mathrm{a}}, f^{\mathrm{b}}, c_{1:3}, s_{1:3}) = p(f^{\mathrm{a}})p(f^{\mathrm{b}}) \prod_{i=1}^{3} p(s_i)p(c_i|s_i, f^{\mathrm{a}}, f^{\mathrm{b}}). \tag{2.5}$$

Now, we define a generic pdf $p(c_i|f^s)$,

$$p(c_i|s_i, f^{\mathrm{a}}, f^{\mathrm{b}}) = \begin{cases} p(c_i|f^{\mathrm{a}}, s_i = \mathsf{a}) \\ p(c_i|f^{\mathrm{b}}, s_i = \mathsf{b}) \end{cases},$$

where we substitute the state of a person indicated by label s_i. This approach relies on a single pdf $p(c_i|f^s)$ and can be easily generalized to problems with more persons or pictures. Additionally, we separated the states from the measured features, therefore a state f^s can be chosen as a low-dimensional, compact representation of the person's appearance. Figure 2.2 illustrates the assumed factorization of the complete model.

The example demonstrates that introducing additional variables to the model does not necessarily make the model more complicated if appropriate conditional independence assumptions can be identified. Furthermore, the additional variables often ease specification of conditional probabilities related to causal/generative relations. Consequently, in problems presented in the rest of the thesis we will often introduce "imaginary" hidden variables in order to elicit suitable generative relations.

Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) are a class of directed models that represent probability distributions for time-sequences of random variables (Dean and Kanazawa, 1989; Ghahramani, 2001; Murphy, 2002). In such time-series graphs, the causal links point forward in time reflecting a natural assumption that variables further in time are caused by (a subset of) past variables.

We denote the discrete time index as $t = 1, 2, \ldots$, and a set of variables associated with index t as x_t. A sequence of variables $(x_1, \ldots, x_T)$ will be denoted as $x_{1:T}$. Formally, a DBN is defined on a semi-infinite set of variables $(x_1, x_2, \ldots)$. The definition includes two models: a prior model $p(x_1)$ for the variables at the initial time index, and a conditional probability $p(x_t|x_{t-1})$. Given a fixed time horizon T, the definition implies a joint distribution for a sequence $x_{1:T}$

$$p(x_{1:T}) = p(x_1) \prod_{t=1}^{T} p(x_t|x_{t-1}). \tag{2.6}$$

A DBN can also be viewed as an algorithm that constructs a regular directed graph for $p(x_{1:T})$ by first drawing a graph for $p(x_1)$ and extending it with a sequence of concatenated graphs corresponding to $p(x_t|x_{t-1})$.

DBNs offer a convenient framework for reasoning about latent time-processes that, at every time instance, produce an observable output (Kröse et al., 2004; Zajdel et al., 2005a). Here, the set of variables x_t at a single time step includes two terms: a hidden variable z_t that represents the state of the process in question, and an observed variable y_t that represents the output.

In particular, DBNs apply very naturally to multi-object tracking problems, where we can consider every object as a latent process. Depending on a modeling assumptions, the hidden state of the process might include object's identity or various other characteristics. The measurements of object's characteristics (e.g. appearance features) obtained from a camera correspond to the observations of the latent process.

First-order systems Commonly there are two assumptions about the latent process. First, the hidden state variable evolves with first order Markovian dynamics, that is, the state z_t is caused exclusively by z_{t-1}. Second, the observed variable y_t is caused exclusively by the corresponding state z_t. Therefore a model for such a process includes: a state *transition* model $p(z_t|z_{t-1})$, an *observation*, also known as a *sensor* or a *measurement*, model $p(y_t|z_t)$, and a prior model $p(z_1)$ for the initial state. These models yield a DBN, where $x_t = \{z_t, y_t\}$ and

$$\begin{aligned} p(x_1) &= p(z_1, y_1) = p(z_1)p(y_1|z_1) \\ p(x_t|x_{t-1}) &= p(z_t, y_t|z_{t-1}, y_{t-1}) = p(z_t|z_{t-1})p(y_t|z_t). \end{aligned} \tag{2.7}$$

The simplification in (2.7) follows from the assumptions about the transition and observation models in first-order Markov systems.

The most important instances of models based on the above assumptions are hidden Markov models (HMMs) and linear dynamical systems (LDSs). In the case of HMM models (Ghahramani, 2001), terms z_t and y_t are multinomial (i.e. discrete) variables. The prior, transition and observation models take the form of probability tables. In the case of LDS models (Roweis and Ghahramani, 1999), also known as Kalman filter models (KFMs), the terms $z_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^n$ are continuous vectors. The transition and observation models assume linear dependencies and additive noise;

$$z_t = Az_{t-1} + w_t \qquad\qquad y_t = Cz_t + v_t,$$

where $w_t \sim \mathcal{N}(0, Q)$, $v_t \sim \mathcal{N}(0, R)$, i.e., the noise is Gaussian distributed. For computational simplicity, the prior model is chosen Gaussian. Therefore

$$p(z_1) = \mathcal{N}(z_1|m_0, V_0) \qquad p(z_t|z_{t-1}) = \mathcal{N}(z_t|Az_{t-1}, Q) \qquad p(y_t|z_t) = \mathcal{N}(y_1|Cz_t, R),$$

where m_0, V_0 are the prior mean and covariance, A, C are, respectively, transition and observation matrices, and Q, R are covariance matrices describing the noise. The covariances R and Q can be assumed diagonal without loss of generality (Roweis and Ghahramani, 1999).

Higher-order systems One may be tempted to consider DBNs as purely first-order Markovian systems, due to the transition model formulated as $p(x_t|x_{t-1})$. However, by augmenting the set x_t with suitable variables we can express arbitrary fixed-order Markov processes. For example, a second-order model emerges when we define $x_t = \{y_t, z_t, z_{t-1}\}$ and let

$$p(x_t|x_{t-1}) = p(y_t, z_t, z_{t-1}|y_{t-1}, z'_{t-1}, z_{t-2}) = p(y_t|z_t)\delta(z_{t-1} - z'_{t-1})p(z_t|z_{t-1}, z_{t-2}),$$

where $\delta(a)$ is a Dirac-delta function, defined as $\delta(a) = 1$ (or 0) iff $a = 0$ (otherwise). In a similar way we obtain models where the current observation y_t depends on a sequence of past data $y_{t-h:t-1}$, where $h > 0$ is a fixed range. An example of such a system is a mixed-memory Markov model (Saul and Jordan, 1999).

2.2.3 Factor Graphs

We have shown the undirected and directed graphical models as two languages to represent factorial multi-variate probability distributions. These languages usually facilitate finding out factorization of the joint distribution in an intuitive way. Below, we present another graphical formalism, so called *factor graphs*, which provides a convenient language for studying factorial expressions in inference problems. In fact, factor

graphs make a common backbone for unification of various inference algorithms that have been independently developed for the directed and undirected models.

A factor graph (Kschischang et al., 2001) represents a factorial expression, that is, a product of several functions (factors), as an undirected graph with two types of nodes: *variable nodes* that correspond to random variables, and *factor nodes* that correspond to individual functions. The edges in the graph connect functions with their arguments (variables). Therefore a factor graph is a *bipartite* graph. Figure 2.3 shows a factor graph that encodes the factorial model (2.5) of the picture-recognition problem.

We will denote the set of factor nodes as F, and variable nodes as V. For every factor node $f \in F$ we let ψ_f denote the associated function, and $n(f) \subset V$ denote the adjacent variable nodes. In this notation $x_{n(f)}$ gives a set of variables that are arguments to the function ψ_f. Therefore, the associated factorial distribution is encoded as

$$p(x_V) = \frac{1}{Z} \prod_{f \in F} \psi_f(x_{n(f)}), \tag{2.8}$$

with Z denoting a normalization term. With these notation it is quite straightforward to translate factorial expressions associated with directed or undirected models to a factor graph. In the directed case (see equation (2.4)): for every variable x_v we introduce one factor f and we define the factor function $\psi_f(x_v, x_{\pi(v)}) = p(x_v|x_{\pi(v)})$, where $\pi(v) \subset V$ is a (possibly empty) set of parents for variable x_v. In the undirected case (see equation (2.3)): we associate a single factor f with every clique C, and define the factor function $\psi_f(x_C) = \psi_C(x_C)$, where x_C is the set of variables included in the clique $C \subset V$.

As the next section will show, various standard inference algorithms can be viewed as instances of a generic "message-passing" scheme that exploits factorization encoded by a factor graph. From inference point of view, it is important to distinguish between cyclic and cycle-free (also known as singly-connected or tree-structured) factor graphs. (A graph is cycle-free iff it is a tree, i.e., there is only a single path connecting any pair of nodes. Otherwise the graph has a cycle.) The distinction is important because cycles in a graph make inference significantly more difficult.

2.3 Inference methods

2.3.1 Overview

Probabilistic inference is a general term that refers to a class of computational problems related to the probabilistic framework. Given a model $p(x_H, x_E)$ and the observed data x_E we want to solve one of the following tasks.

(a) Computing a marginal posterior distribution $p(x_v|x_E)$, where $v \in H$. This is the basic distribution that conveys information about a hidden variable inferred from

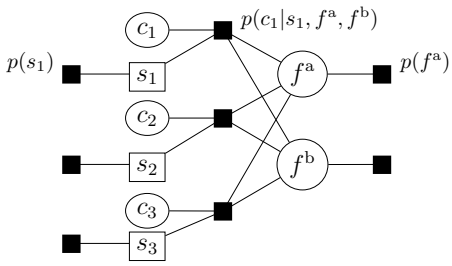


Figure 2.3: A factor graph encoding the factorial model (2.5) defined for the example picture-recognition problem. Transparent nodes depict random variables; filled nodes depict individual factors in the joint probability distribution.

the evidence. In the context of graphical models, we will refer to the distribution $p(x_v|x_E)$ as a *belief* at node x_v, denoted as $q(x_v)$.

(b) Computing probability of the observed data $p(x_E)$.(This quantity is also known as the data likelihood or the evidence.) In case when we have several models competing to explain the data, the data likelihood is the primary criterion to select the best model (model selection). In case when we have several data sets and a single model, the data likelihood allows to find the data set that best matches the model (classification).

(c) Computing various expectations $\langle f(x_v)\rangle = \int f(x_v)p(x_v|x_E)\,\mathrm{d}x_v$, where f is a given function. The expectations are typically required for learning algorithms, but might be of interest itself depending on an application. This task becomes relatively simple after computing $p(x_v|x_E)$. However, frequently exact computation of $p(x_v|x_E)$ is intractable, therefore one might directly approximate expectations.

(d) Computing the MAP (maximum a-posteriori) value $\arg\max_{\mathcal{X}_A} p(x_A|x_E)$, where $A \subset H$ denotes a collection of hidden variables.

Computing a marginal density and computing the data likelihood (tasks a and b) are two instances of a common problem, which boils down to integrating out a subset of variables from the joint $p(x_H, x_E)$. Consider first computing the data likelihood

$$p(x_E) = \sum_{x_H \in \mathcal{X}_H} p(x_H, x_E),$$

where we sum over the space $\mathcal{X}_H$ of all configurations of hidden variables. Next, we express the marginal posterior density using the Bayes' rule:

$$p(x_v|x_E) = \sum_{\sim\{x_v\}} p(x_H|x_E) = \sum_{\sim\{x_v\}} \frac{p(x_H, x_E)}{p(x_E)} = \frac{1}{p(x_E)} \sum_{\sim\{x_v\}} p(x_H, x_E),$$

where "$\sim \{x_v\}$" denotes all configurations of the form $\{x'_H \in \mathcal{X}_H | x'_v = x_v\}$. Therefore, in both cases the key task is to integrate function $p(x_H, x_E)$ over a subset of variables.

In practice, the key challenge of probabilistic inference is computational complexity of integration. For example, suppose we have a model with N hidden multinomial variables $x_{1:N}$, each $x_i \in \{1, \ldots, K\}$. A "naive" way to compute a marginal $p(x_i|x_E)$ requires summing over K^{N-1} configurations of the remaining hidden variables. In case of continuous variables summation is replaced with integration and the problem (in general) becomes even harder. For the same reason, computing the MAP state for a subset of hidden variables (see task d) is generally intractable since it entails maximization over exponentially many configurations $x_A \in \mathcal{X}_A$.

Many, either exact or approximate, algorithms have been developed to overcome the intractability of probabilistic inference. These algorithms fall roughly into two classes: stochastic methods and variational methods. Stochastic methods replace analytical integration with Monte-Carlo approximations (Andrieu et al., 2003; Gilks et al., 1996). Variational methods estimate the target posterior distribution by minimizing a related quantity, known as free energy. By relaxing the minimization problem in various ways (e.g. dropping certain constraints), one can construct various approximate inference algorithms. A general overview of inference methods can be found in (Jordan, 1998; Murphy, 2002).

In the next section we focus on a class of variational methods commonly summarized as "message-passing" or "belief propagation" algorithms. As we show, all inference techniques applied in the thesis are instances of this class of methods. (Another major classes of variational methods are: *cluster* variational methods (Yedidia et al., 2005) and so called "mean-field" approximations (Jaakkola, 2001; Jordan et al., 1998) — although these are sometimes also presented as message-passing algorithms (Winn, 2003).)

2.3.2 Message-passing algorithms

Alternatively to the variational derivation, the message-passing algorithms can be obtained by applying the distributive law for efficient integration of factorial expressions. We begin with such a derivation as it proves much more intuitive. Since various forms of factorized models are conveniently encoded by a factor graph, we present these algorithms as message-passing operations in a factor graph.

We consider the following algorithms. First, we present the sum-product algorithm as a general method for fast and exact integration in cycle-free graphs with discrete or Gaussian variables. Second, we consider graphs with cycles and describe the approximate loopy-belief propagation algorithm. Third, we move on to graphs that include continuous variables and discuss the expectation-propagation algorithm. Finally, we present variational framework as a theoretical foundation for the presented methods.

Sum-product algorithm

Message-passing algorithms can be viewed as instances of a general computational rule known as the sum-product algorithm. The sum-product algorithm integrates factorized functions of multiple variables by exploiting the "distributive law" to limit the number of computations (Aji and McEliece, 2000). In the simplest form the law says that $ac+ab = a(c+b)$, thus it reduces three operations on the left side to only two operations on the right side. Consider now integrating variables x_1, x_2, x_4, x_5 from the following product

$$q(x_3) = \frac{1}{Z} \sum_{x_1,x_2,x_3,x_4} \psi_{\mathrm{b}}(x_2,x_3)\psi_{\mathrm{a}}(x_1,x_2)\psi_{\mathrm{d}}(x_5,x_2)\psi_{\mathrm{c}}(x_3,x_4). \tag{2.9}$$

where every variable is a multinomial with K states. Distributive law allows us to "push" sums into the product, therefore splits the summation into several intermediate results

$$q(x_3) = \frac{1}{Z} \sum_{x_2} \psi_{\mathrm{b}}(x_2,x_3) \underbrace{\underbrace{\sum_{x_1} \psi_{\mathrm{a}}(x_1,x_2)}_{s_{\mathrm{a}}(x_2)} \underbrace{\sum_{x_5} \psi_{\mathrm{d}}(x_5,x_2)}_{s_{\mathrm{d}}(x_2)}}_{\pi_{\mathrm{b}}(x_2)} \underbrace{\sum_{x_4} \psi_{\mathrm{c}}(x_3,x_4)}_{s_{\mathrm{c}}(x_3)}}_{s_{\mathrm{b}}(x_3)}, \tag{2.10}$$

Despite that there are K^4 configurations of the integrated variables, we now have to add only $4K$ terms. There intermediate results can be classified as (i) partial sums indicated as $s_{\mathrm{a}}(x_2)$, $s_{\mathrm{d}}(x_2)$, $s_{\mathrm{c}}(x_3)$, $s_{\mathrm{b}}(x_3)$, and (ii) partial products, indicated as $\pi_{\mathrm{b}}(x_2)$.

Interestingly, every intermediate result can be associated with an edge in the factor graph that represents the integrated expression. Consider Fig. 2.4, which presents factor graph for the discussed example. We use $f \in \{\mathrm{a},\ldots,\mathrm{d}\}$ to index factors and $v \in \{1,\ldots,5\}$ to index variables. We observe in (2.10) that every partial sum $s_f(x_v)$ is a function of a single variable x_v and involves a single ψ_f. Therefore one can associate this sum with an edge (f,v) in the factor graph. It is convenient to denote the partial sum as $s_f(x_v) = m_{f\to v}(x_v)$ and view it as a message from the factor node f to the adjacent variable node v. Analogously, each partial product $\pi_f(x_v)$ is a function of a single variable x_v, and includes partials sums $s_g(x_v)$ from all but one factor, denoted as f. Therefore we associate this product with an edge (f,v) and view it as a message $\pi_f(x_v) = m_{v\to f}(x_v)$ form the variable node v to the factor f.

Given the graphical interpretation the sum-product algorithm can be viewed as a bookkeeping scheme that efficiently computes all messages in a factor graph. The messages are initialized to unit functions, and updated ("passed" or "sent") as

$$m_{v\to f}(x_v) = \prod_{f'\neq f} m_{f'\to v}(x_v) \tag{2.11}$$

$$m_{f\to v}(x_v) = \sum_{\sim\{x_v\}} \psi_f(x_{n(f)}) \prod_{v'\neq v} m_{v'\to f}(x_{v'}), \tag{2.12}$$

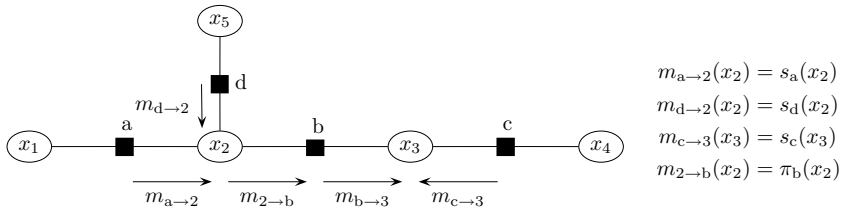


Figure 2.4: (Right) A factor graph corresponding to factorial expression (2.9). The graph includes four factor nodes $f \in F = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}\}$ indicating the functions $\psi_{\mathrm{a}}, \psi_{\mathrm{b}}, \psi_{\mathrm{c}}, \psi_{\mathrm{d}}$, and five variable nodes x_v, $v \in V = \{1, \ldots, 5\}$. (Left) The correspondence between message passed along edges in the graph and intermediate results in the equation (2.10).

for all edges (v, f), where $n(f) \subset V$ indicates variable nodes adjacent to factor node f. These update rules generalize message definitions observed in the example (2.10). To understand the role of messages suppose we remove an edge (f, v) from the factor graph. We obtain two disconnected graphs: $\mathcal{G}_1$ that includes node f and $\mathcal{G}_2$ that includes node v. A message $m_{f\to v}(x_v)$ represents all information about x_v that can be inferred from graph $\mathcal{G}_1$. A message $m_{v\to f}(x_v)$ represents all information about x_v that can be inferred from graph $\mathcal{G}_2$. When computing $m_{v\to f}(x_v)$ in (2.11) we collect information from all adjacent factors f' different that f. When computing $m_{f\to v}(x_v)$ in (2.12) we collect information about other variables v', combine it with the local factor function ψ_f and integrate the other variables to obtain information about x_v. Given the messages, we compute the belief as

$$q(x_v) = \frac{1}{Z} m_{f\to v}(x_v) m_{v\to f}(x_v), \tag{2.13}$$

where and f is any factor adjacent to the node v.

The above procedure provides exact inference for cycle-free factor graphs, where we can choose any order for passing messages. A particularly efficient is the following one. Any vertex, say u, waits until it receives messages from all but one neighbor, denoted as w. Next, the vertex u sends a message to w and waits for a return message from w. When the message arrives, u sends messages to all remaining neighbors. The procedure is started at leaves (vertices that have only a single neighbor) and stopped when a message passes once along every edge in each direction.

The sum-product algorithm is mainly applied to models with exclusively discrete variables and generalizes many well-known algorithms developed for models that can be presented as instances of such graphs (Kschischang et al., 2001; Yedidia et al., 2005). Examples are: the *belief propagation* algorithm (for general Bayes networks), the *forward-backward* procedure (for HMMs). The sum-product algorithm can also be executed in graphs with continuous variables, as long as the integrals in (2.12) have analytical solution. An important instance of such an algorithm is the *Kalman filtering-smoothing* procedure for LDS models (here messages, beliefs and factor functions are always Gaussian functions). We use Kalman-filter equations in Chapter 3.

Loopy-belief propagation

Unfortunately, in general the sum-product algorithm cannot compute exact marginals for models, where the underlying factor graph contains cycles (Kschischang et al., 2001). There are two approaches for inference in such models. One possibility, which we discuss below, is to run the sum-product algorithm despite the cycles and compute approximate marginals. Another possibility entails converting the graph with cycles to a corresponding cycle-free graph. (This is what effectively the *junction-tree* algorithm (Pearl, 1988) does. The algorithm is exact, however, it may have running time exponential in the number of nodes.)

Loopy-belief propagation (LBP) is an algorithm obtained by iterative executing the sum-product rule on a cyclic factor graph (Murphy et al., 1999). Note, that each "message-passing" operation (either (2.11) or (2.12)) is *local* in the sense that it involves only quantities related to the sending and receiving nodes. Therefore regardless of the global cycles one may pass messages without changing the update equations. However, modifications are required for the termination condition. LBP initializes all messages to uniform functions (also known as uniform potentials), and considers (2.11)–(2.12) as update equations that in each iteration re-estimate the messages. The algorithm terminates either when the messages do not improve beyond a certain threshold or simply after executing a fixed number of updates.

The LPB algorithm often yields surprisingly accurate results (Murphy et al., 1999), and has become a standard computationally tractable technique for computer vision problems (Freeman et al., 2000). However, in some cases the sum-product iterations fail to converge, and the precise convergence conditions are in general not known (Heskes, 2003; Welling and Teh, 2001). In this thesis we apply the LBP algorithm for inference in a variety of MRF models presented in Chapter 5.

Expectation Propagation

The expectation-propagation (EP) algorithm (Minka, 2001b) can be viewed as another refinement of the sum-product rule that facilities approximate inference in hybrid models (with both continuous and discrete variables) or purely continuous-variable models. Inference in such models requires solving integrals that often become either very complicated or non-analytical. Intuitively, EP assumes a simple parametric family of functions that represent the messages. When computing a message the algorithm approximates a complicated integral with the "closest" function from the assumed family.

To derive EP we modify the sum-product procedure in the following way. Based on the relation (2.13) we eliminate the variable-to-factor messages $m_{v \to f}$ in favor of beliefs. Next, we reformulate the update equations in such a way that the belief is expressed directly as a solution to the integral. Consequently, we can directly approximate beliefs rather than messages. These modifications lead to the EP algorithm, which iterates the

following recursion

$$m^{\text{new}}_{f\to v}(x_v) = \frac{Zq^{\text{new}}(x_v)}{q(x_v)} m_{f\to v}(x_v) \tag{2.14}$$

$$q^{\text{new}}(x_v) = \arg\min_q \mathrm{KL}(\tilde{q}||q) \tag{2.15}$$

$$\tilde{q}(x_v) = \frac{1}{Z} \sum_{\sim\{x_v\}} \psi_f(x_{n(f)}) \prod_{v'\in n(f)} \frac{q(x_{v'})}{m_{f\to v}(x_v)}, \tag{2.16}$$

where $\mathrm{KL}(\tilde{q}||q)$ denotes the Kullback-Leibler (KL) divergence[2] between the "true" belief $\tilde{q}$ and the approximating belief q. Minimization in (2.15) essentially means that the new belief q^{new} preserves the moments of the actual belief $\tilde{q}$. It is quite easy to show that for a discrete variable x_v, the EP update rule is identical with the sum-product or LBP algorithms, because for multinomial distributions minimization of KL-divergence yields $q^{\text{new}}(x_v) = \tilde{q}(x_v)$.

The key feature of the EP algorithm is that beliefs and messages are represented with a suitable approximating family. Typically, this family is chosen such that the integral (2.16) has an analytical solution, denoted as a temporary belief $\tilde{q}$. However, in most cases the belief $\tilde{q}$ cannot be represented in the assumed form. In (2.15) we replace it with the closest approximating belief. We note further that approximating beliefs rather than messages (i.e., delaying approximation) often yields better results since a belief includes total information available at a given node, while a message only partial (Heskes and Zoeter, 2002; Minka, 2001b). Moreover, in contrast to a message, a belief is *always* a probability distribution, and we can use an established criterion — KL divergence — to find the best approximation.

Exponential family Exponential family (Brown, 1986; Wainwright and Jordan, 2003) encapsulates a class of probability distributions that are particularly suitable for implementing the EP algorithm. The general form of the family is

$$q(x|\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^\top \mathbf{g}(x) - A(\boldsymbol{\theta})),$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^\top$ is a vector of *canonical* parameters, $\mathbf{g}(x) = (g_1(x), \ldots, g_n(x))^\top$ are functions, so called, *sufficient statistics* of x, and $A(\boldsymbol{\theta}) < \infty$ is a normalization term. Distributions from this family are sometimes represented using an alternative formulation, based on *mean* parameters $\boldsymbol{\mu}(\boldsymbol{\theta}) = (\mu_1(\boldsymbol{\theta}), \ldots, \mu_n(\boldsymbol{\theta}))^\top$. The mean parameters are defined as

$$\mu_i(\boldsymbol{\theta}) = \int_x g_i(x) p(x|\boldsymbol{\theta})\, \mathrm{d}x = \langle g_i(x)\rangle_{\boldsymbol{\theta}}\,.$$

[2]The KL divergence is defined as $\mathrm{KL}(p||r) = \int_x p(x)\log(p(x)/r(x))$ and is non-symmetric. When p denotes the "true" distribution, and r an approximating distribution, then $\mathrm{KL}(p||r)$ is sometimes known as "the other KL-divergence".

Importantly, converting between the canonical and the mean parameters is relatively simple (for most of the standard distributions from the family).

Many popular probability density functions turn out to belong to the exponential family. Examples are: Gaussian (Normal), multinomial, Gamma, Bernoulli, Poisson distributions. To see the relationship between the usual mean parametrization and the canonical counterpart, consider the Gaussian density for a scalar variable x

$$\begin{aligned}\mathcal{N}(x|m,v) &= (2\pi v)^{-\frac{1}{2}} \exp(-\frac{1}{2v}(x-m)^2) \\ &= \exp(\theta_1 g_1(x) + \theta_2 g_2(x) - A(\theta_1,\theta_2)),\end{aligned}$$

where m is the expected value and v is the variance. In the canonical parametrization the sufficient statistics are $g_1(x) = x$ and $g_2(x) = -\frac{1}{2}x^2$, and the canonical parameters $\theta_1 = m$, $\theta_2 = 1/v$, $A(\theta_1,\theta_2) = -\frac{1}{2}(\theta_1^2\theta_2 + \log(2\pi/\theta_2))$. A similar relation can be established for a multivariate Gaussian density.

The suitability of exponential family for EP inference follows from the fact that the minimization of KL-divergence can be achieved by exploiting the following property

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathrm{KL}\left(\tilde{q}(x) \middle|\middle| \exp(\boldsymbol{\theta}^\top \mathbf{g}(x) - A(\boldsymbol{\theta}))\right) \qquad \text{iff} \qquad \boldsymbol{\mu}(\boldsymbol{\theta}^*) = \langle \mathbf{g}(x) \rangle_{\tilde{q}}\,.$$

Thus, given any belief $\tilde{q}$ we find the closest belief $q(x|\boldsymbol{\theta})$ from the assumed exponential family by first computing the expectations of the sufficient statistics $\langle \mathbf{g}(x) \rangle_{\tilde{q}}$. Next we solve a system of equations (sometimes called *link function*) $\boldsymbol{\mu}(\boldsymbol{\theta}) = \langle \mathbf{g}(x) \rangle_{\tilde{q}}$ to find parameters $\boldsymbol{\theta}$. For many densities in the exponential family this system of equation can be solved exactly (e.g., Normal). For some other densities (e.g., Wishart) we have to find an approximate solution. Interestingly, the minimization of KL-divergence ensures that the expectations of sufficient statistics are the same under beliefs $\tilde{q}$ and q. (Therefore one also refers to the above procedure as "moment matching".)

Another advantage of using exponential family to represent messages and beliefs is that multiplication (or division) of potentials within this family corresponds to summation (or subtraction) of their canonical parameters. Moreover, within the family one can easily define an uniform (and unnormalized) potential by setting $\theta_i = 0$ and assuming $A(\boldsymbol{\theta}) = 0$. Therefore, the exponential family facilitates convenient implementation of message-passing algorithms.

In this thesis we present two problems that have been solved by applying the EP algorithm together with approximating distributions from exponential family. In Chapter 4 we the use the assumed-filtering algorithm (ADF), which can be considered as a simple EP variant designed for online inference in time-series models. Chapter 5 presents an image segmentation algorithm obtained by executing EP inference in a suitable probabilistic model.

Variational formulation

We have presented the loopy-belief propagation and the expectation-propagation algorithms as a series of seemingly heuristic modifications to the exact sum-product algorithm. However, these algorithms can also be derived in a much more principled way, as instances of so called *Bethe variational problem*. Below we briefly summarize this formulation to show that the message-passing algorithms are a well-motivated class of approximate inference techniques.

Inference can be considered as a problem of approximating the joint posterior distribution $p(x_H|x_E)$ with a factorial distribution $q(x_H) = \prod_{v \in H} q(x_v)$. In the variational framework we find the distribution $q(x_H)$ by minimizing the *variational free energy* (also known as Gibbs free energy)

$$\mathcal{F}(q) = \sum_{x_H} q(x_H) \log \frac{q(x_H)}{\prod_{f \in F} \psi_f(x_{n(f)})},$$

where ψ_f are factor functions defined by the model (see (2.8)). One can show that $\mathcal{F}(q)$ is minimized when $q(x_v) = p(x_v|x_E)$ for all $v \in H$.

The message-passing algorithms relax the minimization problem in two ways. First, the minimized energy is approximated by a simpler quantity, so called *Bethe* free energy. The Bethe free energy is a function of single-node beliefs $q(x_v)$ and *region* beliefs $q(x_{n(f)})$, i.e., joint beliefs defined on variables adjacent to a factor f. Second, the minimization algorithm takes into account only two types of constraints: (i) beliefs $q(x_v)$ have to be normalized, and (ii) beliefs $q(x_v)$ have to be *locally* consistent with region beliefs, that is $q(x_v) = \sum_{\sim\{x_v\}} q(x_{n(f)})$ for any node v adjacent to a factor f. (Note that, unless a factor graph is cycle-free, local consistency does not guarantee *global* consistency, that is, the algorithm might arrive at a set of region beliefs that do not correspond to any valid joint distribution.)

Variational formulation provides a unifying theoretical foundation for various message-passing methods. The sum-product or the LBP algorithms turn out to be instances of the Lagrange-multiplier method applied for solving the above constrained minimization problem (Heskes, 2003; Yedidia et al., 2001). In a similar way one can derive the EP algorithm (Minka, 2001a). (Here, the local consistency constraints require that the moments of single-node beliefs are equal to the moments of the region beliefs.) Moreover based on the variational insight one can construct potentially improved methods. An example are the generalized belief propagation algorithms, based on so called *Kikuchi approximations* to the free energy (Yedidia et al., 2005), or more sophisticated algorithms with convergence guarantees (Heskes, 2003; Welling and Teh, 2001; Yuille, 2002).

2.4 Learning methods

2.4.1 Overview

Probabilistic learning refers to a class of methods that, to a certain extent, allow automating model design on the basis of "training" data. Within the graphical framework, a model consist of (i) an associated graph and (ii) factor functions defined on subsets of vertices. Therefore we distinguish between two problems: *structural* learning and *parameter* learning. In the former, the goal is to estimate the topology of the graph associated with a model. In the second problem we first define appropriate graph and specify a parametric family that will represent each factor function. Here the goal is to estimate parameters of these functions from the data.

Structural learning problems naturally arise in multi-object tracking applications. Recall, that in multi-object tracking the goal is to infer identities of objects given measurements from cameras. Typically the measurements corresponding to different objects (i.e. identities) are independent (or at least conditionally independent given some global environment properties). Therefore, multi-object tracking can be viewed as a search for a set of independence relations among measurements. In the graphical formalism, this task maps to the search for a set of disconnected graphs, each representing a model for measurements of a distinct person. Such an approach and the related structure learning algorithms are discussed in Chapter 3.

In this section we focus on methods for parameter learning, since it is a common task in many applications, including the tracking problems presented in the thesis. We begin with examples of parametric models, and subsequently discuss two major approaches for parameter learning: the *frequentist* and the *Bayesian* learning. Through the section we denote parametric models as $p(x_V|\theta)$, where θ is a collection of parameters for all factor functions. Through the section we let $y_{1:N}$ denote the set of N training data, where each y_i is an instance of the observed variables x_E. We assume that the training data does not include instances of the hidden variables.

Example A simple example of a parametric model is the multinomial distribution. Suppose that we define $p(x_a|x_b)$, where x_a, $x_b \in \{1, \ldots, K\}$ are multinomial variables. The distribution is parametrized as $p(x_a = i|x_b = j, \theta) = A(i, j)$, where the parameters $\theta = \{A_{i,j}|i, j = 1, \ldots, K\}$ satisfy $A_{i,j} > 0$ and $\sum_{i=1}^{K} A_{i,j} = 1$ for all j. Another example are LDS models, where the parameters are $\theta = \{\mu_0, V_0, A, C, Q, R\}$ (see Section 2.2.2.)

2.4.2 Frequentist learning

Frequentist learning methods estimate a single parameter configuration θ^* that optimally fits the training data. The key role in this approach plays a fitness function that is

used as a criterion to evaluate parameters. The maximum-likelihood and the maximum a-posteriori methods are two popular instances of this approach.

The maximum-likelihood (ML) methods find θ^* by maximizing the likelihood of the training data $p(y_{1:N}|\theta)$. Since maximization is often easier in the log-domain, the ML methods maximize log-likelihood, that is,

$$\theta^*_{\mathrm{ML}} = \arg\max_\theta \mathcal{L}(\theta) \qquad\qquad \mathcal{L}(\theta) = \log p(y_{1:N}|\theta).$$

The likelihood usually follows from an assumption that all data y_n in the training set are independent and identically distributed samples from the model. Therefore

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \log p(y_n|\theta) = \sum_{n=1}^{N} \log \sum_{x_H} p(x_H, x_E = y_n|\theta),$$

where we integrate all hidden random variables from the model.

The maximum-a-posteriori (MAP) methods find θ^* by maximizing the posterior probability distribution $p(\theta|y_{1:N})$. In this approach the parameters are considered as random variables with a prior distribution $p(\theta)$. Given the prior and the likelihood $p(y_{1:N}|\theta)$ we can express the posterior distribution as $p(\theta|y_{1:N}) \propto p(\theta)p(y_{1:N}|\theta)$ by using the Bayes' rule. Therefore the MAP parameters follow from

$$\theta^*_{\mathrm{MAP}} = \arg\max_\theta \log p(\theta|y_{1:N}) = \arg\max_\theta \big(\log p(\theta) + \mathcal{L}(\theta)\big),$$

where we see that the only difference between MAP and ML fitness functions is the additive prior $\log p(\theta)$. We can consider the prior as a "regularization" term that penalizes degenerate parameters (e.g. due to singularities in the likelihood function).

