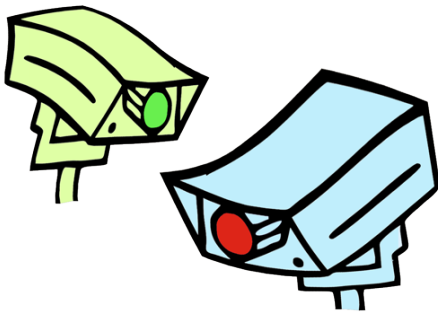
Multi-Observations Newscast EM for Distributed Multi-Camera Tracking

Thomas Mensink

Multi-Observations Newscast EM for Distributed Multi-Camera Tracking



Thomas Mensink

Supervisors:
Dr. ir. B.J.A. Kröse
Dr. W.P. Zajdel

August 2007

Universiteit van Amsterdam

Multi-Observations Newscast EM for Distributed Multi-Camera Tracking

Master's Thesis in Artificial Intelligence

T. E. J. Mensink

tmensink@science.uva.nl
thomas@mensink.nu
0113301

Supervisors:
Dr. ir. B.J.A. Kröse
Dr. W.P. Zajdel

August 2007

UNIVERSITEIT VAN AMSTERDAM
Department of Computer Science
Intelligent Autonomous Systems Group
Kruislaan 403 1098SJ Amsterdam
The Netherlands

Abstract

Visual surveillance in wide areas (e.g. airports) relies on multiple cameras which observe non-overlapping scenes. The focus of this thesis is multi-person tracking, where the task is to maintain a person's identity when he or she leaves the field of view of one camera and later re-appears at another camera. While current wide-area tracking systems are central systems, we propose to use a distributed system; where every camera learns from both its own observations and communication with other cameras.

Multi-person tracking can be seen as a data association problem, where the observations of the same person, gathered from different cameras, have to be clustered into trajectories. For this correspondence between a person's identity and an observation, we use appearances features (such as colour or height) and spatial-temporal features (such as the motion of a person from one camera to another). Under the assumption that all appearances of a single person are Gaussian distributed, the appearance model in our approach consists of a Mixture of Gaussians.

The Expectation-Maximization algorithm can be used to learn the parameters of the Mixture of Gaussians. In this thesis we introduce Multi-Observations Newscast EM to learn the parameters of the Mixture of Gaussians from distributed observations. Each camera learns its own Mixture of Gaussians model. Multi-Observations Newscast EM uses a gossip-based protocol for the M-step. We provide theoretical evidence, and using experiments show, that in an M-step each camera converges exponentially fast to the correct estimates. We propose to initialize Multi-Observations Newscast EM with a distributed K-Means to improve the performance. We found that Multi-Observations Newscast EM performs equally to a standard EM algorithm.

In this thesis we present two probabilistic models for multi-person tracking. The first model uses only appearance features, while the second model also uses spatial-temporal models. Both models are implemented in a central system and in a distributed system. The distributed system uses Multi-Observations Newscast EM, while the central system uses a standard EM.

The two models are tested on artificial data and on a collection of real-world observations gathered by several cameras in the university building. The results show that the central system and the distributed system perform equally well. While the more elaborate model, which uses appearance features and spatial-temporal features, outperforms the simple model.

Acknowledgements

I would like to thank my supervisors Ben Kröse and Wojciech Zajdel. Both have been immensely helpful with the research and the writing of this thesis.

Ben helped me from the start with converting my initial thoughts into a real project, and organizing almost everyone from the IAS-group to help me. During the research we have had many meetings to order my thoughts and writing.

Wojciech has explained to me the details of (Dynamic) Bayesian Networks, and probabilistic inference in time series. We have had quite some conversations until we both ran out of time. He also read my complete theses and provided it with detailed comments.

Moreover, without their high expectations and accompanying enthusiasm this thesis would have never become what it is now. Ben and Wojciech, thank you!

Contents

Chapter 1

Introduction

The use of camera surveillance in public is increasing. In the Netherlands 20% of the municipalities, including all major cities, use camera surveillance for public safety in social areas, shopping malls and bussiness parks. The images are either viewed live by human operators, or the images are recorded and stored (Dekkers and Homburg 2006).

For example, in 2004 the Amsterdam Nieuwmarkt district started using camera surveillance; 31 camera's were sparsely distributed throughout the district, see figure 1.1. The images shot by these camera's are only viewed live by a human operator during the evening hours. The daytime images are recorded on tape and only viewed in the case of an incident. Reports state that these recorded images are used approximately twice a week for the prosecution of suspects (Flight and Hulshof 2006).

Figure 1.1: Public camera surveillance in Amsterdam, *left* a map of the Nieuwmarkt district with locations of surveillance cameras, and *right* an example of a surveillance camera.

Public camera surveillance is used as a means to decrease crime and increase safety. To reach these goals, an operator must react on suspicious activities immediately. While trained human operators are very capable at this, they can do so only for a limited number of cameras simultaneously, and only for a limited amount of time (Weitenberg et al. 2003).

The fact that there are too many cameras and too few people to monitor them, inspires

research on intelligent computer systems to support the human operators. These intelligent systems should take over some routine tasks from the human operator, such as tracking people and detecting suspicious behaviour.

1.1 Automated visual surveillance

Humans are capable of direct interpretation of video data, however for a computer it is a complex task. Since it involves analysing raw video data in order to find relatively high-level summaries of the video content. The main stages of processing in an intelligent visual surveillance system often follow a hierarchical approach. From low-level goals like assigning pixels to moving objects, to high level goal like scene classification.

An example of a high level goal is human action recognition (Gavrila 1999; Masoud and Papanikolopoulos 2003); where a single camera observes a single person and classifies the current image into an action like walking, jumping, etc. Another high level goal is aggression detection (Zajdel et al. 2007), a single camera observes a scene with multiple persons and classifies the level of aggression.

Aside from the high-level goals, all visual surveillance applications need to be able to track objects of interest. Tracking an object means estimating the location of that object in subsequent frames of the video sequence. Detecting an object within a single frame of a camera is often based on foreground / background segmentation. Frequently used methods for this segmentation are background subtraction and temporal difference images (Valera and Velastin 2005).

When tracking involves multiple objects simultaneously present in the field of view of a camera, the correspondence between objects in the current frame with objects in the previous frame has to be resolved. Tracking multiple objects can be performed using, for example, the mean-shift or em-shift method (Zivkovic and Krose 2004).

Tracking multiple objects with multiple cameras requires additional bookkeeping of variables. An object may appear simultaneously on multiple streams. The appearances of the object on the different streams have to be associated to the same object. In-house tracking to control the safety of elderly people is considered in (Cucchiara et al. 2005), where a single person is tracked using several cameras.

Typically in a wide area, such as airports or office buildings, exhaustive coverage of the entire area of interest is impossible due to the large size of the monitored terrain. Therefore wide-area camera surveillance commonly relies on many cameras, where the field-of-view of one camera covers only a relatively small scene that does not overlap with the scenes observed by the other cameras. In this particular setting, tracking persons across all cameras is difficult, because someone is first observed by one camera, subsequently he is out of sight, and later he re-appears on another camera. The question we attempt to answer is whether the person now seen at this camera is the same person who was seen a while ago at the other camera.

A Dynamic Bayesian Network for tracking multiple persons over multiple sparsely distributed cameras was presented by (Zajdel 2006; Kettnaker and Zabih 1999). The presented Dynamic Bayesian Network uses two types of cues. First, appearance cues, because it is assumed that observations of the same person at various cameras are

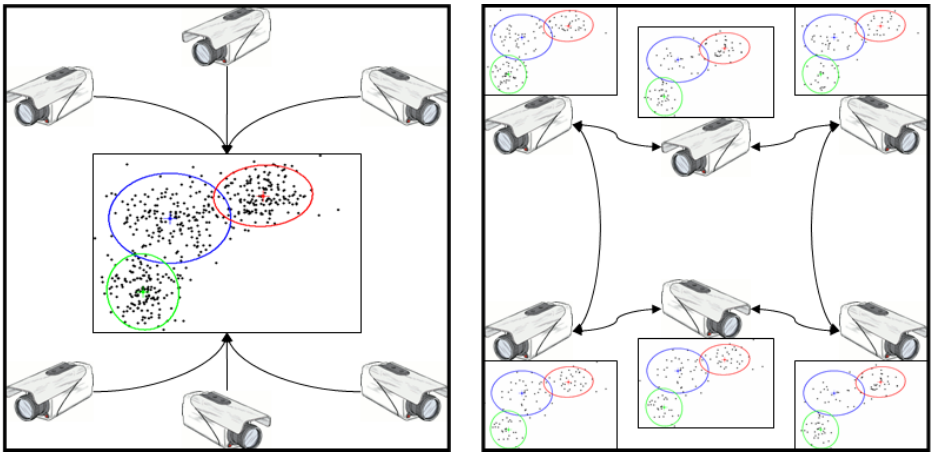


Figure 1.2: Central and Distributed System, *left* a central system, where all cameras send their observations to a single computer which learns a certain model, and *right* a distributed system, where all cameras learn the same model by inference over their own observations and by communication with the other cameras. In this distributed system communication between all cameras is possible, however in one communication cycle only some communication paths are used. This is a snapshot of a specific communication cycle.

similar. Second, spatial-temporal cues, because it is assumed that the camera-to-camera movement of a person is constrained by the environment.

All systems currently used for wide area tracking are central systems. In this setup the cameras send their observations to a central computer, which processes the gathered observations, see figure 1.2 (*left*). Problems of a central system approach include: privacy issues, because all observations are sent over a network and are centrally stored; network bottleneck, because all observations are sent to the same node; and the sheer risk of having one central system. A possible solution to these problems is the use of a distributed system.

1.2 Visual surveillance as a distributed system

A distributed multi-agent system is a collection of agents, where each agent observes local information and all agents have the ability to communicate with each other. The agents are considered to be autonomous entities, such as software programs, robots or intelligent cameras. Their interactions can be either cooperative, they share a common goal; or selfish, they pursue their own interest (Russell and Norvig 2003).

Some characteristics of a distributed multi-agent system are, (1) each agent has incomplete information to solve the problem, (2) there is no global system control, (3) data is stored decentralized and (4) computation is asynchronous (Sycara 1998).

Wide-area camera surveillance systems share many characteristics with a distributed multi-agent system. Imagine a tracking system where each camera is a single agent,

which together with the other agents has the common goal of tracking people. In this approach each camera is a stand-alone tracking unit, which stores its own observations and exchanges only limited data with other cameras. There is no central processing unit nor a central repository and there is no all-to-all broadcasting communication. This system is shown in figure 1.2 (*right*)

Contrary to a centralized system, a distributed system is able to: (1) solve problems that are too large for a centralized agent to solve because of resource limitations or the sheer risk of having one central system; (2) provide solutions that efficiently use information sources that are spatial distributed; (3) use less expensive sensors and therefore a larger number of sensors may be deployed over a greater area; (4) improve (among others), computational efficiency, robustness, reliability and bandwidth usage (Valera and Velastin 2005; Remagnino et al. 2004; Sycara 1998). These arguments should motivate to choose a distributed system over a centralized system.

Most of the current research on automated surveillance involves distributed systems, the need for this kind of systems in automated surveillance is emphasised in (Valera and Velastin 2005). Examples of distributed tracking systems, where multiple cameras observe overlapping field-of-views and together estimate someone's trajectory, are presented in (Remagnino et al. 2004; Nguyen et al. 2003; Nakazawa et al. 1998). However, distributed systems with non-overlapping field-of-views of cameras have received low interest in academic research so far.

1.3 Focus of thesis

In this thesis a distributed visual surveillance system is proposed, where multiple cameras collectively track people using the observations provided by all cameras, as in figure 1.2 (*right*) The focus will be on tracking multiple persons in a wide-area with non-overlapping cameras. The goal is to re-identify persons when they disappear, and later re-appear in the field of any camera. We use an off-line tracking algorithm, which means we use all observations and solve the re-identification after the images are being recorded.

Similar to other approaches a probabilistic framework is used (Zajdel 2006; Kettnaker and Zabih 1999). The probabilistic model, as will be fully presented in chapter 4, consists of hidden variables, and observed variables. The hidden variables define the identity and location of all tracked persons. The observed variables define the current observation seen at a single camera, consisting of appearance features and of spatial-temporal features.

The probabilistic model incorporates an *observation* model and a *transition* model. The transition model captures probabilistic dependencies on spatial-temporal features such that various constraints on motion are enforced. For example, if there is no path between one camera and another camera the transition model can restrict a direct movement between those cameras. The observation model uses the appearance properties such as the observed height of a person or his observed colour. The model defines a probabilistic relation between the appearances of the same person.

A person will appear differently each time he is observed because of changes in illu-

mination and in pose, therefore the appearance features are modelled as a stochastic variable. It is assumed that these observations are samples drawn from a Gaussian distribution with person specific parameters, which are constant over time. In a system where P persons are monitored, observations of all persons are generated by a Mixture of Gaussians (MoG) with P kernels.

An often used method for learning the parameters of an MoG is the Expectation-Maximization (EM) algorithm (Dempster et al. 1977). The standard EM method for mixture learning requires all observations for parameter estimation. However, in this research we have a distributed setup, without a central repository where all observations are stored. Each camera needs its own representation of the model, learned from the local available observations and from communication with other cameras.

The use of the MoG model allows us to adopt recently developed algorithms for distributed learning of the parameters (Nowak 2003; Kowalczyk and Vlassis 2005). A distributed system can efficiently compute the MoG's parameters without the need to gather all observations in a central repository. In this thesis we propose Multi-Observations Newscast EM to learn the parameters of the MoG at the distributed cameras. Multi-Observations Newscast EM is a generalisation of Newscast EM (Kowalczyk and Vlassis 2005). While Newscast EM requires each node to have exactly one observation, Multi-Observations Newscast EM loosens this constraint.

In this thesis two versions of a distributed visual surveillance system are presented. The first version takes into account only appearance features of people, and disregards motion information. The parameters of the MoG are learned from the appearance features. Given the learned MoG, each observation is assigned to the most likely person. A track of a person is considered to be the collection of all observations assigned to that person.

The second version also uses spatial and temporal features, it employs both the observation model for appearance features and the transition model for spatial-temporal features. By assigning a person to an observation, the previous location of that person is taken into account, which will result in more accurate tracks of persons.

A part of this thesis, involving the Multi-Objects Newscast EM algorithm, and its use in distributed appearance based tracking will be presented on the 1st ACM/IEEE International Conference on Distributed Smart Cameras (Mensink et al. 2007).

1.3.1 Thesis Outline

Chapter 2 provides a background to probabilistic models; these will be used throughout this thesis. It discusses the basics of Graphical Models with emphasis on (Dynamic) Bayesian Networks and Factor Graphs. Next it considers inference in Graphical Models and learning the parameters of a Mixture of Gaussians. In Chapter 3 we present Multi-Observations Newscast EM, an extended version of Newscast EM (Kowalczyk and Vlassis 2005); an algorithm for distributed learning of the parameters of a Mixture of Gaussians using the EM algorithm.

Chapter 4 introduces the distributed visual tracking systems. First a model is explained that only uses appearance based cues. Second an enhanced model is discussed, which

includes spatial and temporal features as well. In Chapter 5, experiments with both systems are presented. The performance of the systems is evaluated on both artificially generated and real data. The results of both systems are compared to each other and to centralized/standard implementations of the models.