Generally, finding a closed-form solution to the above maximization problems proves difficult because the likelihood function is either complicated or non-analytical (due to the integration of hidden variables from the model). Therefore one usually has to settle with local maximization techniques, like gradient ascent methods. Another approach, which we discuss below, is iterative maximization of the lower-bound of the likelihood.

Expectation-Maximization

Expectation-maximization (EM) is a classical method to solve maximization problems arising in probabilistic parameter estimation problems (Bilmes, 1997; Dempster et al., 1977; Neal and Hinton, 1998). EM exploits Jensen's inequality to obtain a lower bound on the log-likelihood

$$\mathcal{L}(\theta) = \log \sum_{x_H} p(y_{1:N}, x_H|\theta) \geq \sum_{x_H} q(x_H) \log \frac{p(y_{1:N}, x_H|\theta)}{q(x_H)} = \mathcal{L}_{\mathrm{B}}(q, \theta)$$

where $q(x_H)$ is an arbitrary probability distribution. The key insight here is that the Jensen's inequality allows us to "push" logarithm into the integral. As a result, the bound $\mathcal{L}_{\mathrm{B}}(q, \theta)$ often becomes a relatively simple function that can be analytically maximized w.r.t parameters θ.

EM can be viewed (Neal and Hinton, 1998) as an iterative gradient ascent in the space of parameters θ and distributions q. The algorithm starts with some initial parameter estimates $\theta^{(0)}$ and in every iteration re-estimates the parameters in two steps:

$$\begin{aligned} q^{(i)}(x_H) &= \arg\max_q \mathcal{L}_{\mathrm{B}}(q, \theta^{(i)}) = p(x_H|\theta^{(i)}, y_{1:N}) \\ \theta^{(i+1)} &= \arg\max_\theta \mathcal{L}_{\mathrm{B}}(q^{(i)}, \theta) = \arg\max_\theta \sum_{x_H} q^{(i)}(x_H) \log p(y_{1:N}, x_H|\theta) \end{aligned}$$

The first step, so called E step, maximizes the lower bound w.r.t distribution q. The best, i.e. tight bound $\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{B}}(\theta, q)$, can be achieved when q equals the posterior of hidden variables given the data and current parameters estimates. The second step, so called M step, finds new estimates by maximizing the current bound w.r.t θ. The maximizing parameters $\theta^{(i+1)}$ will depend on various expectations of hidden variables computed from $q(x_H)$. Thus, in fact the E-step is a regular inference problem, where EM implementations compute only the necessary expectations of the hidden variables.

An important feature of the EM procedure is that the estimates $\theta^{(0)}, \theta^{(1)}, \ldots$, always converge to a maximum of the likelihood function. However, the estimated parameters might correspond to various local maxima, depending on the initial parameters $\theta^{(0)}$. Further, the algorithm itself does not rely on extra parameters which need fine-tuning (there is no "step-size" as in gradient methods that directly optimize likelihood). Finally, we note that the basic EM can be extended in various ways (Murphy, 2002; Neal and Hinton, 1998), e.g. by approximating the E step or the M step, deterministic annealing of the lower-bound, etc.

2.4.3 Bayesian learning

Bayesian learning offers an alternative approach for dealing with unknown model parameters. Intuitively, this approach does not find point-estimates of parameters, but considers parameters as regular hidden random variables. If the parameters are of interest themselves, Bayesian methods compute their posterior distribution by regular inference. If the parameters are not explicitly required, Bayesian methods perform standard inference on the remaining hidden variables, thereby integrating the parameters from the model.

More formally, given a parametric model $p(x_H, x_E|\theta)$ we assume prior probability distribution for parameters $p(\theta)$ and construct an augmented model

$$p(x_H, x_E, \theta) = p(\theta)p(x_H, x_E|\theta). \tag{2.17}$$

Suppose now that we have a training set $y_{1:N}$ of N realizations of the observed variables x_E and want to compute posterior distribution $p(\theta|y_{1:N})$. We can compute this distribution by solving a series of inference problems

$$p(\theta|y_1) \propto p(\theta) \sum_{x_H} p(x_H, x_E = y_1|\theta),$$
$$p(\theta|y_{1:n}) \propto p(\theta|y_{1:n-1}) \sum_{x_H} p(x_H, x_E = y_n|\theta),$$

where $n = 2, \ldots, N$ and each posterior $p(\theta|y_{1:n})$ distribution becomes prior for the next data point y_{n+1}. The above algorithm follows from the assumption that all data $y_{1:N}$ are independent samples from the model. In some (rare) cases, exact inference in such augmented models will be possible. Therefore it is convenient to use a so called *conjugate* prior, which guarantees the distributions $p(\theta)$, $p(\theta|y_{1:n})$ are represented with the same parametric family (Gelman et al., 1995). However, in most problems inference in the augmented models is intractable. Here it is convenient to use an expectation-propagation method, where all models $p(\theta)$, $p(\theta|y_{1:n})$ are approximated with the same parametric family.

Once the learning is complete, we have a posterior distribution $p(\theta|y_{1:N})$. This distribution can be plugged in the augmented model (2.17) in place of the prior $p(\theta)$. In this way all subsequent inference on hidden variables x_v, $v \in H$ will be implicitly conditioned on parameters acquired from the training data.

Interestingly, Bayesian formulation unifies learning and inference. For example, we can use the model even without training data. Given a single piece of evidence x_E we can just rely on the prior distribution $p(\theta)$ in (2.17) and estimate at the same time posterior distributions $p(x_v|x_E)$, $v \in H$ and $p(\theta|x_E)$.

Finally we note that, in practice Bayesian methods are often difficult to use because the inference in augmented models turns out very challenging. Factor functions that involve parameters treated as random variables become hard to integrate. However, there exist parametric models suitable for Bayesian parameter learning. In Chapter 4 of the thesis we consider a time-series model were most of the parameters are estimated within the Bayesian framework without much additional computational cost.

2.5 Summary

Probabilistic graphical models provide a convenient methodology for studying large-scale multivariate probability distributions. In the graphical framework — by introducing suitable assumptions — a probability distribution is expressed as a product and converted to a graph. Due to correspondence between many algebraic and graph-theoretic concepts, the underlying graph becomes a basis for solving various computational tasks related to probabilistic methods. In particular, probabilistic inference, that is the task of

computing appropriate marginal distributions can be solved by recursive procedures that propagate "messages" along the edges in the graph. Similarly, probabilistic learning, that is, the task of finding (parameters of) a probability distribution given training data can be solved by finding (parameters of) a suitable graph.

The described methodology forms common underpinnings of the visual surveillance techniques considered in the thesis. These techniques are presented as inference algorithms applied to specific graphical models. In fact, the inference algorithms applied in the thesis are instances of the general message-passing scheme. The scheme does not only offer a unified framework for the following chapters, but also immediately suggests ways to construct potentially improved tracking techniques (by applying potentially more accurate inference methods). Importantly, probability computations in the message-passing framework are local, in the sense, that they involve a limited subsets of variables (a subset of a graph). Therefore these methods often scale very well with the size of the model and potentially offer efficient, although sometimes approximate, solution to inference in complex problems.

CHAPTER 3

SEQUENTIAL DATA ASSOCIATION FOR MULTI-OBJECT TRACKING WITH SPARSE CAMERAS

This chapter focuses on data association problems that arise in tracking multi-object with sparsely distributed cameras.[1] Sparse camera locations introduce gaps between the areas monitored by the cameras. Multi-object tracking is such a setup requires reidentification of an object when it leaves one field of view, and later appears at some other. Essentially, the presented solution defines a dynamic Bayes network (DBN) as a probabilistic model for the observations of a single object. The edges of the network define the correspondences between observations of the same object. Accordingly, we derive an approximate EM-like method for selecting the most likely structure of DBN and learning model parameters.

3.1 Introduction

Visual tracking in wide areas, such as airports or motorways, relies on a network of closed-circuit TV (CCTV) cameras. Typically, exhaustive coverage of the entire area of interest is impossible due to the prohibitively large size of the monitored terrain. Therefore, the surveillance cameras are sparsely distributed, that is, the field-of-view of one camera covers only a relatively small scene that does not overlap with scenes observed by the other cameras. In this chapter we address the problem of reconstructing trajectories of multiple objects (people, cars) observed at such sparsely distributed camera systems. We refer to this problem as wide-area tracking.

The key problem in any multi-object tracking application is data association, i.e., estimating the number of trajectories and association of observations with the trajectories. In wide-area tracking this task corresponds to reidentification of an object when it disappears, and later appears in any the of fields of view. Typical sensors, such as cameras,

[1] The material of this chapter has appeared in *International Journal of Pattern Analysis and Applications* as (Zajdel and Kröse, 2005). Parts have been published in (Zajdel and Kröse, 2002).

cannot directly observe the identity of an object. Instead, cameras provide various location and appearance-related measurements about an object. Since these measurements do not uniquely identify an object, one often employs probabilistic models to encode ambiguous relation between the measurement and the identity of an object. Due to the inherent ambiguity, the number of possible trajectories and associations grows rapidly with the number of observations. Therefore, exact calculation of the most likely association is intractable (Bar-Shalom and Li, 1993) and one has to resort to approximate techniques.

A popular approximate approach is multiple hypothesis tracking (MHT) (Cox and Hingorani, 1994). In the context of wide-area tracking, MHT was applied for campus monitoring (Collins et al., 2001). MHT maintains a list of high probability associations (hypotheses) which are updated online. It is usually a fast and easy to implement method. On the other hand, MHT is a greedy-search heuristic that requires careful choice of thresholds for pruning low-probability associations. Importantly, most of the standard MHT applications assume probabilistic models which are predefined rather than learned from the data.

Alternatively, data association for multi-object tracking can be solved using stochastic approximations: sequential Monte Carlo ("particle filters") (Doucet et al., 2001; Hue and Cadre, 2002) or Markov Chain Monte Carlo (MCMC) (Gilks et al., 1996), e.g. motorway surveillance (Pasula et al., 1999). Stochastic methods have the attractive theoretical property that with increased computational resources, on average, they are guaranteed to give better solutions. In practice, the promised accuracy requires very high computational cost.

Another approach for wide-area tracking is to simplify data association by modeling a trajectory with an observable, first-order Markov chain (Huang and Russell, 1998; Javed et al., 2003; Kettnaker and Zabih, 1999). In this case, the assignment of a new observation depends only on the latest observation in each trajectory, rather than all possible past assignments. Unfortunately, when the sensor noise increases, the performance of such models deteriorates significantly faster than the performance of latent-state models (Pasula et al., 1999).

Wide-area surveillance can also be approached by splitting the entire area of interest into tightly aligned regions, such that each region is only partially covered by a sensor. A possible choice of sensors are cameras, but photocells (movement detectors) are also applicable (Nicholson and Brady, 1994). In this approach rather than finding correspondences between measurements from detectors, one computes posterior probability distributions on the regions where the objects interest might be present. On the practical side, the starting positions and the number of objects have to be known a-priori.

Multi-object tracking is also an active research subject for problems, where objects are tracked within a single field of view (Cai and Aggarwal, 1999; Dockstader and Tekalp, 2001; Hager and Belhumeur, 1998; Isard and Blake, 1998; Jang et al., 1997; Koller et al., 1994; Wren et al., 1997). In single-scene scenarios tracking takes place within a contin-

uous area, where smooth motion provides essential cues for resolving association ambiguities. Smooth position changes allow to apply continuous-motion models, such as Kalman filters (Bar-Shalom and Fortmann, 1988; Cox, 1993) for predicting object future positions from past observations. Smooth motion cues are also exploited by systems with multiple cameras observing the same scene (Gavrila and Davis, 1996; Li et al., 2002; Pedersini et al., 2001; Ruiz-Alzola et al., 2000). Various related approaches involve reducing the overlap area to a minimum guaranteeing smooth camera-to-camera transitions (Cai and Aggarwal, 1999) or pan-tilt active cameras that select the best overlap section (Ukita and Matsuyama, 2002).

Importantly, methods developed for single-scene tracking (using either a single camera or multiple cameras) are not immediately applicable to the sparse multi-camera problem (Huang and Russell, 1998). Any single-scene data association technique exploits the overlap in the viewing field and smooth motion for data association. Such cues are not available for tracking between discontinuous scenes, where objects are observed asynchronously within potentially long time intervals.

In this chapter we describe a probabilistic method for on-line association of observations and simultaneous learning of environment parameters (e.g. travel times). We assume a generative model for data in the form of a Dynamic Bayes Network with hidden states that represent appearance-related properties of objects. Our inference algorithm couples iterative reconstruction of trajectories with the Expectation Maximization approach for learning model parameters. In this way, we combine the learning ability of MCMC-based trackers and the ability to recover individual trajectories of the MHT approach.

3.2 Overview

We consider the problem of camera-to-camera tracking multiple persons observed with sparsely distributed cameras in an office-like environment. We do not assume or constrain the number of tracked objects. Although our goal resembles the motorway scenario (Pasula et al., 1999), there are two key differences. Firstly, we relax the assumption of unidirectional, constant-velocity vehicle movements. In a building people walk in all directions, possibly making long stops between camera locations. Secondly, we seek individual trajectories rather than average traffic parameters.

Since our method focuses on camera-to-camera trajectories we do not analyze the maneuvers of an object within a field of view of a single camera. We derive a "virtual" observation y_i from the sequence of frames that describe the complete pass of an object through the camera viewing field (as in Fig. 3.1). Further, the duration of object's presence in a viewing field is assumed to be significantly shorter than travel times between cameras. Therefore we will represent the interval within a camera field as a single timestamp. Chapter 4 describes an approach, where the timestamps of entering and leaving the field of view are explicitly taken into account.

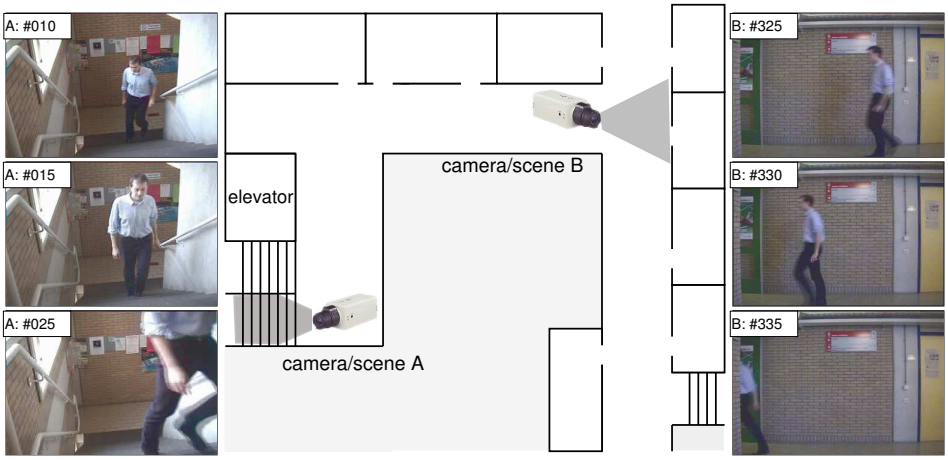


Figure 3.1: Camera-to-camera association of observations. The left and right columns are examples of frames taken from two passes through a camera viewing field. Each complete pass is considered as a single observation of a person.

Formally, an observation $y_i = \{a_i, t_i, l_i, e_i, d_i\}$ includes: the time of observation t_i — represented as a single moment, the summarized object's appearance (e.g. color statistics) a_i, the frame border of entering (e_i) and leaving (d_i) the field of view (this can be: right, left, top, down), and a discrete indicator l_i of the camera which observed the object. Note, that l_i indicates the area uniquely associated with the camera, therefore we can consider l_i as a location data about the object. The time t_i is computed as the average from the times of entering and leaving the field of view. We process observations from all cameras centrally, and treat i as a central index. We also assume that the observations are time ordered, i.e., $t_i < t_j$ iff $i < j$.

Given the set of N observations $Y = \{y_i; i = 1, \ldots, N\}$ we want to identify which observations belong to the same object. We are looking for a partition ω of observations from Y into several trajectories $Y_k \subset Y$ (subsets of Y) such that each trajectory collects observations believed to come from a single person. A valid partition expresses the set of all observations as an exhaustive union of trajectories: $Y = Y_1 \cup \ldots \cup Y_{K_\omega}$, where K_ω is the number of objects proposed by a partition ω. The trajectories must be mutually exclusive: $Y_l \cap Y_k = \emptyset$, when $l \neq k$. We note, that a partition corresponds to a hypothesis about trajectories of objects (this term is typically encountered in the nomenclature related to multiple-hypothesis trackers).

We consider a partition as a discrete-valued, random variable $\omega \in \Omega$, where Ω is the space of valid partitions. Conditioned on the data Y we find the posterior probability distribution of partitions, and select the most likely a-posteriori (MAP) ω. In Section 3.3 we define a probabilistic model $p(Y|\omega)$, which becomes a basis for computing partition posterior distribution. The model includes (initially unknown) parameters that describe

our environment, like the average travel times between camera locations. Accordingly, selecting the best partition has to be coupled with learning the parameters. Both tasks are difficult to solve exactly, due to an intractable number of possible partitions. Our approximate method (Section 3.4) is based on an EM (Dempster et al., 1977) algorithm. It starts with an initial guess on parameters and a tractable partition space obtained using a few initial observations. Each iteration refines the parameters and updates the limited partition space with new observations, essentially using an MHT-like greedy search. Therefore the algorithm can be summarized as a "learn-able" MHT, or alternatively, as an EM learning procedure where the E-step is approximated with an MHT-based inference. The performance of the method on a real data set is evaluated in Section 3.5

3.3 Probabilistic generative model

A generative model $p(Y|\omega)$ provides the basic probabilistic relation that ties the available data Y and the sought partition ω. It gives the likelihood of data Y under the assumption that Y was generated according to the partition ω. Given the data and the model, the partition posterior probability follows from the Bayes' rule: $p(\omega|Y) = \alpha p(\omega)p(Y|\omega)$, where α is a scale factor. By assumption, all partitions are equally likely before the observations arrive, so we take an uniform prior $p(\omega)$.

Let ω define K_ω trajectories $Y = Y_1 \cup \ldots \cup Y_{K_\omega}$. A common assumption is that the tracked objects move independently (Cox, 1993; Pasula et al., 1999), therefore the generative model factorizes into a product of models for individual trajectories:

$$p(\omega|Y) = \alpha p(Y|\omega) = \alpha \prod_{k=1}^{K_\omega} p(Y_k|\omega), \tag{3.1}$$

where $p(Y_k|\omega)$ is the model for the kth trajectory. The trajectory model defines the joint probability of the selected observations under the assumption that they describe the same person $p(Y_k|\omega) = p(y_1^{(k)}, \ldots, y_{N_k}^{(k)}|\omega)$, where the selection $Y_k = \{y_1^{(k)}, \ldots, y_{N_k}^{(k)}\}$ is determined by the partition ω, and N_k is the trajectory length. Since trajectories differ in length, and each observation has several components, we will express the trajectory model as a dynamic Bayes network. In the rest of this section we discuss a model for only one person and omit the superscript (k).

As discussed in Chapter 2, Bayesian networks (BNs) offer a convenient way to express complex probability density functions (pdfs) as a product of simpler conditional pdfs (Jensen, 2001; Pearl, 1988). A BN is a directed acyclic graph denoting variables as vertices and probabilistic dependencies as edges (Fig. 3.2). Dynamic Bayesian networks (DBNs) represent models for discrete-time series of variables with a series of interconnected subgraphs. DBNs generalize other probabilistic models that describe time series, like hidden Markov models or Kalman filter models (Ghahramani, 2001).

The graph of Fig. 3.2 identifies a set of dependencies between the variables in the sequence $\{y_1, \ldots, y_{N_k}\}$. Each column of transparent nodes (subgraph) represents a single observation y_i. The gray node denotes a hidden variable f that describes the time-invariant, intrinsic properties related to object's appearance. When an object enters a viewing field we get noisy observations of this variable. Thus, the appearance a_i given f is independent of all past and future appearances a_j of this person. In this way we have saved the effort of constructing an explicit joint model $p(a_{1:N_k})$. To complete the definition of our DBN, we provide a pdf for every vertex conditioned on its parents, and a prior pdf for vertices that do not have parents:

- $p(a_i|f, l_i)$; Distribution on appearance a_i of an object observed at camera l_i. The appearance features a_i depend on intrinsics f and various object- and camera-specific factors that affect the measurement process. The former introduce distortions related to changes in body pose or viewing angle. The latter include camera-specific illumination conditions, jitter, etc. To minimize the distortions due the object-specific pose we use appearance features that are invariant to body-part configuration. The camera-specific distortions are considered as a stochastic noise, therefore we denote the distribution as $p(a_i|f, \theta(l_i))$, where $\theta(l_i)$ are noise parameters of the camera at location l_i. Our heuristic choice for this model is a linear Gaussian distribution: $\mathcal{N}(a_i|f + b(l_i), S(l_i))$. In the experiments with real-world data we verify the ability of such a simple distribution to capture the observation noise. The parameters $\theta(l_i) = \{b(l_i), S(l_i)\}$ will be learned from the data.
- $p(l_i, e_i|l_{i-1}, d_{i-1})$; Probability of arriving at location l_i via border e_i when departing from location l_{i-1} via border d_{i-1}. We specify this discrete distribution using prior knowledge of the building's layout. In contrast to (Bui et al., 2001) we do not define a separate motion model for each object. The distribution $p(l_i, e_i|l_{i-1}, d_{i-1})$ is a property of the environment, rather than an object. This is a fairy simple and generic approach, suitable for problems with an unknown number of objects. (Note, that object-specific motion models require more elaborate analysis, which typically involves learning higher-order Markovian dynamics individually for every object (Bui et al., 2001).)
- $p(t_i|l_{i-1}, l_i, t_{i-1})$; This pdf models the time of appearing at location l_i knowing that an object left l_{i-1} at time t_{i-1}. We use a truncated Normal distribution:

$$p(t_i|l_{i-1}, l_i, t_{i-1}) = \begin{cases} 0 \text{ iff } t_i < t_{i-1} \\ \alpha\mathcal{N}(t_i|\delta(l_i, l_{i-1}) + t_{i-1}, R(l_i, l_{i-1})) \text{ otherwise,} \end{cases}$$

  where $\delta(l_i, l_{i-1})$ and $R(l_i, l_{i-1})$ are the expected travel time between the two locations and the variance of this distribution, and α denotes a normalization term. These parameters will be learned. Truncating the density function at zero prevents camera-to-camera transition with negative travel times. The Normal distributions facilities simple learning of the travel parameters, and have been successfully applied in (Huang and Russell, 1998; Pasula et al., 1999). (Alternatively, one might consider the gamma distribution, which does not require truncating).

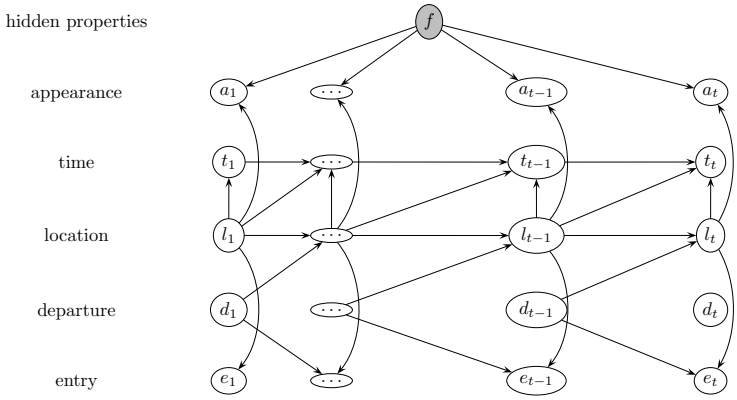


Figure 3.2: An object as a hidden process. The graph presents an intermediate snapshot of the DBN for a single object. The gray node (variable f) describes an object's hidden, time-invariant appearance. The arcs show dependencies between the variables. When a new observation is associated, the network is extended with a new column of nodes.

- $p(f)$; Prior distribution of the intrinsic properties of any object. We use a normal density: $p(f) = \mathcal{N}(f|\mu, R)$. The parameters μ and R will be learned.
- $p(l_1)$, $p(e_1|l_1)$; Distributions on locations and entry sides where a trajectory may start in the building. This distributions are given by the user since they follow from the layout of cameras in the building.
- $p(d_i)$; Prior probability of observing a particular departure side. We take it the same for every value of d_i, so the term $p(d_i)$ becomes a scaling factor.
- $p(t_1|l_1)$; Prior probability distribution on the time of object's first visit to some camera location. We assume that within the considered period a trajectory may start at any time. Thus, $p(t_1|l_1)$ is a scale term.

The joint probability of the variables in the graph is the product of pdfs associated with every vertex (Jensen, 2001). In the case of Fig. 3.2:

$$p(f, y_1, \ldots, y_{N_k}) = \\ p(l_1)p(e_1|l_1)p(t_1|l_1)\prod_{i=1}^{N_k} p(d_i)\prod_{i=2}^{N_k}\left[p(l_i, e_i|l_{i-1}, d_{i-1})p(t_i|t_{i-1}, l_i, l_{i-1})\right] p(f)\prod_{i=1}^{N_k} p(a_i|f, l_i).$$

The variable f is unknown, thus we integrate it in the final expression. The scaling terms $p(t_1|l_1)$, $p(d_i)$ are omitted:

$$p(Y_k|\omega, \Theta) = p(y_1, \ldots, y_{N_k}|\omega) \propto p(l_1)p(e_1|l_1) \\ \times \left\{\prod_{i=2}^{N_k} p(l_i, e_i|l_{i-1}, d_{i-1})p(t_i|t_{i-1}, l_i, l_{i-1})\right\}\int_f p(f)\prod_{i=1}^{N_k} p(a_i|f, l_i)\,\mathrm{d}f. \qquad (3.2)$$

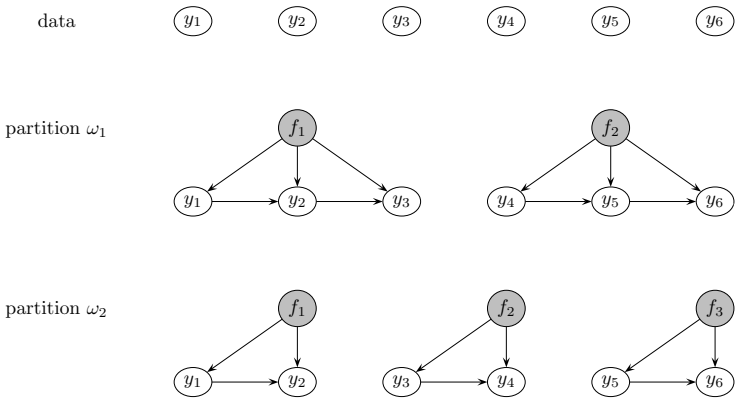


Figure 3.3: Selection of a Bayes network structure as a data association problem. The six observations could be generated from many different models (structures). Two hypothetical structures ω_1 and ω_2 are shown. A single network (e.g. $\{f_1, y_1, y_2\}$) is a compressed version of the corresponding full network as in Fig. 3.2.

The product of Gaussian appearance models $p(a_i|f, l_i)$ is another Gaussian function, therefore the integral in (3.2) has an analytical solution. See Section 3.7 for a discussion on how to compute this term.

Conditioning on Θ in (3.2) follows from the dependency of a trajectory likelihood on the unknown parameters: $\Theta = \{\mu, R, \delta(q, r), R(q, r), b(q), S(q)\}$, where q and r are camera locations. Consequently, this dependency reappears in the posterior probability (see (3.1)), and we have to learn Θ before or during selection of the best ω.

3.4 Associating observations

In the previous section we described a trajectory of a single person with a dynamic Bayes network. Accordingly, to describe multiple persons, we construct several disconnected DBNs as in Fig. 3.3. In this way, every partition ω translates to a structure of a larger network that comprises multiple disconnected subgraphs, each for every proposed object. In this section we present a deterministic approximate method for estimating the sought partition by recovering the structure that explains the data best.

Typical structure selection procedures for BNs compare candidate structures on the basis of a criterion (such as BIC/MDL) that tries to balance the complexity of a structure against its fit to the data (data likelihood) (Murphy, 2001). Such criteria prevent favoring very complex structures that naturally fit the data better. In our case every candidate ω uses the same parameters Θ, but may propose a different number of hidden variables f. These variables increase the expressive power of the candidate model. However,

we treat f in a Bayesian manner, i.e., we set a prior on every f and integrate it from $p(\omega|Y,\Theta)$. Such an approach is shown (MacKay, 1992) to sufficiently penalize over-complex models, therefore the posterior $p(\omega|Y,\Theta)$ is a valid criterion.

Another issues are the exponential number of possible structures (i.e., partitions) that can be defined for a dataset (Jerrum and Sinclair, 1996), and the fact that the parameters Θ need to be learned. Most of the approximate methods for these problems apply the EM algorithm (Murphy, 1998). For a tractable structure space, one could apply the "Structural EM" method that assumes that the structure is another parameter and exhaustively searches for the optimal value (Friedman, 1998). When the structure space is intractable, EM considers the structure as a hidden random variable. However, EM relies on expected values over the hidden variables. These expectations become difficult to compute for large structure spaces. In such cases the expected values are approximated with Markov chain Monte Carlo (MCMC) sampling methods.

In practice, applications using MCMC may suffer from slow convergence (so called *mixing times*), e.g. (Giudci and Castelo, 2001), where 10^5 MCMC iterations are used to find a distribution over the space of $3 \cdot 10^6$ structures with only 6 nodes. The success of MCMC for motorway surveillance partially follows from constraining the partition space by a natural assumption that on a motorway vehicles travel in one direction (thus, the next locations of the objects are roughly known) (Pasula et al., 1999). Apart from the fact that our domain is less constrained, we not only have to find expected values over this space, but also need the most likely partition. MCMC is useful only for approximating the expected value. Therefore, we are looking for a different approximate execution of the EM, such that it would also estimate the most likely structure in the space of all possible structures.

3.4.1 EM for a tractable structure space

Below we discuss a theoretical application of EM for learning of parameters Θ, based on the assumption that the structure space is tractable. The parameter estimates will be only locally optimal due to the approximate nature of EM. From this method, we will later derive an approximate learn-search algorithm for intractable spaces.

EM takes an initial value of Θ at random and improves it iteratively executing two steps per iteration. In the E step it finds the posterior pdf on all hidden variables given the current parameter estimates $\Theta^{(n)}$. In our application there is a hidden discrete-valued variable ω and as many hidden continuous-domain variables f as objects proposed by a particular ω. We denote the variables $f_1, \ldots, f_{K_\omega}$ as $f_{1:K_\omega}$. The interesting posterior is:

$$q(\omega, f_{1:K_\omega}) = p(\omega|Y,\Theta^{(n)}) \prod_{k=1}^{K_\omega} p(f_k|Y_k,\omega,\Theta^{(n)}), \tag{3.3}$$

where $p(f_k|Y_k,\omega,\Theta^{(n)})$ is the posterior distribution on the hidden properties of the kth object proposed by the partition ω. The above factorization followed from conditional

independence of trajectories given ω. In the M step, EM finds improved parameters $\Theta^{(n+1)}$ by maximizing the expected log-likelihood of the observations under $q(\omega, f_{1:K_\omega})$

$$\Theta^{(n+1)} = \arg\max_\Theta \sum_{\omega\in\Omega} \int_{f_{1:K_\omega}} \log p(\omega, f_{1:K_\omega}, Y|\Theta) q(\omega, f_{1:K_\omega}) \tag{3.4}$$

Section 3.7 provides the details of solving the above maximization problem. The EM procedure is guaranteed to find Θ that locally maximize the data likelihood. Given these parameters we can use $p(\omega|Y, \Theta)$ to find the MAP partition.

3.4.2 Approximate EM for an intractable structure space

In practice, the space of partitions Ω is intractable, and one cannot maximize (3.1) or execute the summation in (3.4) for every possible partition ω. Our method exploits the time order of observations to approximate the posterior distribution $p(\omega|Y), \omega \in \Omega$ over the full space Ω with a tractable subspace Ω_T of the H most likely partitions:

$$p^*(\omega|Y) = \begin{cases} \gamma p(\omega|Y) & \text{iff} \quad \omega \in \Omega_T \\ 0 & \text{otherwise} \end{cases}, \tag{3.5}$$

where γ is a normalization constant. We construct the subspace Ω_T iteratively as a part of the EM procedure.

Figure 3.4 presents an iterative scheme to process the observations sequentially. First we process an initial subset $Y^{(0)}$ of N_0 observations, and construct a tractable partition space $\Omega^{(0)}$. Since the parameters Θ were randomly initialized we need to execute several EM iterations. When EM stops we a have posterior distribution on partitions of observations from $Y^{(0)}$. At this point we build a subset $\Omega_T^{(0)}$ by selecting H partitions $\omega \in \Omega^{(0)}$ that are the most likely and remove the remaining partitions from $\Omega^{(0)}$. The subsequent observations will be used for refining the parameters and for updating the retained partitions. Each of the preserved partitions $\omega \in \Omega_T^{(0)}$ defines some K_ω trajectories. By extending every trajectory with a single observation or creating new trajectory, a single partition gives rise to $(K_\omega + 1)$ new partitions. Accordingly, after extending every preserved ω there is a new space of partitions $\Omega^{(1)}$. The new space is tractable, since we created it from a tractable $\Omega_T^{(0)}$. It is further extended with the subsequent observations until we process a batch of R observations. (R is a parameter.) The new tractable space $\Omega^{(1)}$ becomes a basis for updating the parameters with EM and searching for a new subset of the most likely partitions.

The described procedure combines the iterative EM algorithm with a greedy search algorithm for the best partition. By updating Θ after processing each batch of R observations we enforce that Θ follows changes in the noise environment. However, our scheme is not exact, because we discard unlikely partitions from Ω without considering all observations from Y. One should also note, that we estimate maximum-likelihood

$Y^{(0)} := y_{1:N_0}$
build $\Omega^{(0)}$ space of partitions of obs. in $Y^{(0)}$
use EM to find parameters $\Theta^{(0)}$,
build $\Omega_T^{(0)}$ by taking H $\omega \in \Omega^{(0)}$ with the highest $p(\omega|y_{1:N_0}, \Theta^{(0)})$

$n = N_0 + 1$; $s = 1$
while $n < N$
    $Y^{(s)} := y_{n:n+R}$
    build $\Omega^{(s)}$ by extending every trajectory of $\omega \in \Omega_T^{(s-1)}$ with $y \in Y^{(s)}$
    $\Theta^{(s)}$:= updated $\Theta^{(s-1)}$ with EM (single iteration)
    build $\Omega_T^{(s)}$ by taking H $\omega \in \Omega^{(s)}$ with the highest $p(\omega|y_{1:n+R}, \Theta^{(s)})$
    $n := n + R$; $s := s + 1$

Figure 3.4: Pseudo code for the learn-search procedure. Symbol $y_{n:m}$ denotes a set $\{y_n, \ldots, y_m\}$.

parameters from an incomplete data sequence (partial dataset). In order to avoid overfitting, it is recommended to run only a single EM iteration after each batch of R data.