Finally the conclusions are presented in chapter 6, which also includes possibilities for future research and proposed improvements of the used techniques.

Chapter 2

Probabilistic Models

Uncertainty exists in the domain of visual surveillance. In multi-person tracking the goal is to infer the identity of an observed person from the measurements provided by a camera. However a camera cannot directly observe the identity of a person, it only obtains a noisy observation of a person's characteristics. Therefore, tracking relies on related, but often ambiguous properties that can be measured, such as appearance features. The relation between the identity of a person and its properties contains uncertainty (Zajdel 2006).

Uncertainty is common, it arises from imprecise and incomplete data, and from unknown or intractable physical mechanisms which are approximated. Uncertainty also results from the use of a finite dataset in a infinite set of possibilities (Russell and Norvig 2003; Bishop 2006). In video streams incomplete and imprecise data are partly due to: sensor noise of a camera (like jitter); differences in pose of a person; variations in illumination circumstances; and differences in viewing angle.

Probability theory offers a mathematically consistent way to formulate inference algorithms when reasoning under uncertainty. For many computer vision problems, probabilistic models are preferred over other uncertainty enabled methods like default reasoning, fuzzy logic, and rule based systems. This is because of scaling, and because of chaining of evidence (Russell and Norvig 2003). Besides, probabilistic methods approximate intractable problems in a very formal way (Zajdel 2006).

A probabilistic model encodes relations between several variables relevant for a given problem. A convenient way to represent these relations between variables are graphical models, which are discussed in the next section. Followed by a section about inference in probabilistic models and a section on learning in probabilistic models.

2.1 Graphical Models

Graphical models offer some useful properties for studying large-scale probabilistic models. A graphical model represents the dependencies between variables and it gives

a concise specification of the joint probability distribution. In a graphical model, the joint probability distribution is represented as a graph; a collection of nodes and edges (links). Each node represents a random variable, and a link expresses the probabilistic relationships between two variables. Graphical models also visualize the structure of the model, which can be used to design and motivate (new) models (Russell and Norvig 2003; Bishop 2006).

In this thesis only graphical models are considered in the form of (Dynamic) Bayesian networks, and in the form of Factor Graphs. (Dynamic) Bayesian Networks are described in section 2.1.1. Factor Graphs, which are preferred for inference in probabilistic models, are discussed in section 2.1.2.

2.1.1 Bayesian Networks

A Bayesian Network (also known as Bayesian Belief Network or Belief Network) is a directed acyclic graph consisting of a set of random variables (the nodes) and a set of direct probabilistic relations (the links) leading from a parent node to a child node. The graphical model captures the causal process by which the observed data is assumed to be generated. For this reason such models are often called probabilistic generative models. This section contains a concise overview of Bayesian Networks, and presents some topics from (Russell and Norvig 2003; Bishop 2006).

In an exhaustive probability model all nodes will be connected to each other. The joint probability of such a model with N binary variables already requires $2^N - 1$ parameters. In the case of multinomial or continuous variables the number of parameters even increases.

Theoretically an exhaustive definition of a model would be necessary if all variables were directly dependent on each other. In practice, however, this assumption can be weakened using conditional independence. Both the structure of a model, and the computations needed to perform inference and learning under that model, will be simplified.

Consider three random variables a, b and c. Variable a is *conditionally independent* of b given c, if the conditional distribution of a, given b and c does not depend on the value of b (2.1). With the conditional independence, the joint distribution of a and b conditioned on c, can be factorized into (2.3).

$$p(a|b,c) = p(a|c) \tag{2.1}$$

$$\begin{aligned} p(a,b|c) &= p(a|b,c)\ p(b|c) && (2.2)\\ &= p(a|c)\ p(b|c) && (2.3) \end{aligned}$$

In a graphical model this will be represented by 3 nodes with two links: one from node c to node a and the other from node c to node b; but without a link from node b to node a or vice versa. This is shown in figure 2.1. So the absence of links in the graph conveys interesting information about the properties of the model.

A Bayesian Network uses only directed links between nodes. The graph of a Bayesian Network can thus be seen as a hierarchy, from variable nodes without parents to variable

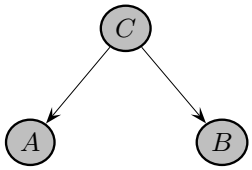


Figure 2.1: Conditional Independence, the graphical representation of variable a is conditionally independent of b given $cvar$.

nodes without children. For each node y_n the probabilistic distribution $p(y_n|\mathbf{parents}_{y_n})$ is defined. In the case of a node without parents this is the prior distribution $p(y_n)$.

The joint probability distribution defined by a Bayesian Network is given by the product (over all nodes), of the conditional distribution of a node given its parents. Thus for a graph with N nodes, where $\mathbf{y} = \{y_1, \ldots, y_N\}$, the joint distribution is given by (2.4).

$$p(\mathbf{y}) = \prod_{n=1}^{N} p(y_n|\mathbf{parents}_{y_n}) \tag{2.4}$$

In a Bayesian Network the nodes are divided between hidden (or unobserved or latent) nodes and observed (or evidence) nodes. It is assumed that the observed values are generated according to the defined probabilistic model. An observed node could be the reading of a sensor, the corresponding hidden node could be the real value. This relation is a convenient way to deal with noisy measurements of actual quantities.

Given some observed variables, the conditional distribution for the hidden node y_n, $p(y_n|\mathbf{observed})$ can be calculated using probabilistic inference. Often the observed variables are the child nodes, with some of the hidden variables as parents.

Example

Imagine a pile of pictures taken at different squares in Amsterdam that have to be classified. The squares are labelled *Spui, Dam, Muntplein, Koningsplein and Leidseplein.* Each square has its own unique real colour, which will not change over time, but which is unknown. The observed colour of a picture depends on both the real colour of the square where it is taken, and on the weather at the time the picture was taken. We assume that we can derive some cues about the real weather from the picture, however we cannot surely derive the real weather at the time the picture was taken. We model this uncertainty between the observed colour and the real colour, and between the observed weather and the real weather.

The label of a specific square is P, the real colours of all squares are represented in RC, the colour of a specific square P is represented as RC_P. Given the ideas from above, we assume that the value of the observed variable colour (C), depends on the square (P), the real colour of that square (RC_P) and on the real weather (RW). We also observe from the picture some clues about the weather, which are quantified in W, the value of W depends on the real weather (RW). The Bayesian Network for this

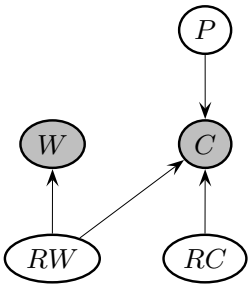


Figure 2.2: The Bayesian Network for the labelling pictures example. It defines the probabilistic relations between the label (P) of the square where the picture was taken, the colour of the picture (C), the weather observed from the picture (W), the real weather at the time when the picture was taken (RW), and the real colour of the square where the picture was taken (RC_P).

example is shown in figure 2.2. In the network the shaded nodes are observed nodes (C, W), and the others are hidden (P, RC, RW).

To classify a picture we have to compute the posterior $p(P|C, W)$. Given the observed variables (C, W), we would like to infer which label yields the highest value of $p(P|C, W)$. This is done using probabilistic inference, discussed in section 2.2.

In the Bayesian Network of figure 2.2, C is conditionally independent of W given RW, because if the value of RW is known, W will not add any information for the value of C. The joint probability distribution of the 5 variables is

$$p(\mathbf{y}) = p(W|RW)\ p(C|P, RC, RW)\ p(RW)\ p(RC)\ p(P) \tag{2.5}$$

Dynamic Bayesian Networks

The discussed Bayesian Networks represent models for static data, where each random variable has a single fixed value. In this section Dynamic Bayesian Networks (DBN) are discussed, which allow us to represent models for sequential data. These models come from measurements of time series, such as speech data or video streams. However sequential data also arise in other domains like base pairs along a strand of DNA or a sequence of characters in a sentence.

The sequential model can be viewed as a series of snapshots, where each snapshot describes the state of the world at a particular time (Russell and Norvig 2003). Each snapshot (or time slice) contains a set of random variables, some are observable and some are hidden. To express time relations in a probabilistic model it is assumed that the current state y_t depends only on the previous states $\{y_0, \ldots, y_{t-1}\}$, where t is a discrete time index, $t = \{0, 1, \ldots\}$. The structure of such model is shown in figure 2.3. This allows us to express the joint distribution as follows:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t|y_0, \ldots, y_{t-1}) \tag{2.6}$$

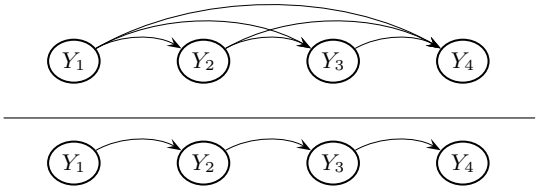


Figure 2.3: Time series in probabilistic models, the *upper* figure shows a full order Markov model, where the current state y_t depends on all previous nodes, and the *lower* figure shows a first order Markov model, where the current state y_t depends exclusively on the previous state y_{t-1}.

It is often assumed that the current state y_t exclusively depends on the previous state y_{t-1}, such a model is a *first-order Markov Model*, shown in lower part of figure 2.3. The conditional distribution for the current state in a first-order Markov Model is given by (2.7). This distribution is also called a *transition* model. Whether the first-order Markov assumption, as a transition model, is a reasonable approximation of the problem depends on the application domain. If the approximation proves too inaccurate, either the order of the Markov process, or the number of variables could be increased.

$$p(y_t|y_0, \ldots, y_{t-1}) = p(y_t|y_{t-1}) \tag{2.7}$$

DBN's could also be used for reasoning over latent time-processes, where the state of a time slice cannot directly be observed, but it produces a (possibly) noisy observable output (Kröse et al. 2004). In this case, every single time slice t consist of one (or more) hidden variables $\mathbf{z}_t$ that represent the current hidden state of the process, and one or more observed variables $\mathbf{y}_t$.

This is particularly useful in probabilistic models where the observations are high dimensional. In that case the transition model $p(y_t|y_{t-1})$ is difficult to learn. It becomes easier by introducing a hidden variable for each observation, which could be of different dimensionality to the observed variable, and by defining the transition model over the hidden variables.

The transition model is usually a first-order Markov Model. The observed variables $\mathbf{y}_t$ are assumed to be caused exclusively by the current state $\mathbf{z}_t$. The conditional distribution $p(\mathbf{y}_t|\mathbf{z}_t)$ is called the *observation* model, because it describes how the observations are affected by the corresponding state.

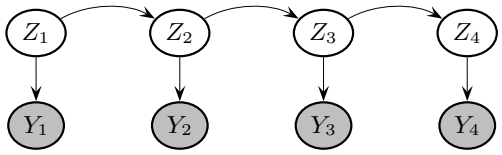


Figure 2.4: State space model, a graphical structure which is the foundation for the Hidden Markov Model and Kalman Filter Models.

This results in the state space model, which is the foundation of Hidden Markov Models and Kalman Filters Models (Murphy 2002). The graphical structure of the state space model is shown in figure 2.4.

The state space model is a convenient framework for multi-person tracking. We assume that each person walks in a first-order transition model through the monitored area and the trajectories of the persons are latent processes. Depending on the tracking problem, the hidden state of the processes might include the identity of the person or various other characteristics. The measurement of a person's characteristics obtained by the cameras correspond to the observations of the latent process.

Example

Let us return to the picture labelling example of the previous section, where we have to label a pile of pictures. Each picture has to be assigned to the square in Amsterdam where it is taken (P), given its average colour (C) and the weather (W) observed on the picture. To deal with the uncertainty we use two hidden nodes to define the real colour (RC) and real weather (RW).

If we assume that the photographer walked through Amsterdam, and took a picture every time he arrived at a square, then the order of the pictures also contains information about the square it is taken. We can define a transition model on the possible routes between the squares, by assigning probabilities to each transition. For example, the probability the photographer walks directly from the *Leidseplein* to the *Spui* without passing by another square is very small, because the main route leads through the *Koningsplein*. Thus the transition from *Leidseplein* to *Koningsplein* is extremely probable.

In figure 2.5, the Dynamic Bayesian Network of the example is shown. We assume a first order transition model $p(P_t|P_{t-1})$, and an observation model $p(C_t, W_t|P_t, RC_t, RW_t)$.

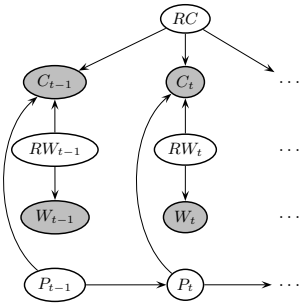


Figure 2.5: The Dynamic Bayesian Network for the labelling pictures example

2.1.2 Factor Graphs

The directed graphical models discussed above allow the joint probability function of several variables to be expressed as a product of factors over subsets of those variables. Factor graphs (Kschischang et al. 2001; Bishop 2006) make this decomposition explicit

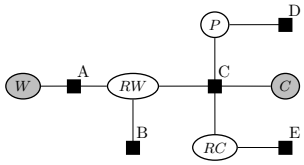


Figure 2.6: Factor Graph for inference problems, the factor graph shown is taken from the labelling pictures example

by introducing additional nodes for the factors themselves, next to the nodes representing the variables. This makes factor graphs convenient for inference problems in graphical models.

A factor graph represents a factorial expression as an undirected graph with two types of nodes: variable nodes representing random variables, and factor nodes that correspond to individual functions of the factorial expression. In figure 2.6 the factor graph for our example Bayesian Network for labelling pictures is shown; the squares represent factor nodes and the circles represent variable nodes.

The joint distribution over a set of variables is written as a product of factors (2.8), where $\mathbf{y}_s$ denotes a subset of the variables. Each factor f_s is a function of its corresponding variables $\mathbf{y}_s$.

$$p(\mathbf{y}) = \prod_s f_s(\mathbf{y}_s) \tag{2.8}$$

The joint distribution of the camera example, see figure 2.6, is :

$$p(\mathbf{y}) = f_a(W, RW)\ f_b(RW)\ f_c(P, RW, RC)\ f_d(P)\ f_e(RC)$$

In section 2.2.2 an inference method is discussed that exploits factorization encoded by a factor graph.

2.2 Probabilistic Inference

Probabilistic inference is a general term that refers to a class of computational problems related to the probabilistic framework. Given the structure of a generic temporal model we can formulate the following basic inference tasks: *filtering*, computing the posterior distribution (or belief state) over the current state given all evidence to date; *prediction*, computing the posterior distribution over the future state given all evidence to date; *smoothing*, computing the posterior distribution over a past state given all evidence up to the present; and *maximum a-posteriori* (MAP), computing the most likely sequence of states that generated the observed sequence of variables (Russell and Norvig 2003).

In multi-person tracking, the hidden state suppresses the identity and location of a person at the current time. To associate the current observation with the most likely person, we should use filtering to compute the posterior distribution over the current state given all evidence up to now.

Next, filtering in time series is considered. In section 2.2.2 we discuss the sum-product algorithm, which is an inference algorithm for dealing with complicated global models.

2.2.1 Filtering in time series models

In a time-series model, such as a Dynamic Bayesian Network, probabilistic filtering can be done with all observations gathered to date. Let us define the current time slice t, with the accompanying hidden variables $\mathbf{z}_t$, and observed variables $\mathbf{y}_t$. Given the result of filtering up to time $t-1$, it is possible to compute the result for t given the new evidence $\mathbf{y}_t$.