Another important property of our procedure is that does not require a separate training data to fit the parameters of the models (indicated in Fig. 3.2). The parameters for color measurement models and travel-time models are estimated online from the actual test data, simultaneously with the tracking algorithm. A limited number of location-to-location distributions (denoted in Fig. 3.2 as $p(l_i, e_i|l_{i-1}, d_{i-1})$) is fixed by the user beforehand on the basis of the layout of camera network.

Computational cost The computational cost of evaluating the likelihood of a single partition ω is $\mathcal{O}(D^3 K_\omega)$, where D is the dimensionality of appearance features, and K_ω denotes the number of persons postulated by ω. After k observations we can have at most k persons, thus $K_\omega < k$. Since the algorithm maintains at most H partitions, the likelihood evaluation at the kth step costs $\mathcal{O}(HD^3k)$ in the worst case. Note that is is very unlikely that any partition will propose k persons from k observations; in most cases the number of persons will grow rather slowly with the number of observations. Additionally, the algorithm updates model parameters. If we have N cameras, there are N covariance matrices, each of D^2 elements, that model the noise in appearance features. Next, there are N^2 means and variances models for camera-to-camera travel times. A single update step for these parameters costs $\mathcal{O}(N^2 + ND^2)$ per partition. Assuming a single parameter update per observation, the total cost is $\mathcal{O}(H(D^3k + N^2 + ND^2))$, that is linear is the number of preserved partitions.

3.4.3 Relation to other methods

The key problem faced by any data association method stems from the intractability of the partition space. An alternative solution to this problem applies MCMC sampling algorithms (mostly used Metropolis-Hastings) (Pasula et al., 1999) to obtain a limited, sample-based representation of the partition space. The strength of MCMC methods follows from the fact the one can easily sample from the posterior $p(\omega|Y)$ that incorporates the *complete* dataset. However, when the partition space is unconstrained, the slow mixing time becomes a disadvantage. Our algorithm evaluates trajectories using partial datasets. In this way we keep the partition space tractable, but reject certain solutions on the basis of partial data.

Our method resembles a Multiple Hypothesis Tracker (MHT), because one may interpret a partition ω as a hypothesis about underlying trajectories. MHT evaluates $p(\{y^*, Y_k\}|\omega)$ in order to decide which of the trajectories Y_k proposed by some ω should be extended with a new observation y^*. We first consider all options by building a full new partition space derived from the current ω, learn the parameters and then prune unlikely partitions. In most of the standard MHT applications parameters are predefined rather than learned.

Similarly as MHT methods, the procedure of Fig. 3.4 might also be considered a simple form of a (Rao-Blackwellized) particle filter (PF) (Arulampalam et al., 2002). In visual tracking PFs are used for approximating intractable filtering densities, typically, on object's position (Isard and Blake, 1998). PF represents the density of interest with a set of samples (particles), which are resampled when new observations become available. In our case the hidden variable is the partition ω with a discrete, finite state space. We represent the posterior distribution with a few most likely elements (particles), because the whole state space is too large for exact representation. These particles are not however a result of resampling, but greedy search. In most of the PF applications for visual tracking the observation model is assumed to be known. In our method we assume a parametric observation model and we learn the model parameters.

3.5 Experiments

In a series of experiments we measure the performance of our method. First, we study the performance as a function of the observation model. In Section 3.3 this model was set to a linear Gaussian distribution with bias: $p_{\mathrm{B}}(a_i|f, l_i) = \mathcal{N}(a|f + b(l_i), R(l_i))$. The camera dependent bias $b(l_i)$ accounts for local illumination and pose of objects relative to the camera. Since the choice of this model is arbitrary we also try its simpler version, without the bias $p_{\mathrm{U}}(a_i|f, l_i) = \mathcal{N}(a|f, R(l_i))$. The next experiment studies the effect of the parameter H which determines the size of the approximate partition space Ω_T. Finally, we compare our method with an MCMC-based algorithm.

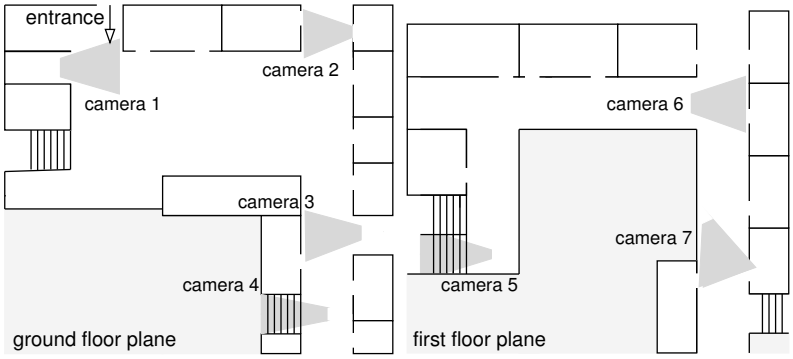


Figure 3.5: Building plan were the observations were taken. The seven gray areas indicate viewing fields of the seven cameras.

Setup We test our method on real-world human observations that were collected at seven disjoint locations at the ground and first floor of an office building as in Fig. 3.5. In total we gathered 70 observations of 5 persons, with an equal number of observations per person. For this set we manually resolved the data association to obtain the "ground truth" partition. Because the layout of the building is known, we can set the probability $p(l_i, e_i|l_{i-1}, d_{i-1})$ of arriving at location l_i at the side e_i when an object departs from position l_{i-1} through side d_{i-1}. This distribution is sketched in Fig. 3.6. In the building, we also manually set the models $p(l_1)$ and $p(e_1|l_1)$ that describe starting location and entry side of a trajectory.

As already indicated, an observation y_i summarizes object appearances captured by a sequence of frames during object's presence in a camera field of view. Given such a sequence, we have computed the appearance features a_i by first manually selecting a single frame where the object was clearly visible, and then manually segmenting pixels in the frame into object and non-object classes. In this way we constructed a data set that is independent from various inaccuracies introduced by automated segmentation. The appearance features a_i were set to a 9D vector containing 3D color means (RGB) computed over three regions selected in the object's image. The regions (see Fig. 3.7 for details) are a heuristic choice based on the assumption that we observe people. These features provide a simple way to summarize color while preserving partial information about geometrical layout. The resulting features are to certain a extent invariant to variations in person's pose. To suppress the effect of the illumination color on the object's observed color, we transformed the original RGB representation to a channel-normalized space (Drew et al., 1998).

Evaluation criteria In order to quantitatively compare various methods, one can view multi-object tracking as an *unsupervised* classification problem, where observations of

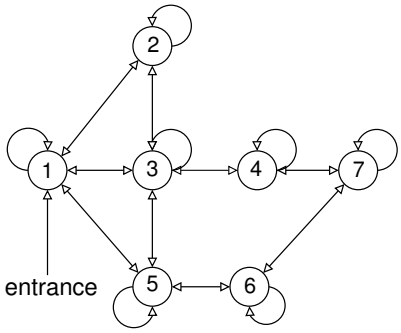


Figure 3.6: An illustration of camera-to-camera movements in the considered environment. The numerical labels correspond to cameras from Fig. 3.5, the arrows indicate possible transitions as defined by distribution $p(l_i, e_i|l_{i-1}, d_{i-1})$. In the drawing we omitted the entry/departure side indication.

the same person have to be clustered together (i.e., a cluster represents a trajectory), and the number of clusters is unknown. For the available dataset we construct a "ground-truth" partition that indicates the actual partitioning (or clustering) of the observations. Analogously as in the classification problems, our evaluation criteria reflect two aspects of the proper partition. It is desirable that: (i) observations within a single reconstructed trajectory correspond to a single person, and (ii) all observations of a single person are clustered in a single reconstructed trajectory.

The first criterion is the partition *accuracy*, denoted as q_ω,

$$q_\omega = \frac{1}{K_\omega} \sum_{k=1}^{K_\omega} q_{k,\omega} \qquad\qquad q_{k,\omega} = \frac{\max_i \mid \bar{Y}_i \cap Y_k \mid}{|Y_k|} \cdot 100\%, \tag{3.6}$$

where K_ω is the number of trajectories defined by ω, and $|.|$ is the number of observations in a set. The term $q_{k,\omega}$ indicates how many observations of a *proposed* subset Y_k are present in a single subset $\bar{Y}_i$ of the *true* partition. Let the "true" partition be $\bar{\omega} = \bar{Y}_1 \cup \ldots \cup \bar{Y}_N$. Note that the considered tracking problem is unsupervised, therefore the trajectories in the true and proposed partitions are arbitrarily ordered. To deal with this ambiguity, we find a true trajectory $\bar{Y}_i$ that best matches a proposed trajectory Y_k, and define $q_{k,\omega}$ as the accuracy of the best match.

The second criterion is the partition *recall*, denoted as ρ_ω,

$$\rho_\omega = \frac{1}{K_{\bar{\omega}}} \sum_{i=1}^{K_{\bar{\omega}}} \rho_{i,\omega} \qquad\qquad \rho_{i,\omega} = \frac{\max_k \mid \bar{Y}_i \cap Y_k \mid}{|\bar{Y}_i|} \cdot 100\%, \tag{3.7}$$

where $K_{\bar{\omega}}$ is the number of trajectories defined by the true partition $\bar{\omega}$. The term $\rho_{i,\omega}$ indicates how many observations of a *true* subset $\bar{Y}_i$ are present in a single subset Y_k of

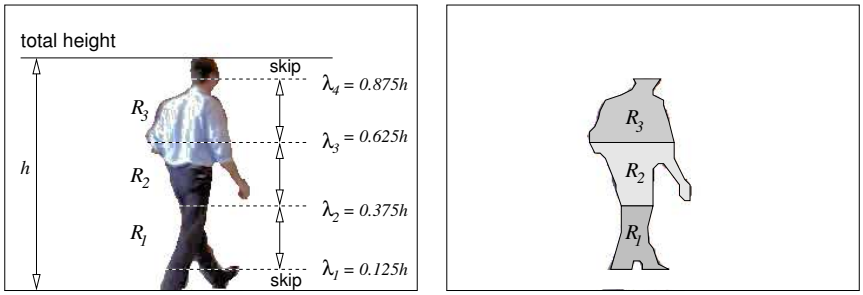


Figure 3.7: (Left) Computation of appearance features. For every pixel we denote the RGB triple as $I_{(m,n)}$, where m and n are, respectively, the vertical and horizontal coordinates. We let $r_{(m,n)} \in \{0, 1\}$ denote a binary label that indicates whether the pixel represents an object ($r_{(m,n)} = 1$). The image is split into three horizontal stripes (regions), defined as $R_i = \{I_{(m,n)} | r_{(m,n)} = 1, \lambda_i < m < \lambda_{i+1}\}$, $i = 1, 2, 3$, where the thresholds λ are indicated in the figure. Within every region, we compute a 3-dimensional (RGB) average color; in total obtaining a 9-dimensional feature vector. The "skip areas" correspond to image regions that usually provide little information. (Right) The resulting regions.

the *proposed* partition (that is how completely the true trajectory has been recovered). Analogously to the previous criterion, given a true trajectory $\bar{Y}_i$ we select the proposed trajectory Y_k that best matches $\bar{Y}_i$.

Importantly, the above criteria have to be considered jointly, particularly in case of degenerate partitions. For example, proposing one trajectory $Y_k = \{y_k\}$ per observation would always give a 100% accuracy. Similarly, clustering all observations into a single trajectory yields a 100% recall. To provide a more intuitive assessment of a partition in such degenerate cases, we also report the *number of reconstructed trajectories*. Comparing this number with the actual number of trajectories provides additional insight to the properties of various tracking methods.

Results A sample run of our method with the bias-enabled linear Gaussian model $p_{\mathrm{B}}(a_i|f, l_i) = \mathcal{N}(a|f + b(l_i), R(l_i))$ is shown in the top panel of Fig. 3.8. The parameters were $H = 10, N_0 = 6, R = 2$. Each dot represents a partition from the estimated subspace Ω_T. The line in the figure connects the partitions that maximize the posterior probability. After an initial period of correct associations, the performance levels at around 60%, meaning that only 6 out of 10 observations assigned to a single trajectory really describe the same person. The average accuracy of MAP ω indicated in the figure is the average of q_ω for MAP partitions after processing each batch of R observations.

The bottom panel of Fig. 3.8 presents the corresponding run of our method with the bias-free Gaussian noise model $p_{\mathrm{U}}(a_i|f, l_i) = \mathcal{N}(a|f, R(l_i))$. All parameters were kept the same as in the previous run. In the case of the simpler model the method provided nearly perfect assignments for approximately the first 36 observations, whereas in the

Table 3.1: Tracking accuracy of the described method, averaged from 10 runs. Parameters $N_0 = 6$, $R = 2$. The actual number of objects was 5. Computation times correspond to a MATLAB implementation and an 1 GHz desktop PC.

parameter	bias-free noise model				bias-enabled noise model			
H	accuracy [%]	recall [%]	objects	time [s]	accuracy [%]	recall [%]	objects	time [s]
1	48	55	5	54	44	50	5	102
5	77	70	5	258	57	66	5	471
10	78	79	5	515	54	66	5	943
20	76	79	5	1036	46	63	5	1892

case of the general model – for only 16. For the subsequent observations (starting at around observation 54) we observe that the MAP partition coincides with the highest-accuracy partition. Figure 3.9 presents selected trajectories as defined by the estimated MAP partition after processing 70 observations.

The evaluation of our method with different noise models and varying setting of parameter H is given in Tab. 3.1. The results are averages from 10 runs. We observe that keeping only a single ($H = 1$) partition was not enough to obtain an accurate solution. When the number of preserved ω's was slightly higher ($H = 5, 10$) the performance improved. However increasing H further did not improve the results, or even the performance deteriorated in case of the general observation model. Note, that the experiments confirm the linear dependency of computational cost on the parameter H.

There are two main reasons why the bias-free noise model performs better on the considered dataset. First, the EM method for parameter estimation is prone to finding only the locally optimal estimates. The more parameters assumed by the model, the more local maxima exist in the data likelihood function. In our application this could be especially the case since we always run EM on a partial dataset. The bias estimates found from the initial observations may not fit well to the subsequent data. The second reason follows from our image pre-processing step. The camera-specific bias parameter might be unnecessary because we suppress camera-dependent noise artifacts by using pose-invariant features and a channel-normalized color space. Finally, we note that the above results comply with the recent findings that the modern CCD cameras do not introduce bias into color measurements (Withagen et al., 2005).

Finally, we note that the appearance features are computed from multiple frames, where an object was visible during his/her pass through the field of view. For individual frames, the appearance features might be strongly affected by various noise artifacts (e.g. due to a bad segmentation or occlusions). Nevertheless, given a complete pass, such artifacts can be easily eliminated as "outliers".

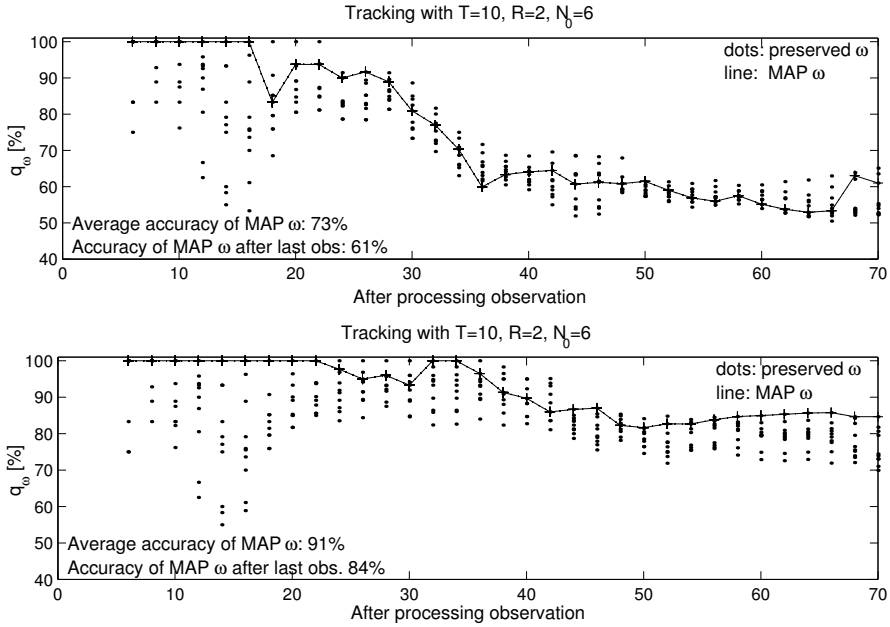


Figure 3.8: Results of tracking with normal prior distribution on features f. The parameters of the distribution were randomly initialized and learned from data. (Top) Observation model is linear Gaussian with bias parameter estimated by EM. (Bottom) Observation model is a bias-free Gaussian (with linear bias parameter fixed to zero).

MCMC-based algorithm Another series of experiments evaluate the MCMC-based approximations to the partition space. We implemented the Metropolis-Hastings sampling algorithm (Pasula et al., 1999). In essence, the partitions $\omega_i, i = 1 \dots N$ are sampled as follows: a new ω_{i+1} is obtained from ω_i by random selection of two observations assigned to different trajectories and swapping their assignment. Then we accept ω_{i+1} at random with probability proportional to $p(\omega_{i+1}|Y)/p(\omega_i|Y)$. In the motorway scenario (Pasula et al., 1999), construction of the initial sample exploits the natural motorway constraint that objects (vehicles) move in a single direction. In our office scenario, the objects (people) move without constraints. Therefore, we consider two different ways to initialize sampling. First, we build the initial sample in a purely random way. Second, we obtain the initial sample as a result of a greedy search, using a simple MHT tracker with only one hypothesis.

We use the bias-free observation model $p_{\mathrm{U}}(a_i|f, l_i) = \mathcal{N}(a|f, R(l_i))$, and prior models the same as in our algorithm. Since we cannot average over the sampled partitions, as the final solution we took the sample that maximized the data likelihood $p(Y|\Theta, \omega_i)$. Table 3.2 shows the evaluation results of the partitions found by MCMC runs with varying number of samples N and different methods for sample initialization.

Table 3.2: Tracking accuracy of the MCMC based-algorithms. The results are averages over 5 runs. The actual number of objects was 5.

	random initialization			
samples	accuracy [%]	recall [%]	objects	time [s]
10^2	57 ± 9.8	31 ± 9.4	18.8 ± 8.0	89 ± 4
10^3	66 ± 5.3	33 ± 2.8	13.8 ± 2.1	789 ± 16
10^4	78 ± 2.1	53 ± 3.8	9.6 ± 0.8	7689 ± 53
10^5	86 ± 3.4	79 ± 6.0	7.0 ± 0.7	76631 ± 50

	MHT initialization			
samples	accuracy [%]	recall [%]	objects	time [s]
10^2	62 ± 7.7	49 ± 8.4	7.4 ± 1.0	97 ± 0
10^3	73 ± 4.9	60 ± 3.9	7.2 ± 0.4	759 ± 3
10^4	87 ± 1.4	77 ± 3.2	6.6 ± 0.5	7401 ± 19
10^5	88 ± 1.2	81 ± 1.9	6.0 ± 0.0	73992 ± 17

In comparison with the Tab. 3.1 it turns out that the MCMC approximation is less suited for finding a single reliable partition ω in an office-like tracking scenario. In particular, it is difficult to estimate the correct number of objects. The original application of MCMC for tracking largely exploited the specific motorway traffic properties for disambiguating the number of distinct objects (Pasula et al., 1999). In that case a vehicle enters a motorway in one of the known locations, exits it only in specific off-ramps, and never (implicit assumption) visits the same location twice. We also note that the MCMC runs tend to be computationally more intensive, and their results highly depend on the initial sample.

3.6 Conclusions

We proposed a data association algorithm for tracking people observed with sparsely distributed cameras in an office environment. In particular, we have focused on re-identification of a person when he/she leaves the viewing field of one camera and later appears at some other camera. The algorithm assumes a probabilistic relation between the available object observations and the sought association. Given the data, our approximate inference procedure iteratively reconstructs a limited space of a-posteriori plausible associations. The proposed scheme is derived from the EM algorithm in order to combine learning model parameters with the search through the association space.

The described EM-based algorithm for learning model parameters resembles the recent approaches used by the MCMC-based trackers. However, in contrast to the methods employing MCMC sampling, our method is able to recover a single reliable partition

Figure 3.9: Trajectories defined by a partition ω estimated by our algorithm. Every image represents a frame from a person's pass through the filed of view of some camera. The "true labels" indicate the ground-truth association. The numerical camera indicators correspond to Fig. 3.5. From the true labels, we see that the trajectories #1, #2, #5 were reconstructed correctly, the trajectories #3 and #4 are mixed.

(association) of the observation sequence into individual trajectories. The method could be also viewed as an extension to Multiple Hypothesis Tracking with on-line learning of model parameters. Therefore, it is particularly suited for on-line applications.

The presented greedy search for the optimal association offers an alternative to the methods employing stochastic approximations of the association space. This space is particularly difficult to approximate in the indoor tracking domain, which lacks strong traffic constraints. The presented experiments suggest that for this domain, sequential search for trajectories from an incomplete dataset provides better results than sampling associations from the complete dataset.

3.7 Appendix

EM for Bayes network parameters Below we provide implementation details for finding maximum-likelihood model parameters Θ with EM. In the E step we find posterior distribution on all hidden variables:

$$\begin{aligned} q(\omega, f_{1:K_\omega}) &= p(\omega, f_{1:K_\omega}|Y, \Theta^{(n)}) = \\ &= p(\omega|Y, \Theta^{(n)}) p(f_{1:K_\omega}|Y, \omega, \Theta^{(n)}) = p(\omega|Y, \Theta^{(n)}) \prod_{k=1}^{K_\omega} p(f_k|Y_k, \omega, \Theta^{(n)}), \end{aligned} \tag{3.8}$$

where $p(f_k|Y_k, \omega, \Theta^{(n)})$ is the posterior distribution on the hidden properties of the kth object proposed by a partition ω. The factorization in (3.8) followed from trajectory conditional independence given ω (the subgraphs of Fig. 3.3 are disconnected). The term $q(\omega) = p(\omega|Y, \Theta^{(n)})$ is given by (3.1). To find the function $q(f_k|\omega) = p(f_k|Y_k, \omega, \Theta^{(n)})$ we exploit the independencies implied by the trajectory model, as in Fig. 3.2

$$q(f_k|\omega) = p(f_k|Y_k, \omega, \Theta^{(n)}) \propto p(f_k) \prod_{i=1}^{N_k} p(a_i^{(k)}|f_k, l_i^{(k)}), \tag{3.9}$$

where N_k is the number of observations assigned to the kth object, $a_{1:N_k}^{(k)}$ and $l_{1:N_k}^{(k)}$ are sequences of appearance features and locations of the kth object as proposed by ω. Since all models in (3.9) are linear and Gaussian, the posterior pdf will be a Normal density, what makes (3.9) a special case of the Kalman filter.

In the M step we solve the following maximization problem:

$$\Theta^{(n+1)} = \arg\max_\Theta \sum_{\omega \in \Omega} \int_{f_{1:K_\omega}} \log p(\omega, f_{1:K_\omega}, Y|\Theta) q(\omega, f_{1:K_\omega}).$$

We express the joint $p(\omega, f_{1:K_\omega}, Y|\Theta)$ as a product of $p(f_{1:K_\omega}, Y|\omega, \Theta)$ and $p(\omega|\Theta)$. The latter is fixed uniform, so it is omitted. We solve

$$\Theta^{(n+1)} = \arg\max_\Theta \sum_{\omega \in \Omega} \int_{f_{1:K_\omega}} \log p(f_{1:K_\omega}, Y|\Theta, \omega) q(\omega, f_{1:K_\omega}).$$

The joint pdf $p(f_{1:K_\omega}, Y|\Theta, \omega)$ factorizes into a product of trajectory models (see Fig. 3.3). As a result the maximized term becomes:

$$\Theta^{(n+1)} = \arg\max_\Theta \sum_{\omega\in\Omega} \int_{f_{1:K_\omega}} \sum_{k=1}^{K_\omega} \log p(f_k, Y_k|\Theta, \omega) q(\omega, f_{1:K_\omega}).$$

Solving the integral for every component of the sum yields:

$$\Theta^{(n+1)} = \arg\max_\Theta \sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \int_{f_k} \log p(f_k, Y_k|\Theta, \omega) q(\omega, f_k).$$

The above expression is our starting point for finding $\Theta^{(n+1)}$.

As an example we show how to update the prior distribution of object hidden properties. The distribution was assumed normal $p(f) = \mathcal{N}(f|\mu, R)$. Thus we find parameters μ and R. Since $\log p(f_k, Y_k|\Theta, \omega) = \log p(f_k) + \log p(Y_k|f_k\Theta, \omega)$ and $p(Y_k|f_k\Theta, \omega)$ does not depend on the sought μ and R; we maximize:

$$\sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \int_{f_k} \log p(f_k) q(\omega, f_k) = \sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \int_{f_k} \log \mathcal{N}(f_k|\mu, R) q(\omega, f_k). \tag{3.10}$$

In order to maximize (3.10) w.r.t μ we take the derivative w.r.t μ, set it to zero and solve for μ:

$$\mu^{(n+1)} = \frac{\sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \int_{f_k} f_k q(\omega, f_k)}{\sum_{\omega\in\Omega} K_\omega q(\omega)} = \frac{\sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \mu_k q(\omega)}{\sum_{\omega\in\Omega} K_\omega q(\omega)},$$

where the joint pdf $q(\omega, f_k)$ was expressed as a product: $q(f_k|\omega)q(\omega)$ (see the E step). The posterior $q(f_k|\omega)$ is a Normal pdf, with mean μ_k and covariance R_k. The integral $\int_{f_k} f_k q(f_k|\omega) = \mu_k$ is the expected value of this distribution (we have it from the E step).

Similarly one can find the R that maximizes (3.10):

$$R^{(n+1)} = \frac{\sum_{\omega\in\Omega} \sum_{k=1}^{K_\omega} \left(R_k + \mu_k\mu_k^\top - \mu_k{\mu^{(n)}}^\top - \mu^{(n)}\mu_k^\top + \mu^{(n)}{\mu^{(n)}}^\top \right) q(\omega)}{\sum_{\omega\in\Omega} K_\omega q(\omega)},$$

where $\mu^{(n)}$ is the past estimate of parameter μ, R_k is the posterior covariance of $q(f_k|\omega)$ and $\mu^\top$ indicates transposition. Note, that the summation over ω is tractable because our approximation replaces the full partition space Ω with a tractable subspace Ω_T as in (3.5). In a similar way we find the observation model parameters (per camera).

Finding updated parameters of a model for travel times (per camera pair) is slightly more complicated since the assumed model is a truncated Gaussian distribution (therefore is non-Gaussian). We find the updated parameters (mean, covariance) as if the model was Gaussian and next truncate the updated model. Although, not formally exact, such an approach works well in practice (see also Pasula et al. (1999)).

CHAPTER 4

A HYBRID GRAPHICAL MODEL FOR ONLINE MULTICAMERA TRACKING

This chapter presents another approach for multi-object tracking with sparsely distributed cameras.[1] In contrast to the previous chapter, we define a single graphical model for all objects. For each object, the model embeds a hidden continuous state variable and a hidden discrete label that uniquely identifies the object. The model provides a joint probability distribution for observations, states and labels of all objects. Given the model and the observations, we apply the assumed-density filtering algorithm to compute marginal posterior distributions on labels, which resolve association ambiguities. The presented approach facilitates a principled estimation of the number of objects and various model parameters by Bayesian inference.

4.1 Introduction

As we have discussed in the previous chapter, visual surveillance in wide areas relies on a network of sparsely distributed cameras. In this setup we are interested in tracking various objects (like people) on the basis of their observations provided by the cameras. Similarly as in the previous chapter, we focus on reidentification of an object when it disappears, and later appears in the field of view of any camera.

Most of the existing approaches for this task, including the algorithm described in Chapter 3, view reidentifiaction as a problem of partitioning a set of observations of objects into subsets, such that each subset includes all observations of one object. This approach requires searching for an optimal partition in the space of possible partitions of a given dataset. Most important algorithms from this family include: a method based on MCMC sampling (Pasula et al., 1999), various exhaustive search methods (Huang and Russell, 1998; Javed et al., 2003; Kettnaker and Zabih, 1999) or MHT-based methods (Collins et al., 2001; Zajdel and Kröse, 2005) (Zajdel and Kröse, 2003a) (see Chapter 3

[1] The material of this chapter has been submitted to *Pattern Recognition* Journal (Zajdel et al., Submitted, 2005a). Parts have been published in (Zajdel et al., 2004) and (Zajdel et al., 2005b).

for an overview). On one hand, all these methods are relatively simple to implement since every candidate partition is explicitly evaluated according to some criterion (typically, by computing partition likelihood according to a probabilistic model). On the other hand, except from some limited cases, finding good candidates in the space of possible partitions is computationally intractable.

Intuitively, the method discussed in this chapter is an attempt to avoid explicit search for the optimal partition. We assume that every object can be identified with a unique, imaginary label. Since a camera cannot directly observe the label, it is considered as a hidden random variable. We infer the labels from observations by computing *approximate* label probabilities and selecting the most likely label for every observation. In this way we approximately identify objects without constructing partitions.

From a more formal viewpoint, the presented method is an instance of an approximate, deterministic inference algorithm applied to a specific *hybrid* probabilistic model. The model explicitly identifies objects with hidden *discrete* labels and treats observations as noisy measurements of objects' *continuous* hidden states. More specifically, we model multiple objects by considering a mixture density, where each mixture component is parametrized by the state of a different object. Since the number of objects is unknown, the model allows new components with new observations, analogously to the "infinite" mixture models (Rasmussen, 2000).

The above formulation allows us to associate observations with trajectories by Bayesian inference. We compute posterior probability densities of labels and states given the observations and find the most likely label for each observation. These densities, however, cannot be computed exactly and we approximate them using an assumed-density filtering (ADF) algorithm. ADF replaces complicated probability densities with simpler functions, while preserving the moments of the original density. Such an approach is motivated by successful applications of ADF to many other hybrid probabilistic models (Murphy, 2002).

In the next section we present assumptions about our tracking environment and define the probabilistic model. In Section 4.3 we describe an inference algorithm applied to the model. In Section 4.4 the resulting tracking algorithm is demonstrated and quantitatively compared with the MCMC and MHT methods using real-world data. Sections 4.5 and 4.6 conclude the chapter.

4.2 Probabilistic Generative Model

As already mentioned, we focus on tracking objects as they move between sparsely distributed cameras. In this chapter we are not interested in the problem of monitoring an object within a single field of view. This step is considered as a "measurement" that yields an observation of an object whenever the object passes the field of view of a camera.

Our tracking approach follows from an idea to describe objects with unique labels and "guess" from observations of objects. We define a probabilistic relation between the (unknown) label and the observation by introducing an intermediate unknown term, called a hidden state of an object. For example, the actual color histogram of a person could be considered as the hidden state, and the measured color histogram as an observation. The relation between labels, observations and states takes the form of a joint probability distribution, which can be *informally* denoted as $p(\text{Obs}, \text{States}, \text{Labels}) = p(\text{Obs}|\text{States}, \text{Labels})p(\text{States}, \text{Labels})$, where the last expression is an assumed prior distribution for unknown states and labels. We define this distribution using the formalism of Bayesian networks (generative models). Given the model, we compute posterior distribution

$$p(\text{Labels}|\text{Obs}) \propto \int_{\text{States}} p(\text{Obs}|\text{States}, \text{Labels})p(\text{States}, \text{Labels}), \tag{4.1}$$

and find the most likely labels. In this way associate observations with objects. In the rest of this section we first define a model for a single observation, then the prior density, and finally a model for a sequence of observations. The integration problem is an instance of Bayesian inference task. In Section 4.3 we will apply an established inference technique — the assumed-density filtering algorithm (ADF) — to compute the posterior densities of interest.

4.2.1 Model for a single observation

We denote an observation (a measurement vector) as $y_k = \{o_k, d_k\}$, where o_k denotes an r-dimensional appearance vector and d_k are spatio-temporal features. The appearance vector o_k is a collection of color features computed from various body parts of a person (see Section 4.4). The spatio-temporal features are $d_k = \{\ell_k, t_k^{\text{e}}, t_k^{\text{q}}, b_k^{\text{e}}, b_k^{\text{q}}\}$, where ℓ_k is a discrete camera location indicator, $t_k^{\text{e}}, b_k^{\text{e}}$ ($t_k^{\text{q}}, b_k^{\text{q}}$) are the time stamp and frame border of entering (or leaving) the scene. We process observations from all cameras centrally, and treat $k = 1, 2, \ldots$ as a central index that preserves time-order of observations.

Appearance features Our model assumes that the observed, r-dimensional vector o_k is a noisy measurement of a hidden random variable x_k. We call the variable x_k a hidden state of a person (or just a state). The state can be interpreted as a numerical representation of some constant, intrinsic property of an object. The noise typically arises from factors like camera jitter or variations in object's pose relative to the camera. For the simplicity of modeling, we assume that the noise is Normal distributed. The state x_k is considered as the parameter vector $x_k = \{\mathbf{m}_k, \mathbf{V}_k\}$ of the Normal probability density that generated y_k. Thus, the state $x_k = \{\mathbf{m}_k, \mathbf{V}_k\}$ is a $(r + r^2)$-dimensional random vector that is composed of two parts: $\mathbf{m}_k$ (r-dimensional mean vector) and $\mathbf{V}_k$ (which is constrained to represent a positive-definite $r \times r$ matrix). The hidden state x_k and the

Figure 4.1: An example of the considered tracking problem, with three cameras: 'A', 'B' and 'C' that observe non-overlapping scenes. Every image depicts a complete pass of a person through a camera viewing field. The goal of wide-area tracking is to reconstruct global trajectories between the cameras/scenes.

observed vector o_k are related as

$$o_k|x_k \sim \mathcal{N}(\mathbf{m}_k, \mathbf{V}_k),$$

where $\mathbf{m}_k$ and $\mathbf{V}_k$ are, respectively, the expected value and the covariance matrix of the Normal probability density function (pdf).

The above definition of state x admits a natural interpretation. The mean vector $\mathbf{m}$ describes the person-specific appearance features. The covariance $\mathbf{V}$ corresponds to person-specific noise amplitude. For instance, the appearance of somebody dressed uniformly is relatively independent of pose, so his/her noise level is low. In contrast, the appearance of a person wearing non-uniform colors, is very sensitive to pose changes, therefore the noise level will be higher. (Generally, the noise amplitude corresponds to the sum of eigenvalues of $\mathbf{V}$.)