This process is often called recursive estimation, because it can be separated into two parts: (1) the state distribution $\mathbf{z}_{t-1}$ is projected forward to t, and (2) then this is updated using the new evidence $\mathbf{y}_t$. This recursive estimation emerges from:

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{y}_{1:t}) &= p(\mathbf{z}_t|\mathbf{y}_t, \mathbf{y}_{1:t-1}) \\ &= \frac{1}{\alpha} p(\mathbf{y}_t|\mathbf{z}_t, \mathbf{y}_{1:t-1}) p(\mathbf{z}_t|\mathbf{y}_{1:t-1}) \end{aligned}$$

Under the first-order Markov assumption (see section 2.1.1, equation (2.7)), we assume $p(\mathbf{y}_t|\mathbf{z}_t, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t|\mathbf{z}_t)$. Therefore we can rewrite the equation to:

$$p(\mathbf{z}_t|\mathbf{y}_{1:t}) = \frac{1}{\alpha} p(\mathbf{y}_t|\mathbf{z}_t) p(\mathbf{z}_t|\mathbf{y}_{1:t-1})$$

The second term, $p(\mathbf{z}_t|\mathbf{y}_{1:t-1})$, represents a one-step prediction of the next state. While the first term, $p(\mathbf{y}_t|\mathbf{z}_t)$ updates this with the new evidence. With $p(\mathbf{y}_t|\mathbf{z}_t)$ taken directly from the observation model, the one step prediction for the next state is obtained by conditioning on the state $\mathbf{z}_{t-1}$:

$$p(\mathbf{z}_t|\mathbf{y}_{1:t}) = \frac{1}{\alpha} p(\mathbf{y}_t|\mathbf{z}_t) \sum_{\mathbf{z}_{t-1}} p(\mathbf{z}_t|\mathbf{z}_{t-1}) p(\mathbf{z}_{t-1}|\mathbf{y}_{1:t-1})$$

Within the summation the first term is the transition model, and the second term is the current state distribution. Hence, the probability of $p(\mathbf{z}_t|\mathbf{y}_{1:t})$ obeys recursion, because it depends on $p(\mathbf{z}_{t-1}|\mathbf{y}_{1:t-1})$. The probability distribution $p(\mathbf{z}_{t-1}|\mathbf{y}_{1:t-1})$, might be difficult to represent, since it is often high dimensional. Therefore we will approximate the posterior probability of $\mathbf{z}_{t-1}$, with a simpler expression, and refer to it as belief $q(\mathbf{z}_{t-1})$.

The belief of the previous state can be seen as a message that is propagated forward along the sequence, modified by each transition and updated by each new observation. For computing the belief q from time slice $t-1$ to t, the sum-product algorithm is used.

2.2.2 Sum-Product Algorithm

The sum-product algorithm (Kschischang et al. 2001; Bishop 2006) is a generic message passing algorithm, that operates on a factor graph. It can deal with complicated global

functions of many variables by exploiting the way in which a global function factors into a product of simpler local functions. A wide variety of algorithms can be derived from the sum-product algorithm, with appropriately chosen factor graphs. For example the Viterbi algorithm, the Kalman filter, and Pearl's belief propagation algorithm for Bayesian Networks.

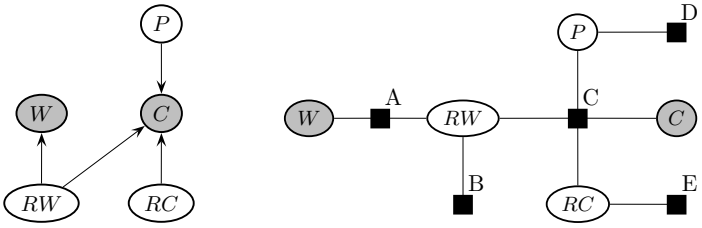


Figure 2.7: The Bayesian Network and Factor Graph from the labelling pictures example

The sum-product algorithm efficiently computes various marginal functions derived from the global function. This is done by exploiting the factorization of the global function, and by re-using intermediate values of partial sums. The Bayesian network and factor graph of the labelling pictures example are shown in figure 2.7. The joint function g of the variables can be expressed as a product of factors:

$$g(W, C, RW, RC, P) = f_a(W, RW)\ f_b(RW)\ f_c(C, P, RW, RC)\ f_d(P)\ f_e(RC).$$

With inference in a (Dynamic) Bayesian Network, one of the marginal functions $g_i(x_i)$ of interest is computed. In the example where a picture is assigned to a label, this is the marginal function $g_P(P)$. To compute this marginal function the variables RW, RC are integrated (2.9). In this example the variables W and C are observed variables, therefore there is no need to integrate over them.

$$g_P(P) = \sum_{RW,RC} f_a(W, RW)\ f_b(RW)\ f_c(C, P, RW, RC)\ f_d(P)\ f_e(RC) \tag{2.9}$$

$$= f_d(P) \sum_{RC} f_e(RC) \left(\sum_{RW} f_b(RW)\ f_a(W, RW)\ f_c(C, P, RW, RC) \right) \tag{2.10}$$

Each edge in the factor graph can be associated with an intermediate result of this integration, and can be either classified as a partial sum or a partial product. For example the edge between factor node b and variable node RW corresponds with the partial sum $\sum_{RW} f_b(RW)$.

Intermediate results can be seen as messages between variable nodes and factor nodes and vice versa, let m be such a message. A message from a variable node x to a factor node f is denoted as $m_{x \to f}(x)$. It contains the product of all incoming messages, except for the one giving the direction the message is sent to (2.11). The message from a factor node f to a variable node x is denoted as $m_{f \to x}(x)$, and equals the sum of the factor over all variables, multiplied by the incoming messages (2.12).

In equation (2.11) and (2.12), all neighbours of node v, except of one node w is represented as $n(v) \setminus \{w\}$, and the sum over all variables of a function, except of variable

x, is represented as $\sum_{\sim x}$. A message between two nodes is always a function of the variable associated with the variable node that the edge connects to.

$$m_{x\to f}(x) = \prod_{h\in n(x)\setminus\{f\}} m_{h\to x}(x) \tag{2.11}$$

$$m_{f\to x}(x) = \sum_{\sim x}\left(f(x)\prod_{y\in n(f)\setminus\{x\}} m_{y\to f}(y)\right) \tag{2.12}$$

The marginal function $g_P(P)$ (2.9) can be inferred from the following series of messages:

$$\begin{aligned} m_{a\to RW}(RW) &= f_a(W,RW) & \qquad m_{d\to P}(P) &= f_d(P)\\ m_{b\to RW}(RW) &= f_b(RW) & \qquad m_{e\to RC}(RC) &= f_e(RC) \end{aligned}$$

$$\begin{aligned} m_{RW\to c}(RW) &= m_{a\to RW}(RW)\; m_{b\to RW}(RW) = f_a(W,RW)\; f_b(RW)\\ m_{RC\to c}(RC) &= m_{e\to RC}(RC) = f_e(RC) \end{aligned}$$

$$\begin{aligned} m_{c\to P}(P) &= \sum_{\sim P} f_c(C,P,RW,RC)\; m_{RW\to c}\; m_{RC\to c}\\ &= \sum_{RC}\sum_{RW} f_c(C,P,RW,RC)\; f_a(W,RW)\; f_b(RW)\; f_e(RC) \end{aligned}$$

At variable P, the marginal function is calculated as the product of the incoming messages: $g_P(P) = m_{c\to P}(P)\; m_{d\to P}(P)$.

The marginal function of any variable $g_i(x_i)$ can be computed with the messages (2.11) and (2.12) according to the following rule. Start at the nodes with only one neighbour, each variable node sends an identity function message to its neighbour, and each factor node f sends a description of f to its neighbour. Next, each node v waits for messages from all of its neighbours except of one (node w), before computing a message to be sent to w. A variable node sends a message according to (2.11) and a factor node sends a message according to (2.12). The computation terminates at the desired node x_i, where $g_i(x_i)$ is obtained as the product of all received messages.

In many circumstances it may be of interest to compute $g_i(x_i)$ for more than one value of i. To do so, the described procedure has to be altered. Any node v waits until it receives messages from all but one neighbour (w), when all messages are received, v sends a message to w and waits for a return message from w. When this return message has been received, v sends a message to all its other neighbours. The procedure stops when a message has passed once along every edge in both directions.

The sum-product algorithm can be applied to probabilistic models with discrete and continuous variables, as long as the integrations have analytical solutions. The algorithm could handle arbitrary factor graphs, cycle free or not. A cycle free graph has only single paths connecting any pair of nodes, for cycle free graphs the sum-product algorithm computes the marginal function exactly. Inference in graphs containing cycles is significantly more difficult. It can be done using an approximate version of the sum-product algorithm, known as loopy-belief propagation (Kschischang et al. 2001; Bishop 2006). In cyclic graphs the marginal function is approximated.

2.3 Learning

Learning in graphical models involves algorithms that effectively estimate a probabilistic model from training data. There is a distinction between *structural* learning and *parameter* learning. With structural learning, the network structure is not known in advance, and the goal is to estimate it from the given training data. With parameter learning, the network structure and parametric family for each factor is given, and the goal is to estimate the corresponding parameters from the given training data.

In our wide-area tracking problem, we assume that observations are generated according to a (Dynamic) Bayesian Network with a fixed structure. A person's appearance is assumed to be a sample drawn from a Gaussian, with person specific parameters. All persons together are modelled with a Mixture of Gaussians.

In this setting, we are interested in learning the parameters of a Mixture of Gaussians. We have chosen to use the Expectation-Maximization (EM) algorithm for this. The results of the EM algorithm depends highly on the initialization. It is common to run the K-Means algorithm to find a suitable initialization for a Mixture of Gaussians that is subsequently adapted using EM (Bishop 2006).

First, we briefly revise the Mixture of Gaussian (MoG) distribution, followed by the EM algorithm for learning the parameters of an MoG in section 2.3.2. The K-Means algorithm is described in section 2.3.3, as initialization function for the EM algorithm.

2.3.1 Mixture of Gaussians

The Mixture of Gaussians (MoG) distribution, is a linear combination of several, say K, Gaussian components, providing a rich class of density models, which can be written as:

$$p(x) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2.13}$$

Where π_k is the mixing coefficient, μ_k the mean and Σ_k the covariance matrix of kernel k. The parameter π_k must satisfy $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$.

Depending on the kind of mixture, it can be either easy or hard to estimate the parameters so that the model fits our data as good as possible. "Fits our data as good as possible" is in our case defined as assigning the maximum log-likelihood to the data (Verbeek 2004). If we use a simple single Gaussian distribution, we can set the derivative of the log-likelihood to zero and solve the parameter estimation analytically. However, for many problems, it is not possible to find such analytical expressions and we must use more elaborate techniques, (Bilmes 1997). Below we present the standard Expectation Maximization algorithm to estimate the parameters.

2.3.2 Expectation Maximization for Mixture of Gaussians

The Expectation-Maximization (EM) algorithm (Dempster et al. 1977; Webb 2002; Bishop 2006) is a general method for finding the maximum-likelihood estimate of pa-

rameters of an underlying distribution from data. The EM algorithm is used in two cases. First, when the data has missing values the EM algorithm can be used to estimate these. Second, when the log-likelihood function is analytically intractable, but can be simplified by assuming the existence of additional but missing parameters, for example class labels (Bilmes 1997). The task in multi-person tracking is to associate data, to find all observations of the same person, making the additional parameter the identity of a person.

The standard EM algorithm assumes that the number of kernels is known in advance. However, there exist several alternations of the EM algorithm which gradually yield more kernels, these are called *greedy* EM algorithms (Verbeek et al. 2003; Vlassis et al. 2005).

Suppose we have a data set $x_1, \ldots, x_N$, which we will model using a K-component Mixture of Gaussians. The data are assumed to be independent and identically distributed (i.i.d.), thus $p(x_1, \ldots, x_N) = \prod_{n=1}^{N} p(x_n)$, where $p(x_n)$ is a Mixture of Gaussians. The EM algorithm for data association aims to estimate the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ of the K kernels that maximize the log-likelihood (2.14) of the data.

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right) \tag{2.14}$$

The EM algorithm is an iterative procedure, in each iteration two steps are performed to maximize a lower bound on (2.14). This bound is calculated from the current parameters θ and a set of responsibility distributions $\{q_n(k)\}$, for $n = 1, \ldots, N$. Each $q_n(k)$ corresponds to a data point x_n and defines for each kernel the responsibility that component k takes for explaining observation x_n (Bishop 2006).

In the Expectation (E) step of the EM algorithm, the responsibility $q_n(k)$ for each data point x_n is set to the posterior probability $p(k|x_n)$ given the current parameter estimate θ (2.15).

$$q_n(k) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{2.15}$$

In the Maximization (M) step, the parameters mean, covariance and mixing coefficients are updated by maximizing the log-likelihood for fixed $q_n(k)$. By a maximum of the log-likelihood, its derivative is zero. The update equations for one of the parameters is obtained by setting the derivative of the log-likelihood with respect to this parameter to zero. For the mean μ_k of kernel k the following derivative is obtained:

$$0 = -\sum_{n=1}^{N} \ln \left(\frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \Sigma_k^{-1} (x_n - \mu_k) \right) \tag{2.16}$$

Note that the derivative contains the equation for calculating the responsibility $q_n(k)$ given by (2.15). In order to get the maximization update function for μ_k, the derivative

(2.16) has to be rearranged and multiplied with Σ_k, which results in[1]:

$$\mu_k = \frac{1}{\sum_{n=1}^{N} q_n(k)} \sum_{n=1}^{N} q_n(k) x_n \tag{2.17}$$

The mean μ_k for component k of the Gaussian Mixture is obtained by taking a weighed mean of all data points in the set. The weighting factor for data point x_n is the posterior probability $q_n(k)$ that component k was responsible for generating x_n.

The update function for Σ_k follows a similar line of reasoning. The derivative of the log-likelihood with respect to Σ_k is set to zero and rearranged. This results in the following update function for Σ_k for component k:

$$\Sigma_k = \frac{1}{\sum_{n=1}^{N} q_n(k)} \sum_{n=1}^{N} q_n(k)(x_n - \mu_k)(x_n - \mu_k)^T \tag{2.18}$$

Finally, the log-likelihood with respect to the mixing coefficients π_k has to be maximized. Hereby the constraint $\sum_k \pi_k = 1$ should be taken into account. This can be achieved by using a Lagrange multiplier and maximizing the following quantity:

$$\ln p(X|\pi, \mu, \Sigma) + \gamma \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{2.19}$$

We find that $\gamma = -N$. After rearranging (2.19), and eliminating γ, we obtain:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} q_n(k) \tag{2.20}$$

The mixing coefficient π_k for component k is the average responsibility which this components takes for explaining the data points.

The derived update functions for the parameters (2.17), (2.18) and (2.20) do not have closed-form solutions because the responsibilities $q_n(k)$ depend on those parameters in a complex way through (2.15).

The EM algorithm for Mixture of Gaussian parameter estimation, starts with some initial values for either the responsibilities $q_i(k)$ or for the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$. Then it alternates between the E-step (2.15) and M-step (2.17,2.18,2.20), to update the responsibilities and parameters according to the equations above.

Each update of the parameters resulting from an E-step followed by a M-step is guaranteed to increase the log-likelihood function. Therefore, the EM algorithm is guaranteed to converge to a maximum of the log-likelihood function. However, this may be a local, sub-optimal, maximum depending on the initial parameters. The initial parameters or responsibilities could be set at random or by using an initialization function. A common initialization function is the K-Means algorithm (Bishop 2006).