Spatio-temporal features Another component of the observation y_k are the spatio-temporal features d_k. In our, and many other approaches (Huang and Russell, 1998; Javed et al., 2003; Kettnaker and Zabih, 1999; Pasula et al., 1999), these features are assumed noise-free. A sequence of such features defines a path of a person between the scenes. It is convenient to assume that each path feature d depends only the directly preceding feature. To define a probabilistic model that generates d_k we express this dependency as a distribution

$$d_k \sim p_\delta(d|d_{\text{prec}(k)}), \tag{4.2}$$

where $\text{prec}(k)$ is the index of the last observation assigned to the same person as observation k. If d_k starts a new path, then it is generated by an initial distribution $p_{\delta_0}(d)$. In fact, this model describes a path of any person as a first-order Markov chain.

Intuitively, the distribution p_δ reflects constraints on motion between scenes. For example, consider Fig. 4.1 and assume that spatio-temporal features include only scene indicators $\ell \in \{\text{'A'}, \text{'B'}, \text{'C'}\}$. In this case, $p_\delta(\ell_\text{b}|\ell_\text{a})$ gives probabilities of direct transition from scene ℓ_a to scene ℓ_b. One might set $p_\delta(\text{'A'}|\text{'A'}) = p_\delta(\text{'B'}|\text{'A'}) \gg p_\delta(\text{'C'}|\text{'A'})$ because it is unlikely to move from 'A' to 'C' without being observed at camera 'B'. In practice, the model also reflects temporal constraints, like minimum travel time (see Section 4.4).

4.2.2 Prior density for states

The Bayesian approach requires a prior probability density $\pi(x)$ for the hidden states. To select a suitable density we recall that a state x represents the parameters (i.e., mean $\mathbf{m}$ and covariance $\mathbf{V}$) a Normal pdf. We follow the standard approach for Bayesian estimation of these parameters and apply "Normal-Inverse Wishart" density as the prior (Gelman et al., 1995). The density, denoted as

$$\phi(x|\theta) = \mathcal{N}(\mathbf{m}|\mathbf{a}, \kappa\mathbf{V})\mathcal{IW}(\mathbf{V}|\eta, \mathbf{C}) \tag{4.3}$$

represents mean $\mathbf{m}$ using a Normal pdf, (where $\mathbf{V}$ is interpreted as a parameter), and covariance $\mathbf{V}$ using an "Inverse Wishart" $\mathcal{IW}$ pdf. The Inverse Wishart is a multivariate generalization of the standard Inverse Gamma density. The term $\theta = \{\mathbf{a}, \kappa, \eta, \mathbf{C}\}$ denotes all parameters of this family. If we fix matrix $\mathbf{V}$, then the vector $\mathbf{a}$ is the expected value of $\mathbf{m}$, and $\kappa\mathbf{V}$ is the covariance of $\mathbf{m}$. Matrix $\mathbf{C}$ and scalar η define moments of the Inverse Wishart density Press (1972), e.g., the expected value of $\mathbf{V}$ is $\frac{\mathbf{C}}{\eta-r-1}$. We denote the prior as $\pi(x) = \phi(x|\theta_0)$, and in Section 4.4 define appropriate parameters θ_0.

4.2.3 Model for a sequence of observations

Below we define a probabilistic model that embeds observations of all objects provided by all cameras. The model is constructed by using the previously defined models — the model for a single observation and the state prior — as basic building components. Our approach for designing the model follows from the fact that in the considered scenario objects are observed *asynchronously*, that is, observations from cameras typically arrive one at a time. (Of course, it might happen that two or more observations will arrive at the same time; in this case we can order the observations arbitrarily, keeping in mind that the wall-clock timestamps are preserved anyway.)

Given the above intuition, we organize the model as a discrete-time series, where a single "time" step corresponds to a single observation y_k, and $k = 0, 1, \ldots$ is a time index, augmented with a zeroth state. For every time step k, the model assumes variables $\{y_k, x_k, h_k\}$, where y_k is the observation, x_k is the state of the object represented by y_k and h_k collectively denotes various variables for "bookkeeping" in the model. Since we consider a single model for all objects, the bookkeeping variables are necessary for

maintaining information about which observations and states correspond to the same object. Naturally, these variables will include a label that indicates of the identity of the object represented by y_k. Figure 4.2 indicates the basic structure of the model. In the rest of this chapter, we will refer to the variables $\{y_k, x_k, h_k\}$ at a single time step as a *slice*. Importantly, the bookkeeping variables and the state variables are hidden (i.e., we do not know object identities and states).

Bookkeeping variables The bookkeeping variables h_k include the following terms $h_k \equiv \{s_k, c_k, z_k^{(1)}, \ldots, z_k^{(k)}\}$. For every slice, we maintain a discrete label s_k that denotes the person represented by y_k. The label takes values in a set $s_k \in \{1, \ldots, k\}$, because within the first k observations there may be at most k different people. (Recall, that a single slice k corresponds to an observation of a single person.) Additionally, the model maintains a set of auxiliary variables: a counter c_k, and pointers $z_k^{(1)}, \ldots, z_k^{(k)}$. The counter, $c_k \in \{1, \ldots, k\}$, indicates the number of persons encountered up to slice k. The pointer $z_k^{(n)}$ denotes the slice when the nth person was *last* observed *before* slice k, so $z_k^{(n)} \in \{0, \ldots, k-1\}$, where "0" indicates that person has not yet been observed.

It is useful to realize that the auxiliary variables deterministically follow from the labels, $c_k = \max_{0<n\leq k}\{s_n\}$, $z_k^{(i)} = \arg\max_{0<n<k}\{s_n | s_n = i\}$. The role of these variables is twofold. First, the auxiliary variables provide an instant reference to the information that is encoded by a the complete sequence $s_{1:k}$, and required by the conditional model (4.2). Second, as we show below, the sequence $h_{1:k}$ can be described with a first-order Markov model (in contrast to a sequence $s_{1:k}$).

Model definition The model definition naturally follows from an analysis of an imaginary process that generates the observations. We assume that the observations are generated one-by-one, with the counter initialized as $c_0 = 0$. Generating y_k starts by setting the label s_k. We choose uniformly at random between one of the known c_{k-1} persons or a new person who will receive the next available label;

$$s_k \sim \text{Uniform}(1, \ldots, c_{k-1}, c_{k-1} + 1). \tag{4.4}$$

Given the label we deterministically update the counter and pointers,

$$c_k = c_{k-1} + [s_k > c_{k-1}] \tag{4.5}$$

$$z_k^{(k)} = 0 \tag{4.6}$$

$$z_k^{(n)} = z_{k-1}^{(n)}[s_{k-1} \neq n] + (k-1)[s_{k-1} = n], \tag{4.7}$$

where $n = 1, \ldots, k-1$ and $[f]$ is an indicator function; $[f] \equiv 1(0)$ iff the binary proposition f is true (false). If the label indicates a new person, $s_k = c_{k-1} + 1$, then the counter increases, as in (4.5). The kth person cannot be observed before slice k, so the pointer to

his/her last observation $z_k^{(k)}$ is set to zero. If the nth person was observed at previous slice, then $z_k^{(n)}$ points to this slice, otherwise $z_k^{(n)}$ does not change, as in (4.7).

Next, we generate the state x_k of the person indicated by s_k. Recall that the state $x_k = \{\mathbf{m}_k, \mathbf{V}_k\}$ represents intrinsic object properties, which by our assumption does not change. Therefore, we set $x_k = x_j$, where $j = z_k^{(s_k)}$ points to the slice when the person was previously observed. If the person has not been yet observed, then we sample the state from prior. Let $j = z_k^{(s_k)}$;

$$x_k = x_j[j > 0] + x^{\text{new}}[j = 0], \tag{4.8}$$

$$x^{\text{new}} \sim \pi(x). \tag{4.9}$$

Given the state $x_k = \{\mathbf{m}_k, \mathbf{V}_k\}$ (i.e. the parameters of a Normal density) and the pointer to the last instance of the current object $j = z_k^{(s_k)}$; the model generates the current observation $y_k = \{o_k, d_k\}$;

$$o_k \sim \mathcal{N}(\mathbf{m}_k, \mathbf{V}_k) \qquad\qquad d_k \sim p_\delta(d_k|d_j), \tag{4.10}$$

where the notation was slightly abused $p_\delta(d_k|d_0) \equiv p_{\delta_0}(d_k)$.

In summary, equations (4.4)–(4.10) define a probabilistic time-series model that binds a sequence of observed variables $y_{1:k}$ with hidden variables. The variables include a sequence of states $x_{1:k}$, a sequence of labels and auxiliary variables $h_{1:k}$. We finalize the description of the model by providing model's graphical representation, which is a basis of further developments.

4.2.4 Graphical representation

Dynamic Bayesian Networks (DBNs) provide a convenient framework for graphical representation of probabilistic time-series models. A DBN represents a model as a directed graph, where the variables become nodes, and conditional probability distributions correspond to directed edges (Murphy, 2002) (see also Chapter 2). Figure 4.2 shows the graph representing our model as a series of slices, where each slice is a column of nodes. The variables $\{h_k, x_k, y_k\}$ at slice k depend on previous slices. The edges incoming to h_k correspond to a conditional density $p(h_k|h_{k-1})$, which is defined by equations (4.4)–(4.7). The edges incoming to x_k correspond to a density $p(x_k|x_{1:k-1}, h_k)$ which is defined by (4.8)–(4.9). Finally, the edges incoming to y_k correspond to a density $p(y_k|y_{1:k-1}, x_k, h_k)$, which follows from (4.10).

4.3 Online tracking

In our probabilistic formulation solving the data association problem corresponds to Bayesian inference on objects' labels. As the Bayes rule (eq. (4.1)) informally indicates

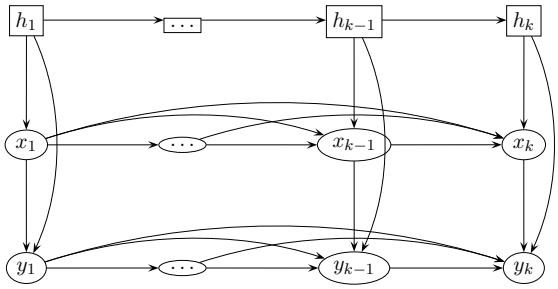


Figure 4.2: A graphical representation of the generative model. The continuous variables are shown as ovals, the discrete – as rectangles.

we are interested in computing posterior density on labels given observations. Depending on the tracking setup, this generic rule can be specialized in a number of ways. In this chapter we consider densities of the form $p(s_k|y_{1:k})$, which correspond to *online* tracking, where we find the label s_k of the observation y_k on the basis of the current data $y_{1:k}$. Alternatively, one might compute densities of the form $p(s_k|y_{1:k+\tau})$ or $p(s_k|y_{1:T})$. These choices correspond to variants of *offline* tracking, where the association of y_k is delayed until τ future observations arrive, or where one first collects a sequence of observations $y_{1:T}$ and later finds individual labels inferred from the *complete* sequence.

4.3.1 Probabilistic filtering

The sought density $p(s_k|y_{1:k})$ can be expressed as a marginal of the joint posterior density $p(x_{1:k}, h_{1:k}|y_{1:k})$ on all hidden variables conditioned on the observations. Therefore, we first apply the Bayes rule to find joint $p(h_{1:k}, x_{1:k}|y_{1:k}) \propto p(y_{1:k}|x_{1:k}, h_{1:k})p(x_{1:k}, h_{1:k})$ (notice the correspondence with the informal rule (4.1)). Next, we integrate every hidden variable except for the label

$$p(s_k|y_{1:k}) \propto \int_{\{x_{1:k}\}} \sum_{\{h_{1:k}\}\setminus s_k} p(y_{1:k}|x_{1:k}, h_{1:k})p(h_{1:k}, x_{1:k}), \tag{4.11}$$

Due to a large number of integrated variables, efficient integration in (4.11) requires a special approach. We apply so called *probabilistic filtering* procedure, which recursively computes marginal probability distribution in time-series models (Boyen and Koller, 1998; Lerner and Parr, 2001). The procedure is also interpreted as a dynamic programming approach for exact solving large-scale integration problems (Murphy, 2002).

At the kth slice the filtering procedure maintains a so called *filtering* density. In the case of our model the filtering density takes the form $p(x_{1:k}, h_k|y_{1:k})$. Given the filtering

density at the preceding slice $k-1$, the filtering density at slice k is estimated in two steps. First, we compute *predictive density*

$$p_{\mathrm{r}}(x_{1:k}, h_k|y_{1:k-1}) = p(x_k|x_{1:k-1}, h_k) \sum_{h_{k-1}} p(h_k|h_{k-1}) p(x_{1:k-1}, h_{k-1}|y_{1:k-1}), \tag{4.12}$$

which represents our knowledge about hidden variables that could be inferred from the data $y_{1:k-1}$. Here, the integration in (4.12) involves hidden variables h_{k-1} from just one slice. Second, the new filtering density is computed by applying Bayes rule

$$p(x_{1:k}, h_k|y_{1:k}) = \frac{1}{L_k} p(y_k|x_k, h_k, y_{1:k-1}) p_{\mathrm{r}}(x_{1:k}, h_k|y_{1:k-1}) \tag{4.13}$$

where L_k is a normalization constant. By integrating $x_{1:k}$ and summing over h_k (except for label s_k) we can find the sought posterior density on labels.

Unfortunately, the exact computation of the filtering density is intractable. One can show that by repeating the summation in (4.12) for consecutive slices k, the resulting predictive density will become a weighted sum of $\mathcal{O}(k!)$ functions. In this way the inevitable complexity of data association shows up in our Bayesian formulation.

4.3.2 Approximate filtering

The intractability of filtering is typical for models with hidden discrete and continuous variables (Lerner and Parr, 2001) and there exist a number of approximation techniques (Murphy, 2002). We follow the assumed-density filtering (ADF) approach for it is suited for online implementations. ADF (Boyen and Koller, 1998; Minka, 2001b) executes the standard filtering steps (4.12)-(4.13) and approximates the filtering density with a simpler function from some assumed parametric family. At every step, the approximating function is chosen by minimizing a "distance" to the filtering density (4.13), expressed as the Kullback-Leibler (KL) divergence.

4.3.3 Algorithm

We approximate the filtering density with a factorial family

$$p(x_{1:k}, h_k|y_{1:k}) \approx q_k(s_k, c_k) \prod_{i=1}^{k} \phi(x_i|\theta_{i,k}) q_k(z_k^{(i)}), \tag{4.14}$$

where q_k denotes a probability table for appropriate discrete variable; and $\phi(x_i|\theta_{i,k})$ represent densities on states with the same family as the prior (4.3).

We initialize the filtering density at slice $k=0$ by setting $q_0(s_0=0, c_0=0)=1$. The general algorithm for recomputing the filtering density is as follows. At slice $k-1$ we

maintain an approximation to the filtering density in the factorial family (4.14). After executing (4.12)-(4.13), the expression (4.13) has to be approximated with the closest function that can be expressed as a product. It is a standard result (Cover and Thomas, 1991), that for a given density $p(a,b)$, the factorial density $q(a,b)$ that minimizes the KL-divergence KL$(q||p)$ is a product $q(a,b) = \sum_b p(a,b) \sum_a p(a,b)$ of marginal densities. Therefore we find the individual terms in (4.14) by computing appropriate marginal densities from (4.13).

The marginal filtering density on the counter and label evaluates to

$$q_k(s_k, c_k) = \frac{1}{L_k} \sum_{j=0}^{k-1} \lambda_j p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)} = j) \tag{4.15}$$

$$\lambda_j = p_\delta(d_k|d_j) \int_x p(o_k|x_k = x) \phi(x_j = x|\theta_{j,k-1}), \tag{4.16}$$

where $\theta_{0,k-1} \equiv \theta_0$. Section 4.7 provides the detailed equations. For $j \geq 1$, the term λ_j is the likelihood of assigning y_k to a trajectory that ends with y_j; and λ_0 is the likelihood of introducing a new person. Integrating the prior density $\phi(x|\theta_0)$ yields an almost uniform, but low likelihood λ_0 for a wide range of o_k. Integrating the density $\phi(x|\theta_{j,k-1})$ yields likelihood λ_j that is peaked around o_j. Thus, despite that the model proposes a new trajectory (state), the filtering density penalizes introduction of a new label unless the appearance does not "match" with any of the past appearances.

The marginal filtering density on the ith auxiliary pointer is given by

$$q_k(z_k^{(i)}) = \frac{1}{L_k} \sum_{s_k=1}^{k} \sum_{c_k=1}^{k} \sum_{j=0}^{k-1} \lambda_j p_{\mathrm{r}}(s_k, c_k, z_k^{(i)}, z_k^{(s_k)} = j). \tag{4.17}$$

In Section 4.7 we provide an efficient way to compute the above summations.

The marginal filtering density on the current state x_k evaluates to a weighted sum (i.e., a mixture) of Normal-InverseWishart functions $\phi(x_k|\theta^j)$

$$p(x_k|y_{1:k}) = \sum_{j=0}^{k-1} w_j \phi(x_k|\theta^j) \tag{4.18}$$

$$w_j = \frac{1}{L_k} \lambda_j \sum_{s_k=1}^{k} \sum_{c_k=1}^{k} p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)} = j),$$

where the parameters $\theta^j = \{\kappa^j, \eta^j, \mathbf{a}^j, \mathbf{C}^j\}$ are obtained from the parameters $\theta_{j,k-1}$ of the

filtering density at previous slice

$$\begin{aligned}\kappa^j &= 1/(1 + 1/\kappa_{j,k-1}) \\ \mathbf{a}^j &= (o_k + \frac{1}{\kappa_{j,k-1}}\mathbf{a}_{j,k-1}) \\ \eta^j &= 1 + \eta_{j,k-1} \\ \mathbf{C}^j &= \mathbf{C}_{j,k-1} + \frac{1}{1+\kappa_{j,k-1}}(o_k - \mathbf{a}_{j,k-1})(o_k - \mathbf{a}_{j,k-1})^\top.\end{aligned}$$

The mixing weights w_j can be efficiently computed together with (4.15). Note, that our assumed family represents density on state x_k as a single function $\phi(x_k|\theta_{k,k})$. We find the parameters $\theta_{k,k}$ by minimizing the KL-divergence to the mixture

$$\theta_{k,k} = \arg\min_\theta \mathrm{KL}\Big(\phi(x_k|\theta)\Big|\Big|\sum_{j=0}^{k-1} w_j \phi(x_k|\theta^j)\Big).$$

The minimizing θ is derived in Section 4.7.3, where the derivation hinges on the fact that Normal-InverseWishart density belongs to so called exponential family.

The marginal filtering density $p(x_j|y_{1:k})$, $j = 1, \ldots, k-1$, evaluates to a mixture of two Normal-InverseWishart functions. We replace this mixture with the closest function in the assumed family $\phi(x_j|\theta_{j,k})$ (see Section 4.7.3)

$$\theta_{j,k} = \arg\min_\theta \mathrm{KL}\Big(\phi(x_j|\theta)\Big|\Big|w_j\phi(x_j|\theta^j) + (1-w_j)\phi(x_j|\theta_{j,k-1})\Big).$$

The function $\phi(x_j|\theta^j)$ is the same as the jth function in (4.18), the term $\phi(x_j|\theta_{j,k-1})$ is just the previous marginal filtering density on the state x_j.

4.3.4 Limiting memory and computational costs

Memory cost Potential limitations of the filtering procedure follow from memory requirements. The assumed filtering family requires $\mathcal{O}(k^2)$ parameters to store marginal density $q_k(c_k, s_k)$, $\mathcal{O}(k^2)$ parameters to represent k marginal densities on pointers $q_k(z_k^{(i)})$, and $\mathcal{O}(k)$ parameters $\theta_{1:k,k}$. Moreover, to evaluate (4.16) we need to preserve $\mathcal{O}(k)$ spatio-temporal features $d_{1:k}$.

We partially overcome the memory-size problems by setting a limit K on the number of tracked persons. Appropriately chosen K will not seriously affect the tracker, since typically the number of persons is much lower than the number of observations. In this way, the density $q_k(c_k, s_k)$ requires only $\mathcal{O}(K^2)$ parameters, and there are only K marginal densities $q_k(z_k^{(i)})$.

To further restrict the memory demand we set a limit M on the number of parameters θ and features d preserved in memory. After processing observation k, we store $\{d_k, \theta_{k,k}\}$, and if $k > M$ we prune $\{d_\nu, \theta_{\nu,k}\}$, where

$$\nu = \arg\min_{1 \le j < k} \sum_{c_k=1}^{K} \sum_{s_k=1}^{K} p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)} = j). \tag{4.19}$$

Intuitively, this criterion selects a slice ν that is the least likely to be an end-point of any trajectory. More formally, the sum (4.19) measures the total weight attributed to the term λ_ν in (4.15) where the marginal filtering density on labels is computed. After pruning $\{d_\nu, \theta_{\nu,k}\}$ we cannot compute λ_ν, thus λ_ν has to be weighted by zero during future filtering steps. For this purpose, we set $q_k(z_k^{(n)} = \nu) = 0$ and re-normalize, for all $n = 1, \ldots, K$. In this way, there are at most M non-zero entries in the table of $q_k(z_k^{(n)})$ and representing these densities for $n = 1, \ldots, K$ requires a constant $\mathcal{O}(KM)$ memory.

Computational cost The computational cost of the filtering procedure is the sum of costs necessary to update each factor in the assumed filtering density. Updating the term $q(x_k)$ costs $\mathcal{O}(Mr^3)$, where r is the dimensionality of appearance features. At slice k, there are at most M densities $q(x_i)$, $i < k$ which cost $\mathcal{O}(Mr^3)$ to update. (The r^3 component arises during inversion of $r \times r$ matrices in the "moment matching" for Normal-Inverse Wishart densities; see Appendix 4.7.) Updating the term $q(s_k, c_k)$ and all $p(z_k^{(i)})$ costs $\mathcal{O}(K^3)$. Therefore the total cost is $\mathcal{O}(Mr^3 + K^3)$, that is linear in the "memory size" parameter M, and cubic in the maximal number of tracked persons.

4.4 Experiments

In this section a series of experiments demonstrates the algorithm applied to tracking people in an office building. The first experiment compares the tracking quality our approach with methods based on MCMC and MHT approximations. The second test involves tracking in difficult environments, where either appearance or spatio-temporal measurements are very weak. These experiments are based on semi-manual data preprocessing, analogously to the previous chapter. The final experiment demonstrates a fully automated tracking system.

Data set Our data set includes 70 observations representing 5 persons who were observed in an office building with 7 disjoint scenes. Figure 4.3(a) shows a map of camera locations and their fields of view. Observations in this set are extracted from video streams manually in order to construct a basis for comparison that is free from artifacts introduced by automated video preprocessing. From a video fragment with a pass of a person we selected a single frame. Next, color values of pixels within the person's

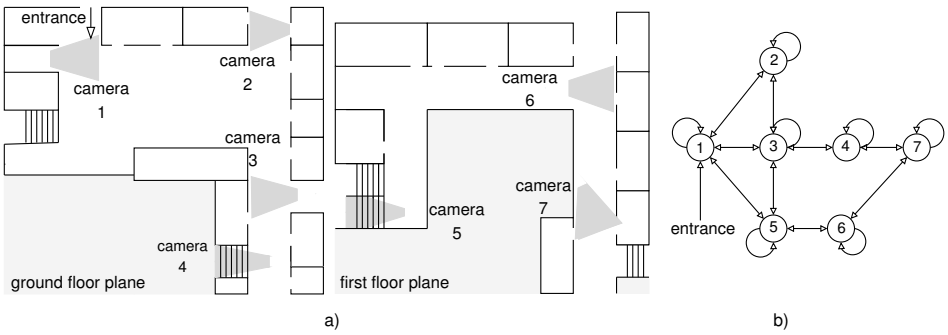


Figure 4.3: (a) Building plan were the observations were taken. The gray areas show camera viewing fields (scenes). (b) A graph showing movement constraints assumed by distributions p_δ and p_{δ_0}. The numbered nodes indicate scenes, the edges indicate allowed scene-to-scene transitions.

silhouette were transformed into a channel-normalized space to suppress the effects introduced by the color of the illuminating light (Drew et al., 1998). The appearance features o_i have been defined as a 9-dimensional vector of color statistics, analogously as described in Chapter 3 (see Fig. 3.7 for details). Unlike histograms, such appearance vectors provide a compact summary of color content and its geometrical layout.

The parameters $\theta_0 = \{\mathbf{a}_0, \kappa_0, \eta_0, \mathbf{C}_0\}$ of the prior (4.3) density were: $\mathbf{a}_0 = \mathbf{0}_{9\times 1}$ (9-dimensional zero vector); $\kappa_0 = 100$; the degrees of freedom $\eta_0 = 9$ (data vector size), $\mathbf{C}_0 = 10^{-3}\mathbf{I}_{9\times 9}$, where $\mathbf{I}_{9\times 9}$ is a 9×9 identity matrix. Parameter $\mathbf{C}_0$ defines prior preferring covariances $\mathbf{V}$ with relatively small eigenvalues. To assure that the prior does not significantly prefer any mean $\mathbf{m}$ we set κ_0 to a large value.

As already mentioned, the densities p_δ, p_{δ_0} for spatio-temporal features reflect motion constraints. We set $p_\delta(d_k|d_i) = p(\ell_k, b_k^{\mathrm{e}}|\ell_i, b_i^{\mathrm{q}})p(t_k^{\mathrm{e}}|t_i^{\mathrm{q}}, \ell_k, \ell_i)$, where the first factor gives the probability of entering scene ℓ_k via border b_k^{e} after quitting ℓ_i via border b_i^{q}. Given the prior knowledge about the building, we set the allowed transitions as indicated in Fig. 4.3(b). The second factor gives the probability of travel time between two scenes. We used a simple density that only prevents zero or negative travel times. The density $p_{\delta_0}(d_k) = p(\ell_k, b_k^{\mathrm{e}})$ gives the likelihood of first appearance in at scene ℓ_k and border b_k^{e}. It was only possible via left border of scene "1".

Experiment 1: Quantitative evaluation In order to quantitatively compare various methods we apply the evaluation criteria defined in the previous chapter. These criteria follow from the view of multi-object tracking as an unsupervised classification problem, where observations of the same object have to be assigned to a single cluster (i.e., a trajectory). The estimated trajectories are evaluated against the ground-truth trajectories. The first criterion (*accuracy*) indicates the percentage of observations in a

reconstructed trajectory that indeed correspond to a single person. The second criterion (*recall*) indicates the percentage of observations of a single person recovered in a single reconstructed trajectory. See equations (3.6)–(3.7) of Chapter 3 for formal definitions. Additionally, we report the number of reconstructed trajectories.

We compare the presented ADF-based tracker with three alternative methods for multi-object tracking. The first compared tracker (Pasula et al., 1999) defines a probabilistic model of trajectories and relies on similar assumptions as our model (Gaussian noise, first-order Markov motion). The method estimates the association of observations with trajectories using an MCMC sampling algorithm, where a single sample defines trajectories for all objects. Originally, this method does not find the most likely association, but some averaged traffic parameters. In order to compare with our method, we used the trajectories defined by the most likely sample. The second compared method is an MHT association algorithm applied to our probabilistic model. In this setup, a hypothesis defines a sequence of labels. Computing the hypothesis likelihood is a tractable problem since, given the labels, the models for trajectories are independent. The third compared method is the algorithm presented in Chapter 3, indicated as MHT-EM. Note, that the comparison with this algorithm is not fully fair, since the methods of chapter 3 tries to learn expected travel-time parameters (with EM procedure). The other methods assume known and fixed minimum travel times.

Table 4.1 summarizes the evaluation results of the compared algorithms. The experiment evaluated tracking quality as a function of: (i) memory depth M for our model and the ADF approximation, (ii) hypotheses buffer size H for the MHT and (iii) sample size S for the MCMC. The evaluation scores for the MCMC are indicated as means and standard deviations obtained from ten runs. The reported run-times correspond to Matlab implementations executed on a 1 GHz PC. Figure 4.4 demonstrates a sample run of the ADF-based tracker where $M = 10$ and $K = 10$. Columns presents subsequent distributions on label $q_k(s_k)$ (top) and counter $q_k(c_k)$ (bottom), with probability given in gray scale. An example of an estimated trajectory is shown in Fig. 4.5.

We observe that for sufficient memory depth M, our model together with ADF algorithm found the correct number of persons and returned nearly exact trajectories. The accuracy and recall of MCMC and MHT algorithms is much lower, although, these methods improve with the increasing memory and time resources. Importantly, MCMC sampling always indicated 6 or 7, and MHT 8 distinct persons. On the other hand, clearly MHT is the fastest algorithm. The method of Chapter 3 (indicated as MHT-EM) cannot compete with the other algorithms (neither in terms of speed nor in terms of accuracy). This property follows from the fact that the MHT-EM algorithm does not assume known travel-times parameters and tries to estimate these from the data. Alternative algorithms assume known minimum travel times.

Experiment 2: Difficult environments This series of experiments evaluates various tracking methods in difficult environments where either appearance features or spatio-temporal features provide weak cues for data association. We consider two scenarios.

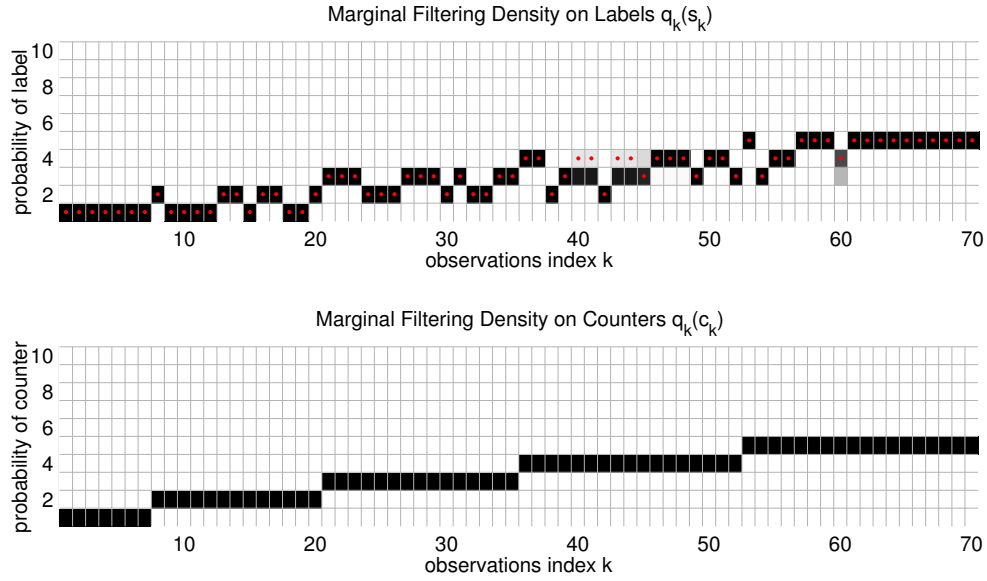


Figure 4.4: Filtering densities during tracking with our model, with $K = 10, M = 10$. Columns correspond to filtering steps; $k = 1, ...70$. (Top) Each column shows the filtering distribution on label; $q_k(s_k)$. The true labels are indicated as dots. (Bottom) Each column shows the filtering distribution on counter; $q_k(c_k)$.

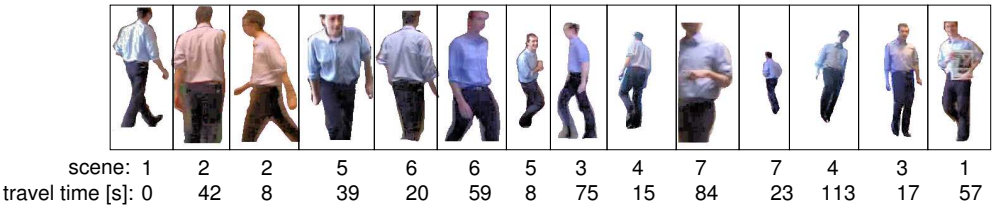


Figure 4.5: An example of a recovered trajectory. The images represent appearances at the indicated scenes. The travel times correspond to differences between times of visiting subsequent scenes. The trajectory includes such observations y_i that have the estimated labels $s_i = 5$ (see the top panel of Fig. 4.4).

Table 4.1: Performance summary for the compared of methods. MHT-EM denotes the method of Chapter 3.

parameter	ADF			
M	accuracy [%]	recall [%]	objects	time [s]
05	65.5	84.3	5	07
10	95.6	94.3	5	13
20	95.6	94.3	5	22
30	95.6	94.3	5	30
40	95.6	94.3	5	37

parameter	MHT			
H	accuracy [%]	recall [%]	objects	time [s]
05	74.4	55.7	8	02
10	69.7	52.9	8	04
20	77.3	58.6	8	08
30	86.1	70.0	8	11
40	85.1	70.0	8	15

parameter	MHT-EM (chapter 3)			
H	accuracy [%]	recall [%]	objects	time [s]
1	48	55	5	54
5	77	70	5	258
10	78	79	5	515
20	76	79	5	1036

	MCMC			
S	accuracy [%]	recall [%]	objects	time [s]
10^2	74.4 ± 7.8	68.7 ± 7.8	6.6 ± 0.7	26 ± 0.1
10^3	79.0 ± 10.2	79.0 ± 5.6	6.4 ± 0.7	209 ± 1.2
10^4	89.7 ± 4.5	84.7 ± 6.1	6.8 ± 0.8	2045 ± 13.2

In the first scenario we simulate weak or unknown motion constraints, which undermine the discriminative power of the spatio-temporal features. Recall, that the camera-to-camera motion is represented by spatio-temporal features d_n, and the motion constraints are embedded in the probabilistic model for a sequence of such features attributed to some person. The model takes the form of a first-order Markov chain, and includes a prior distribution $p_0(d_1)$ and a set of conditional distributions $p_\delta(d_n|d_{n-1})$, where d_n and d_{n-1} are two subsequent features associated with the same person. Appro-

Table 4.2: Tracking in environments with varying difficulty level (indicted by the entropy value, see text for explanation). The parameters were set as follows: $M = 10$ (for ADF), $H = 20$ (for MHT), $S = 10^4$ (for MCMC).

	accuracy [%]			recall [%]			objects		
entropy	ADF	MHT	MCMC	ADF	MHT	MCMC	ADF	MHT	MCMC
1.17^a	96	77	90	94	59	85	5	8	7
1.21^a	65	81	90	83	33	84	4	18	7
1.43^a	72	83	91	83	30	78	5	21	8
2.23^a	66	79	90	79	24	60	4	21	13
2.77^b	61	71	77	60	20	46	5	24	13
1.17^c	59	71	70	63	68	70	5	5	6

[a] Tracking based on spatio-temporal and appearance features.

[b] Tracking based exclusively on appearance features. The entropy value corresponds to uniform distributions p_δ and p_0.

[c] Tracking based exclusively on spatio-temporal features.

priate low-entropy (or "peaked") distributions will decrease the likelihood of camera-to-camera paths that violate motion constraints specific to some environment. High-entropy (or "flat") distributions effectively remove motion constraints since neither hypothetical path will be penalized significantly. In the limit, uniform distributions simulate tracking based exclusively on appearance features. As a consequence, various motion constraints can be simulated by various distributions p_0 and p_δ, and the "strength" of the constraints can be measured in terms of average entropy of these distributions.