[1] In the appendix a more elaborate derivation of the update equations is presented.

2.3.3 The K-Means algorithm

K-Means is an iterative clustering procedure to partition data into K clusters. Each cluster is a group of data points, whose inter-point distances are small compared to the distances between them and other cluster means. K-Means is often used as initialization for the EM algorithm when it is applied to a Mixture of Gaussians. Because K-Means is a very simple algorithm, and requires less computational costs then the EM algorithm.

$$J = \sum_{n}^{N} \sum_{k}^{K} r_n(k) \|x_n - \mu_k\|^2 \tag{2.21}$$

The quality of the current clusters is measured by the distortion measurement J (2.21). Each cluster k has a cluster mean μ_k, and each data point n has a vector $r_n(k)$, which is 1 for the cluster k the data point belongs to and 0 for all other clusters. The goal is to find values for $r_n(k)$ and μ_k to minimize J. The distortion measurement is the sum of the squared distance between each data point and its assigned cluster mean, this measurement could be seen as the sum of squared error.

The optimization of the distortion measurement can be done with an EM-like procedure, which successively assigns each observation to the closest cluster mean and update the cluster means according to the assigned observations. In the Expectation step, each data point x_n is assigned to the closest cluster mean:

$$r_n(k) = \begin{cases} 1 & \textbf{if } n = \arg\max_j \|x_n - \mu_j\|^2 \\ 0 & \textbf{otherwise} \end{cases} \tag{2.22}$$

And, in the Maximization step, for each cluster k, the cluster mean is updated to:

$$\mu_k = \frac{\sum_n r_n(k) x_n}{\sum_n r_n(k)} \tag{2.23}$$

Both steps are repeated until convergence of the parameters or of the distortion function. With the clusters found by the K-Means algorithm we can conveniently initialize the parameters $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ for the Mixture of Gaussians. The mean μ_k of each component is, of course, set to the mean obtained by the K-Means algorithm for cluster k. The covariance matrix Σ_k can be initialized to the sample covariance of cluster k. The mixing coefficient π_k, is set to the fraction of data points assigned to cluster k.

In fact the K-Means algorithm is equal to EM for learning the parameters of a Mixture of Gaussians, where the covariance matrices of all mixture components are fixed to the identity matrix I. However, the responsibilities in K-Means use a hard assignment for the closest cluster mean, $r_m(k) \in \{0, 1\}$, where the responsibility in EM algorithm is a soft assignment, $q_m(k) \in \mathbb{R}(0, 1)$.

2.4 Summary

In this chapter some of the basics of the probabilistic framework were explored. In the first section Dynamic Bayesian Networks were introduced, as well as the Factor

Graph representation. Next, inference in DBNs using the Sum-Product Algorithm was described. In the last section, parameter estimation for Mixture of Gaussians was discussed. With these basic principles explained, distributed learning of the MoG's parameters will be discussed in the next chapter. Followed by a chapter about the probabilistic model for tracking people in a distributed visual surveillance system.

Chapter 3

Multi-Observations Newscast EM

Learning a Mixture of Gaussians from distributed data

In the previous chapter we have described probabilistic models, inference and learning methods. As we will see in chapter 4, tracking can be formulated as a data association problem, where data (the observations of persons) have to be associated into trajectories. In our setup, the observations are generated from a Mixture of Gaussians, with unique parameters for each person. We need to learn this mixture model from the observations.

In our proposed distributed tracking system the observations are gathered by the cameras, and locally stored at the cameras. The distributed tracking system is seen as a network of nodes, each with a number of observations. In this network arbitrary point-to-point communication is possible between all nodes. The aim is to build a distributed learning method, where the nodes use their own observations and communication with other nodes to learn the parameters of the Mixture of Gaussian.

3.1 Distributed Learning a Mixture of Gaussians

A Mixture of Gaussians could be learned from the data itself, or from sufficient statistics of the data. The sufficient statistics for a Mixture of Gaussians are the *weight*, *mean* and *covariance* for each mixture component. These sufficient statistics can be computed efficiently by a distributed system, without the need to gather all observations at a central repository.

A frequently used approach for distributed parameter estimation is one where all nodes compute the sufficient statistics over their local data and send these to a central node. The central node merges the sufficient statistics into a global model and broadcasts

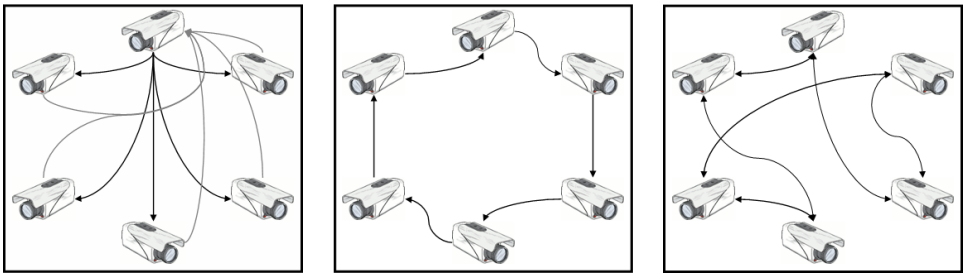


Figure 3.1: Different possible communication schemes in order to learn the parameters of a Mixture of Gaussians: (*left*) all nodes send their sufficient statistics to a central node, which broadcasts the global model to all nodes; (*middle*) the nodes communicate and update the partial sufficient statistics according to a fixed routing scheme; (*right*) the nodes communicate in cycles. In each cycle communication paths are chosen at random, and nodes update their sufficient statics by weighed averaging. A snapshot of the communication paths of a specific communication cycle is shown.

the global model to all nodes (Lin et al. 2005; Tasoulis and Vrahatis 2004; Forman and Zhang 2000). This approach is visualised in figure 3.1*(left)*, where the camera at the top of the figure acts as central node.

Another approach exploits a fixed communication scheme along all nodes (Nowak 2003; Blatt and Hero 2004). One node at a time carries out computations, it updates the (partly) sufficient statistics. Next the node sends the sufficient statistics to the following node in the schema. These algorithms monotonically converge to a local maximum, however they may converge to suboptimal maxima more often than a standard EM, because they operate much like a coordinate-ascent type algorithm (Nowak 2003). This approach is shown in figure 3.1*(middle)*, where no central node is used, but the order and direction of communication is fixed.

The third approach uses gossip-based methods (Kowalczyk and Vlassis 2005; Bandyopadhyay et al. 2006; Nikseresht and Gelgon 2006). These are different from the previous approaches, because they do not rely on a fixed communication scheme, but on randomization. Random pairs of nodes repeatedly exchange their local sufficient statistics and combine them by weighed averaging. In a communication cycle each node initiates a single contact. After multiple communication cycles the local sufficient statistics are converged to the global sufficient statistics. A snapshot of the communication paths in a specific communication cycle is shown in figure 3.1*(right)*. In another communication cycle the communication paths will be different.

Using a gossip-based method, the local statistics of each node converge exponentially fast to the global sufficient statistics. Gossip-based methods yield strong performance guarantees as a result of randomization. An advantage is that all nodes are simultaneously carrying out computations.

In this research we decided to use a gossip-based approach because it performs equally well as standard EM, and it does not rely on a fixed routing scheme. In this chapter we present the Multi-Observations Newscast EM algorithm, which is an extension of the Newscast EM algorithm (Kowalczyk and Vlassis 2005). The updates of the

parameters of the Mixture of Gaussians in the M-step of EM are averages over the data. Multi-Observations Newscast EM uses a gossip-based protocol to average the data, this protocol is Multi-Observations Newscast Averaging.

In section 3.2 Multi-Observations Newscast Averaging is presented, which is a protocol to compute the mean of a set of observations distributed over some nodes. In section 3.3, Multi-Observations Newscast EM is presented, which uses Multi-Observations Newscast Averaging.

3.2 Multi-Observations Newscast Averaging

Newscast is a gossip-based protocol for distributed computing, it applies to networks where arbitrary node-to-node communication is possible. The protocol involves repeated data exchange between nodes using randomization (Jelasiy et al. 2003; Kowalczyk and Vlassis 2005). As with other gossip based protocols, Newscast can be used for computing (among others) the mean of a set of values that are distributed over a network (Kempe et al. 2003).

In this section, we first present the conventional Newscast Averaging algorithm. Second in section 3.2.2, we introduce Multi-Observations Newscast Averaging as an extension of the original algorithm. Third in section 3.2.3, we prove the convergence rate for the proposed algorithm. Finally we present an experiment, where we test different distributions of the values over the nodes.

3.2.1 Newscast Averaging

The Newscast Averaging algorithm, as presented in (Jelasiy et al. 2003), assumes that every node in the network holds exactly one data point. It computes the mean of the data using communication between nodes.

Suppose that observations $o_1, \ldots, o_n$ are stored in the nodes of a network, with one value per node. To compute $\mu = \frac{1}{n}\sum_{i=1}^{n} o_i$, each node i initially sets $\hat{\mu}_i = o_i$ as its local estimate of μ. Then the node runs the following protocol for a number of cycles:

1. Contact node j, which is chosen uniformly at random from $1, \ldots, c$.
2. Nodes i and j update their estimates as follows: $\hat{\mu}'_i = \hat{\mu}'_j = \frac{\hat{\mu}_i + \hat{\mu}_j}{2}$.

Using this protocol, the mean of the local estimates is always the correct mean: $\mu = \frac{1}{n}\sum_{i=1}^{n} \hat{\mu}_i$. While, with each communication cycle the local estimates of node i and node j tend to the mean μ. After a number of cycles, the local estimates are converged to the mean μ, as is proven in (Kowalczyk and Vlassis 2005).

3.2.2 The Multi-Observations Newscast Averaging Algorithm

In our setting, where we learn from the observations gathered by the different cameras, the constraint of having exactly one observations per node is not useful. We

propose Multi-Observations Newscast Averaging (MON-Averaging), as an extension of Newscast Averaging. MON-Averaging does not have a constraint on the number of observations stored at a node. It uses additional variables to take into account the number of observations at a node.

Suppose that observations $o_1, \ldots, o_n$ are arbitrarily distributed over a network of nodes $c_1, \ldots, c_c$. Each node i in the network locally stores a number n_i of observations. The observations at node i are $o_{i,1}, \ldots, o_{i,n_i}$. The mean of all observations is given by (3.1), to make the distribution of the data over the nodes explicit, the mean is written as (3.2).

$$\mu = \frac{1}{n} \sum_{n=1}^{n} o_n \tag{3.1}$$

$$= \frac{1}{\sum_{i=1}^{c} n_i} \sum_{i=1}^{c} \sum_{m=1}^{n_i} o_{i,m} \tag{3.2}$$

To compute this mean in a distributed way, each node i sets $\hat{\mu}_i = \frac{1}{n_i} \sum_{m=1}^{n_i} o_{i,m}$ as its local estimate of μ, and it sets $w_i = n_i$ as its local estimate of n/c. The weighed average of all local estimates of $\hat{\mu}_i$ (3.3) equals the mean μ (3.2).

$$\mu = \frac{1}{\sum_i w_i} \sum_i w_i \hat{\mu}_i \tag{3.3}$$

After this initialization, each node runs the following update steps for a number of cycles:

1. Contact node j, which is chosen uniformly at random from $1, \ldots, c$.
2. Update nodes i and j estimates as follows $\hat{\mu}'_i = \hat{\mu}'_j = \frac{\hat{\mu}_i w_i + \hat{\mu}_j w_j}{w_i + w_i}$.
3. Update nodes i and j weights as follows $w'_i = w'_j = \frac{w_i + w_j}{2}$.

The local estimates $\hat{\mu}'_i$ and $\hat{\mu}'_j$ represents $w_i + w_j$ observations, after the communication between node i and node j. However the estimates are stored at two nodes, thus each node represents $(w_i + w_j)/2$ observations, this is included in the third step of the protocol.

With this protocol each node's estimate of the mean rapidly converges to the correct mean μ. After each update of the local estimates, the weighed mean of the local estimates remains the correct mean μ. This guarantees that the algorithm does not introduce errors. Figure 3.2 demonstrates that the mean of the local estimates is always the correct mean. It also shows the exponential convergence of all local estimates to the correct mean.

3.2.3 Variance Reduction

The variance of the local estimates represents how large the differences between the mean μ and the local estimates $\hat{\mu}_i$ are. Since the weighed mean of the local estimates

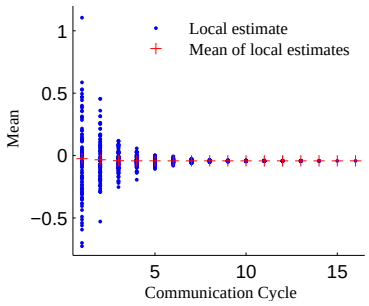


Figure 3.2: Estimated Mean at each cycle of Newscast Averaging

always equals the correct mean, the number of update cycles needed for convergence depends on on the decrease of variance. When the variance tends to zero, the local estimations are converged to the correct mean.

We define the un-normalized variance of the local estimates $\hat{\mu}_i$, at cycle t as: $\Phi_t = \sum_{i=1}^{c}(\hat{\mu}_i - \mu)^2$. This definition of the variance of the local estimates is equal to the variance in the Newscast Averaging algorithm (Kowalczyk and Vlassis 2005).

Lemma In each cycle of Multi-Observations Newscast Averaging the variance of the local estimates decrease on average by γ, with $\gamma \leq \frac{1}{2\sqrt{e}}$.

Proof Suppose that the mean of the observations is μ, and that within cycle t, the nodes initiate contact in the order $c_1, c_2, \ldots, c_c$. The current local estimate at node c_i is $\hat{\mu}_i$, and the current variance is $\Phi_{t,0}$. To calculate the variance $\Phi_{t,1}$ after node c_1 contacts node c_i, we take $\Phi_{t,0}$ subtract the variance of local estimates $\hat{\mu}_1$ and local estimate $\hat{\mu}_i$ before communication, and add the variance of the local estimates after the communication $\hat{\mu}'_1$ and $\hat{\mu}'_i$:

$$\Phi_{t,1} = \Phi_{t,0} - (\hat{\mu}_1 - \mu)^2 - (\mu_i - \mu)^2 + 2\left(\frac{w_1\mu_1 + w_i\mu_i}{w_1 + w_i} - \mu\right)^2$$

This can be rewritten to[1]:

$$\Phi_{t,1} = \Phi_{t,0} - \frac{2w_1^2}{(w_1+w_i)^2}(\hat{\mu}_1 - \mu)^2 - \frac{2w_i^2}{(w_1+w_i)^2}(\hat{\mu}_i - \mu)^2 + 4\frac{w_1\, w_i}{(w_1+w_i)^2}(\hat{\mu}_1 - \mu)(\hat{\mu}_i - \mu)$$

Taking the expectation of the variance, and using the facts that (1) $E[w_i] = E[w_j] = n/c$ for all nodes c_i and c_j; (2) all nodes are contacted with an equal probability of $\frac{1}{c}$; and (3) for all nodes $E[\hat{\mu}_i] = \mu$, gives:

$$\begin{aligned} E[\Phi_{t,1}|\Phi_{t,0} = \phi] &= \phi - \frac{1}{2}(\hat{\mu}_1 - \mu)^2 - \frac{1}{2c}\underbrace{\sum_{i=1}^{c}(\mu_i - \mu)^2}_{\phi} \\ &= \left(1 - \frac{1}{2c}\right)\phi - \frac{1}{2}(\mu_1 - \mu)^2 \end{aligned}$$

[1] In the appendix the complete derivation is shown.

After c such updates the complete communication cycle has passed. The variance of the next cycle Φ_{t+1} is on average:

$$E[\Phi_{t+1}|\Phi_t = \phi] = \left(1 - \frac{1}{2c}\right)^c \phi - \frac{1}{2}\sum_{i=1}^{c}\left(1 - \frac{1}{2c}\right)^{c-i}(\mu_i - \mu)^2$$

Bounding the term $\left(1 - \frac{1}{2c}\right)^{c-i}$ by $\left(1 - \frac{1}{2c}\right)^c$ and using the fact that $\left(1 + \frac{z}{n}\right)^n \leq e^z$ (Stewart 2001) results finally in:

$$E[\Phi_{t+1}|\Phi_t = \phi] \leq \frac{1}{2}\left(1 - \frac{1}{2c}\right)^c \phi \leq \frac{\phi}{2\sqrt{e}}.$$

Thus after t cycles of Multi-Observations Newscast Averaging, the original variance Φ_0 is reduced to $\Phi_t \leq \frac{\Phi_0}{(2\sqrt{e})^t}$ on average. The variance reduction is at an exponential rate, which means that the nodes converge to the correct mean extremely fast. For MON-Averaging and Newscast Averaging, the variance reduction rate is equal to each other. The variance reduction rate of Newscast Averaging is proven in (Jelasiy et al. 2003; Kowalczyk and Vlassis 2005).

With the variance reduction rate we can derive a maximum bound on the number of update cycles needed to guarantee, with high probability, that all nodes have estimated the correct mean (within some specific accuracy).

We use the assumption that the initial variance is $\Phi_0 = n\sigma^2/w = c\sigma^2$, where $w = n/c$, n the number of observations, c the number of nodes, and σ^2 the data variance. We obtain $E[\Phi_t] \leq \frac{c\sigma^2}{(2\sqrt{e})^t}$ as the expected variance of the local estimates after cycle t. If the variance $E[\Phi_t] \leq \epsilon^2\delta$, then we can proof with Markov inequality that with probability of at least $1 - \delta$ holds that $\Phi_t \leq \epsilon^2$. In this case, each of the local estimates must hold $|\mu_i - \mu| \leq \epsilon$, because the variance is the sum of local terms.

Set the maximum number of cycles to $\tau = \log\left(\frac{n\sigma^2}{\epsilon^2\delta} / \log(2\sqrt{e})\right)$, which will result in $E[\Phi_\tau] \leq \epsilon^2\delta$. Thus, with probability $1-\delta$ after $\tau = \frac{1}{\log 2\sqrt{e}}[\log c + 2\log\sigma + 2\log\frac{1}{\epsilon} + \log\frac{1}{\delta}]$ cycles MON-Averaging has converged, with precision ϵ.

While the variation reduction rate of MON-Averaging equals the original algorithm, the bound of the number of cycles could be lower. This is because of the lower initial variance Φ_0, when an average weight of $w > 1$ is used.

The maximum number of communication cycles for 100 nodes with unit variance data, $\epsilon = 10^{-5}$, and $\delta = .05$, is 16. This number grows only log-linear to the number of nodes.

3.2.4 Experiment

In the previous section we have presented MON-Averaging, a generalisation of the Newscast Averaging. In contrast to the original algorithm, MON-Averaging allows to have any number of observations at a node. We want to know if the performance of MON-Averaging is influenced by the distribution of the observations over the nodes.

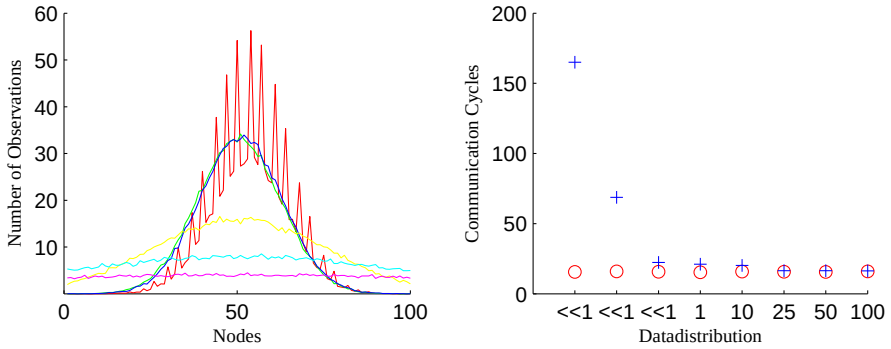


Figure 3.3: Different distribution of the data over the nodes. *Left* the different data distributions, for each node the number of observations at that node is shown for several standard deviation values. *Right*, the number of communication cycles to converge (with $\delta = .05$ and $\epsilon = 10^{-5}$) for several standard deviation values. The $\circ$ marks indicates the MON-Averaging with updating weights, and the $+$ mark indicates the algorithm without updating weights.

For this experiment we have generated several sets of 1000 unit variance random values, distributed over 100 nodes. The distribution over the nodes is a Gaussian function, with mean 50 and different standard deviation (sd) values. Using low sd values results in a peaky distribution of the values over the nodes. While when using high sd values the distribution is almost uniform. This is shown in figure 3.3 on the left.

As evaluation criteria we have taken the number of update cycles needed to converge, within an accuracy of $\epsilon = 10^{-5}$. In figure 3.3, the number of update cycles needed before convergence is shown for several sd values.

We have compared our MON-Averaging algorithm with weight updating (step 3 of the protocol), to an implementation without these updates. Without a weight update, the sum of the weights keeps also equal, and therefore the weighed average of all local estimates equals μ (3.3).

Updating the weights requires more bandwidth, but the performance increase of updating the weights is enormous for low sd values. This can be explained by the following example: a node (A) with a huge weight contacts a node (B) with a small weight. After the update both nodes have an estimate of the parameters which is calculated over a huge part of the observations. In the next cycle node B contacts another node (C) with a small weight. If the weights are not updated, the estimates from B and C are almost equally taken into account for the new estimate. However, when the weights are updated, B has a larger weight, which will yield a more accurate new estimate for both nodes B and C.

As can be seen in figure 3.3 the presented MON-Averaging algorithm, with updating weights, will always converge quickly. The performance of MON-Averaging is not affected by the distribution of the data over the nodes.

3.3 Multi-Observations Newscast EM

In the previous section we have introduced MON-Averaging; a protocol to calculate the mean of a set of observations, which are arbitrary distributed over a set of nodes. In this section we present Multi-Observations Newscast EM (MON-EM), a gossip-based distributed implementation of the EM algorithm for Gaussian mixture learning. MON-EM is a direct application of the MON-Averaging algorithm for the M-step of EM.

MON-EM is a generalisation of Newscast EM (Kowalczyk and Vlassis 2005), where the requirement of exactly one observation per node from Newscast EM is relieved.

In our distributed tracking system, the task is to learn the parameters of the Mixture of Gaussians (MoG), from the observations distributed over the cameras. The cameras should use their local observations and communications with other nodes to learn the parameters of the MoG. Therefore, all learning steps should be performed locally at the nodes and should involve as little communication as possible.

The MON-EM algorithm is described in section 3.3.1. We propose to initialize MON-EM with a distributed K-Means algorithm, which is explained in section 3.3.2. In section 3.3.3, we compare the computational complexity and bandwidth usage of a standard EM algorithm and MON-EM. Finally, in section 3.3.4, we present some experimental results of our proposed MON-EM algorithm.

3.3.1 The Multi-Observations Newscast EM Algorithm

The EM algorithm for Mixture of Gaussians is used to find the optimal parameters to model a set of observations. The standard EM algorithm (see section 2.3.2) requires that all observations are at a central repository. MON-EM learns the parameters of the Mixture of Gaussians from a set of distributed values.

The main difference between MON-EM and a standard EM algorithm is the M-step. This step is implemented in a distributed manner, using a sequence of gossip-based cycles. The E-step is identical for each node to standard EM, the responsibilities $q_i(k)$ for all data points are calculated according to (3.4). Each node computes the posterior distribution for the local available data, given the current estimates of the parameters. This can be done by all nodes simultaneously.

In the M-step of the EM algorithm the parameters, mean (μ_k), covariance (Σ_k), and mixing coefficient (π_k) for each kernel k, are updated according to (3.5-3.7). These parameter updates are mainly averages over all observations from the data set. As shown in the previous section MON-Averaging can be used to calculate the average over a set of observations distributed over several nodes. The M-step is implemented as series of MON-Averaging over the parameters.

$$q_n(k) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{3.4}$$

$$\mu_k = \frac{1}{\sum_{n=1}^{N} q_n(k)} \sum_{n=1}^{N} q_n(k) x_n \tag{3.5}$$

$$\Sigma_k = \frac{1}{\sum_{n=1}^{N} q_n(k)} \sum_{n=1}^{N} q_n(k)(x_n - \mu_k)(x_n - \mu_k)^T \tag{3.6}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} q_n(k) \tag{3.7}$$

Suppose we are given a set of observations $x_1, \ldots, x_n$ which are arbitrarily distributed over a network of nodes $c_1, \ldots, c_c$. Each node i in the network locally stores a number n_i of observations. The observations at node i are $x_{i,1}, \ldots, x_{i,n_i}$.

The observations are assumed to be independent samples from a common K-component mixture of Gaussians $p(x)$ with (unkown) parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$. In the E-step each node i computes the responsibilities $q_{i,m}(k)$ for every local observation $x_{i,m}$, given the current parameters θ, according to (3.4).

In the M-step each node i starts with a local estimate $\hat{\theta}_i$, of the correct parameter vector θ (correct according to EM and for the current EM iteration). This is followed by a number of communication cycles, where each node contacts another node at random, and both nodes update their local estimates by their (weighed) averages. The local parameter estimates converge quickly to the correct parameter vector θ.

This results in the following algorithm, which runs identically and in parallel at each node:

1. **Initialization** set the responsibilities $q_{i,m}(k)$ random or with K-Means.
2. **M-Step** initialize node i's local parameter estimates by setting $w_i = n_i$, and for each component k set the estimates to:

$$\hat{\pi}_{i,k} = \frac{1}{n_i} \sum_{m=1}^{n_i} q_{i,m}(k)$$

$$\hat{\mu}_{i,k} = \frac{1}{\hat{\pi}_{i,k}} \sum_{m=1}^{n_i} q_{i,m}(k)\ x_{i,m}$$

$$\hat{C}_{i,k} = \frac{1}{\hat{\pi}_{i,k}} \sum_{m=1}^{n_i} q_{i,m}(k)\ x_{i,m}\ x_{i,m}^T$$

   Then repeat for t cycles :

   (a) Contact node j, randomly chosen from $1, \ldots, c$.

(b) Update the estimates of node i and j for each component k as follows:

$$w_i' = w_j' = \frac{w_i + w_j}{2}$$
$$\hat{\pi}_{i,k}' = \hat{\pi}_{j,k}' = \frac{\hat{\pi}_{i,k} w_i + \hat{\pi}_{j,k} w_j}{w_i + w_i}$$
$$\hat{\mu}_{i,k}' = \hat{\mu}_{j,k}' = \frac{\hat{\pi}_{i,k} \hat{\mu}_{i,k} w_i + \hat{\pi}_{j,k} \hat{\mu}_{j,k} w_j}{\hat{\pi}_{i,k} w_i + \hat{\pi}_{j,k} w_j}$$
$$\hat{C}_{i,k}' = \hat{C}_{j,k}' = \frac{\hat{\pi}_{i,k} \hat{C}_{i,k} w_i + \hat{\pi}_{j,k} \hat{C}_{j,k} w_j}{\hat{\pi}_{i,k} w_i + \hat{\pi}_{j,k} w_j}$$

3. **E-Step** compute for each component k the new responsibilities for the local available observations $q_{i,m}(k)$(3.4), using the M-step estimates $\pi_{i,k}$, $\mu_{i,k}$, $\Sigma_{i,k} = C_{i,k} - \mu_{i,k}\ \mu_{i,k}^T$.
4. **Loop** repeat the M-step and E-step until a stopping criterion is satisfied.

An illustration of the M-update step of the MON-EM algorithm is shown in figure 3.4. The local parameter estimates are shown before any communication and after several communication rounds. We see that the local estimates converge quickly to each other.

Some aspects of the MON-EM algorithm are worth mentioning. First, the initialization of the M-step of MON-EM is equal to the M-update step for standard EM, however it is computed at each node, and thus only over the locally available data. Both the initialization of the M-step and the E-step are completely local to each node. Similarly, a stopping criterion based on the parameters could be implemented locally, this only requires that each node stores its previous parameter estimates. So, except for the M-step update, all steps can be implemented locally.

$$\mu_k = \frac{\sum_{i=1}^{c} w_i \pi_{i,k} \mu_{i,k}}{\sum_{i=1}^{c} w_i \pi_{i,k}} \tag{3.8}$$

Second, it is important to point out that the weighed averages of the local estimates are always the EM-correct estimates. For the mean this is shown in equation (3.8), but it holds for all parameters in θ. So in each communication cycle within the M-step update, the parameters converge at an exponential rate to the correct values.

A problem associated with the Maximum Likelihood framework for Mixture of Gaussians is the presence of singularities (Bishop 2006). Singularities will occur whenever one of the Gaussian kernels collapses onto a specific observation. In a standard EM algorithm, it is possible to check the covariance matrix after the M-step for singularities. The covariance matrix is only updated, when the new covariance matrix does not contain singular values.

In the distributed setting this is a bit more challenging, however the same procedure can be done as well. After the M-update step, just before the E-step, each node should calculate $\Sigma_{i,k} = C_{i,k} - \mu_{i,k}\ \mu_{i,k}^T$ and it should check this covariance matrix for singularities. If it contains singular values, then the node will use its previous covariance matrix. As a results all nodes will use their previous covariance matrix in the case where a covariance matrix contains singular values.