In the second scenario we consider tracking based exclusively on spatio-temporal features without using the appearance features. This case simulates environments with strong or multimodal appearance noise (e.g. due to inaccurate video preprocessing).

Table 4.2 summarizes qualitative results of the experiment. Figure 4.6 presents the density on labels obtained when filtering used only: (top) the appearance features, and (bottom) the spatio-temporal features. We observe (three rightmost columns of Tab. 4.2) that the ADF-based tracker is much better at estimating the correct number of objects than the alternative approaches. This property is reflected by high recall scores. The other approaches achieve higher accuracy scores at the cost of introducing superfluous trajectories.

Experiment 3: Kernel parameters Below we compare the Gaussian kernels inferred from the appearance features in two cases; (i) when the association was based exclusively on appearance features, and (ii) when the association was based on appearance and spatio-temporal features.

After processing 70 observations our algorithm maintains $M = 10$ Gaussian kernels representing the latent color features (M is the assumed practical limit; in principle we

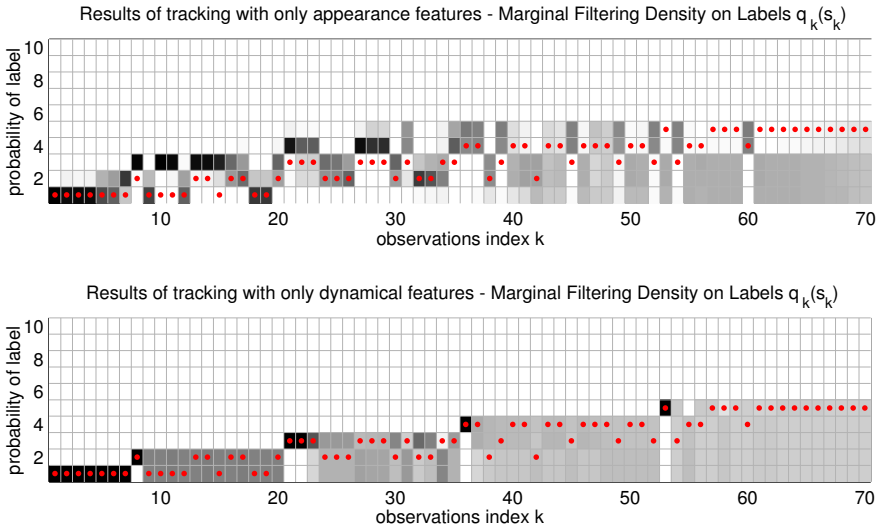


Figure 4.6: (Top) Tracking based *only* on the appearance features. (Bottom) Tracking based *only* on the spatio-temporal features. In both cases, columns show the density $q_k(s_k)$, for $k = 1, ...70$ obtained by probabilistic filtering in our model. The true labels are indicated as dots.

should have 70 kernels). Every kernel x_j, $j = 1, \ldots, M$, is represented by a posterior distribution $\phi(x_j|\theta_{j,70})$ conditioned on all 70 observations, because filtering updates distributions on all kernels in memory. Given the parameters $\theta_{j,70}$ we find the expected mean $\mathrm{E}[\mathbf{m}_j]$ and the expected covariance $\mathrm{E}[\mathbf{V}_j]$ of the jth kernel; $\mathrm{E}[\mathbf{m}_j] = \mathbf{a}_{j,70}$ and $\mathrm{E}[\mathbf{V}_j] = \mathbf{C}_{j,70}/(\eta_{j,70} - d - 1)$. The kernels and the appearance features have $d = 9$ dimensions. For visualization, we find a 2-dimensional PCA projection of the original data and apply it to the expected kernels. Figure 4.7 presents the projections of the observed appearance features (as points) and the expected kernels (as ellipses). Note, that both plots present 10 kernels; some kernels overlap in the figure.

Figure 4.7 shows that the appearance features alone do not distinguish between the persons. In the right panel of the figure, we observe that the ten kernels inferred using only appearance similarities form only two clusters. This corresponds to the top plot in Fig. 4.6, where the estimated labels of observations 45–70 fall into two groups. In the left panel of Fig. 4.7 we see that when the spatio-temporal and appearance features supported tracking, then the estimated kernels closely match to distributions of appearance features for each of the tracked persons.

Experiment 4: Automated tracking demonstration Below we present a wide-area tracking experiment with automated detection of objects and feature extraction. Our object detection relies on static background scene at each camera (Zivkovic, 2004). To

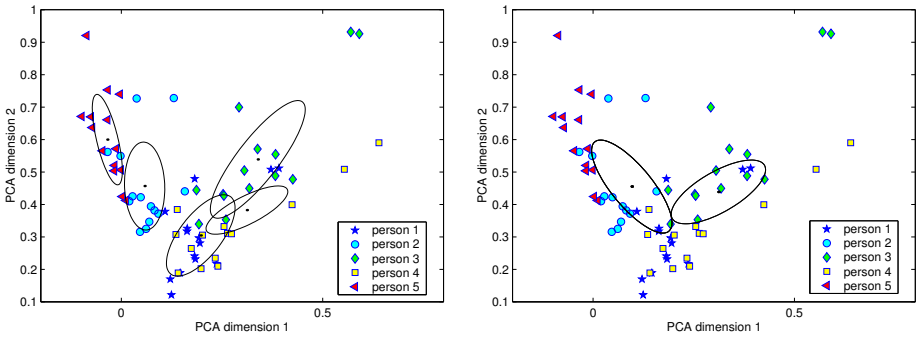


Figure 4.7: The expected kernels (expected mean and expected covariance) shown as a 2D PCA projection of the original 9D feature space. (Left) The kernels obtained when tracking was based on spatio-temporal and appearance features. (Right) The kernels obtained when tracking was based only on appearance features.

eliminate false detections (e.g. resulting from shadows) we limit the object's minimal valid bounding box to 30×30 pixels (in 240×320 frames). We also eliminate spurious appearances that are shorter than 0.5 s (at 30 fps).

In the experiment we track five persons observed at three disjoint scenes. An observation is reported after a person has been visible for 2 s in a scene. The appearance vectors and the prior state density are the same as previously. The model of motion constrains allows a trajectory to start at any scene; and allows direct movements between all scenes. The only constraint is that a person cannot be observed at different scenes at the same time.

Figure 4.8 shows selected frames from the three simultaneous video streams. Bounding boxes indicate detected objects, and numerical labels — estimated identities. Through the sequence, the person indicated as "1" is clearly distinguished from the others, probably due to distinctive appearance. The persons "2" and "4" are difficult to separate from each other, and their identities were confused. The overall accuracy was 91%, recall 80%, and the number of persons was correctly estimated, despite relatively weak motion constraints.

4.5 Discussion

The developed approach for wide-area tracking bears relationship with other multi-object tracking methods and many machine learning techniques. In this section we discuss these relationships, present ideas to improve our tracker and point other tracking scenarios where the technique is potentially applicable.

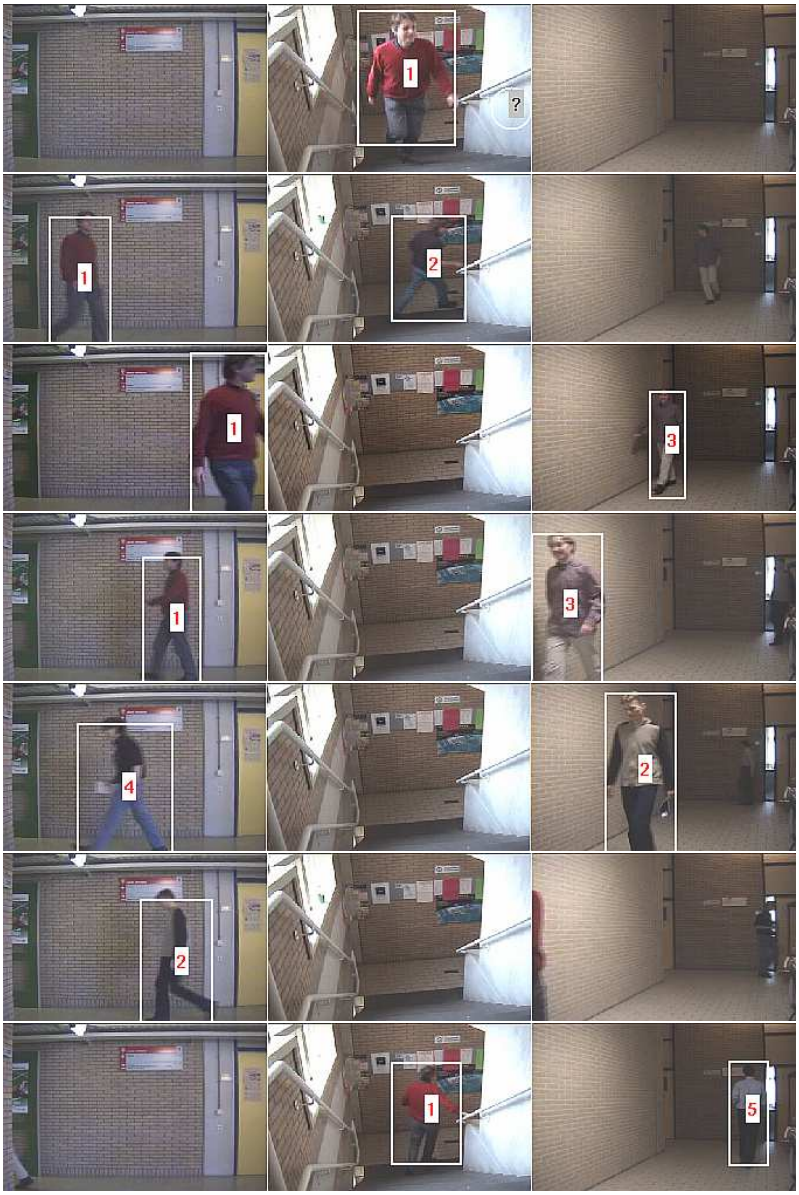


Figure 4.8: Frames selected from simultaneous video streams of three cameras involved in the experiment. The length of each stream was 2000 frames (at 30 fps). Detected object are shown with bounding boxes. The numerical labels indicate estimated associations.

Our multi-object tracking technique resembles joint probabilistic data association filters (JPDAFs) (Bar-Shalom and Li, 1993). For example, JPDAFs are applied for multiple aircraft tracking based on radar readings, which are considered as noisy measurements of a hidden aircraft position. Similarly to our algorithm, JPDAFs deal with association ambiguity by approximating densities of states with an assumed parametric family (usually Gaussian). Typically, JPDAFs rely on smooth motion and dense measurements of the tracked objects. In contrast to our method, JPDAFs estimate the number of trajectories using application-specific heuristics (e.g., by detecting motion discontinuities).

One explanation to the success of JPDAF and ADF algorithms for multi-object tracking follows from a theoretical error-bound property. Approximating the filtering density with a product of simpler marginal densities introduces an error at every time step. In time-series models with exclusively discrete variables this error is bounded (Boyen and Koller, 1998). Although such a result is not available for models with continuous variables, in our experiments ADF outperforms the heuristic MHT algorithm.

To a certain extent, our model resembles a Dirichlet process (DP) mixture model (Antoniak, 1974). In machine learning applications, DP models are applied for estimation of mixture distributions with an unknown number of components. For every data vector, a DP model introduces a new mixture component that is either sampled from a prior or equal to one of the other components. Our model, for every observation introduces a new state that is either sampled from the prior $\pi(x)$ or equal to some past state.

The relationship with Dirichlet processes becomes important when we realize that a difficult aspect of wide-area tracking is estimation of the number objects. In machine learning, this task resembles problems, where one has to select the number of model parameters necessary to explain the data. Recently, Dirichlet processes have been applied for estimation of *infinite* models (Beal et al., 2002; Rasmussen, 2000), i.e., models that do not constrain the number of parameters. Overfitting in such models is avoided due to Bayesian treatment of parameters (in our case: states), which are considered as random variables and integrated.

Analogously to many tracking or machine learning problems, we assumed Gaussian noise in the measured quantities. We considered mean and noise covariance jointly as a hidden state, represented with a Normal-InverseWishart density. In many applications the covariance is not a part of the state, but rather a parameter. Consequently, one needs training data to estimate this parameter, e.g., according to the maximum-likelihood criterion with EM algorithm as in (Pasula et al., 1999; Zajdel and Kröse, 2003b). Incorporating the covariance as a part of the state allows us to estimate it by Bayesian inference and avoids "training" of the model.

A potential extension of our algorithm is to delay estimation the label of the current observation until a limited number of future observations become available. This extension fits well to our probabilistic framework since ADF is an instance of a more general inference scheme – Expectation Propagation(EP) (Minka, 2001b). EP iteratively improves approximation of filtering densities by incorporating, so called, smoothing

densities. EP has been already shown successful for fixed-lag smoothing in time-series models (Qi and Minka, 2003) and infinite mixture models (Minka and Ghahramani, 2003). Finally, the developed technique can be also applied to different problems involving data association, like tracking multiple persons from a mobile robot equipped with a single camera (Zajdel et al., 2005b).

4.6 Conclusions

We developed a technique for tracking multiple objects with sparsely distributed cameras, where the primary problem is the association of observations with trajectories when objects enter and leave disjoint fields of view. Our technique relies on a probabilistic model that assumes an unique, but hidden label for every tracked object. The model defines a probabilistic relation between the observations of objects an their labels. Given such a model and a sequence of observations, we approximate the posterior probability densities of labels, and find the most likely label for each observed object.

Our algorithm applies a principled and unified framework – Bayesian inference – for approximating solution to several intractable problems arising in multi-object tracking. First, the Bayesian assumed-density filtering (ADF) algorithm solves data association by approximating probability density on hidden object labels. Second, the algorithm estimates the appropriate number of trajectories by combining ideas inspired by machine learning models (Dirichlet processes) with Bayesian inference. Finally, we have shown how the parameters – mean and covariance – of popular Gaussian noise models can be estimated by Bayesian inference under data association ambiguity.

We have presented real-world experiments, where the deterministic ADF approximation performed superior to the multiple-hypothesis tracker (MHT) and a tracker based on stochastic inference (MCMC). The developed technique offers a practical trade-off between the speed of MHT-based methods and the potential accuracy of trackers employing stochastic approximations.

4.7 Appendix

This section provides derivations of the marginal filtering densities from the joint filtering density (4.13). We use the following abbreviations: $z_k^{(1:k)} \equiv \{z_k^{(1)}, \ldots, z_k^{(k)}\}$; $z_k^{(\neg i)} \equiv z_k^{(1:k)} \setminus z_k^{(i)}$; and $p_{\mathrm{r}}(x_{1:k}, h_k) \equiv p(x_{1:k}, h_k | y_{1:k-1})$.

4.7.1 Computing marginals of the filtering density

We begin with the marginal on label and counter $q_k(s_k, c_k)$. From (4.13):

$$q_k(s_k, c_k) = \frac{1}{L_k} \sum_{z_k^{(1:k)}} \int_{x_{1:k}} p(y_k|h_k, x_k, y_{1:k-1}) p_{\mathrm{r}}(x_{1:k}, h_k)$$

The likelihood simplifies as $p(y_k|h_k, x_k, y_{1:k-1}) = p_\delta(d_k|d_j)p(o_k|x_k)$ after (4.10), where $j = z_k^{(s_k)}$. Therefore, the multiple integration reduces immediately to

$$q_k(s_k, c_k) = \frac{1}{L_k} \sum_{j=0}^{k-1} p_\delta(d_k|d_j) \int_{x_k} p(y_k|x_k) p_{\mathrm{r}}(x_k, s_k, c_k, z_k^{(s_k)} = j).$$

Using (4.8)–(4.9) we can simplify the predictive density

$$p_{\mathrm{r}}(x_k, s_k, c_k, z_k^{(s_k)} = j) = \phi(x_k|\theta_{j,k-1}) p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)} = j),$$

where $\theta_{0,k-1} \equiv \theta_0$. When $j = 0$ then the state is sampled from $\pi(x_k) = \phi(x_k|\theta_0)$, otherwise $x_k = x_j$, where x_j is a past state distributed as $q_{k-1}(x_j) = \phi(x_j|\theta_{j,k-1})$. With these simplifications, we have

$$q_k(s_k, c_k) = \frac{1}{L_k} \sum_{j=0}^{k-1} \lambda_j p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)} = j)$$

$$\lambda_j = p_\delta(d_k|d_j) \int_{x_k} p(o_k|x_k) \phi(x_k|\theta_{j,k-1})$$

The Normal pdf $p(o_k|x_k) = \mathcal{N}(o_k|\mathbf{m}_k, \mathbf{V}_k)$ is conjugate to the Normal-Inverse Wishart $\phi(x_k|\theta_{j,k-1})$, and the above integral evaluates to a standard result (Gelman et al., 1995):

$$\lambda_j = p_\delta(d_k|d_j) \mathcal{T}(o_k|\mathbf{a}_{j,k-1}, (1 + \kappa_{j,k-1})\mathbf{C}_{j,k-1}, 1 + \eta_{j,k-1}),$$

where $\mathcal{T}$ denotes a multivariate T-distribution. Finally, we recover L_k by finding an unnormalized distribution and then applying the normalization constraint. The predictive density $p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)})$ is discussed in Section 4.7.2.

Below we find marginals on auxiliary pointers $q_k(z_k^{(i)})$, only for $i = 1, \ldots, k-1$, because the pointer $z_k^{(k)}$ is deterministic (see (4.6)). Begin with (4.13)

$$q_k(z_k^{(i)}) = \frac{1}{L_k} \sum_{s_k, c_k} \sum_{z_k^{(\neg i)}} \int_{x_{1:k}} p(y_k|h_k, x_k) p_{\mathrm{r}}(x_{1:k}, h_k).$$

As in the previous case, all pointers other than $z_k^{(i)}$ and $z_k^{(s_k)}$ marginalize to unity. For a fixed $z_k^{(s_k)} = j$, the integral over x_k evaluates to λ_j, and all other states integrate to

one. We split the summation over s_k into two cases: (1) $s_k = i$; (2) $s_k \neq i$. The term n, $0 \leq n \leq k-1$, enumerates the domain of $z_k^{(i)}$.

$$q_k(z_k^{(i)} = n) = \frac{1}{L_k}\lambda_n \sum_{c_k} p_{\mathrm{r}}(s_k = i, c_k, z_k^{(s_k)} = n) + \beta_i(n),$$

where $\beta_i(n)$ denotes the sum over $s_k \neq i$:

$$\beta_i(n) = \frac{1}{L_k} \sum_{s_k \neq i, c_k} \sum_{j=0}^{k-1} \lambda_j p_{\mathrm{r}}(s_k, c_k, z_k^{(i)} = n, z_k^{(s_k)} = j).$$

We first calculate a special case $\beta_i(k-1)$:

$$\beta_i(k-1) = \frac{1}{L_k} \sum_{s_k \neq i, c_k} \sum_{j=0}^{k-1} \lambda_j p_{\mathrm{r}}(s_k, c_k, z_k^{(i)} = k-1, z_k^{(s_k)} = j),$$

where the predictive density is very simple (see 4.7.2). Since the normalization term L_k is already available from the previous marginal, we exploit the normalization constraint to efficiently find $\beta_i(n)$, $0 \leq n \leq k-2$. We obtain the following relation

$$\beta_i(n) = q_{k-1}(z_{k-1}^{(i)} = n)\Big(1 - \sum_{c_k} q_k(s_k = i, c_k) - \beta_i(k-1)\Big),$$

The second factor does not depend on n, hence we compute it only once.

4.7.2 Predictive density

We have shown, that our filtering algorithm requires only two marginals from the predictive density: $p_{\mathrm{r}}(s_k, c_k, z_k^{(s_k)})$ and $p_{\mathrm{r}}(s_k, c_k, z_k^{(i)} = k-1, z_k^{(s_k)})$. These quantities are straightforward to compute because the discrete variables evolve as a Hidden Markov Model (see (4.4)–(4.7)). Moreover, the computation is efficient since $z_k^{(i)}$, c_k are deterministic given s_k and the previous-slice variables.

4.7.3 Moment matching in Normal Inverse-Wishart family

The Normal-Inverse Wishart density is a member of exponential family of densities. Therefore we can conveniently minimize $\mathrm{KL}\Big(\phi(x|\theta)\Big|\Big|\sum_{i=1}^{N} w_i\phi(x|\theta_i)\Big)$ w.r.t θ by setting the moments of $\phi(x|\theta)$ equal to those of the mixture (Cover and Thomas, 1991) and

solving for $\theta = \{\kappa, \eta, \mathbf{a}, \mathbf{C}\}$ (Gelman et al., 1995)

$$\begin{aligned}
\mathbf{a} &= \left(\sum_{i=1}^{N} w_i \eta_i \mathbf{a}_i \mathbf{C}_i^{-1}\right)\left(\sum_{i=1}^{N} w_i \eta_i \mathbf{C}_i^{-1}\right)^{-1}, \\
\kappa &= \sum_{i=1}^{N} w_i \kappa_i, \\
\mathbf{C} &= \eta \sum_{i=1}^{N} w_i \eta_i \mathbf{C}_i^{-1}.
\end{aligned}$$

The analytical solution for η is difficult, therefore we use η from the most likely mixture component: $\eta \approx \eta_n$, where $n = \mathrm{argmax}_i w_i$.

CHAPTER 5

A MODEL OF SPATIAL PIXEL CORRELATIONS FOR BACKGROUND SEGMENTATION

This chapter is focused on the problem of assigning to every pixel of an image a binary label that indicates whether the pixel represents a static background or a foreground object in motion.[1] Popular approaches for this problem consider the labels as random variables with an appropriate prior distribution, which ensures that the labeled pixels form spatially coherent regions. Typically, the prior takes the form of a Markov Random Field (MRF) model that, due to difficulties with learning, assumes only generic short-range coupling of pixel labels. This chapter presents an alternative model that takes spatial correlations into account in a more flexible way. The model can be easily learned from exemplary data to incorporate correlations characteristic for foreground objects in a given scene (e.g. human silhouettes). For inference in the model we derive an approximate, but computationally efficient Expectation-Propagation algorithm.

5.1 Introduction

Detecting moving objects in static visual scenes has emerged as an important preprocessing step for a variety of computer-vision applications, like tracking (Javed et al., 2003; Toyama and Blake, 2001; Zajdel et al., 2004), surveillance (Haritaoglu et al., 2000; Zhao and Nevita, 2004) or vision-based control (Wren et al., 1997). In this problem, known as *background segmentation*, one aims to assign pixels of video frames either to a background segment that represents the static scene or to a foreground segment that represents the objects in motion relative to the scene.

Background segmentation relies on two types of cues: temporal cues following from the assumption of a static background scene and spatial cues following from an additional assumption that in typical scenes the pixels assigned to the foreground segment form spatially coherent groups.

[1] The material of this chapter has been submitted for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Zajdel et al., Submitted, 2005b).

Intuitively, the assumption of a static background scene allows to detect individual foreground pixels as short-term deviations from a long-term stationary image that represents the scene. More formally, probabilistic approaches (Friedman and Russell, 1997; Kato et al., 2002; Stauffer and Grimson, 1999) attempt to estimate a probability density for the background color of every pixel and detect foreground pixels as outliers from this density. In practice (see Fig. 5.1) segmentation based on temporal cues only is prone to errors due to factors like: variations in the background scene (e.g. motion of trees due to wind), illumination changes, camera jitter or cases when the appearance of a foreground object coincides with the background.

A common approach for introducing spatial cues is by "postprocessing" the pixels detected earlier as outliers from the background density. The postprocessing can either enforce only local pixel connectivity (e.g., morphological operators (Stauffer and Grimson, 1999)) or/and apply specialized models depending on the objects of interest (e.g., humans). The models take the form of elliptical blobs (Wren et al., 1997; Zhao and Nevita, 2004), splines (Blake et al., 1995) templates (Gavrila and Giebel, 2001; Lim and Kriegman, 2004).

Alternatively, one can consider the spatial and temporal cues jointly. Here, probabilistic formulations assume a prior distribution for class of each pixel and define such priors that prefer segmentations with correlated pixels. A natural choice is a Markov Random Field (MRF), where a positive coupling is assumed between pixels within an explicitly defined neighborhood (Freeman et al., 2000; Rittscher et al., 2000; Wang et al., 2002). Other models found in the literature include epitomes (Jojic et al., 2003) or fixed-basis transform (Sullivan et al., 2001).

An important limitation of the existing approaches for spatial correlations is the inability to automatically learn mid- and long-range correlations typical for a given scene and a camera angle. On one hand, approaches that construct custom 2D object models (e.g., spatially arranged collections of blobs or ribbons) are not immediately applicable to arbitrary class of objects. On the other hand, typical applications of MRFs (and morphological operators) consider only generic short-range coupling within a pixel neighborhood. In case of long-range coupling, inference and especially learning in MRFs tend to be difficult (Geman and Geman, 1984).

In this chapter we describe a probabilistic model that facilitates easy learning of spatial correlations between pixels at an arbitrary range. The learning procedure relies exclusively on a set of exemplary images that exhibit correlations characteristic for some scene, e.g. corresponding to human silhouettes. In this way the model collects information about typical object shapes. Such specific information allows handling difficult segmentation problems, such as cases where the appearance of a foreground object partially coincides with the background scene. (See Fig. 5.1 for an example scene, where the person's torso coincides with the brick pattern of the background wall.)

The idea underlying our approach is to assume a set of fictitious continuous variables which can be easily correlated by imposing a Gaussian distribution. By quantizing the

Figure 5.1: (Left) A frame from video sequence that presents a static scene and a person in motion. (Right) Results of segmentation based only on temporal cues for individual pixels. Note incorrect segmentation of image regions (torso) where the color of the background coincides with the object.

correlated continuous variables we obtain a set of correlated binary foreground/background pixel labels. Similar models have been proposed for unsupervised learning of static patterns (Lee and Sompolinsky, 1998), visualization (Tipping, 1998), and for classification (Williams and Barber, 1998). Such an approach allows us to easily learn correlations of the continuous variables by estimating a Gaussian distribution for these variables, e.g. with the EM algorithm (Neal and Hinton, 1998). For inference we derive an expectation-propagation (Minka, 2001b) algorithm that considers the continuous variables as hidden random variables. The algorithm works in linear time in the number of pixels, regardless of the correlation structure between pixels.

The chapter is structured as follows. In Section 5.2 we briefly review the probabilistic framework for modeling pixel correlations. In Section 5.3 we define our model, and describe the learning and inference algorithms. In Section 5.4 we experimentally validate the proposed model and compare it with MRF models. The experiments demonstrate learning, segmentation and a simple tracking application based on the model. Section 5.5 concludes the chapter.

5.2 Probabilistic modeling of pixel correlations

In this section we review the idea of probabilistic models for reasoning about pixel correlations in segmentation problems. The concept is illustrated with an MRF model.

For every pixel we define a binary indicator $r_k \in \{-1, 1\}$, where $k = (i, j)$ with i and j corresponding to vertical and horizontal spatial indices. The pixels with $r_k = 1$ will belong to the foreground, and the pixels with $r_k = -1$ will belong to the background. To simplify the notation we treat k as a linear index and let $k = 1 \dots K$ where K is the number of pixels in an image. We use a term *mask* $\mathbf{r} = r_{1:K}$ to denote a collection of indicators.

5.2.1 Probabilistic framework

In a probabilistic formulation we consider the sought segmentation of an image (i.e. a sought mask) as a hidden random vector $\mathbf{r}$. Our aim is to estimate this mask given an observed image $\mathbf{y}$. The estimation is based on a joint probability distribution $p(\mathbf{r}, \mathbf{y}) = p(\mathbf{r})p(\mathbf{y}|\mathbf{r})$, where the *prior model* $p(\mathbf{r})$ assigns high probabilities to such masks $\mathbf{r}$ that exhibit desired spatial correlations between pixels, and *observation model* $p(\mathbf{y}|\mathbf{r})$ defines a probabilistic dependency between the measured and the hidden vectors.

The process of estimating the hidden mask corresponds to probabilistic inference. Given an image $\mathbf{y}$ we compute the posterior probability distribution of the hidden mask

$$p(\mathbf{r}|\mathbf{y}) = p(\mathbf{y})^{-1} p(\mathbf{y}, \mathbf{r}) \tag{5.1}$$

$$p(\mathbf{y}) = \sum_{\mathbf{r}} p(\mathbf{y}, \mathbf{r}) \tag{5.2}$$

and select appropriate mask according to $p(\mathbf{r}|\mathbf{y})$. One possibility is to select the MAP (maximum a-posteriori) mask $\mathbf{r} = \arg\max_{\bar{\mathbf{r}}} p(\mathbf{r} = \bar{\mathbf{r}}|\mathbf{y})$. An alternative option is to select so called max-marginal mask, where the individual indicators r_k are obtained as $r_k = \arg\max_{\bar{r}_k} p(r_k = \bar{r}_k|\mathbf{y})$, where $p(r_k|\mathbf{y})$ denotes a marginal posterior distribution.

Typically, the models $p(\mathbf{r})$ and $p(\mathbf{y}|\mathbf{r})$ depend on a set of (unknown) parameters. Probabilistic learning is the process of finding the parameters on the basis of training data. Learning algorithms (Neal and Hinton, 1998) adjust these models by maximizing the *evidence* term $p(\mathbf{y})$ as given by (5.2). For many types of models the exact maximization is intractable and one typically resorts to the Expectation Maximization (EM) algorithm (Neal and Hinton, 1998).

In this chapter our focus lies in the prior model $p(\mathbf{r})$, which can be used with various types of observed images $\mathbf{y}$ and observation models $p(\mathbf{y}|\mathbf{r})$. Specifically, one can consider the following cases.

- The vector $\mathbf{y}$ can be a binary image (see Fig. 5.1) where individual foreground pixels are detected as outliers from the static background image. This is the primary type of observed images considered in this chapter. This choice simplifies the derivation of the learning procedure. Moreover, the resulting algorithm is already practical since there exist many fast techniques for segmentation of individual pixels (e.g. (Stauffer and Grimson, 1999; Zivkovic, 2004)).
- Sometimes we have access to an empirical distribution $q(\mathbf{y}) = \prod_{k=1}^{K} q(y_k)$, where $q(y_k = 1) \in \langle 0, 1 \rangle$ indicates the probability that the kth pixel is assigned to the foreground. Such distributions arise mainly as posteriors in probabilistic models for segmentation of individual pixels (Friedman and Russell, 1997). Rather than thresholding $q(\mathbf{y})$ to obtain a "crisp" input $\mathbf{y}$, it is better to preserve the uncertainty in the labels, and consider $q(\mathbf{y})$ as the input. As we show in Section 5.6.3, the EP framework handles this case with little extra implementation effort.

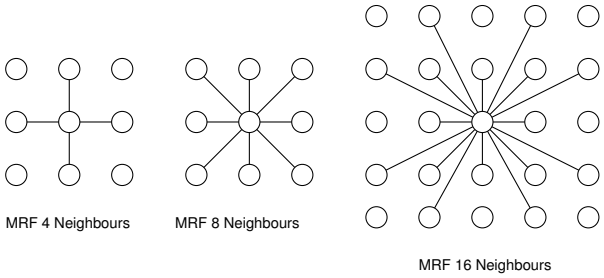


Figure 5.2: Three MRF implementations tested in the chapter. In the graphs, a circular node represents binary mask of a single pixel, and edges indicate neighbor pixels. The graphs show different correlation structures, where a pixel is related to its 4, 8, or 16 neighboring pixels. Note, that in general MRFs express correlations in terms of maximal cliques (see Chapter 2). Therefore the above graphs correspond to slightly constrained MRFs.

- The vector $\mathbf{y}$ can also be a color image, where every pixel is represented as a point in a suitable color space. In Section 5.3 we briefly mention the corresponding observation model. A detailed approach for this type of observed images is discussed in (Cemgil et al., 2005).

5.2.2 Markov Random Fields

Markov random field models (e.g. (Freeman et al., 2000; Geman and Geman, 1984)) provide a particular way to define the probability distributions $p(\mathbf{r})$ and $p(\mathbf{y}|\mathbf{r})$, which take the form

$$p(\mathbf{y}|\mathbf{r}) = \frac{1}{Z_{\mathrm{y}}} \prod_{k=1}^{K} \psi_k(r_k, y_k) \qquad p(\mathbf{r}) = \frac{1}{Z_{\mathrm{r}}} \prod_{k=1}^{K} \prod_{n \in V(k)} \phi_{kn}(r_k, r_n),$$

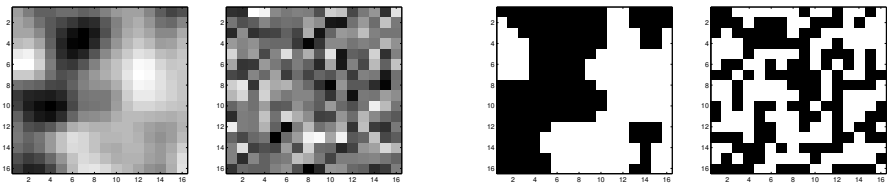
where $V(k)$ is a set of neighbors of the kth pixel, and Z_{y}, Z_{r} are normalization constants (in practice for inference these do not have to be known). The functions ϕ_{kn} determine the strength of coupling between the adjacent pixels, and ψ_k specifies the noise in the observed pixels. One can control the mask correlations by choosing appropriate neighborhood structures and terms ϕ_{kn}.

Figure 5.2 presents three MRF models with different neighborhood sets that include 4, 8 or 16 neighbors per pixel. An example choice of the functions ψ_k, ϕ_{kn} is

$$\psi_k(r_k = y_k) = \alpha \qquad \phi_{kn}(r_k = r_n) = \beta \tag{5.3}$$

$$\psi_k(r_k \neq y_k) = 1 - \alpha \qquad \phi_{kn}(r_k \neq r_n) = 1 - \beta, \tag{5.4}$$

where $\alpha, \beta \in (0, 1)$ are parameters. Increasing α reduces the chance that the hidden and observed labels will differ. Increasing β makes the pixels in the mask $\mathbf{r}$ more correlated. Later in the chapter we demonstrate mask reconstruction based on these models.



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

Figure 5.3: (a,b) Each plate represents a vector $\mathbf{x}$ drawn from a Gaussian $\mathcal{N}(0, \mathbf{S})$, where the vector elements have been arranged on a 16×16 grid. (c,d) Binary masks $\mathbf{y}$ obtained by clipping the corresponding vectors $\mathbf{x}$. We have parametrized $\mathbf{S} = (s_{k_1,k_2})$ as $s_{k_1,k_2} = \theta_2 \exp(-\frac{|k_1-k_2|^2}{\theta_1^2}) + \theta_3\delta(k_1 - k_2)$ where the coupling between pixels (k_1, k_2) depends on their weighted Euclidean distance, and $\delta()$ denotes the Dirac-delta function. The weights were $\theta_2 = 5$, $\theta_3 = 0.01$, (a) $\theta_1 = 2$, and (c) $\theta_1 = 0.5$.