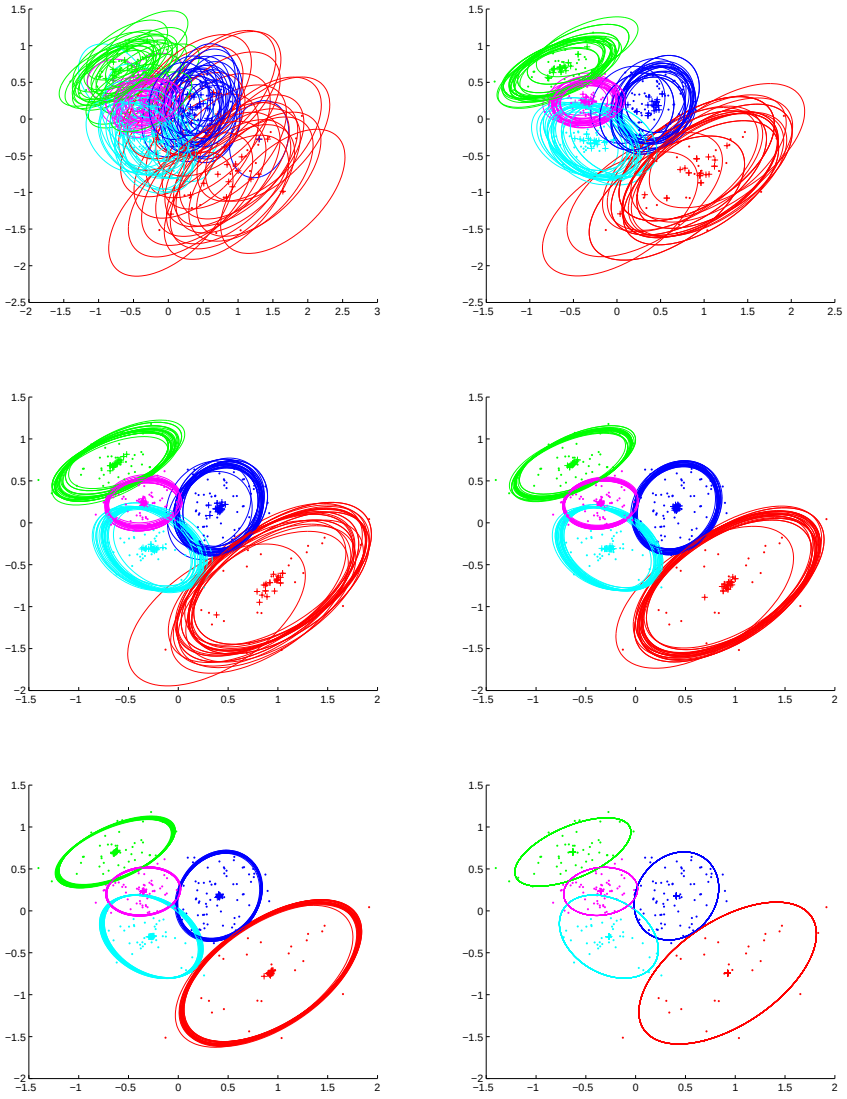


Figure 3.4: In this figure the Multi-Observation Newscast EM algorithm is shown after different communication cycles within one EM iteration. The figures show a data set with two dimensions, generated from a Mixture of Gaussians with 5 kernels. Each figure, shows the local parameter estimates from 25 nodes during the M-update step. The figures show the result before communication and after $\{1, 2, 3, 4, 10\}$ communication rounds respectively. Each colour indicates a different cluster, each data point is represented by a **.**, each mean by a $+$, and each covariance matrix by a ellipse. The figure shows that the estimates of the nodes converge quickly to each other.

3.3.2 Initialization with Multi-Observations Newscast K-Means

As discussed in section 2.3, the performance of the EM algorithm for Mixture of Gaussians, depends on the initialization. The Newscast EM algorithm uses a random initialization for the responsibilities. The responsibilities are set to a random positive value and then normalized, in order for $\sum_{k=1}^{K} q_i(k) = 1$. However, randomly initialized EM's yield a suboptimal solution more often than K-Means initialized EM's. Therefore, we propose Multi-Observations Newscast K-Means (MON-KM), a distributed gossip-based implementation of the K-Means algorithm.

K-Means is an iterative clustering procedure. In the first step for each observation x_n and for each cluster k, $r_n(k)$ is set to 1 if x_n belongs to cluster k and zero otherwise (3.9). In the second step, the cluster mean μ_k is updated according to all observations x_m which are assigned to cluster k (3.10). These steps are repeated until convergence.

$$r_n(k) = \begin{cases} 1 & \textbf{if } k = \arg\min_m \|x_n - \mu_m\|^2 \\ 0 & \textbf{otherwise}. \end{cases} \tag{3.9}$$

$$\mu_k = \frac{1}{\sum_{n=0}^{n} r_n(k)} \sum_{n=0}^{n} x_n \, r_n(k) \tag{3.10}$$

To execute the K-Means algorithm in a network with observations distributed over a set of nodes, we modify it in a similar way as the MON-EM algorithm. The first step of K-Means (3.9) is completely local at each node, it considers only the local available data, and the current estimates of the means. Each node calculates $r_{i,m}(k)$ for each local available observation $x_{i,m}$. The second step (3.10) of MON-KM is implemented as a number of gossip-based update cycles, because it is an average over all observations, with the following procedure:

1. **initialization** The local parameters for the weight w_i, the local means $\hat{\mu}_{i,k}$ and the responsibilities $\hat{\pi}_{i,k}$ has to be initialized as follows:

$$\begin{aligned} w_i &= n_i \\ \hat{\mu}_{i,k} &= \frac{1}{\sum_{m}^{n_i} r_{i,m}(k)} \sum_{m}^{n_i} r_{i,m}(k) \, x_{i,m} \\ \hat{\pi}_{i,k} &= \frac{\sum_{m}^{n_i} r_{i,m}(k)}{n_i} \end{aligned}$$

2. **gossip-based update** for a number of cycles, contact node j, randomly chosen from $1, \ldots, c$ and update the estimates of both nodes as follows:

$$\begin{aligned} w_i' = w_j' &= \frac{w_i + w_j}{2} \\ \hat{\mu}_{i,k}' = \hat{\mu}_{j,k}' &= \frac{w_i \, \hat{\pi}_{i,k} \, \mu_{i,k} + w_j \, \hat{\pi}_{j,k} \, \mu_{j,k}}{w_i \, \hat{\pi}_{i,k} + w_j \, \hat{\pi}_{j,k}} \\ \hat{\pi}_{i,k}' = \hat{\pi}_{i,k}' &= \frac{w_i \, \hat{\pi}_{i,k} + w_j \, \hat{\pi}_{j,k}}{w_i + w_j} \end{aligned}$$

Several experimental results have shown that MON-KM performs identically to a central K-Means implementation.

3.3.3 Computational Complexity and Bandwidth usage

In this section Newscast EM is compared to standard EM considering the computational complexity and bandwidth usage.

Computational Complexity

The time required by an algorithm to be executed is proportional to the number of *basic operations* that it performs. Some examples of basic operations are, arithmetic operation, binary test (e.g., $x == 0$), and read or write. The Big $\mathcal{O}$ notation could be used for describing the number of basic operations of an algorithm.

When considering EM for learning the parameters of a Mixture of Gaussians, all basic operations are performed during the E-step and the M-step. The overhead of the EM algorithm is relatively low.

For a standard EM algorithm, the E-Step has to calculate the responsibility for each data point and for each cluster k. Therefore the number of basic operations is in the order of $\mathcal{O}(KDN)$, where K is the number of clusters, N is the number of observations, and D is the dimensionality of the observations. The M-step has to compute the new parameters for all clusters from all data, therefore it is also $\mathcal{O}(KDN)$. The summed complexity of standard EM for Mixture of Gaussians, is $\mathcal{O}(KDN)$, (Verbeek 2004).

The MON-EM algorithm basically completes the same functions, however these functions are distributed over a set of nodes (C), and the M-step is divided into an initialization and an update procedure. We are considering the complexity of MON-EM over all nodes. The complexity of the E-step over all nodes is the same as for standard EM, $\mathcal{O}(KDN)$. Also the complexity of the M-step initialization over all nodes is the same as for standard EM, $\mathcal{O}(KDN)$.

The M-update step of MON-EM does not exist in the standard EM algorithm. In every cycle of the M-step update, each node adjusts, on average, twice their parameter estimations for each kernel k. Each update of the parameters approximately has a cost which is quadratic in the number of dimensions. The complexity therefore is, $\mathcal{O}(KCD^2)$. The summed complexity for MON-EM is $\mathcal{O}(KDN)$ if $N \geq CD$ and $\mathcal{O}(KCD^2)$ otherwise. With $\mathcal{O}(KDN)$ as the complexity of standard EM, we can state that the global complexities of MON-EM and standard EM are comparable.

Bandwidth Usage

Another important bottleneck in many camera surveillance systems could be the bandwidth usage. The bandwidth usage for a central server system is the sum of the bits sent from all nodes to the central computer for every observation. The total bandwidth usage in a system with N observations each with D dimensions usage is : $N * D * b$, where b represents the number of bits needed for a data point.

The bandwidth usage in MON-EM is completely different; observations are not shared, but kept locally. Only estimated responsibilities, means, and covariance matrices are

sent from node to node. The total bandwidth depends on the number of EM iterations and communication cycles, and not on the number of observations.

In a MON-EM system with K kernels, the communication of the parameters for kernel k requires b for the mixing coefficient, bD for the mean and $bD(D+1)/2$ for the covariance matrix. For the covariance matrix holds that $\Sigma_k = \Sigma_k^T$, therefore it is sufficient to sent only the upper part of the covariance matrix instead of the complete matrix. The update of the parameters between two nodes requires that both nodes send their parameters $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ and their weight estimate w. This has a total bandwidth usage of $b2(1+K\,(1+D+D(D+1)/2))$. Thus all communication within a single cycle of the M-update step require a total of $b2C(1+K\,(1+D+D(D+1)/2))$, where C is the number of nodes.

The number of update cycles is $EM\,iterations * MaxCycles$. While the number of $MaxCycles$ is the number of cycles to guarantee convergence, and is known, the number of $EM\,iterations$ is unknown in advance.

Assume a system with 100 cameras, tracking 10 people, each camera has 1000 observations, and each observation is a 9-dimensional appearance feature vector. The $MaxCycles$ is set to 16, and the number of EM-iterations is 25. In this case, the number of bits sent for both algorithms are:

- **Newscast System** : $b * 25 * 16 * 2 * 100 * (1 + 10 * [1 + 9 + 45]) = 44 * 10^6 * b$.
- **Central System** : $b * 100 * 1000 * 9 = .9 * 10^6 * b$.

So in this case the central system uses less bandwidth than Newscast system. Of course, this depends mainly on the number of observations, nodes and dimensionality. In a system with many observations, Newscast EM can be more economical with bandwidth usage.

Although in a central system the total bandwidth usage is lower, there could still arise a bottleneck at the central computer, because it has to receive $.9 * 10^6 * b$ bits. While in the Newscast EM system, each node on average receives only $.44 * 10^6 * b$ bits.

Also, in the current computation of the bandwidth cost, the distance of communication is not taken into account. In systems where the distance between nodes is a cost component for the bandwidth usage, the equations differ completely. Newscast EM can be implemented in a way where it only communicates with nearby nodes. This is not possible for a central system algorithm.

3.3.4 Experiments

In this section we describe our experiments with MON-EM and present the obtained results. The experiments compare the performance of MON-EM to a standard EM algorithm.

We have performed several experiments on synthetic data, to assess the performance differences between MON-EM and standard EM. The experiments investigate different initialization methods, different characteristics of the dataset, possibilities to use only

local communication, possibilities to reduce communication and the reliability of both methods when communication fails.

Data For each experiment data were generated according to a Mixture of Gaussians, with several parameters to influence the dataset. Examples of these parameters are the number of kernels, and the number of nodes. Because all data is generated at random, for each set of parameters, 10 datasets are created. Each dataset is tested 10 times, so the results of testing a set of parameters are based on 100 tests.

The c-separation value and eccentricity (Dasgupta 1999, 2000) indicates how difficult a dataset is, these values are also used in (Verbeek et al. 2003). Two Gaussians are c-separated if :

$$\|\mu_1 - \mu_2\| \leq c\sqrt{\max\{trace(\Sigma_1), trace(\Sigma_2)\}}$$

A mixture of Gaussians is c-separated if all its components are pairwise c-separated. The eccentricity of a Gaussian is defined as the square root of the ratio of the largest eigenvalue γ_n of the covariance (Σ) and the smallest eigenvalue γ_1 :

$$ecc = \sqrt{(\gamma_n/\gamma_1)}$$

When eccentricity increases or c-separation decreases, the difficulty of the recognition problem grows accordingly.

For several experiments the same dataset is used, this set is referred to as the standard set. This standard set contains 100 observations, generated from a 5 component MoG, distributed over 25 nodes. The data has a dimensionality of 5, the c-separation value is 1, and the maximum eccentricity 5.

Evaluation Criteria The evaluation criteria should allow us to compare the performance differences between MON-EM and standard EM. Therefore we use two measurements. First the number of EM iterations needed for convergence. Both the standard EM as the MON-EM algorithm use a stopping criterion based on the parameters. As long as the new parameters after an M-step differ from the previous parameters with a threshold of 10^{-3}, the EM algorithm will continue. Second, we use conditional entropy (Verbeek 2004) to evaluate the quality of the clustering discovered by the algorithm.

Once the parameters of the Mixture of Gaussians were learned, we clustered the data by assigning each observation to the mixture component having the highest posterior probability. This yields a confusion matrix between the real clustering and the estimated clustering. Conditional entropy measures how predictable the true label is, given the estimated label.

The conditional entropy of two variables X, Y is defined as:

$$\mathcal{H}(X|Y) = -\sum_{x \in X} p(X = x) \sum_{y \in Y} p(Y = y|X = x) \log p(Y = y|X = x) \tag{3.11}$$

The conditional entropy measures the uncertainty in X (true label) given the value of the Y (estimated label). The probabilities are estimated from the confusion matrix.

Experiment 1: Initialization In this experiment we have used the standard set to test the influence of the initialization method on the performance of EM. In table

Table 3.1: Initialization; the performance on the standard set by random and K-Means initialization.

	Standard EM	MON-EM
Conditional Entropy		
Random	1.02 ± .22	.95 ± .22
K-Means	.42 ± .15	.48 ± .15
EM-iterations		
Random	33 ± 11	34 ± 12
K-Means	11 ± 6	13 ± 10

3.1 we show the performance of MON-EM and standard EM initialized randomly and with the K-Means algorithm.

As expected MON-EM and standard EM perform almost equally on conditional entropy and on number of EM-iterations. The results show that K-Means initialized EM outperforms random initialization. The number of EM-iterations needed for convergence is also lower, but the iterations of the K-Means algorithm are not taken into account.

Experiment 2: Characteristics of the dataset In this experiment different data sets are used to investigate their influence on the performance of MON-EM and standard EM. We have generated data sets of 100 observations in $\mathbb{R}^5$. The data was drawn from a MoG with $k \in \{2, \ldots, 20\}$ components, c-separation values $c \in \{.1, \ldots, 2\}$, and the maximum eccentricity allowed was 5. The observations were distributed over $n \in \{10, \ldots, 100\}$ nodes.

In table 3.2(a), the results of MON-EM and standard EM are shown, when the c-separation value increases. The results show that both algorithms perform almost identically on the dataset, compared to conditional entropy and compared to the number of EM iterations. The results also show that, when the dataset becomes easier (i.e the c-separation value increases) the conditional entropy becomes lower, which implies that the clustering is better.

The results of MON-EM and standard EM with an increasing number of components (k) in the mixture, are shown in table 3.2(b). As expected the conditional entropy and the number of EM iterations increase for both algorithms when the number of components are increased. But, compared to each other, the algorithms perform equally well.

In table 3.2(c) the results from when the system consists of an increasing number of nodes are shown. The observations are randomly distributed over the nodes. Note that for the standard EM algorithm this does not change the learning problem. The results make clear that the number of nodes in the system does not have an influence on the performance of MON-EM, compared to standard EM. The difference in conditional entropy can probably be explained by little differences in the dataset, and not by the number of nodes used.

Experiment 3: Specific Communication In certain cases, for example when wireless communication is used, communication over a small distance is preferred above communication over a larger distance. Therefore we have experimented with MON-EM

Table 3.2: Conditional Entropy and Number of EM iterations needed to converge for standard EM and MON-EM for three experiments. In the first experiment datasets with an increasing c-separation value are generated. For the second experiment several datasets from a Mixture of Gaussian with an increasing number of kernels are generated. In the third experiment, the generated datasets are distributed over different number of nodes.

(a) C-Separation

C-separation Value	.1	.2	.3	.4
Conditional Entropy				
Standard EM	2.01 ± .07	1.78 ± .25	1.61 ± .24	1.29 ± .25
MON-EM	2.01 ± .07	1.77 ± .29	1.61 ± .22	1.31 ± .29
Number of EM iterations				
Standard EM	40 ± 17	43 ± 22	38 ± 18	31 ± 14
MON-EM	41 ± 20	39 ± 16	34 ± 15	30 ± 16

C-separation Value	.5	.75	1	2
Conditional Entropy				
Standard EM	.94 ± .34	.70 ± .38	.31 ± .27	.17 ± .19
MON-EM	.92 ± .34	.70 ± .36	.31 ± .32	.13 ± .18
Number of EM iterations				
Standard EM	30 ± 14	29 ± 22	13 ± 10	4 ± 4
MON-EM	30 ± 21	32 ± 23	13 ± 9	3 ± 3

(b) Number of Kernels

Number of Kernels	5	10	15	20
Conditional Entropy				
Standard EM	.45 ± .28	.37 ± .16	.46 ± .10	.51 ± .09
MON-EM	.43 ± .27	.38 ± .17	.44 ± .13	.50 ± .09
Number of EM iterations				
Standard EM	14 ± 8	10 ± 5	13 ± 8	18 ± 9
MON-EM	14 ± 8	11 ± 6	14 ± 6	17 ± 7

(c) Number of Nodes

Number of nodes	10	25	50	100
Conditional Entropy				
Standard EM	.41 ± .26	.37 ± .19	.31 ± .18	.37 ± .23
MON-EM	.39 ± .23	.37 ± .23	.31 ± .20	.38 ± .23
Number of EM iterations				
Standard EM	17 ± 12	16 ± 13	13 ± 7	14 ± 8
MON-EM	14 ± 9	16 ± 14	13 ± 9	13 ± 8

in a setting where only local communication is allowed. This is done by restricting the nodes with which communication is possible.

In this experiment MON-EM is tested on the standard set, but with different connection degrees. The connection degree is the percentage of nodes to which communication is allowed, a connection degree of .1 means that a node (on average) is allowed to communicate with 10% of all nodes. For each connection degree value, several communication matrices are generated, defining which nodes are allowed to communicate which each other. Each communication matrix is checked on consistency, in order to exclude isolated nodes. The matrix is consistent when each node can reach all other nodes directly, or via other nodes.

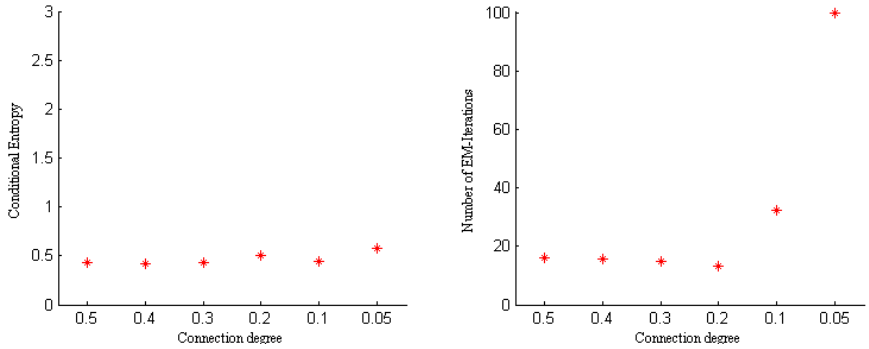


Figure 3.5: Allow specific communication, the F1 measure when only communication is allowed to a small subset of the cameras.

The experimental results are shown in figure 3.5, the results show that the conditional entropy is hardly influenced by the connection degree. However, by very low values ($\leq .1$) the number of EM iterations increases.

Experiment 4 Reduce Communication MON-EM uses quite a lot of bandwidth for each iteration of EM. This is influenced by the number of communication cycles in each EM-iteration. In this experiment we have tested MON-EM on the standard set with different values for the maximum number of communication cycles. The algorithm is not altered, only the maximum number of communication cycles is set differently.

In figure 3.6 we show the experimental results of the performance, when the maximum number of communications is limited. It is quite clear that MON-EM needs the communications to converge, however the results on conditional entropy are comparable. The differences between the parameters of each round are probably just too large to converge, while small enough to gain a comparable performance.

The MON-EM algorithm in this experiment is not altered to deal with the lack of communication. In (Kowalczyk and Vlassis 2005), a *partial* Newscast EM is presented, which is able to correct for the un-converged parameters.

Experiment 5 Reliability In a system with a lot of network traffic, it is likely that some traffic will fail. In this experiment, we tested MON-EM and standard EM while we introduced a percentage of communication errors.

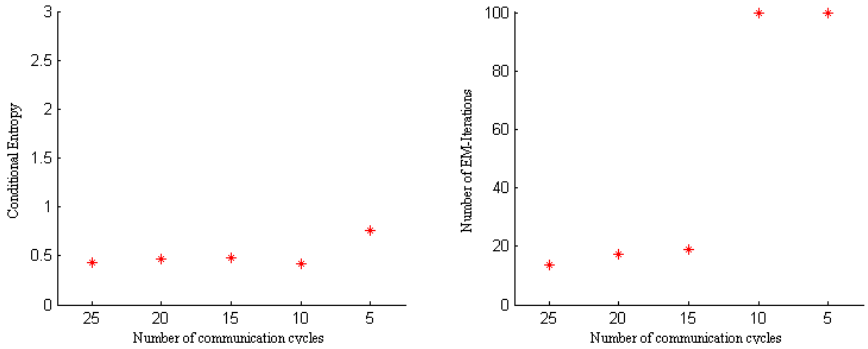


Figure 3.6: Reduce Communication, the F1 measure and number of EM iterations when the number of communication cycles in each round is reduced.

We only considered errors where the communication package was completely lost. We assume that the sender of the package does not know this, while the receiver just did not receive any communication package.

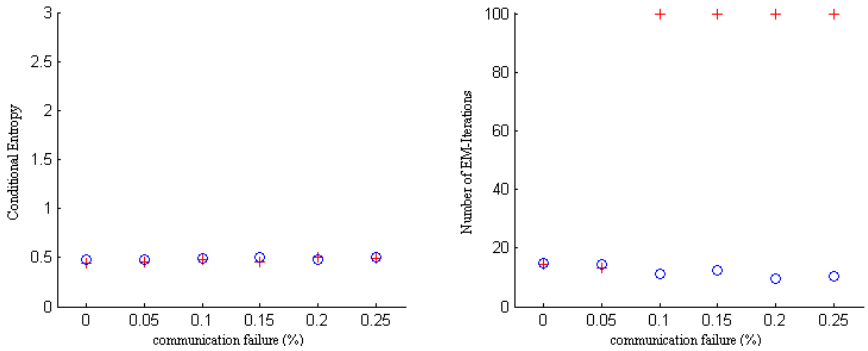


Figure 3.7: Reliability, the F1 measure and number of EM iterations when communication fails

For the standard EM algorithm, the communication failures result in less observations. As can been seen in figure 3.7, up to 25% loss of communication can be handled without a decrease of performance.

For the MON-EM algorithm a communication failure means that an update of the parameters can not be performed. Therefore the expectation is, that it will take longer before the nodes converge to the correct parameters, therefore the number of EM iterations will increase.

The results in figure 3.7 show that MON-EM will not converge within 100 EM-iterations when there is some communication failure, while the performances on conditional entropy are still comparable to standard EM. As discussed in the previous experiment, the differences between the parameters of each round are probably just too large to converge, while small enough to gain a comparable performance.

Both algorithms yield stable results with an increasing loss of communication up to 25%.

3.4 Conclusions

In this chapter we have discussed how to learn the parameters of a Mixture of Gaussian over distributed data.

First, Newscast Averaging was described, which is a gossip-based protocol for computing the mean of a distributed set of observations very quickly. Second, we have presented MON-Averaging, a more generalised version of Newscast Averaging, which allows any number of observations at a node. Third, we have proposed MON-EM, an algorithm for distributed learning of the parameters of a Mixture of Gaussians, which uses Multi-Observations Newscast Averaging. Fourth, we presented MON-KM a gossip-based implementation of the K-Means algorithm. We have used MON-KM to initialize MON-EM. The performance of K-Means initialized EM is significantly better than random initialized EM.

In the last section we have compared MON-EM to a standard EM algorithm. The experimental results show that both algorithms perform equally well.

In chapter 4 we will discuss two probabilistic models for tracking persons along multiple cameras. The models are implemented in both a central system, and in a distributed system. The central system will make use of a standard EM algorithm. The distributed system will make use of the MON-EM algorithm described in this chapter.

Chapter 4

Distributed Multi-Camera Tracking in a Wide-Area

In this chapter distributed wide-area tracking with sparsely placed cameras is considered. Here the field-of-view of one camera covers only a relatively small scene which does not overlap with scenes observed by other cameras. The focus is on camera-to-camera trajectories, therefore the manoeuvres of a person within the field-of-view of a single camera are not analysed. In figure 4.1 a sample of a wide-area video tracking setting is shown, on the left the camera layout is presented, and on the right a graph of allowed camera-to-camera paths is shown.

Cameras cannot directly observe the identity of a person, but they provide appearance measurements (like colour or height of a person) and spatial-temporal cues (like position or time) about an observation of a person. Since these measurements do not identify a individual person, a probabilistic model is used to encode the ambiguous relation between the measurement and the identity of a person.

In current systems, a single computer collects all observations from all cameras and estimates the identity and trajectory of a person from all gathered observations. In our

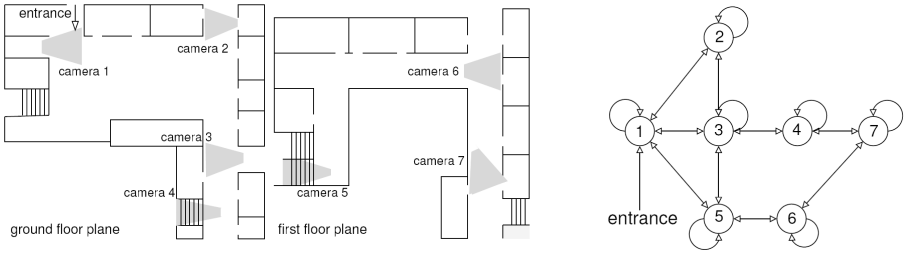


Figure 4.1: An example of wide-area tracking with multiple cameras with non-overlapping field-of-view, *left* the layout of the cameras in the building, and *right* a graph of the allowed camera-to-camera paths.

proposed distributed system, there is no central processing unit, no central repository, and no all-to-all broadcasting communication. Each camera is a stand-alone tracking unit, which stores its own observations and exchanges only limited data with other cameras. The identity and trajectory of a person are estimated by all cameras by using both their local observations and by their communication with other cameras about their observations.

In this chapter we present two models for wide-area tracking, the first is a simple appearance-based model, and the second also incorporates spatial-temporal features. Both models are off-line tracking models. We use all observation to estimate the parameters of the model, and to solve the tracking. The first model only uses appearance features for the identification of people who appear in one field-of-view and later re-appear in another field-of-view. The second model, is more elaborate and also uses spatial-temporal features for the re-identification. This allows us to enforce constraints on the motion, like excluding impossible camera-to-camera paths of a person.

Both models described have been implemented in a central and in a distributed system. In chapter 5 both models and both implementations are compared on performance. The first model, which uses only appearance features, is discussed in the section 4.1. In section 4.2 the more complex tracking model, which uses also discrete spatial-temporal features, is presented.

4.1 Tracking with Appearance Features only

In this section a probabilistic model for visual tracking is presented, which uses only appearance features. Similar to other approaches (Zajdel 2006; Kettnaker and Zabih 1999), appearance cues such as the observed average colour, or length of a person are used to find the correspondence between camera observations and persons. Because of changes in illumination and pose a person will appear differently each time he is observed. To account for these differences, we assume that the observations are samples drawn from a Gaussian distribution with person specific parameters which are constant over time. In a system where P persons are tracked, observations from all persons are generated by a Mixture of Gaussians (MoG) with P-kernels.

This distribution can be considered as the *observation* model which, together with the *transition* model, is needed to find a person's most likely track given a sequence of observations. In this section only the appearance model, as an observation model, is considered. The parameters of the MoG are learned with the Expectation-Maximization (EM) algorithm, from the appearance features. In the central implementation these parameters are learned with a standard EM algorithm (Dempster et al. 1977; Bishop 2006), while in the distributed setting these parameters are learned with the Multi-Observations Newscast Algorithm, see section 3.3. Given the learned MoG each observation is assigned to the most likely person. A track is the collection of all observations assigned to a single person.

In this section the probabilistic model for appearance based tracking is presented. The section also provides some assumptions about the tracking environment and the inference rule for assigning observations to persons.

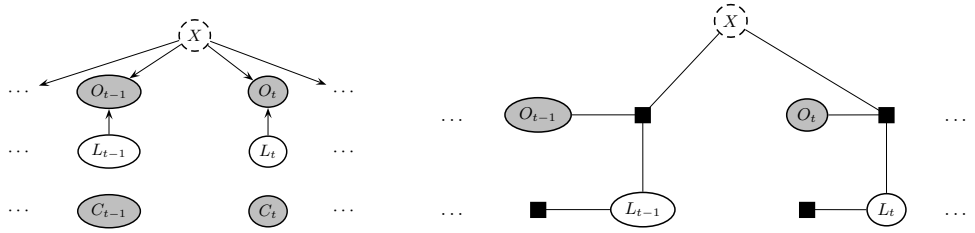


Figure 4.2: The Bayesian Network for Appearance Based Tracking (left) and the accompanying factor graph model (right)

4.1.1 Probabilistic Generative Model

In this thesis the focus is on tracking persons as they move between cameras. Therefore, the path of a person moving within the field-of-view of a single camera is not analyzed. The presence of a person in the field-of-view of a single camera results in a single observation. Such an observation, at time slice t, consists of an appearance part (O_t), and a discrete camera identifier (C_t).

The appearance features O_t denotes an r-dimensional appearance vector, which is a collection of colour features computed from various body parts of a person. In our model it is assumed that O_t is a noisy reading, generated from an MoG with parameters X. The MoG has as many kernels as there are persons tracked by the system, each person's parameters are composed of three parts: q_p, the mixing coefficient or the weight; μ_p, the r-dimensional mean vector; and Σ_p, the $r \times r$ covariance matrix. So X for P persons consists of $X = \{q_p, \mu_p, \Sigma_p\}_{p=1}^{P}$.

In the model L_t denotes the hidden variable, also called the label of a person, which identifies the person appearing at t's observation. The probability of the appearance features given the model (X) and the label (L_t) is:

$$O_t|X, L_t \sim \mathcal{N}(O_t|\mu_{L_t}; \Sigma_{L_t})$$

The second part (C_t) is the camera location where the person is seen, which is assumed to be noise free. We will call this model the *appearance model*, it defines the observation model and the probabilistic relations on the appearance features.

A Bayesian Network is a convenient way in which probabilistic relations between different variables in a system can be expressed. The generative model, which is used by the central and distributed implementation of our Appearance Based Tracking, system is shown in figure 4.2.

In this model, camera specific variations like illumination and jitter, are ignored, thus there are no links between the camera identifier and any other nodes. Also it is assumed that the number of persons tracked (P), and the number of camera's (C) in the system are known in advance.