Despite the apparent simplicity, in the MRF framework exact inference of the hidden mask $\mathbf{r}$ is usually intractable (Geman and Geman, 1984). However, in many cases posterior distributions $p(r_k|\mathbf{y})$ can be approximated using the Loopy-Belief Propagation (LBP) algorithm — a popular message-passing method for inference in intractable probabilistic models (Yedidia et al., 2001). More importantly, learning of the prior model $p(\mathbf{r})$ is difficult since it entails estimating an optimal neighbor set $V(k)$ for every pixel k (i.e., learning corresponds to optimization over the space of graph topologies, see Fig. 5.2). Additionally, learning requires estimation of the normalization terms, in order to avoid degenerate solutions. Therefore usually MRFs assume a generic, predefined neighborhood structures, which involve only short-range couplings of pixels.

5.3 Clipped factor analysis model

In this section we present a Clipped Factor Analysis (CFA) model as an alternative way to express the prior density $p(\mathbf{r})$ and provide two examples of observation models $p(\mathbf{y}|\mathbf{r})$ that can be conveniently used with the prior.

Our approach follows from an observation that signs of correlated Gaussian variables are also correlated. Therefore we introduce a random K-dimensional real-valued vector $\mathbf{x}$ that is a-priori Gaussian distributed, and obtain binary masks $\mathbf{y}$ by taking the sign of the corresponding entries in $\mathbf{x}$; $y_k = \mathrm{sgn}(x_k)$. Accordingly, correlations of the continuous variables introduce couplings between binary indicators. We can control these correlations by choosing an appropriate covariance matrix for the Gaussian distribution. The idea is illustrated in Fig. 5.3, which presents two masks obtained by clipping vectors $\mathbf{x}$ that have been drawn from a Gaussian $\mathcal{N}(0, \mathbf{S})$ with different covariances $\mathbf{S}$. We note however, that this is only a toy example since an explicit definition of covariance (see caption of Fig. 5.3) requires $\mathcal{O}(K^2)$ parameters.

A more suitable way to induce a Gaussian density is by a factor analysis (FA) model. The FA model (Roweis and Ghahramani, 1999) assumes that the K-dimensional vector $\mathbf{x}$ is obtained as a noisy, linear projection of a latent d-dimensional vector $\mathbf{z}$ ($d \ll K$), which is a-priori Gaussian distributed with unit covariance. We express the model as:

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \tag{5.5}$$

$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{a}, \mathbf{R}) \tag{5.6}$$

$$\mathbf{y}|\mathbf{x} = \mathrm{sgn}(\mathbf{x}), \tag{5.7}$$

where $\mathbf{a}$ is a $K \times 1$ mean vector, $\mathbf{W}$ is a $K \times d$ linear projection matrix, and $\mathbf{R}$ is a diagonal $K \times K$ matrix. By integration over $\mathbf{z}$, it is easy to see that this model induces on $\mathbf{x}$ a Gaussian distribution $\mathcal{N}(\mathbf{a}, \mathbf{W}\mathbf{W}^\top + \mathbf{R})$ with a constrained (but in general full) covariance defined by $\mathcal{O}(dK)$ parameters. We refer to this model as Clipped Factor Analysis (CFA) model.

The CFA model does not explicitly define the distributions $p(\mathbf{r})$ and $p(\mathbf{y}|\mathbf{r})$. However, by relating the mask $\mathbf{r}$ to the random vector $\mathbf{z}$, one can obtain an implicit prior $p(\mathbf{r})$. We assume the sought mask can be approximated as a "noiseless" projection of the hidden vector $\mathbf{z}$

$$\mathbf{r}|\mathbf{z} = \mathrm{sgn}(\mathbf{W}\mathbf{z} + \mathbf{a}).$$

Such a formulation allows us to estimate hidden mask $\mathbf{r}$ by probabilistic inference on vector $\mathbf{z}$ from $\mathbf{y}$ without explicit definition of $p(\mathbf{r})$. The directed graphical model in Fig. 5.4 illustrates the relation between vectors $\mathbf{r}$,$\mathbf{z}$,$\mathbf{x}$ and $\mathbf{y}$.

Note that alternative observation models can be defined by relating an observed image $\mathbf{y}$ to the vector $\mathbf{x}$, which is a "noisy" projection of $\mathbf{z}$. For example, if $\mathbf{y}$ represented a color image, we could consider the following observation model

$$y_k|x_k \sim \begin{cases} p_{\mathrm{f}}(y_k) & \text{if } \mathrm{sgn}(x_k) \geq 0 \\ p_{\mathrm{b}}(y_k) & \text{if } \mathrm{sgn}(x_k) < 0 \end{cases} \quad \text{for } k = 1 \ldots K,$$

where $p_{\mathrm{f}}(y_k)$ and $p_{\mathrm{b}}(y_k)$ model the color of, respectively, foreground and background pixels. This model is further presented in (Cemgil et al., 2005).

Analogously to the standard FA models, we can interpret columns of $\mathbf{W}$ as vectors that span a subspace of masks, and $\mathbf{z}$ as a latent mask coordinate in this subspace. For instance consider Fig. 5.5, where $d = 2$, $\mathbf{z} = [z_1, z_2]^\top$ and the columns of $\mathbf{W}$ have been chosen to produce circular shapes. In the bottom panel we demonstrate the effect of noise on the masks generated by or model. Since the noise covariance matrix $\mathbf{R}$ is diagonal, the distortions of individual pixels are uncorrelated, and do not affect the general structure of the masks.

In summary, our model includes vectors $\mathbf{z}$, $\mathbf{x}$ $\mathbf{y}$ with a joint distribution $p(\mathbf{z}, \mathbf{x}, \mathbf{y}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{x})$, where the individual densities $p(\mathbf{z})$, $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{y}|\mathbf{x})$ are defined by, respectively, (5.5)–(5.7). In the remainder of this section we discuss the inference and learning for the model. The former tries to estimate an underlying mask from an observed image $\mathbf{y}$. The later finds the optimal subspace (matrix $\mathbf{W}$) and noise variance $\mathbf{R}$ from a set of training images.

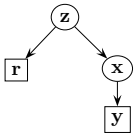


Figure 5.4: A directed graph (a Bayes network) representing the CFA model. Discrete variables are indicated as rectangles, real-valued— as ovals.

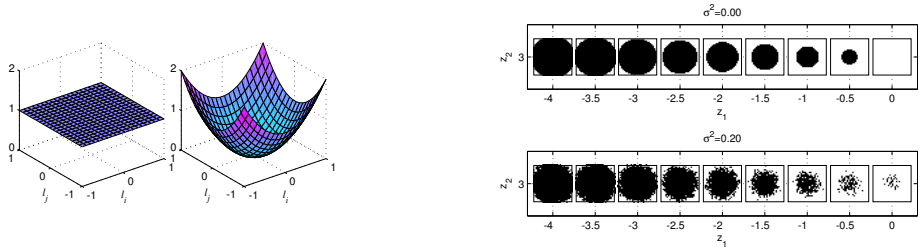


Figure 5.5: Two dimensional subspace. (Left) Two columns of matrix $\mathbf{W} = [W^{(1)}, W^{(2)}]$, where each column contains 400 entries, arranged on a 20×20 grid. We set $W^{(1)}_{i,j} = 1$, and $W^{(2)}_{i,j} = l_i^2 + l_j^2$, where l_i and l_j are 20 uniformly spaced numbers from $(-1, 1)$. (Right, Top) Every rectangular box represents a 20×20 mask $\mathbf{r} = \mathrm{sgn}(\mathbf{W}\mathbf{z} + \mathbf{a})$, where $\mathbf{z} = [z_1, z_2]^\top$ are the subspace coordinates corresponding to the box center, and $\mathbf{a} = 0$. (Right, Bottom) The corresponding noisy masks $\mathbf{y} = \mathrm{sgn}(\mathbf{x})$, $x \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{a}, 0.2\mathbf{I})$.

5.3.1 Inference

Our model assumes that the hidden mask $\mathbf{r}$ can be computed deterministically from the mask coefficients $\mathbf{z}$. Thus, our inference algorithm estimates the mask by finding coefficients $\mathbf{z}$ on the basis of posterior density $p(\mathbf{z}|\mathbf{y})$

$$p(\mathbf{z}|\mathbf{y}) = p(\mathbf{y})^{-1} \int p(\mathbf{z}, \mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x}$$
$$p(\mathbf{y}) = \iint p(\mathbf{z}, \mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{z},$$

where we integrate $\mathbf{x}$ since it is a hidden random variable.

Unlike in the case of linear Gaussian models, neither of the interesting quantities can be computed exactly. The non-linear clipping mechanism renders inference in our model intractable. For instance, when computing $p(\mathbf{y})$ we could integrate $\mathbf{z}$; $p(\mathbf{y}, \mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{W}\mathbf{W}^\top + \mathbf{R})p(\mathbf{y}|\mathbf{x})$. However, the subsequent integration over $\mathbf{x}$ requires computing a Gaussian integral in one of the 2^K orthants specified by binary image $\mathbf{y}$, and this cannot be done in closed-form.

A straightforward approximation would be to sample from $\mathbf{z}$ and then analytically integrate over $\mathbf{x}$. In practice, unless $\mathbf{z}$ is low-dimensional, Rao-Blackwellized Gibbs sam-

pling will be computationally expensive. Alternatively, we could use mean-field approximation. However, due to the hard clipping mechanism, the mean-field with factorized Gaussians as the approximating family would break down (since Kullback-Leibler divergence between any Gaussian and a clipped Gaussian becomes ∞), and one has to relax clipping with sigmoidal threshold or use a more exotic approximating family.

Expectation Propagation

The posterior distributions in our model can be conveniently approximated with the expectation-propagation (EP) algorithm (Minka, 2001b). The algorithm approximates intractable posterior distributions of individual hidden variables with simple parametric functions — called beliefs. From the beliefs we find various posterior statistics that are necessary for inference and learning.

As discussed in Chapter 2, EP can be viewed as a message-passing algorithm on a *factor graph* that represents the joint probability distribution of the model. Recall, that a factor graph (Kschischang et al., 2001) comprises (i) factor nodes that represent individual functions in the joint, and (ii) variable nodes that denote random variables. Figure 5.6 shows a factor graph of our model, where the nodes A, B, C correspond to functions $p(\mathbf{z})$, $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{y}|\mathbf{x})$. The function represented by factor f will be denoted as $\phi_f(V_f)$, where V_f is a set of variables adjacent to factor f, thus $\phi_{\mathrm{A}}(\mathbf{z}) = p(\mathbf{z})$; $\phi_{\mathrm{B}}(\mathbf{z}, \mathbf{x}) = p(\mathbf{x}|\mathbf{z})$, and $\phi_{\mathrm{C}}(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})$. The belief associated with variable v will be denoted as $q(v)$, thus we introduce beliefs $q(\mathbf{z})$ and $q(\mathbf{x})$. In the primary setup the vector $\mathbf{y}$ is observed. In the case when information about $\mathbf{y}$ is given by a distribution $q(\mathbf{y})$ we will consider $q(\mathbf{y})$ as a fixed belief.

The algorithm iteratively estimates beliefs together with intermediate terms, known as messages. We use a variant that can be summarized as a fixed-point iteration between beliefs and messages from factor to adjacent variables, $m_{f\to v}(v)$. The algorithm is presented in detail in Chapter 2; below we provide the general update rule for reference

$$m^{\mathrm{new}}_{f\to v}(v) = \frac{Z q^{\mathrm{new}}(v)}{q(v)} m_{f\to v}(v) \tag{5.8}$$

$$q^{\mathrm{new}}(v) = \arg\min_q \mathrm{KL}(\tilde{q}||q) \tag{5.9}$$

$$\tilde{q}(v) = \frac{1}{Z} \int_{V_f \setminus v} \phi_f(V_f) \prod_{v' \in V_f} \frac{q(v')}{m_{f\to v'}(v')}, \tag{5.10}$$

where Z denotes normalization constant for $\tilde{q}$. We refer to the above equations as a "message-sending" operation from f to v. Intuitively, the algorithm can be viewed as a generalization of the Belief Propagation method with an extra step (5.9). For a continuous variable v the integral $\tilde{q}(v)$ might be a complicated function, whereas a belief $q(v)$ typically is a simple function from the exponential family. In this case, minimizing KL-divergence corresponds to finding such belief $q^{\mathrm{new}}(v)$ whose moments match the moments of $\tilde{q}(v)$ (Cover and Thomas, 1991).

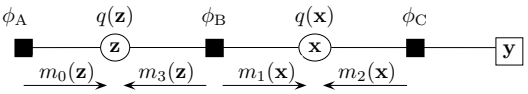


Figure 5.6: A factor graph for our model, where the filled nodes denote factors and empty ovals (or squares) represent continuous (or discrete) variables. The vector $\mathbf{r}$ is not shown, as it does not contribute to the inference.

EP inference algorithm:
1. Initialize
   $q(\mathbf{z}) = p(\mathbf{z})$, $q(\mathbf{x}) = 1$,
   $m_1(\mathbf{x}) = m_2(\mathbf{z}) = m_3(\mathbf{x}) = 1$
2. Iterate
   send message $m_1(\mathbf{x})$ from $\phi_B(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})$ to $\mathbf{x}$
   send message $m_2(\mathbf{x})$ from $\phi_C(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})$ to $\mathbf{x}$
   send message $m_3(\mathbf{z})$ from $\phi_B(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})$ to $\mathbf{z}$

Figure 5.7: The schedule of the message passing algorithm for our model.

In our application, we represent beliefs of hidden vectors $q(\mathbf{z})$, $q(\mathbf{x})$ as Gaussian densities. This is a natural choice since our model builds upon linear Gaussian models. Due to the large dimensionality of vector $\mathbf{x}$ we constrain belief $q(\mathbf{x})$ to be a Gaussian with diagonal covariance matrix. The messages in our model are denoted as $m_1(\mathbf{x}) = m_{B\to x}(\mathbf{x})$, $m_2(\mathbf{x}) = m_{C\to x}(\mathbf{x})$ and $m_3(\mathbf{z}) = m_{B\to z}(\mathbf{z})$. We parametrize these messages as Gaussian potentials. The message $m_0(\mathbf{z}) = m_{A\to z}(\mathbf{z})$ does not require updating because factor A is adjacent only to a single variable (see Fig. 5.6).

We send the messages in the order indicated in Fig. 5.7, where every EP iteration consists of three steps. First, we "predict" $\mathbf{x}$ by sending message $m_1(\mathbf{x})$, then "update" $\mathbf{x}$ by sending message $m_2(\mathbf{x})$, and finally "update" $\mathbf{z}$ by sending message $m_3(\mathbf{z})$. This schedule seems the most natural for our model, given the fact that the generic EP formulations do not advocate any particular order. We provide a detailed implementation of (5.8)–(5.10) for sending messages in our model in the Appendix 5.6.1.

Computational cost The computational cost of a single iteration of the EP procedure is the sum of computational costs of sending the three types of messages. The exact message-update equations are provided in the Appendix 5.6.1. Here we summarize the respective computational costs. The message $m_3(\mathbf{z})$ requires inversion of a $d \times d$ matrix with the associated cost $\mathcal{O}(d^3)$. Sending message $m_1(\mathbf{x})$ requires inversion of a $K \times K$ matrix $\mathbf{G}$. (Recall that K is the number of pixels, and d is the subspace dimension.) Fortunately, this matrix can be decomposed as $\mathbf{G} = \mathbf{A} - \mathbf{C}\mathbf{\Lambda}\mathbf{C}^\top$, where $\mathbf{A}$ is a $K \times K$ diagonal matrix, $\mathbf{\Lambda}$ is a $d \times d$ matrix, and $\mathbf{C}$ is a $K \times d$ matrix. Using the matrix inversion lemma, $\mathbf{G}$ can be inverted in $\mathcal{O}(d^3 + K)$ time. When computing the message $m_1(\mathbf{x})$ every

pixel is treated independently, there the cost of the message is $\mathcal{O}(K)$. Accordingly, with n iterations the EP inference costs $\mathcal{O}(nd^3 + nK)$, that is linear in the number of pixels and cubic in the subspace dimensionality.

5.3.2 Learning

Given the probabilistic framework, a natural question arises about selecting model parameters. Particularly important is the matrix $\mathbf{W}$ whose columns span the space of masks typical for the model. Below, we employ probabilistic learning to estimate the parameters from exemplary masks. In this way obtain a model that is specialized to a class of objects characteristic for a given scene.

Suppose one has a set of M training masks denoted as $\mathbf{y}_n$, $n = 1, \ldots, N$. We assume that each of the masks was independently generated from our model, and attempt to find the model parameters $\theta^* = \{\mathbf{W}^*, \mathbf{a}^*, \mathbf{R}^*\}$ that maximize the data log-likelihood

$$\theta^* = \arg\max_\theta \sum_n \log p(\mathbf{y}_n|\theta), \tag{5.11}$$

with $p(\mathbf{y}_n|\theta)$ denoting the likelihood of mask $\mathbf{y}_n$ conditioned on some choice of parameters θ. Since the exact maximization of log-likelihood is intractable, we resort to the expectation-maximization (EM) algorithm (Neal and Hinton, 1998) (see Chapter 2 for details). In every iteration, the algorithm requires posterior statistics on the latent variables. We approximate these statistics with EP iterations. Although there is no guarantee of convergence, such double-loop EP-EM procedures have been shown successful (Minka and Lafferty, 2002). Alternatively, one might try the convergent double-loop algorithm (Heskes et al., 2004).

Initialization EM finds only locally optimal parameters, therefore it is useful to start with an "educated guess" parameters. For the moment, we treat each $\mathbf{y}_n$ as a continuous variable, and find the sample mean $\bar{\mathbf{y}}$ and sample covariance $\mathbf{S}$ of $\mathbf{y}_{1:N}$. Next, we find eigen-decomposition of matrix $\mathbf{S}$: $\Lambda D = \mathbf{S}\Lambda$, and initialize parameters as a PCA subspace: $\mathbf{a} = \bar{\mathbf{y}}$ and $\mathbf{W} = \Lambda^{(1:d)} D^{1/2}$, where D is diagonal and $\Lambda^{(1:d)}$ are the d eigenvectors that correspond to the largest eigenvalues. The diagonal matrix $\mathbf{R}$ is randomly initialized.

EM iterations Given the current parameters θ, in the E-step, we find beliefs of the latent variables $q_n(\mathbf{z})$ and $q_n(\mathbf{x})$ for every data point $\mathbf{y}_n$ separately by iterating EP until convergence (in practice 3 loops). From the beliefs we compute the following posterior statistics $\langle \mathbf{x}|\mathbf{y}_n \rangle$, $\langle \mathbf{x}\mathbf{x}^\top|\mathbf{y}_n \rangle$, $\langle \mathbf{z}|\mathbf{y}_n \rangle$, and $\langle \mathbf{z}\mathbf{z}^\top|\mathbf{y}_n \rangle$, where $\langle v \rangle = \int v q(v)\, \mathrm{d}v$. Since we represent the beliefs as Gaussian densities, the required statistics can be easily found from the parameters of a Gaussian.

In the M step we reestimate θ by maximizing the expected log-likelihood given the beliefs from the E-step. The maximizing parameters can be easily found by observing that θ affect only the linear Gaussian part of our model. Therefore the M-step is analogous to the standard FA models (Roweis and Ghahramani, 1999). It differs only in the fact that we do not have the access to vectors $\mathbf{x}_n$, which are replaced by their statistics conditioned on $\mathbf{y}_n$. For completeness, we provide the M-step equations

$$\bar{\mathbf{W}} = [\mathbf{W}, \mathbf{a}] = \Big(\sum_n \langle \mathbf{x}|\mathbf{y}_n \rangle \langle \bar{\mathbf{z}}|\mathbf{y}_n \rangle^\top \Big) \Big(\sum_n \big\langle \bar{\mathbf{z}}\bar{\mathbf{z}}^\top|\mathbf{y}_n \big\rangle \Big)^{-1}$$

$$\mathbf{R} = \frac{1}{N} \operatorname{diag} \Big[\sum_n \big\langle \mathbf{x}\mathbf{x}^\top|\mathbf{y}_n \big\rangle - \bar{\mathbf{W}} \sum_n \langle \bar{\mathbf{z}}|\mathbf{y}_n \rangle \langle \mathbf{x}|\mathbf{y}_n \rangle^\top \Big]$$

Here, we update $\mathbf{W}$ and $\mathbf{a}$ jointly, using an augmented vector $\bar{\mathbf{z}}^\top = [\mathbf{z}^\top, 1]$, whose statistics are straightforward

$$\langle \bar{\mathbf{z}}|\mathbf{y}_n \rangle = \begin{bmatrix} \langle \mathbf{z}|\mathbf{y}_n \rangle \\ 1 \end{bmatrix} \qquad \big\langle \bar{\mathbf{z}}\bar{\mathbf{z}}^\top|\mathbf{y}_n \big\rangle = \begin{bmatrix} \big\langle \mathbf{z}\mathbf{z}^\top|\mathbf{y}_n \big\rangle & \langle \mathbf{z}|\mathbf{y}_n \rangle \\ \langle \mathbf{z}|\mathbf{y}_n \rangle^\top & 1 \end{bmatrix}.$$

Note, that the procedure does not require full matrices $\big\langle \mathbf{x}\mathbf{x}^\top|\mathbf{y}_n \big\rangle$ (which are of size $\mathcal{O}(K^2)$), but only their diagonals (which are of size $O(K)$). We also observe, that the EP algorithm provides an estimate of data log-likelihood, which can be used for monitoring convergence of the EM iterations.

5.4 Experiments

In this section we experimentally evaluate various aspects of our model. First, we examine the convergence properties of the EP iterations, since the convergence time determines the practicality of inference with the EP algorithm. Second, we create a dataset of aligned, real-world images and test learning of model parameters. Next, we compare segmentation accuracy and computation speed of our and MRF-based models. The experiments involve a simple set of images with ground-truth segmentation and various real-world video sequence recorded in office-like environments. Finally, the chapter provides evaluation of the presented model when applied for tracking humans in video sequences. These tests involve real-world data recored at an office and airport scenarios.

Throughout the section, we compare the presented model with three variants of MRF, as presented in Fig. 5.2. The models were parametrized according to (5.3)–(5.4) with $\alpha = 0.99$ and $\beta = 0.7$. Inference in the MRF-based models is performed with the Loopy-Belief Propagation (LBP) algorithm. In all the experiments we use the pixel-wise background segmentation algorithm (Zivkovic, 2004) to precompute binary images $\mathbf{y}$ from the video frames, as assumed by the model.

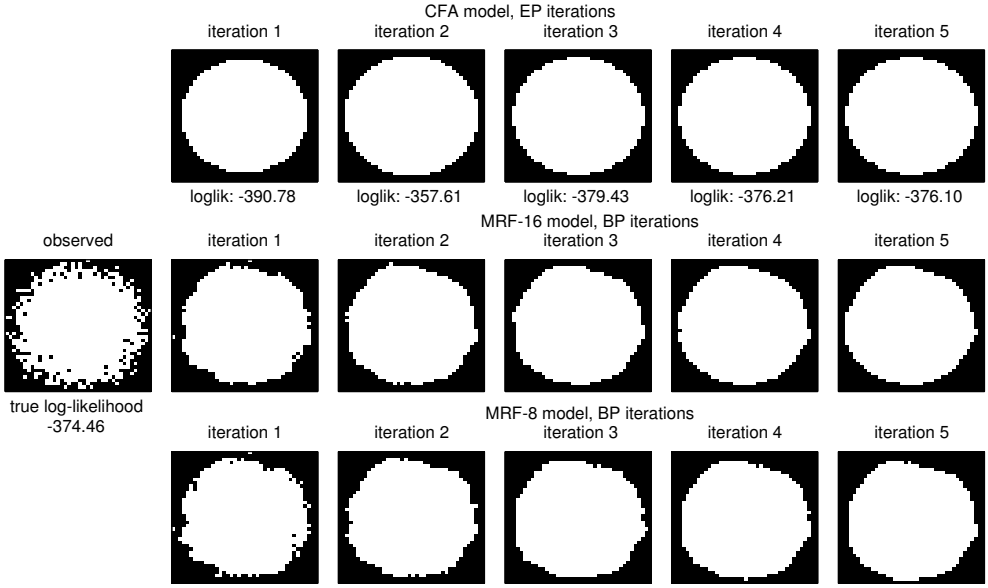


Figure 5.8: EP convergence test. (Left) A random mask $\mathbf{y}$ generated from our model, where the parameters $\mathbf{W}$ and $\mathbf{R}$ were the same as in Fig.2, and $\mathbf{R} = \mathbf{I}$. (Top) Masks $\mathbf{r}$ reconstructed by our model, after subsequent EP iterations; $\mathbf{r} = \mathrm{sgn}(\mathbf{a} + \mathbf{W}\,\langle \mathbf{z}|\mathbf{y} \rangle)$. Below the masks we show the log-likelihood (ll) estimates $\log p(\mathbf{y})$. The true log-likelihood has been computed by sampling 10^8 samples from $p(\mathbf{z})$ and analytical integration over $\mathbf{x}$. (Middle) Masks reconstructed by a MRF-16 model. (Bottom) Masks reconstructed by a MRF-8 model. For both MRF models parameters were ($\alpha = 0.99$, $\beta = 0.7$).

EP convergence A key aspect of the presented model is the convergence of the EP message-passing iterations. The EP inference algorithm in our model (as well as the LBP algorithm in MRF models) lacks a formal convergence guarantee, therefore experimental validation of convergence is necessary. We illustrate the convergence properties in Fig. 5.8. The figure shows a noisy mask $\mathbf{y}$ generated from our model (Top), and a series of reconstructed masks $\mathbf{r}$ after several EP iterations. For each iteration we also show the estimate of log-likelihood $\log p(\mathbf{y})$. In the middle and bottom panels we show the masks reconstructed by MRF-16 and MRF-8 models during subsequent LBP iterations.

In this, and several other experiments, we observe that the EP algorithm converges after $3 - 4$ iterations regardless of the number of pixels and the dimensionality of the mask subspace. This convergence rate is similar to the typical runs of belief propagation for MRFs. Therefore, in all other experiments with our model (or MRF models) we perform three EP (or LBP) iterations.

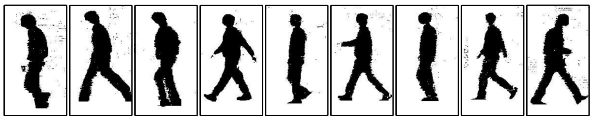


Figure 5.9: Selected binary images (masks) from the training sequence. The image size was 50×30 pixels. Each mask represents a human silhouette, centered within the image.

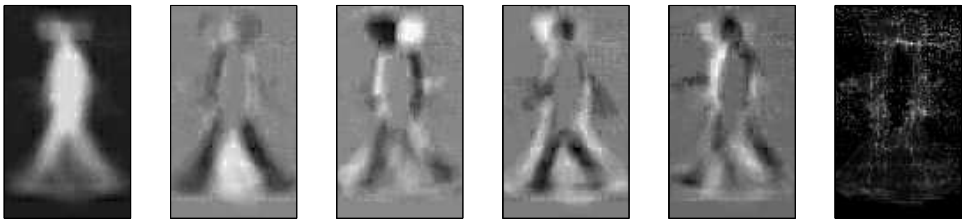


Figure 5.10: Parameters of the CFA model $(\mathbf{a}, \mathbf{W}, \mathbf{R})$ estimated by the EM algorithm from a training set that included human silhouettes. The vector $\mathbf{a}$, the first four columns of matrix $\mathbf{W}$ and the diagonal of matrix $\mathbf{R}$ are shown as 50×30 images.

Learning model parameters In this experiment we illustrate learning of model parameters using the EM algorithm that has been described in Section 5.3.2. Our training data includes 100 noisy, binary images of size 50×30 pixels. The images represent silhouettes of various humans that have been observed walking lateral to the camera plane. The full frames were cropped in order to center the silhouettes within a 50×30 window, as shown in Fig. 5.9.

Given the training data, we run the EM algorithm with various dimensions of the mask subspace d. Figure 5.10 presents the estimated parameters after 10 EM iterations for $d = 15$. We observe that the parameter $\mathbf{a}$ represents the body parts (torso, head) typically present in every silhouette. The first column of $\mathbf{W}$ represents various leg configurations. When the coefficient z_1 is positive the legs are closed, when z_1 is negative the angle between legs increases. Analogously, the second column of $\mathbf{W}$ together with the second coefficient z_2 controls the head swing, and overall orientation of the silhouette (left vs right). Figure 5.11 demonstrates this effect explicitly, where we show masks obtained by various settings of the coefficients $[z_1, z_2]$.

The considered setup assumes that people walk lateral to the camera plane and a camera placed approx. 1.5 meters above the ground. Essentially, in this case there are only three types of shape variations: leg movements, arm movements, pose orientation (face directed to left or right). Therefore 100 training images proved enough to capture the typical variation in the shapes.

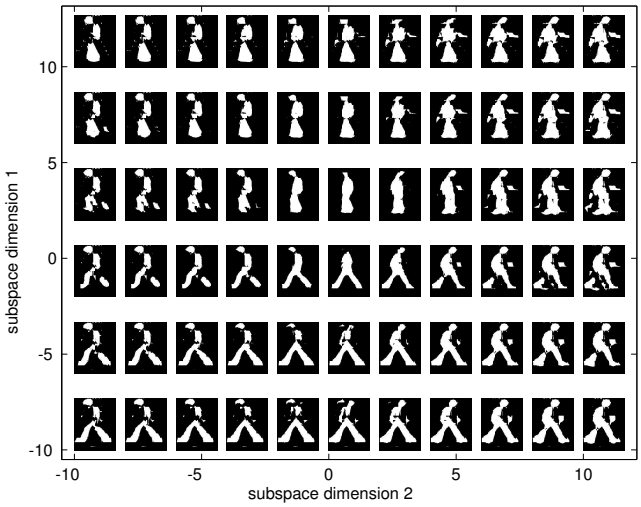


Figure 5.11: Mask subspace spanned by the first two columns of matrix $\mathbf{W}$. Every binary plot shows a mask $\mathbf{r} = \mathrm{sgn}(\mathbf{a} + [W_1, W_2][z_1, z_2]^\top)$, where coefficients $[z_1, z_2]$ correspond to the location of the plot on the grid.

Segmentation Accuracy In this experiment we compare the accuracy of segmentation with the CFA model and the MRF models. Segmentation error is defined as a percentage of pixels that are mis-segmented according to "ground truth" segments (which were obtained by hand). Our test set consists of 60 noisy, binary images. The test data were obtained in the same setup as the training data, however involved different persons (with differently clothes) from those involved in the training set. The tests involve CFA models with varying dimensionality of the masks subspace d, and MRF models with varying number of neighbors per pixel. Aside to the segmentation error we also measure the execution time for the tested methods.

Figure 5.12 summarizes the results. We observe that learning the prior model of spatial correlations results in lower segmentation errors in comparison with the generic correlations assumed by the MRF models. However, increasing the dimensionality d of linear subspace beyond some optimal value leads to overfitting in the case of the CFA model. In the case of MRFs, we observe that the segmentation error was not affected by the number of neighbors correlated with each pixel. This effect follows from the fact that generic correlation structures (like those in Fig. 5.2) cannot describe well pixel correlations specific to images of a particular type of objects.

Mask selection The previous experiments were based on a dataset where the objects were aligned in the image. In practical scenarios, we need to automatically find a window that includes an object of interest within a larger image. For tracking applications,

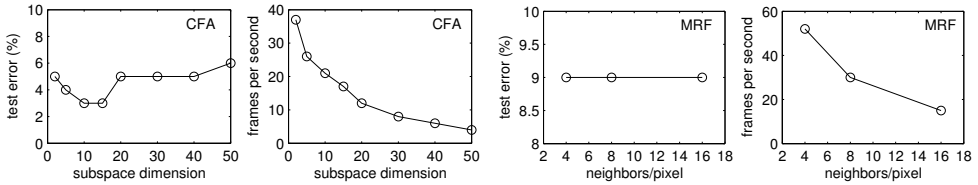


Figure 5.12: (Left) Segmentation error and computation speed for the CFA model. (Right) Segmentation error and computation speed for various MRF models. All computation times are expressed as frame per second (fps), where the frame size was 50×30 pixels, (MATLAB implementations, on a 3GHz PC).

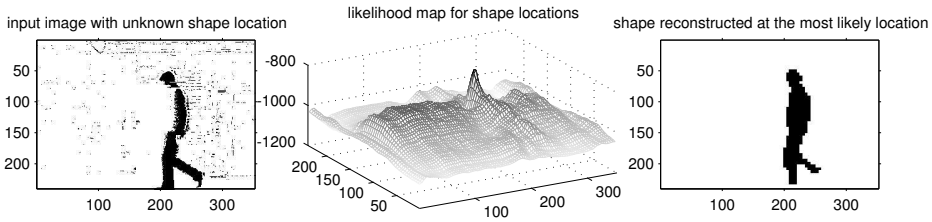


Figure 5.13: Finding a mask of a human in an image. (Left) An input binary image, where we want to find the location of an objects' mask. (Center) A log-likelihood map, where the value at each pixel indicates the log-likelihood of an object present at this location. (Right) The reconstructed mask, positioned at the most likely location.

location and scale of this box correspond to the translation and scaling of the mask. Below, we illustrate a window selection mechanism based on the likelihood computation in our model.

Figure 5.13 shows an experiment illustrating the ability to detect the most likely location of the mask within a video frame. The left panel shows a 240×320 binary image. From this image we construct binary masks $\mathbf{r}_{(i,j)}$ of size 130×70, each centered at a pixel (i, j) in the image (the masks are padded with zeros when necessary). The center panel shows a "likelihood map", where the value at pixel (i, j) gives the log-likelihood of the mask $\log p(\mathbf{r}_{(i,j)})$. We select the mask corresponding to the global maximum of this map, and plot the reconstructed mask in the right panel of Fig. 5.13.

The experiment shows that one can find the actual object location by finding the maximum of the likelihood map. We notice that although the map exhibits a clear global maximum, typically there will be also several local maxima. In many applications, finding global maximum by exhaustive search might turn out intractable. We have to resort to local maximization techniques, which need an "educated guess" initialization in order to avoid finding a local maximum.

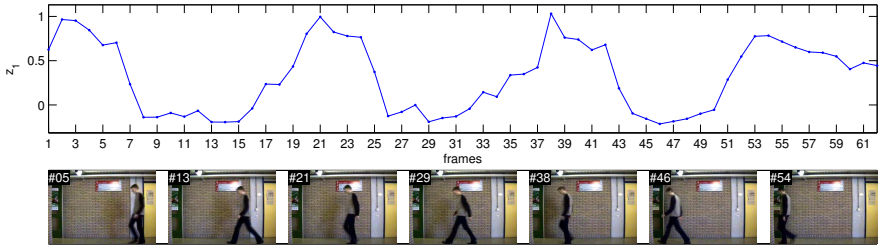


Figure 5.14: Evolutions of mask coefficients during a walking cycle. (Top) Estimated values of coefficient z_1. (Bottom) The corresponding body-part configurations at selected frames. The maxima (or minima) in z_1 correspond to the maximum (or minimal) angle between the legs of the person.

Segmenting video sequences As the final experiment, we consider the problem segmenting video sequence with multiple foreground objects. First, we show how the evolutions of an observed mask correspond to the evolutions of the latent mask coefficients through a sequence of frames. Second, the experiment allows comparing segmentation techniques on a larger dataset without the ground truth segmentation.

In order to segment video sequences we construct a simple tracking algorithm that finds foreground objects in a current frame given their location in a preceding frame. Our tracker finds locations of mask representing foreground objects using the "likelihood maps". First, the tracker predicts the current object location on the basis of previous locations by assuming that objects travel with a constant velocity. Next, the tracker employs a greedy, gradient-based search for a (local) maximum of the "likelihood map" starting at the predicted location. Additionally, new persons are detected by finding areas in the frame, where the number of foreground pixels exceeds a fixed threshold.

Figure 5.14 presents the results of segmenting a sequence that included a single person. The top panel presents the evolutions of the coefficient z_1 in the expected vector $\mathbf{z}$. The bottom panel includes several input frames at the indicated time steps. Note, a strict correspondence between the value of z_1 and pose of the tracked person — the periodical pattern in the top panel represents the periodical leg configurations of the object. (Analogous correspondence occurs between the persons' heading direction and the coefficient z_2 (not shown).) The experiment demonstrates potential applicability of the model to gesture-recognition tasks (Gavrila, 1999; Rao et al., 2002; Yamato et al., 1992).

Below we compare our model with MRF-based models. The experiment involves segmenting video sequences with several people and comparing the quality of the estimated masks. The masks are evaluated against two criteria. The first is the number of disconnected segments within a mask (the lower — the better). The second criterion follows from an assumption, that in the considered scenario the appearance of a person is practically constant during a trajectory. A trajectory, (i.e., a pass through the field of view), consists of a series of estimated masks, one per person and frame. Given the

masks, we compute a corresponding series of person's appearance features, defined as the average color of pixels representing the person. When the estimated masks correctly match with the underlying person, then the estimated features are constant during the trajectory. Therefore, we indirectly evaluate masks by measuring the variation in the appearance features estimated from each trajectory (the lower — the better).

The first experiment was performed on sequence (2000 frames) presenting people walking laterally to the camera plane recorded in an office scenario. Tables 5.1–5.2 summarize the results where we evaluate three approaches: (i) the CFA model with subspace dimension $d = 15$, (ii) various MRF models with parameters $\alpha = 0.99$, $\beta = 0.7$ and, as a reference, (iii) mask smoothing based on mathematical morphology (1 erosion, 2 dilations, 1 erosion). Figure 5.15 presents selected frames and estimated masks by two MRF models (with 8 and 16 neighbors per pixel) and the CFA model with subspace dimension $d \in \{5, 10, 15\}$.

According to Tab. 5.1, the CFA model yielded object masks consisting of significantly lower number of disjoint segments than the MRF models or morphological smoothing (in the ideal case, an object would consist of a single segment). Figure 5.15 illustrates another important property of the CFA model. In certain frames (e.g. 118, 570, 1516) the appearance of a person partially coincides with the background, and the coinciding pixels are missing in the observed mask (leftmost column). The CFA model recovered the missing pixels in the objects mask, therefore allowed us to properly characterize the appearance of each person during the trajectory. The ability to recover stable and exact appearance features is particularly important for applications, where one aims to identify persons on the basis of their appearance. Table 5.2 illustrates the advantage of our model by comparing the variance of color features for various methods (notice tracks $\#1, \#7, \#9, \#10$).

The second experiment was performed on a video sequence recorded in an airport environment, with the camera positioned approx. 7 meters above the ground, pointing downwards in a such way that the lower arm of the viewing angle pointed straight down. Here most of the persons are observed from a top-view and their shapes take to form of circular blobs with limbs stretching out. At the side of an image, the shapes start to vary and take the form of elongated blobs that were tilted at varying angles. Additionally, the silhouettes are distorted by various objects, like bags or luggage trolleys. In this case, the training set consisted of 170 images. (initially, the training set included 70 images, but this number turned out insufficient). Table 5.3 summarizes the results of this experiment. We observe that the advantages of the CFA model are less evident (especially w.r.t to the variation in the appearance features). In specific cases (e.g. tracks $\#5$ or $\#6$) the table indicates better performance of the CFA model, however from a visual inspection of the segments we conclude that in these cases all techniques fail.

Table 5.1: Quantitative comparison of various correlations techniques applied to tracking humans. Each row shows the average number of disconnected fragments in the foreground segment that represents a person.

	average number of segments/trajectory				
track	CFA-15	MRF-4	MRF-8	MRF-16	MORPH
01	1.5	2.7	2.5	2.1	4.1
02	1.1	1.8	1.8	1.4	2.4
03	1.1	2.6	2.3	1.9	3.1
04	1.3	3.6	2.7	2.2	4.7
05	1.4	2.6	2.1	1.7	3.2
06	1.3	2.3	2.2	1.7	2.8
07	1.4	3.0	2.3	1.8	3.5
08	1.4	4.1	3.2	2.6	4.9
09	1.6	4.7	3.9	3.0	5.5
10	1.6	6.3	5.0	3.6	7.9
11	1.2	17.3	7.1	4.2	11.3
12	1.2	13.3	7.2	4.7	13.6
average	1.4	5.4	3.5	2.6	5.6

Table 5.2: Quantitative comparison of various segmentation techniques applied to tracking humans. Each row shows variance in appearance features associated with a single person.

	variance in appearance features/trajectory				
track	CFA-15	MRF-4	MRF-8	MRF-16	MORPH
01	10.4	18.8	18.6	18.7	18.9
02	18.9	18.3	18.5	18.2	16.7
03	7.3	7.0	7.1	7.5	6.5
04	14.9	16.5	16.7	17.6	17.5
05	17.9	15.1	14.9	14.3	14.9
06	14.3	15.0	14.8	14.9	14.3
07	20.4	28.1	27.2	26.8	28.0
08	47.5	34.7	34.6	34.7	33.8
09	22.2	58.2	58.5	61.6	56.1
10	23.2	42.7	42.3	46.2	40.1
11	15.0	13.6	13.3	12.1	11.5
12	40.4	47.9	49.5	48.2	39.0
average	21.0	26.3	26.3	26.7	24.7

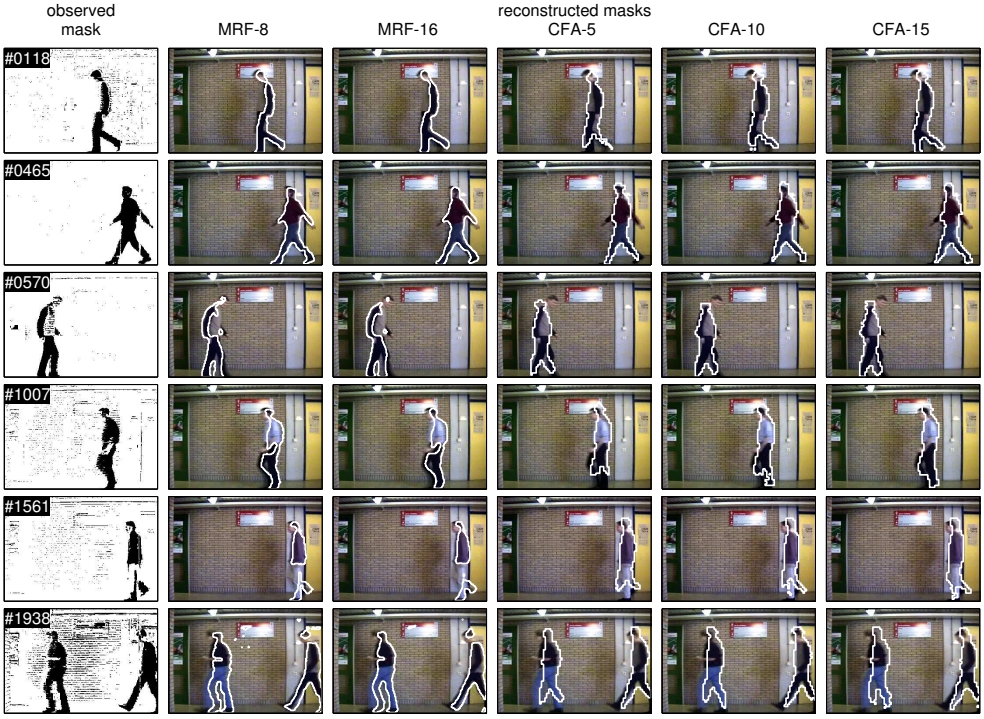


Figure 5.15: Qualitative comparison of various variants of the CFA and MRF models. Every row presents a single frame from a video sequence and masks recovered by the compared methods.

5.5 Conclusions

We considered a visual scene analysis problem, where pixels of an image have to be assigned a binary label that indicates whether the pixel represents a (static) background scene or a (moving) foreground object. An important aspect of this problem is proper characterization of spatial correlations between pixels assigned either to the background or to the foreground segment. We have shown that by incorporating prior knowledge about correlations typical for some scene, segmentation algorithms can handle difficult problems, such as cases where the background scene is difficult to distinguish from a foreground object.

This chapter presented a method for flexible representation and learning of prior knowledge about spatial correlations between binary labels. The method assumes that the labels are random variables with a prior probability distribution, defined as the factor analysis model with quantized output. The prior distribution is particularly suited for modeling long-range spatial couplings between the labels. Further, based exclusively

Table 5.3: Results of the experiment with "airport" video sequence.

	average number of segments			variance in appearance features		
track	CFA-15	MRF-16	MORPH	CFA-15	MRF-16	MORPH
01	1.9	3.8	6.0	29.0	26.0	22.4
02	2.1	2.9	6.8	32.0	17.5	32.2
03	1.1	4.8	6.2	15.0	36.1	30.1
04 [a]	2.3	4.0	8.0	38.7	34.8	38.8
05 [b]	1.9	3.0	4.4	109.6	135.3	130.1
06	2.5	1.3	2.9	26.1	21.2	20.6
07 [c]	2.4	3.7	9.9	95.3	182.3	170.0
08	1.6	3.6	9.3	42.6	7.2	15.0
09 [d]	2.5	2.6	4.3	63.4	25.7	25.0
average	2.0	3.3	6.4	50.1	54.0	53.8

[a] Object carries a backpack

[b] Object carries a bag

[c] Object pushes a luggage trolley

[d] Object pulls a suitcase

on exemplary data, parameters of this distribution can be easily adjusted in order to incorporate knowledge about label couplings typical for some class of foreground objects (e.g. human silhouettes). Such a learned prior offers a significant advantage over alternative prior models that assume only generic short-range correlations of pixel labels.

Background segmentation based on the presented model relies on probabilistic inference of unknown (hidden) pixel labels from an observed image. We have derived a local message-passing algorithm (Expectation Propagation) that approximates required posterior probabilities in a computationally tractable way. The inference algorithm handles arbitrary-range couplings of pixel labels in linear time in the number of pixels. The computation times in the presented model are comparable with the computation times in the popular MRF models.

In many applications background segmentation is not of interest itself, but is a basis for higher-level tasks, like object detection, classification or identification. An attractive feature of the presented model and message-passing algorithm is that they can be readily extended to more elaborate scenarios, like estimation of appearance features of the foreground objects (see (Cemgil et al., 2005)). Another potential applications of the model are gesture recognition problems (Gavrila and Davis, 1996; Rao et al., 2002), where one can interpret the model latent variables (low-dimensional factors) as gesture indicators. As a result background segmentation need not be viewed as an independent pre-processing step, but can be coupled in a consistent way with higher-level techniques.

5.6 Appendix

5.6.1 Implementation

Implementation of the EP algorithm in our model is based on "canonical" Gaussian potentials, which yield slightly easier formulation than the Gaussians represented in the usual moment form. The beliefs for hidden vectors $\mathbf{z}$ and $\mathbf{x}$ are represented as

$$\begin{aligned} q(\mathbf{z}) &= \psi(\mathbf{z}|\mathbf{h}_{\mathrm{z}}, \mathbf{G}_{\mathrm{z}}, g_{\mathrm{z}}) \\ q(\mathbf{x}) &= \psi(\mathbf{x}|\mathbf{h}_{\mathrm{x}}, \mathbf{G}_{\mathrm{x}}, g_{\mathrm{x}}), \end{aligned} \tag{5.12}$$

where $\psi()$ denotes a Gaussian potential, and $\mathbf{h}$, $\mathbf{G}$, g are canonical parameters (see Appendix 5.6.2). To reflect the assumption that Gaussian $q(\mathbf{x})$ has diagonal covariance, we constrain the canonical parameter $\mathbf{G}_{\mathrm{x}}$ to be diagonal.

The factor functions for our model follow from (5.5)–(5.7)

$$\begin{aligned} \phi_{\mathrm{A}}(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) = \psi(\mathbf{z}|\mathbf{0}, \mathbf{I}, d/2\log(2\pi)) \\ \phi_{\mathrm{B}}(\mathbf{z}, \mathbf{x}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mathbf{a}, \mathbf{R}) = \psi([\begin{smallmatrix}\mathbf{z}\\ \mathbf{x}\end{smallmatrix}]|\mathbf{h}_{\mathrm{B}}, \mathbf{G}_{\mathrm{B}}, g_{\mathrm{B}}) \\ \phi_{\mathrm{C}}(\mathbf{x}, \mathbf{y}) &= \prod_{k=1}^{K} [y_k x_k > 0], \end{aligned}$$

where ["text"] is an indicator function, that evaluates to 1 (or 0) if the binary proposition "text" is true (or false). The canonical parameters for potential ϕ_{B} are

$$\mathbf{G}_{\mathrm{B}} = \begin{bmatrix} \mathbf{W}^\top\mathbf{R}^{-1}\mathbf{W} & -\mathbf{W}^\top\mathbf{R}^{-1} \\ -\mathbf{R}^{-1}\mathbf{W} & \mathbf{R}^{-1} \end{bmatrix}, \qquad \mathbf{h}_{\mathrm{B}} = \begin{bmatrix} -\mathbf{W}^\top\mathbf{R}^{-1}\mathbf{a} \\ \mathbf{R}^{-1}\mathbf{a} \end{bmatrix}$$
$$g_{\mathrm{B}} = -\frac{1}{2}\log|2\pi\mathbf{R}| - \frac{1}{2}\mathbf{a}^\top\mathbf{R}\mathbf{a}.$$

and can be easily verified using (5.18)–(5.20) of Appendix 5.6.2.

The messages are represented as canonical Gaussians

$$m_i(\cdot) = \psi(\cdot|\mathbf{h}_i, \mathbf{G}_i, g_i), \text{ for } i = 1, 2, 3.$$

Importantly, the diagonal parameter $\mathbf{G}_{\mathrm{x}}$ of the belief $q(\mathbf{x})$ implies that the matrices $\mathbf{G}_1$ and $\mathbf{G}_2$, which parameterize messages incoming to $\mathbf{x}$ will be diagonal as well. (This can be seen from the generic update formula in Appendix 5.6.2.)

Initialization We initialize the messages to uniform potentials by setting the canonical parameters as $\mathbf{G}_i = 0, \mathbf{h}_i = 0, g_i = 0$ for $i = 1, 2, 3$. Initially, the belief $q(\mathbf{x})$ is uniform. During initialization, we send the message $m_0(\mathbf{z})$ to variable $\mathbf{z}$. As a result we have $q(\mathbf{z}) = \phi_A(\mathbf{z})$. The respective belief parameters are

$$\begin{array}{lll} \mathbf{G}_{\mathrm{x}} = 0 & \mathbf{h}_{\mathrm{x}} = 0 & g_{\mathrm{x}} = 0 \\ \mathbf{G}_{\mathrm{z}} = \mathbf{I} & \mathbf{h}_{\mathrm{z}} = 0 & g_{\mathrm{z}} = d/2\log(2\pi). \end{array}$$

Updates We first consider sending a message from factor B to variable $\mathbf{z}$. According to (5.10) we have to integrate

$$\psi(\left[\begin{smallmatrix}\mathbf{z}\\ \mathbf{x}\end{smallmatrix}\right], \mathbf{G}, \mathbf{h}, g) = \phi_B(\mathbf{x}, \mathbf{z}) \frac{q(\mathbf{x})q(\mathbf{z})}{m_1(\mathbf{x})m_3(\mathbf{z})} \tag{5.13}$$

over variable $\mathbf{x}$. The integrand turns out to be a Gaussian potential. We find its parameters by substituting the parametric beliefs and messages to (5.13), and using the fact the multiplication (or division) of Gaussian potentials corresponds to summation (or subtraction) of their canonical parameters

$$\begin{aligned}
\mathbf{G} &= \mathbf{G}_{\mathrm{B}} + \left[\begin{smallmatrix} 0 & 0 \\ 0 & \mathbf{G}_{\mathrm{x}} - \mathbf{G}_1 \end{smallmatrix}\right] + \left[\begin{smallmatrix} \mathbf{G}_{\mathrm{z}} - \mathbf{G}_3 & 0 \\ 0 & 0 \end{smallmatrix}\right] \\
\mathbf{h} &= \mathbf{h}_{\mathrm{B}} + \left[\begin{smallmatrix} 0 \\ \mathbf{h}_{\mathrm{x}} - \mathbf{h}_1 \end{smallmatrix}\right] + \left[\begin{smallmatrix} \mathbf{h}_{\mathrm{z}} - \mathbf{h}_3 \\ 0 \end{smallmatrix}\right] \\
g &= g_{\mathrm{B}} + g_{\mathrm{x}} - g_1 + g_{\mathrm{z}} - g_3.
\end{aligned}$$

As shown in Appendix 5.6.2, the integral evaluates to a Gaussian potential, denoted as $\tilde{q}(\mathbf{z}) = \frac{1}{Z}\psi(\mathbf{z}|\tilde{\mathbf{h}}, \tilde{\mathbf{G}}, \tilde{g})$. (see the Appendix 5.6.2 for the parameters.) We now move to (5.9). Since the potential $\tilde{q}(\mathbf{z})$ is already in the required parametric form we set $q^{\mathrm{new}}(\mathbf{z}) = \tilde{q}(\mathbf{z})$. Thus,

$$\begin{aligned}
\mathbf{G}_{\mathrm{z}}^{\mathrm{new}} &= \tilde{\mathbf{G}} & \mathbf{h}_{\mathrm{z}}^{\mathrm{new}} &= \tilde{\mathbf{h}} \\
g_{\mathrm{z}}^{\mathrm{new}} &= \bar{\mathrm{g}}(\mathbf{G}_{\mathrm{z}}^{\mathrm{new}}, \mathbf{h}_{\mathrm{z}}^{\mathrm{new}}) & \log Z &= \tilde{g} - g_{\mathrm{z}}^{\mathrm{new}},
\end{aligned}$$

where $\bar{\mathrm{g}}()$ is a normalizing term (see Appendix 5.6.2). Finally, we update the canonical parameters of the message $m_3(\mathbf{z}) = m_{\mathrm{B}\to\mathrm{z}}(\mathbf{z})$ according to (5.8). This is a fairly simple operation, as presented in Appendix 5.6.2.

Next, we consider sending message from factor B to variable $\mathbf{x}$, keeping in mind that the belief $q(\mathbf{x})$ is constrained to have diagonal matrix $\mathbf{G}_{\mathrm{x}}$. By substituting to (5.10) we find that the integrated expression is identical with (5.13), but now we integrate over variable $\mathbf{z}$. As shown in Appendix 5.6.2, the integral is a Gaussian potential, denoted as $\tilde{q}(\mathbf{x}) = \frac{1}{Z}\psi(\mathbf{x}|\tilde{\mathbf{h}}, \tilde{\mathbf{G}}, \tilde{g})$. Importantly, in general the matrix $\tilde{\mathbf{G}}$ will not be diagonal. According to (5.9) we find the closest (in KL-sense) potential to $\tilde{q}(\mathbf{x})$ with diagonal parameter $\mathbf{G}_{\mathrm{x}}^{\mathrm{new}}$. Such a potential can be easily found using the moment representation of Gaussians. Let $\tilde{\mathbf{V}} = \tilde{\mathbf{G}}^{-1}$, and $\tilde{\mathbf{m}} = \tilde{\mathbf{h}}\tilde{\mathbf{V}}$ be the mean and covariance of $\tilde{q}(\mathbf{x})$. It is a standard result (Cover and Thomas, 1991) that

$$(\mathbf{m}^*, \mathbf{D}^*) = \arg\min_{(\mathbf{m},\mathbf{D})} \mathrm{KL}\left(\mathcal{N}(\mathbf{x}|\tilde{\mathbf{m}}, \tilde{\mathbf{V}}) || \mathcal{N}(\mathbf{x}|\mathbf{m}, \mathbf{D})\right),$$

for diagonal covariances $\mathbf{D}$ is attained when $\mathbf{D}^* = \mathrm{diag}(\tilde{\mathbf{V}})$ and $\mathbf{m}^* = \tilde{\mathbf{m}}$ (i.e., $\mathbf{D}^*$ is the diagonal of $\tilde{\mathbf{V}}$). Thus, we have $q^{\mathrm{new}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\tilde{\mathbf{m}}, \mathrm{diag}(\tilde{\mathbf{V}}))$, or in the canonical parametrization:

$$\begin{aligned}
\mathbf{G}_{\mathrm{x}}^{\mathrm{new}} &= \left(\mathrm{diag}(\tilde{\mathbf{G}}^{-1})\right)^{-1} & \mathbf{h}_{\mathrm{x}}^{\mathrm{new}} &= \mathrm{diag}(\tilde{\mathbf{G}}^{-1})\tilde{\mathbf{m}} \\
g_{\mathrm{x}}^{\mathrm{new}} &= \bar{\mathrm{g}}(\mathbf{G}_{\mathrm{x}}^{\mathrm{new}}, \mathbf{h}_{\mathrm{x}}^{\mathrm{new}}) & \log Z &= \tilde{g} - g_{\mathrm{x}}^{\mathrm{new}}.
\end{aligned}$$

Note, that computing $\mathbf{G}_{\mathrm{x}}^{\mathrm{new}}$ and $\mathbf{h}_{\mathrm{x}}^{\mathrm{new}}$ should rely on the "matrix inversion lemma" for efficient computation of quantities dependent on the inverse of $\tilde{\mathbf{G}}$. The message $m_1(\mathbf{x}) = m_{\mathrm{B}\to\mathrm{x}}(\mathbf{x})$ is updated as in the Appendix 5.6.2.

Finally, we update the message $m_2(\mathbf{x})$ from factor C to variable $\mathbf{x}$. Recall first, that we assumed diagonal covariances for Gaussian potentials that represent the belief $q(\mathbf{x})$ and the message $m_2(\mathbf{x})$. Therefore, these potentials factorize into functions defined on individual terms x_k,

$$m_2(\mathbf{x}) = \psi(\mathbf{x}|\mathbf{G}_2, \mathbf{h}_2, g_2) = \prod_{k=1}^{K} \psi(x_k|G_{2,k}, h_{2,k}, g_{2,k})$$
$$q(\mathbf{x}) = \psi(\mathbf{x}|\mathbf{G}_{\mathrm{x}}, \mathbf{h}_{\mathrm{x}}, g_{\mathrm{x}}) = \prod_{k=1}^{K} \psi(x_k|G_{\mathrm{x}k}, h_{\mathrm{x}k}, g_{\mathrm{x}k}),$$

where $G_{\mathrm{x}k} = \mathbf{G}_{\mathrm{x}}(k,k)$ is the kth element on the diagonal of "inverse covariance" $\mathbf{G}_{\mathrm{x}}$, $h_{\mathrm{x}k} = \mathbf{h}(k)$ is the kth element in the vector $\mathbf{h}_{\mathrm{x}}$ and $g_{\mathrm{x}k} = g_{\mathrm{x}}/K$. Analogous parametrization holds for the message $m_2(\mathbf{x})$.

The factorial representation allows us to execute the update equations (5.8)–(5.10) individually for every pixel. Since y_k is fixed, the integration in (5.10) disappears and we have

$$\tilde{q}(x_k) = \frac{1}{Z_k}[x_k y_k > 0]\psi(x_k|h_k, G_k, g_k), \tag{5.14}$$

where $G_k = G_{\mathrm{x}k} - G_{2k}$, $h_k = h_{\mathrm{x}k} - h_{2k}$, and $g_k = g_{\mathrm{x}2} - g_{2k}$. When $y_k = 1$ then $\tilde{q}(x_k)$ is a Gaussian potential truncated to zero for $x_k \in (-\infty, 0)$. When $y_k = -1$, the the potential in truncated for $x_k \in (0, \infty)$.

We find the updated Gaussian belief, $q^{\mathrm{new}}(x_k)$, by minimizing the KL-divergence as indicated by the generic EP rule (5.9). Since Gaussian potentials belong to the exponential family, we find the minimizing potential by matching the moments of q^{new} with the moments of $\tilde{q}$ (see Chapter 2 or (Cover and Thomas, 1991))

$$\langle x_k \rangle_{q^{\mathrm{new}}} = \langle x_k \rangle_{\tilde{q}} \qquad\qquad \left\langle x_k^2 \right\rangle_{q^{\mathrm{new}}} = \left\langle x_k^2 \right\rangle_{\tilde{q}} \tag{5.15}$$

In order to solve the above system of equations we compute first moments of $\tilde{q}$. For this purpose we replace $\psi(x_k|h_k, G_k, g_k)$ with its counterpart in the usual moment form $\alpha_k \mathcal{N}(x_k|\mu_k, v_k)$, where $v_k = 1/G_k$, $\mu_k = h_k v_k$ and α_k is the normalization term. The moments follow from evaluating a one-dimensional Gaussian integral over positive (or negative) half-space. For $y_k = 1$ we compute

$$\begin{aligned}
\langle x_k \rangle_{\tilde{q}} &= \int_{-\infty}^{\infty} x_k \tilde{q}(x_k)\,\mathrm{d}x \\
&= \frac{1}{Z_k}\int_{-\infty}^{\infty} x_k [x_k y_k > 0]\psi(x_k|h_k, G_k, g_k)\,\mathrm{d}x \\
&= \frac{\alpha_k}{Z_k}\int_{0}^{\infty} x_k \mathcal{N}(x_k|\mu_k, v_k)\,\mathrm{d}x_k.
\end{aligned}$$

The terms Z_k and $\langle x_k^2 \rangle_{\tilde{q}}$ are computed analogously. For $y_k = -1$ we integrate over $(-\infty, 0)$. The Gaussians integrals evaluate to standard results, which we denote compactly as functions of y_k

$$\langle x_k | y_k \rangle_{\tilde{q}} = \mu_k + \frac{y_k \eta}{\lambda(y_k)} \qquad \langle x_k^2 | y_k \rangle_{\tilde{q}} = \mu_k^2 + v_k + \frac{y_k \mu_k}{\lambda(y_k)}, \tag{5.16}$$

where

$$\eta = \exp(-\mu_k^2 / 2v_k) \sqrt{v_k / 2\pi} \qquad \lambda(y) = (1 + r)/2 - y/2 \, \text{erfc}(\mu_k / \sqrt{2v_k}).$$

The normalization constant evaluates to $Z_k(y_k) = \alpha_k \lambda(y_k)$. We now express the moments $\langle x_k \rangle_{q^{\text{new}}}$, $\langle x_k^2 \rangle_{q^{\text{new}}}$ using the canonical parameters of belief $q^{\text{new}}(x_k)$

$$\begin{aligned} \langle x_k \rangle_{q^{\text{new}}} &= \mathbf{h}_{\text{x}}^{\text{new}}(k) / \mathbf{G}_{\text{x}}^{\text{new}}(k,k) \\ \langle x_k^2 \rangle_{q^{\text{new}}} &= (\mathbf{h}_{\text{x}}^{\text{new}}(k) / \mathbf{G}_{\text{x}}^{\text{new}}(k,k))^2 + 1 / \mathbf{G}_{\text{x}}^{\text{new}}(k,k). \end{aligned} \tag{5.17}$$

Substituting (5.16) and (5.17) to (5.15) yields the sought parameters

$$\mathbf{G}_{\text{x}}^{\text{new}}(k,k) = \frac{1}{\langle x_k^2 | y_k \rangle_{\tilde{q}} - \langle x_k | y_k \rangle_{\tilde{q}}^2}, \qquad \mathbf{h}_{\text{x}}^{\text{new}}(k) = \frac{\langle x_k | y_k \rangle_{\tilde{q}}}{\langle x_k^2 | y_k \rangle_{\tilde{q}} - \langle x_k | y_k \rangle_{\tilde{q}}^2}.$$

Repeating this steps for all pixels $k = 1, \ldots, K$ yields the diagonal matrix $\mathbf{G}_{\text{x}}^{\text{new}}$ and vector $\mathbf{h}_{\text{x}}^{\text{new}}$. To obtain a normalized belief we set $g_{\text{x}}^{\text{new}} = \bar{\text{g}}(\mathbf{G}_{\text{x}}^{\text{new}}, \mathbf{h}_{\text{x}}^{\text{new}})$. Given the new belief $q^{\text{new}}(\mathbf{x})$ we find the updated message $m_2(\mathbf{x})$ using the standard equations (Appendix 5.6.2), where the joint normalization term Z is

$$\log Z = \sum_{k=1}^{K} \log Z_k.$$

Solution After EP converges, we can find the expected mask coordinates by converting the belief $q(\mathbf{z})$ to the moment form; $\langle \mathbf{z} | \mathbf{y} \rangle = \mathbf{G}_{\text{z}}^{-1} \mathbf{h}_{\text{z}}$. The mask likelihood corresponds to the overall normalization constant

$$\log p(\mathbf{y}) = \sum_{n}^{\text{EP iter}} \sum_{i=1}^{3} \log Z^{(n,i)},$$

where $Z^{(n,i)}$ denotes the normalization constant obtained during the nth EP iteration when sending message $m_i(\cdot)$.

5.6.2 Canonical Gaussian potential

Definition A canonical Gaussian potential of a random, d-dimensional vector $\mathbf{s}$ is defined as

$$\psi(\mathbf{s}|\mathbf{h},\mathbf{G},g) \equiv \exp(\mathbf{s}\mathbf{h}^\top - \frac{1}{2}\mathbf{s}\mathbf{G}\mathbf{s}^\top + g), \tag{5.18}$$

where *canonical* parameters are: $d\times 1$ vector $\mathbf{h}$, $d\times d$ matrix $\mathbf{G}$, and scalar g. The canonical form becomes equivalent to the moment form $\alpha\mathcal{N}(\mathbf{s}|\mathbf{m},\mathbf{V})$, when

$$\mathbf{G} = \mathbf{V}^{-1} \qquad g = \log(\alpha) + \bar{\mathrm{g}}(\mathbf{G},\mathbf{h}) \tag{5.19}$$

$$\mathbf{h} = \mathbf{V}^{-1}\mathbf{m} \qquad \bar{\mathrm{g}}(\mathbf{G},\mathbf{h}) = \frac{1}{2}\log|\frac{\mathbf{G}}{2\pi}| - \frac{1}{2}\mathbf{h}^\top\mathbf{G}^{-1}\mathbf{h} \tag{5.20}$$

with $\mathbf{m}$ and $\mathbf{V}$ denoting respectively, the mean and the covariance. When the potential is normalized, $\alpha = 1$, then $\bar{\mathrm{g}}(\mathbf{G},\mathbf{h})$ becomes the normalization constant for the canonical form. The canonical notation is useful for expressing uniform potentials $\psi(\mathbf{s}|\mathbf{0},\mathbf{0},0) = 1$ and for multiplication $\psi(\mathbf{s}|\mathbf{h}_1,\mathbf{G}_1,g_1)\psi(\mathbf{s}|\mathbf{h}_2,\mathbf{G}_2,g_2) = \psi(\mathbf{s}|\mathbf{h}_1+\mathbf{h}_2,\mathbf{G}_1+\mathbf{G}_2,g_1+g_2)$. In case of division the sums are replaced with subtractions.

Marginalization The following result is useful for deriving the EP updates with linear Gaussian family. Suppose we want to marginalize a joint Gaussian potential, given in the canonical form

$$\psi(\mathbf{z}|\tilde{\mathbf{G}},\tilde{\mathbf{h}},\tilde{g}) = \int \psi(\left[\begin{smallmatrix}\mathbf{z}\\ \mathbf{x}\end{smallmatrix}\right] | \left[\begin{smallmatrix}\mathbf{h}_1\\ \mathbf{h}_2\end{smallmatrix}\right], \left[\begin{smallmatrix}\mathbf{G}_{11} & \mathbf{G}_{12}\\ \mathbf{G}_{21} & \mathbf{G}_{22}\end{smallmatrix}\right], g)\,\mathrm{d}\mathbf{x}.$$

The canonical parameters of the marginal potential are

$$\tilde{\mathbf{G}} = \mathbf{G}_{11} - \mathbf{G}_{12}\mathbf{G}_{22}^{-1}\mathbf{G}_{21} \qquad \tilde{\mathbf{h}} = \mathbf{h}_1 - \mathbf{G}_{12}\mathbf{G}_{22}^{-1}\mathbf{h}_2$$
$$\tilde{g} = g - \frac{1}{2}\log|\frac{\mathbf{G}_{22}}{2\pi}| + \frac{1}{2}\mathbf{h}_2^\top\mathbf{G}_{22}^{-1}\mathbf{h}_2.$$

Analogous result holds for the integration of $\mathbf{x}$.

Message update

Canonical parametrization allows simple implementation of the generic message update formula, given by (5.8). Let $q(v) = \psi(v|\mathbf{G},\mathbf{h},g)$ and $m_{f\to v}(v) = \psi(v|\mathbf{G}_\mathrm{m},\mathbf{h}_\mathrm{m},g_\mathrm{m})$. The parameters of the updated message are:

$$\mathbf{G}_\mathrm{m}^\mathrm{new} = \mathbf{G}^\mathrm{new} - \mathbf{G} + \mathbf{G}_\mathrm{m} \qquad \mathbf{h}_\mathrm{m}^\mathrm{new} = \mathbf{h}^\mathrm{new} - \mathbf{h} + \mathbf{h}_\mathrm{m}$$
$$g_\mathrm{m}^\mathrm{new} = \log Z + g^\mathrm{new} - g + g_\mathrm{m}.$$

5.6.3 Probabilistic foreground images

The message-passing scheme applies quite naturally to problems where the information about mask $\mathbf{y}$ is given by a distribution $q(y_k)$ on individual indicators, $y_k \in \{1, -1\}$. In this case, the EP inference algorithm requires modifications only to the message passing operation between factor C and the variable $\mathbf{x}$.

We modify the message passing operation for $m_2(\mathbf{x})$ and updating belief $q(\mathbf{x})$ as follows. We begin with the equation (5.14), which now takes the form

$$\tilde{q}(x_k) = \frac{1}{Z_k} \sum_{y_k \in \{1,-1\}} [x_k y_k > 0] \psi(x_k | h_k, G_k, g_k) q(y_k),$$

where we integrate y_k because $y_k \in \{1, -1\}$ is now a random variable with a belief $q(y_k)$. The term $\tilde{q}(x_k)$ is a mixture of two Gaussian potentials truncated, respectively, in the negative and the positive half-space. We find the updated belief $q^{\text{new}}(x_k)$ in the Gaussian family by matching the moments of $q^{\text{new}}(x_k)$ with the moments of $\tilde{q}(x_k)$. Below we compute the moments $\langle x_k^2 \rangle_{\tilde{q}}$ and $\langle x_k^2 \rangle_{\tilde{q}}$. The term $\tilde{q}(x_k)$ can be expressed as

$$\tilde{q}(x_k) = \sum_{y_k \in \{1,-1\}} q(y_k) \tilde{q}(x_k | y_k),$$

where $\tilde{q}(x_k|y_k) = \frac{1}{Z(y_k)} [x_k y_k > 0] \psi(x_k | h_k, G_k, g_k)$ denotes a normalized truncated Gaussian potential. The moments of interest become

$$\langle x_k \rangle_{\tilde{q}} = \sum_{y_k \in \{1,-1\}} q(y_k) \langle x_k | y_k \rangle_{\tilde{q}} \qquad \langle x_k^2 \rangle_{\tilde{q}} = \sum_{y_k \in \{1,-1\}} q(y_k) \langle x_k^2 | y_k \rangle_{\tilde{q}},$$

where $\langle x_k | y_k \rangle_{\tilde{q}}$ and $\langle x_k^2 | y_k \rangle_{\tilde{q}}$ are given by (5.16). The normalization term is

$$Z_k = \sum_{y_k \in \{1,-1\}} q(y_k) Z_k(y_k) = \sum_{y_k \in \{1,-1\}} q(y_k) \lambda(y_k) \alpha_k,$$

where $\lambda(y_k)\alpha_k$ are the same as in the case when y_k is known. Given these moments, the update equations are identical with the case when y_k is known.

CHAPTER 6

CONCLUSIONS

The thesis addressed several problems distinctive for visual surveillance in spatially wide areas. Such areas are characterized by the fact that even multiple cameras cannot exhaustively cover the entire region under surveillance. Therefore surveillance employs several cameras that are spatially distributed, i.e., cameras with fields of view covering disjoint scenes scattered over the global region of interest. The primary problem involved tracking multiple objects (humans) with such distributed camera systems. In this problem we focused on techniques for maintaining a trajectory of an object when it temporarily disappeared from the view of a camera, due to the visual gap between the disjoint scenes. The secondary problem involved characterization of appearance of objects in the video data provided by the cameras. Here, our emphasis was on proper estimation of image regions that represent humans.

Both problems have been studied in a single methodological framework of probabilistic graphical models. In either case, our approach included definition of an appropriate probabilistic model. The model encodes dependencies between various known and unknown quantities relevant to a given problem. Given a model, we estimated the unknown quantities of interest by implementing a suitable probabilistic inference algorithms. The inference algorithms considered in the thesis can all be viewed as instances of the general class of belief propagation techniques.

6.1 Summary of conclusions

In Chapter 3 we studied the first of the mentioned problems, where the key task is to partition a set of observations of different people from various cameras into clusters, where a cluster represented a single person. We constructed an iterative optimization procedure (the EM algorithm) that embeds greedy search for the most likely partition (in the E step) with estimation of parameters of a probabilistic model (in the M step). The experiments showed that in the indoor tracking domain, such greedy search algorithm is much more efficient than the algorithm that employs MCMC sampling to find the optimal partition. The sampling-based method turned out two-orders of magnitude

slower than the developed method, when finding a solution of similar quality. This property can be explained by the fact that the compared sampling-based method exploits strong constraints on the motion of objects (e.g. vehicles, which can follow only roads). In indoor scenario, human motion typically lacks such strong constraints.

In Chapter 4 we considered the same problem as in Chapter 3. We developed a probabilistic model that allowed replacing explicit search in the intractable space of partitions with an approximate inference method — assumed-density filtering (ADF). In this way we applied a principled Bayesian framework for approximating solution to the intractable multi-object tracking problem. Moreover, the model combines ideas inspired by machine learning (Dirichlet processes) to properly characterize the number of different persons in the environment. Finally, we have shown how the parameters – mean and covariance – of popular Gaussian noise models can be estimated by Bayesian inference under data association ambiguity. The principled approximation techniques yielded a stable algorithm that avoids degenerate solutions in difficult tracking conditions. For example, when all motion constraints have been discarded from the problem or when appearance of objects has been neglected, the algorithm still correctly estimated the number of objects (while alternative algorithms overestimate this number by more than 100%).

The basic functional difference between the methods of Chapter 3 and 4 are different assumptions on the tracking application and the tracking environment. In Chapter 3 parameters of the environment (travel times) are assumed constant and unknown. In Chapter 4 these parameters are assumed known and provided by the user. Accordingly the algorithm of Chapter 3 learns the parameters from the data. The algorithm is an instance of unsupervised learning procedure that does not require any prior training data, since learning takes place at the same time as tracking. In contrast, the algorithm of Chapter 4 relies on the user who has to supply the minimum travel times. The parameters usually will follow from the prior knowledge about the camera setup; however the parameters might also be estimated with an independent learning method.

A more fundamental difference follows from the approximation technique applied to deal with the intractability of data association in multi-object tracking problems. In both formulations, data association boils down to probabilistic inference on hidden continuous state variables and hidden discrete partitions (assignments of observations to trajectories). The method of Chapter 3 performs approximate inference on the discrete partition space, and exact (conditional) inference on continuous states. In the Chapter 4 a partition is represented as a sequence of discrete labels. Here, the tracking methods performs approximate inference on the continuous states and exact (conditional) inference on the discrete labels. The approximation scheme of Chapter 3 is simpler and easier to implement. However, the explicit search procedure for the optimal partition seems hard to improve. The approximation scheme of Chapter 4 opens many more possibilities for improving the accuracy of approximation by employing more flexible approximation families or employing potentially better approximation techniques (like expectation-propagation or variational mean-field).

In Chapter 5 we considered a scene segmentation problem, where pixels of an image have to be assigned a binary label that indicates whether a pixel represents a (static) background scene or a (moving) foreground object. We have shown that by incorporating prior knowledge about correlations of pixel labels typical for some scene, segmentation algorithms can handle difficult problems, such as cases where the background scene is difficult to distinguish from a foreground object. We viewed the labels as random variables and presented a prior probability distribution (a probabilistic model) that facilitates flexible representation and learning of long-range spatial correlations between the labels. Based on exemplary data, the prior distribution can be easily adjusted in order to incorporate knowledge about label couplings typical for some class of foreground objects (e.g. human silhouettes). Such a learned prior results in more accurate segmentation than alternative prior models, which assume generic short-range correlations of pixel labels. The improved accuracy can be observed according to various criteria (like the number of disconnected regions that represent a foreground object).

6.2 Future research

As indicated in Chapter 1, typical visual surveillance systems analyze video data at various levels of abstraction. Often the results of the analysis at lower levels (like background segmentation) are not of interest themselves but become an input to higher-level tasks (like object tracking). An interesting future research task is to merge the presented probabilistic models into a single model that would encapsulate the low and high level analytical tasks. Such a model would allow to properly characterize uncertainties in the quantities passed between various levels. On the downside, inference in the unified model requires dealing with potentially large number of hidden variables. The discussed message-passing algorithms usually scale well with the size of the model since computation of a belief is always local, in the sense that it involves a limited subset of other beliefs (consider dynamic Bayesian networks as an example of large-scale models, where inference is efficient). Therefore, a promising approach for inference in the unified model is by representing the model as a factor graph and applying one of the message-passing algorithms.

Another research direction is to consider non-homogeneous sensors, like cameras and radars, and develop a probabilistic model that would encapsulate measurements from all sensors. Such non-homogeneous systems are already subject of intense research, mainly in the area of video and audio fusion. However, at present processing measurements from different sensors occurs at independent modules, and only the high-level "events" are fused. Probabilistic generative models allow to view an "event" as a random variable that generates various types of measurements in different sensors at the same time. Inference in such models would immediately merge contributions from different sensors and avoided the need to develop customized merging techniques.

SUMMARY

Visual surveillance systems steadily find they way into our daily lives. Either for security enforcement, congestion monitoring or daily assistance for elderly, cameras are common in schools, homes, airports, shopping centers and other public areas. For example, in UK alone there are estimated 4.2 million CCTV cameras already installed (McCahil and Norris, 2003). According to a study (Norris and Armstrong, 1999) an average person on a busy day in an urban environment could have its image captured by up to 300 cameras.[1]

Progress in communication technology and communication infrastructure allows to develop networks of multiple surveillance cameras. Compared to single-camera systems, such networks provide greater flexibility and wider observation scope. In particular, one may setup a surveillance system for a wide area (like an airport) by focusing individual cameras on distinct, relatively narrow fields of view and merging information from different cameras. A direct consequence of the technological advances in camera networks is an increased need for *intelligent video analysis techniques*. Such techniques automatically interpret video data from individual cameras and fuse data from multiple cameras in order to provide access to global information about events taking place in the considered wide area.

This thesis presents various video analysis techniques that allow automated tracking of multiple persons with spatially distributed cameras. The techniques analyze video data at various levels of abstraction. At the low level, we considered the problem of interpreting individual pixels in video frames in order to detect pixels that represent humans. At a higher-level, we considered the problem of recognizing appearances of the same person from different, spatially distributed cameras. A key challenge, common to both tasks, is the uncertainty of video measurements. For example, when detecting which pixels represent a person one has to deal with the fact that the appearance of the person is similar to some other objects. When recognizing a person one has to deal with the fact that the same person observed at different cameras will look slightly different due to differences in illuminating light, body pose and camera viewing angle.

A characteristic feature of the presented techniques is that they are developed within a single methodological framework of Bayesian networks. This framework has been

[1] Estimates for United Kingdom in 1999.

described in Chapter 2. Bayesian networks provide an elegant tool for dealing with uncertain knowledge, where the input quantities are represented as observed random variables, and the sought output quantities as hidden random variables. The relation between the hidden and the observed variables is encoded in a probability distribution or a so-called probabilistic model. Essentially, in this formalism a given problem is converted to a Bayesian inference problem and solved with Bayesian inference techniques. An appealing property of this approach is that Bayesian networks allow to solve such inference problems to a certain extent automatically.

Chapter 3 and 4 are focused on the high level part of the system. The task of the high level subsystem is to maintain person's identity when he or she leaves the field of view of one camera and later on appears at some other camera. Since the cameras in our system are sparsely distributed, their fields of view are disjoint. Consequently, when a person leaves the field of view we temporarily lose track of it. When the person appears at some other we have to re-identify the person. In this way the system maintains global trajectories of people despite sparse camera locations. The system relies on two types of cues. First, appearances of the same person at various cameras are expected to be similar (although not identical). Second, motion of a person from one camera to some other is constrained by various environment parameters, like minimum travel time.

The thesis presents two probabilistic data association algorithms that gather appearance of the same person from different cameras into a single trajectory. The method developed in Chapter 3 solves the association problem and at the same time tries to estimate environment parameters. This approach can be viewed as an extension of the classical "multiple hypothesis" tracking algorithm. The extension allows to adapt the algorithm to changing environment parameters (e.g., in the rush hours the travel times between some cameras might increase). In contrast, the method developed in Chapter 4 assumes known environment parameters. This online algorithm is based on Bayesian inference in Infinite Gaussian Mixture Models (also known as Dirichlet Process Mixture Models). The emphasis is on accurate estimation of the number of distinct persons and correct association in difficult environments (e.g., characterized by very large illumination variations between cameras).

In Chapter 5 a low level video analysis system is presented. The tasks of the low level system include segmentation of video frame into background and foreground regions and detecting humans. The basic assumption here is that the camera and the background scene remain static. In this case a person passing through the scene can be detected by finding pixels that deviate from the probabilistic model of the background scene. The static background is assumption fairly common, and there exist a variety of methods for object detection and local tracking. However, many of these methods consider individual pixels independent of each other, and thus often provide incomplete or rough object shapes. We have developed a probabilistic model (Clipped Factor Analysis model) of human silhouettes that incorporates pixel correlations characteristic for human shapes. It this way our model allows more accurate detection of pixels that represent a person in a video frame.

The Bayesian approach to video analysis considered in the thesis offers several benefits. First, experimental results indicate improvements in terms of speed or accuracy over existing methods. Results show that our methods are much more robust to illumination conditions, have a better performance in tracking and scale much better with the number of cameras and the number of humans in the system. Second, the Bayesian approach constitutes a generic paradigm for constructing more elaborate video analysis systems that couple the inference at the low and high levels in a principled way. A potential advantage of such systems lies in clarity of formulations and proper characterization of uncertainties in the results of analysis at intermediate levels.

SAMENVATTING

Bewakingscamera's worden steeds meer toegepast in onze samenleving, bijvoorbeeld voor beveiliging, verkeerstoezicht of als hulpmiddel in de ouderenzorg. Camera's zijn op grote schaal te vinden in scholen, huizen, luchthavens, winkelcentra en andere publieke ruimtes. In het Verenigd Koninkrijk alleen al zijn naar schatting 4.2 miljoen bewakingscamera's in gebruik (McCahil and Norris, 2003). Volgens onderzoek (Norris and Armstrong, 1999), zal een gemiddelde persoon door z'n 300 camera's op een drukke dag in stedelijke omgeving geobserveerd kunnen worden.[2]

Innovaties in communicatie technologie en infrastructuur maken het mogelijk om meerdere camera's in een netwerk te bundelen. In vergelijking met een enkele camera bieden zulke netwerken meer flexibiliteit. Als gevolg hiervan groeit de behoefte aan intelligente video analyse methoden die de beelden van individuele camera's analyseren, en de informatie van meerdere camera's op een slimme manier combineren.

Het proefschrift dat voor u ligt beschrijft een aantal video analyse technieken voor het volgen van verschillende personen door meerdere camera's. De beschreven methoden analyseren de videobeelden op verschillende abstractieniveaus. Op het laagste niveau is worden individuele pixels geclassificeerd als persoon of achtergrond. Een hoger niveau behandelt het probleem van het herkennen van dezelfde persoon in de beelden van camera's die zich op verschillende locaties bevinden. Beide taken worden gekenmerkt door dezelfde uitdaging, het omgaan met de onzekerheid van de camera waarnemingen. Bijvoorbeeld, methoden om personen te detecteren moeten rekening houden met het feit dat een persoon er hetzelfde uit kan zien als andere objecten. Verder hebben technieken om personen opnieuw te herkennen te maken met verschillende lichtomstandigheden, houdingen en camera standpunten.

Alle gepresenteerde technieken zijn ontwikkeld binnen hetzelfde Bayesiaanse netwerken formalisme, beschreven in hoofdstuk 2. Bayesiaanse netwerken vormen een elegant hulpmiddel voor het omgaan met onzekere informatie, waarin de invoer gepresenteerd wordt als zichtbare variabelen, en de gewenste uitvoer als verborgen variabelen. De relatie tussen beide typen variabelen is probabilistisch, en wordt gerepresenteerd als een kansverdeling. Binnen dit formalisme wordt een gegeven probleem beschouwd als

[2]Schatting voor het Verenigd Koninkrijk in 1999.

een Bayesiaans inferentie probleem, dat opgelost kan worden met bijbehorende standaardtechnieken. Een aantrekkelijk voordeel is Bayesiaanse netwerken het mogelijk maken zulke inferentie problemen tot op zekere hoogte automatisch op te lossen.

Hoofdstuk 3 en 4 gaan over het hogere niveau van het systeem: het volgen van meerdere personen met camera's op verschillende locaties. Het doel is hier om een persoon die uit het gezichtsveld van een camera loopt en later bij een andere camera weer verschijnt te herkennen. Aangezien de camera's zich op verschillende locaties bevinden zijn hun gezichtsvelden gescheiden, met als gevolg dat wanneer een persoon het gezichtsveld verlaat we hem niet meer kunnen volgen. Wanneer de persoon terug in het beeld van een camera komt moeten we hem of haar opnieuw identificeren. Op deze manier is het mogelijk globale trajecten te berekenen, ondanks het feit dat de camera's slechts op enkele plaatsen staan. Het systeem is gebaseerd op twee typen kenmerken. Ten eerste, het verwacht dat de beelden van dezelfde persoon bij verschillende camera's op elkaar lijken (maar hoeven niet identiek zijn). Ten tweede zijn de mogelijke bewegingen van een persoon tussen camera locaties beperkt door verscheidene omgevingsfactoren, zoals bijvoorbeeld de minimale tijd die nodig is om van de ene naar de andere camera locatie te bewegen.

Dit proefschrift presenteert twee probabilistische algoritmes om globale trajecten te berekenen. De methode beschreven in hoofdstuk 3 berekent deze trajecten en probeert tegelijkertijd de relevante omgevingsfactoren te schatten. De techniek kan gezien worden als een uitbreiding van het klassieke "multiple hypothesis tracking" algoritme, die zich kan aanpassen aan een veranderende omgeving (bijvoorbeeld tijdens de spits kan de reistijd tussen twee camera's toenemen). De techniek beschreven in hoofdstuk 4 daarentegen gaat uit van bekende omgevingsfactoren die niet veranderen. Dit online algoritme is gebaseerd op Bayesiaanse inferentie in Infinite Gaussian Mixture modellen. De nadruk ligt hier op het nauwkeurige schatten van het aantal verschillende personen in moeilijke omstandigheden (bijvoorbeeld wanneer er grote verschillen bestaan in de lichtomstandigheden tussen camera's.

Hoofdstuk 5 beschrijft een video analyse methode voor het lage niveau van het systeem, waarin een videobeeld gescheiden wordt in voor- en achtergrond en personen gedetecteerd worden. De belangrijkste aanname hier is dat de camera en de achtergrond statisch blijven. Een persoon wordt gedetecteerd als de pixels die afwijken van het probalistische achtergrondmodel. De aanname van een statische achtergrond is redelijk gangbaar, en er bestaan vele methoden voor het detecteren en volgen van objecten. De meeste methoden bekijken echter losse pixels onafhankelijk van elkaar, met incomplete en grillige vormen als gevolg. Wij hebben een probabilistische model van menselijke silhouetten ontwikkeld dat rekening houdt met pixel correlaties kenmerkend voor menselijke vormen. Op deze manier maakt ons model een nauwkeurigere detectie mogelijk van pixels die een persoon representeren.

De Bayesiaanse aanpak van video analyse behandeld in dit proefschrift biedt verscheidene voordelen. Ten eerste laten experimentele resultaten aanzienlijke verbeteringen zien qua snelheid en nauwkeurigheid over bestaande methoden. De resultaten tonen

ook aan dan onze methoden robuuster zijn met betrekking tot veranderingen in de lichtomstandigheden en zijn beter in het volgen van personen. Ten tweede is de Bayesiaanse aanpak een erg generiek formalisme voor het bouwen van meer ingewikkelde video analyse systemen, waarbinnen analyse op lagere en hogere niveaus op een consistente manier gekoppeld kunnen worden.

Translated by Mathijs Spaan.

BIBLIOGRAPHY

S. M. Aji and R. J. McEliece (2000). The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43.

C. E. Antoniak (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174.

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp (2002). A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Y. Bar-Shalom and T. E. Fortmann (1988). *Tracking and Data Association*. Academic Press.

Y. Bar-Shalom and X.-R. Li (1993). *Estimation and Tracking: Principles, Techniques and Software*. Artech House.

M. Beal, Z. Ghahramani, and C. Rasmussen (2002). The infinite hidden Markov model. In *Advances in Neural Information Processing Systems 14*.

J. Bilmes (1997). A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Icsi-tr-97-021, University of Berkeley.

A. Blake, M. Isard, and D. Reynard (1995). Learning to track the visual motion of contours. *Artificial Intelligence*, 78(1–2):179–212.

X. Boyen and D. Koller (1998). Tractable inference for complex stochastic processes. In *Uncertainty in Artificial Intelligence*, pages 33–42. Morgan Kaufman.

L. Brown (1986). *Fundamentals of statistical exponential families*. Institute of Mathematical Statistics, Hayward, USA.

H. H. Bui, S. Venkatesh, and G. West (2001). Tracking and surveillance in wide-area spatial environments using the abstract hidden Markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):177–197.

Q. Cai (1997). *Tracking Human Motion in Indoor Environments Using Distributed-Camera System*. Ph.D. thesis, University of Texas at Austin.

Q. Cai and J. K. Aggarwal (1999). Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241–1247.

A. T. Cemgil, W. Zajdel, and B. J. A. Kröse (2005). A hybrid graphical model for robust feature extraction from video. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

D. Chandler (1987). *Introduction to modern statistical mechanics*. Oxford University Press.

R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade (2001). Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477.

T. M. Cover and J. A. Thomas (1991). *Elements of Information Theory*. John Wiley & Sons, New York.

I. J. Cox (1993). A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66.

I. J. Cox and S. L. Hingorani (1994). An efficient implementation and evaluation of Reid's multiple hypothesis tracking algorithm for visual tracking. In *IEEE Int. Conf. on Pattern Recognition*.

T. Dean and K. Kanazawa (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150.

A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via EM algorithm. *Journal of the Royal Statistical Society*, B 39(1):1–38.

S. Dockstader and A. M. Tekalp (2001). Multiple camera tracking of interacting and occluded motion. *Proceedings of IEEE*, 89(10):1441–1455.

A. Doucet, N. de Freitas, and N. Gordon, editors (2001). *Sequential Monte Carlo Methods in Practice*. Springer Verlag.

M. Drew, J. Wei, and Z. Li (1998). Illumination-invariant color object recognition via compressed chromaticity histograms of color-channel-normalized images. In *Int. Conf. on Computer Vision*, pages 533–540.

W. T. Freeman, E. C. Pasztor, and O. T. Carmichael (2000). Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47.

B. Frey and N. Jojic (2005). A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1392–1416.

N. Friedman (1998). The Bayesian structural EM algorithm. In *Uncertainty in Artificial Intelligence*.

N. Friedman and S. Russell (1997). Image segmentation in video sequences. A probabilistic approach. In *Uncertainty in Artificial Intelligence*.

D. Gavrila (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98.

D. Gavrila and L. Davis (1996). 3-d model-based tracking of humans in action: a multi-view approach. In *IEEE Computer Vision and Pattern Recognition*, pages 73–80.

D. Gavrila and J. Giebel (2001). Virtual sample generation for template-based shape matching. In *IEEE Computer Vision and Pattern Recognition*, pages 676–681.

A. Gelman, J. Carlin, H. Stern, and D. Rubin (1995). *Bayesian Data Analysis*. Chapman & Hall.

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

Z. Ghahramani (2001). An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors (1996). *Markov Chain Monte Carlo in Practice*. CRC Press.

P. Giudci and R. Castelo (2001). Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50:127–158.

G. D. Hager and P. Belhumeur (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.

I. Haritaoglu, D. Harwood, and L. S. Davis (2000). W^4: Real-time surveillance of people and their activities. *IEEE Transactions on Paterrn Recognition and Machine Intelligence*, 22(8):809–830.

T. Heskes (2003). Stable fixed points of loopy belief propagation are local minima of the Bethe free energy. In *Advances in Neural Information Processing Systems 15*.

T. Heskes and O. Zoeter (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. In *Uncertainty in Artificial Intelligence*.

T. Heskes, O. Zoeter, and W. Wiegerinck (2004). Approximate expectation maximization. In *Advances in Neural Information Processing Systems 16*.

G. Hinton and T. Sejnowski (1986). Learning and relearning in Boltzmann machines. In D. Rumelhart and J. McCelland, editors, *Parallel distributed processing*. MIT Press.

T. Huang and S. Russell (1998). Object identification: A Bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103(1–2):1–17.

C. Hue and J.-P. L. Cadre (2002). Sequential Monte Carlo methods for multiple target tracking and data fusion. *IEEE Transactions on Signal Processing*, 50(2):309–325.

M. Isard and A. Blake (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28.

T. S. Jaakkola (2001). Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced mean field methods*. MIT Press.

D. Jang, G. Kim, and H. Choi (1997). Model-based tracking of moving object. *Pattern Recognition*, 30(6):999–1008.

O. Javed, Z. Rasheed, K. Shafique, and M. Shah (2003). Tracking across multiple cameras with disjoint views. In *IEEE Int. Conf. on Computer Vision*, pages 952–957.

F. V. Jensen (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York.

M. Jerrum and A. Sinclair (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS Publishing.

N. Jojic, B. Frey, and A. Kannan (2003). Epitomic analysis of appearance and shape. In *IEEE International Conference on Computer Vision*.

M. I. Jordan, editor (1998). *Learning in graphical models*. MIT Press.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1998). An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press.

J. Kato, S. Joga, J. Rittscher, and A. Blake (2002). An HMM-based segmentation method for traffic monitoring movies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1291–1296.

V. Kettnaker and R. Zabih (1999). Bayesian multi-camera surveillance. In *IEEE Computer Vision and Pattern Recognition*, pages 2253–2261.

D. Koller, J. W. Weber, and J. Malik (1994). Robust multiple car tracking with occlusion reasoning. In *European Conf. on Computer Vision*, pages 186–196.

B. J. A. Kröse, N. Vlassis, and W. Zajdel (2004). Bayesian methods for tracking and localization. In *Philips Symposium On Intelligent Algorithms*, pages 27–38.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

S. L. Lauritzen (1996). *Graphical Models*. Clarendon Press, Oxford.

D. D. Lee and H. Sompolinsky (1998). Learning a continuous hidden variable model for binary data. In *Advances in Neural Information Processing Systems*.

U. Lerner and R. Parr (2001). Inference in hybrid networks: Theoretical limits and practical algorithms. In *Uncertainty in Artificial Intelligence*, pages 310–318.

Y. Li, A. Hilton, and J. Illingworth (2002). A relaxation algorithm for real-time multiple view 3d-tracking. *Image and Vision Computing*, 20(12):841–859.

J. Lim and D. Kriegman (2004). Tracking humans using prior and learned representations of shape and appearance. In *IEEE Conf. on Face and Resture Recognition*, pages 869–874.

D. J. MacKay (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

M. McCahil and C. Norris (2003). Estimating the extent, sophistication and legality of CCTV in london. In M. Gill, editor, *CCTV*. Perpetuity Press.

T. Minka (2001a). The EP energy function and its minimization schemes. Technical report, MIT Media Lab.

T. Minka (2001b). *Expectation Propagation for approximate Bayesian inference*. Ph.D. thesis, MIT.

T. Minka and Z. Ghahramani (2003). Expectation propagation for infinite mixtures. In *Advances in Neural Information Processing Systems*. Workshop on Nonparametric Bayesian Methods and Infinite Models.

T. Minka and J. Lafferty (2002). Expectation propagation for the generative aspect model. In *Uncertainty in Artificial Intelligence*.

K. Murphy (1998). Switching Kalman filters. Technical report, University of Berkeley.

K. Murphy (2001). Learning Bayes net structure from sparse data sets. Technical report, University of Berkeley.

K. Murphy (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

K. Murphy, Y. Weiss, and M. I. Jordan (1999). Loopy-belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*.

R. M. Neal and G. E. Hinton (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press.

A. E. Nicholson and J. M. Brady (1994). Dynamic belief networks for discrete monitoring. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11).

C. Norris and G. Armstrong, editors (1999). *The Maximum Surveillance Society: The Rise of CCTV*. Oxford:Berg.

H. Pasula, S. Russell, M. Ostland, and Y. Ritov (1999). Tracking many objects with many sensors. In *Int. Joint Conf. on Artificial Intelligence*, pages 1160–1171.

J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

F. Pedersini, A. Sarti, and S. Tubaro (2001). Multi-camera parameter tracking. *IEE Proceedings: Vision, Image & Signal Processing*, 148(1):70–77.

S. Press (1972). *Applied Multivariate Analysis*. Series in Quantitive Methods for Decision-Making. Holt, Rinehart and Winston, Inc.

Y. Qi and T. Minka (2003). Expectation propagation for signal detection in flat-fading channels. In *IEEE Int. Symposium on Information Theory*.

C. Rao, A. Ylimaz, and M. Shah (2002). View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226.

C. E. Rasmussen (2000). The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12*, pages 554–560.

J. Rittscher, J. Kato, S. Joga, and A. Blake (2000). A probabilistic background model for tracking. In *Proc. of European Conf. Computer Vision*.

S. T. Roweis and Z. Ghahramani (1999). A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345.

J. Ruiz-Alzola, C. Alberola-Lopez, and J. Corredera (2000). Model-based stereo-visual tracking: Covariance analysis and tracking schemes. *Signal Processing*, 80(1):23–43.

L. K. Saul and M. I. Jordan (1999). Mixed memory Markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, 37(1):75–87.

C. Stauffer and W. E. Grimson (1999). Adaptive background mixture modelling for realtime tracking. In *IEEE Computer Vision and Pattern Recognition*, pages 246–252.

J. Sullivan, A. Blake, M. Isard, and J. MacCormick (2001). Bayesian object localization in images. *International Journal of Computer Vision*, 44(2):111–135.

J. Sullivan, A. Blake, and J. Rittscher (2000). Statistical foreground modelling for object localisation. In *Proc. of European Conf. on Computer Vision*, pages 307–323.

M. E. Tipping (1998). Probabilistic visualisation of high-dimensional binary data. In *Advances in Neural Information Processing Systems*.

K. Toyama and A. Blake (2001). Probabilistic tracking in a metric space. In *IEEE International Conference on Computer Vision*, pages 50–59.

N. Ukita and T. Matsuyama (2002). Real-time cooperative multi-target tracking by communicating active vision agents. In *IEEE Int. Conf. on Pattern Recognition*.

M. J. Wainwright and M. I. Jordan (2003). Graphical models, exponential families, and variational inference. Technical report 649, University of California, Berkeley.

D. Wang, T. Feng, H.-Y. Shum, and S. Ma (2002). A novel probability model for background maintenance and subtraction. In *Int. Conf. on Vision Interface*, pages 109–117.

M. Welling and Y.-W. Teh (2001). Belief optimization in binary networks: a stable alternative to loopy belief propagation. In *Uncertainty in Artificial Intelligence*.

C. K. I. Williams and D. Barber (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Ananlysis and Machine Intelligence*, 20(12):1342–1351.

J. Winn (2003). *Variational Message Passing and its Applications*. Ph.D. thesis, University of Cambridge.

P. J. Withagen, F. Groen, and K. Schutte (2005). CCD characterization for a range of color cameras. In *IEEE Instrumentation and Measurement Technology Conference*.

C. R. Wren, A. Azerbayejani, T. Darrel, and A. P. Pentland (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785.

J. Yamato, J. Ohya, and K. Ishii (1992). Recognizing human action in time-sequential images using hidden Markov model. In *IEEE Computer Vision and Pattern Recognition*.

J. S. Yedidia, W. T. Freeman, and Y. Weiss (2001). Understanding belief propagation and its generalizations. In *Int. Joint Conf. on Artificial Intelligence*.

J. S. Yedidia, W. T. Freeman, and Y. Weiss (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(5):2282–2312.

A. Yuille (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14:1691–1722.

W. Zajdel, A. Cemgil, and B. J. A. Kröse (2004). Online multicamera tracking with a switching state-space model. In *IEEE Int. Conf. on Pattern Recognition*, pages 339–343.

W. Zajdel, A. Cemgil, and B. J. A. Kröse (Submitted, 2005a). Hybrid graphical model for online multicamera tracking. *Pattern Recognition*.

W. Zajdel, A. T. Cemgil, and B. J. A. Kröse (Submitted, 2005b). A probabilistic model for spatial pixel correlations for background segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

W. Zajdel and B. J. A. Kröse (2002). Bayesian network for multiple hypothesis tracking. In *Dutch-Belgian Artificial Intelligence Conference*, pages 379–386.

W. Zajdel and B. J. A. Kröse (2003a). Approximate learning and inference for tracking with non-overlapping cameras. In *Int. Conf. on Artificial Intelligence and Applications*, pages 70–75.

W. Zajdel and B. J. A. Kröse (2003b). Gaussian mixture model for multi-sensor tracking. In *Dutch-Belgian Artificial Intelligence Conference*, pages 371–378.

W. Zajdel and B. J. A. Kröse (2005). A sequential Bayesian algorithm for surveillance with non-overlapping cameras. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(8):977–996.

W. Zajdel, N. Vlassis, and B. J. A. Kröse (2005a). Bayesian methods for tracking and localization. In E. Aarts, J. Korts, and W. Verhaegh, editors, *Intelligent Algorithms*, pages 243–258. Kluwer Academic Publishers.

W. Zajdel, Z. Zivkovic, and B. J. A. Kröse (2005b). Keeping track of humans: have I seen this person before? In *IEEE Int. Conf. on Robotics and Automation*, pages 2093–2098.

T. Zhao and R. Nevita (2004). Tracking multile humans in complex situations. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 26(9):1208–1221.

Z. Zivkovic (2004). Improved adaptive Gausian mixture model for background subtraction. In *IEEE Int. Conf. on Pattern Recognition*.

ACKNOWLEDGMENTS

I worked on this thesis pretty much uninterrupted between September 2001 and November 2005. The thesis is a result of support, collaboration and help from many friends and colleagues.

Without doubt, the thesis would have not been written without Ben and Taylan. I am deeply grateful for their guidance, research ideas and other teachings. If there is anything you both failed to teach me then it must be your passion for music. Although Frans was not directly involved in my daily research, his role as a promotor cannot be missed. I thank him for his contributions to the papers and the thesis.

Nikos and Josep gave me a lot of encouragement, which has been very valuable for me. Much appreciation to the remaining group members as well. In particular, Zoran contributed substantial part of the software used in Chapter 5. Leo, Stephan, my roommate Sjaak, fellow AIOs Mathijs (who translated the summary) and Jelle provided for a *gezellig* working environment. I thank the Nijmegen group, especially Tom — the wizard of graphical models always ready to guide the seeker of the truth on Bayesian inference. I thank Arend from Eagle Vision for collaboration and his interest in my work.

Outside office, my endavours in The Netherlands have been made easier by Kamil, Wien, Laurens, Dini, Anushka, Paula and Jan. I am especially indebted to Anushka and Dini. Anushka allowed me to live in effectively two homes for the past two years. If I can say a single Dutch word correctly it must have been Dini's effort.

Magdalena, whatever I say in whatever language . . .. I wouldn't have made it through to the end of this thesis without you. I deeply thank for all the love, patience and the inevitable insights on latest ballet displays in Amsterdam.

Dziękuję moim rodzicom: wasze wsparcie było i jest podstawą wszystkiego. Uściski dla mojej siostry Ewy oraz Szymona, ktorzy nigdy nie szczędzili zaproszeń do Lille! Dziękuję również rodzicom Magdaleny, za to iż zawsze o mnie pamiętali podczas swoich wizyt w Holandii.

Amsterdam, 21 November 2005.