4.1.2 Tracking People

To track persons with the gathered observations, the parameters of the Mixture of Gaussians have to be learned and the observations have to be clustered. The parameters of the MoG are learned by the EM algorithm. For the distributed implementation, the presented MON-EM algorithm is used. Where each camera has its own Bayesian model and Mixture of Gaussians model. Using MON-EM all cameras will converge to the same Mixture of Gaussians. For the central implementation, a standard EM algorithm is used. Both the MON-EM and the standard EM algorithms, are initialized with a K-Means clustering procedure.

To track people, each observation has to be assigned to the most likely person. The factor graph of this model is shown in figure 4.2 (*right*). Given the learned parameters of the MoG (X) and the current observation(O_t), the label (L_t) selected for an observation is:

$$L_t = \arg\max_{l \in L} \left\{ p(O_t|X_{(l)})p(l) \right\} \tag{4.1}$$

The collection of all observations assigned to one labels is considered to be the trajectory of one person.

4.2 Tracking with Appearance and Spatial-Temporal Features

This section presents an extended probabilistic model for multi person tracking with sparsely distributed cameras. Contrary to the previous model, not only appearance based features are considered, but also spatial and temporal features. The tracking model is enhanced with a transition model, which should exclude impossible paths between cameras. For example, in the environment shown in figure 4.1 it is impossible for a person to move from camera one to camera four without being seen by any other camera.

The focus will be on camera-to-camera trajectories, therefore the activity of a person within the field-of-view of a camera is not analysed. The frames that describe the complete pass of a person through the camera's viewing field are summarized into a single observation, this observation includes an appearance vector, a camera identifier and a time stamp.

In this model, the re-identification of a person when he re-appears in the field-of-view of any camera is based on his appearance, the movement from the previous to the current location, and the travel time. Therefore for each person the camera location and the time stamp of the last observation is stored. This can be seen as a Dynamic Bayesian Network, with an observation, and a transition model. The observation model is equal to the observation model described in the previous section. It is assumed that the appearance vector of an observation is a sample from a Mixture of Gaussians (MoG). The parameters of the MoG, have to be learned from the observations. While the transition model, formed from the possible camera-to-camera movements and associated travel times, is given.

The described model is comparable to the model of (Zajdel 2006, chapter 4), both models assume a first order transition model of movements along the cameras for a single person, where the transition model is based on spatial and temporal features. However, the model of Zajdel does not assume the number of persons tracked is known in advance. Therefore, the model is a full-order Markov model, while the described model is a first-order Markov model. Another similarity between the two models is the probabilistic relations between the observations and hidden variables. The model of Zajdel differs because it is only implemented centrally, and estimates the persons intrinsic parameters with a Normal-Inverse Wishart function instead of the EM algorithm for MoGs.

Next, the probabilistic generative model is explained, followed by a section which shows the inference in this model based on the Sum-Product algorithm from section 2.2.

4.2.1 Probabilistic Generative Model

The probabilistic generative model in the appearance-, and spatial-temporal feature tracking system is equal for the central and distributed implementation. The observation at time slice t consists of an r-dimensional appearance vector O_t, and the discrete features $\{C_t, T_t\}$ which represent the camera location and wall clock time.

The hidden variable, L_t is the unique label of the person observed at time slice t. This is the sought variable which has to be estimated from the model and the observations. The parameters of the Mixture of Gaussians ($X = \{q_p, \mu_p, \Sigma_p\}_{p=1}^{P}$) describe each person's intrinsic parameters, which do not change over time.

In the considered tracking scenario, persons are observed asynchronously; the observations from cameras are typically generated one at a time. Therefore the model is organized as a discrete-time series, where a single time slice t corresponds to a single observation $\{O_t, C_t, T_t\}$, where $t = 0, 1, \ldots$ is a discrete time index. The wall clock time T_t, is the time elapsed since the first observation.

Appearance features The observed appearance features vector O_t, is a noisy measurement generated by a Mixture of Gaussians with parameters X. The hidden unique identifier of the person L_t, the parameters X of the MoG and the observation are related as

$$O_t|X, L_t \sim \mathcal{N}(O_t|\mu_{L_t}; \Sigma_{L_t})$$

This appearance-model is equal to the appearance model described in the previous section.

Spatial-Temporal features The other component of the observation consist of the spatial-temporal features. These features are in this model, and in many others (Zajdel 2006; Kettnaker and Zabih 1999), assumed to be discrete and noise-free observed variables. The path of one person between the scenes is defined by a sequence of such features. It is convenient to assume that the current spatial-temporal features of a person rely only on the preceding features of the same person. This can be seen as a fist-order Markov model. To define a probabilistic model that generates the current features, this dependency is expressed as the distribution:

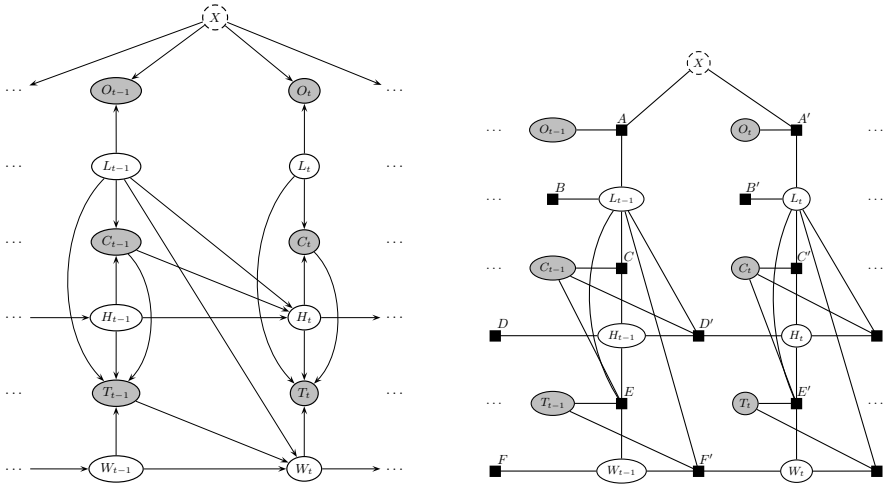


Figure 4.3: Dynamic Bayesian Network and Factor Graph On the *left* the Bayesian Network for Appearance, Spatial and Temporal Tracking and on the *right* the accompanying Factor Graph Model.

$$(C_t, T_t) \sim p(C_t, T_t | C_{prec(t)}, T_{prec(t)})$$

where $prec(t)$ is the index of the last observation of the same person as observation t. We will call this model the *motion model*, it defines the transition model from the spatial-temporal features. This model defines the motion constraints of the persons in the monitored area.

Bookkeeping variables In our model, the bookkeeping variables H_k and W_k store information from previous observations. This enables the use of a first-order motion model for a person. The bookkeeping variables are computed given the label (L_{t-1}), the camera position (C_{t-1}) and the wall clock time (T_{t-1}) of the previous observation $(t-1)$.

The camera location where person p was last observed before time slice t is stored in the hidden variable $h_{t,p}$. The vector of indicators $h_{t,p}$ for all persons is $H_t = (h_{t,1}, \ldots, h_{t,P})$. The wall clock time when person p was last observed before time slice t, is stored in the hidden variable $w_{t,p}$. The accompanying vector of the time-indicators $w_{t,p}$ for all persons is $W_t = (w_{t,1}, \ldots, w_{t,P})$.

The Dynamic Bayesian Network, which entails the defined graphical model, is shown in figure 4.3. To complete the definition of the model, a probability density function (pdf) for every vertex conditioned on its parents, as well as a-priori pdf for all vertices without parents, are provided below.

- $P(L_t)$, the a-priori probability of person p to be observed, a uniform random distribution is used for this.
- $P(O_t|L_t, X)$, the distribution on the appearance O_t of a person observed at time slice t. The appearance features O_t depend on the intrinsic person parameters

$X(L_t)$. However distortions are introduced by differences in pose or viewing angle, therefore the probability density function is modelled with a Gaussian distribution, $\mathcal{N}(O_t; X(L_t))$.

- $P(C_t|H_t, L_t)$, is the probability that person p is seen at camera c when he/she is last seen at camera $h_{t,p}$. We see the probability distribution of arriving at camera c, when departing from camera d, as a property of the environment. Therefore, in our implementation the pdf is given, and not learned from the observations. However, this distribution can also be learned from data in order to capture typical paths in a given environment. With the use of the hidden variable H_t, this is modelled with a first-order Markov motion model, $P(C_t|H_{t,L_t})$.

- $P(T_t|W_t, L_t, H_t, C_t)$, models the time of arriving at camera C_t, knowing that a person left camera $h_{(L_t)}$ at time $w_{(L_t)}$. For this we use a Gamma distribution, $P(T_t|w_{(L_t)}, h_{(L_t)}, C_t) = \Gamma(\Delta t|C_t, h_{(L_t)})$, where $\Delta t = T_t - w_{(L_t)}$, is the travel time. The travel time is the difference between the current wall clock time T_t and the previous wall clock time $w_{(L_t)}$. The expected travel time between two locations is considered as a property of the environment and is given beforehand.

- $H_t|C_{t-1}, L_{t-1}, H_{t-1}$, models the update procedure of the H_t vector given the previous observation. For each person the last location where he is observed is stored in this vector. The value of H_t is determined by:

$$P(H_t|C_{t-1}, L_{t-1}, H_{t-1}) = \begin{cases} 1 & H_{t(p=L_{t-1})} = C_{t-1} \\ 1 & H_{t,(p\neq L_{t-1})} = H_{t-1,p} \\ 0 & otherwise \end{cases}$$

  This models, that H_t is equal to H_{t-1} for all persons, except for the person seen at time $t-1$. For that person H_t should be equal to C_{t-1}.

- $W_t|T_{t-1}, L_{t-1}, W_{t-1}$, models the update procedure of the W vector at time t given the previous observation. For each person the time he was last observed is stored and maintained in this vector, this vector has a great analogy with the H_t vector. The value of W_t is determined by:

$$P(W_t|T_{t-1}, L_{t-1}, W_{t-1}) = \begin{cases} 1 & W_{t,(p=L_{t-1})} = T_{t-1} \\ 1 & W_{t,(p\neq L_{t-1})} = W_{t-1,p} \\ 0 & otherwise \end{cases}$$

4.2.2 Learning and Inference

The Mixture of Gaussians parameters X are learned using the MON-EM algorithm in the distributed implementation and in the central implementation the parameters are learned using standard EM. However, unlike X the vectors H and W are not parameters of the model, but latent variables in the model. Therefore, we will use probability distributions $q(H)$ and $q(W)$ to represent the current belief of these variables, see section 2.2. In the distributed implementation, all nodes will learn the same model parameters X, as a result of using the Multi-Observations Newscast EM algorithm. But to achieve that all cameras also share the same distributions $q(H)$ and $q(W)$, these distributions should be communicated to each other.

First the calculation of the posterior distribution on L, within one time slice is discussed. Secondly the update of the $q(H)$ and $q(W)$ distributions between two time slices is discussed. In the sections below only a brief derivation of the different messages is given. The complete derivations of the different messages are shown in appendix C.

4.2.3 Calculating the posterior distribution

To calculate the posterior probability of variable L_t, we use the sum-product algorithm (Kschischang et al. 2001), which is explained in section 2.2. These calculations use only a part of the complete factor graph. The complete factor graph is shown in figure 4.3, while the used part of the factor graph is shown in figure 4.4. The posterior probability of the hidden label L_t depends only on variables from the current time slice t. Therefore, in this section, the time slice index t is dropped.

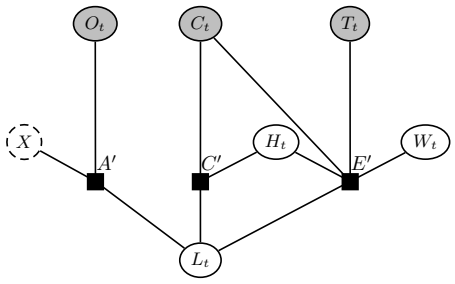


Figure 4.4: Part of Factor Graph from the full tracking model, which shows the factors used within one time slice

The Sum-Product algorithm defines messages that are sent between the nodes in the factor graph. A factor graph contains two kinds of nodes, variable nodes and factor nodes, both nodes send messages (m) to each other about the variable connected by the link. The factor graph in figure (4.4) allows the following factorization of the global function:

$$g(O, C, T, X, H, W, L) = f_{A'}(O, X, L) f_{C'}(C, H, L) f_{E'}(T, C, W, H, L)$$

The posterior probability of L can be computed with the following function:

$$g_L(L) = \underbrace{\sum_X \sum_O f_{A'}(O, X, L)}_{m_{A' \to L}(L)} \underbrace{\sum_C \sum_H f_{C'}(C, H, L)}_{m_{C' \to L}(L)} \underbrace{\sum_T \sum_W \sum_C \sum_H f_{E'}(T, C, W, H, L)}_{m_{E' \to L}(L)}$$

The message $m_{A' \to L}(L)$ from factor node A' to variable node L considers the observed variable O and the parameters X. Since these are given, the message should only exists for the given values. This is modelled with a Dirac-delta probability function, which is only 1 if and only if $O == O_t$ and similar for the parameters X. So, the message from factor node A' becomes:

$$m_{A' \to L}(L) = \mathcal{N}(O | \mu_L; \Sigma_L) \tag{4.2}$$

The message from factor node C' contains the first order motion model, that is, the probability of the current location of a person given his previous location. Variable node C is an observed variable, and therefore modelled with the Dirac-delta pdf. The message is :

$$m_{C' \to L}(L) = \sum_{h \in H} P(C|h_L)\ q(h_L) \tag{4.3}$$

where h_L signifies the entry of person L at location h; $P(C|h_L)$ is the motion model, and $q(h_L)$ is the current belief that person L is at location h.

The message from factor node E' should reflect the current belief given the transition and travel-time model. For the travel time model, the expected travel time for every pair of locations C_i, C_j is given. The variables C and T are observed variables, therefore they are modelled with a Dirac-delta pdf. From the generative model we obtain that for a specific value of $\{T, C, W, H, p\}$ holds that $f_{E'}(T, C, W, H, p) = P(T|W, p, H, C) = \Gamma(T - W|C, H(p))$. This results in the following message for person p:

$$m_{E' \to L}(p) = \sum_{h \in H} \sum_{w \in W} \Gamma(T - w_p|C, h_p)\ q(h_p)\ q(w_p) \tag{4.4}$$

With the messages from factor nodes $\{A', C', E'\}$ the posterior distribution on L can be computed as the product of the messages. The current belief of L for person p is

$$q(L(p)) = m_{f_{A' \to L}}(p)\ m_{f_{C' \to L}}(p)\ m_{f_{E' \to L}}(p), \tag{4.5}$$

The used factor graph seems to contains cycles, and therefore a loopy-belief method (Zajdel 2006; Kschischang et al. 2001) should be implemented. However, it seems reasonable to ignore the cycle connections in the factor graph, because the messages from observed variables $\{O, C, W\}$ will not change. Thus the only node which causes a cycle is node H. This is a latent variable, however, for a specific time slice t it has a given belief of where each person was before. Therefore the message sent from this variable will not change, and so the factor graph can be considered to be cycle free.

4.2.4 Updating the bookkeeping variables

The bookkeeping variables have to be updated with the observations from the previous time slice $t - 1$. The connection between two different time slices comes from factor nodes $f_{D'}$ and $f_{F'}$, see figure 4.3. Those factor nodes are responsible for the update of the H and W vectors.

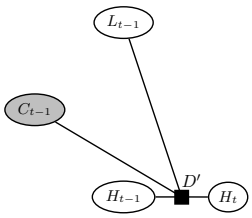


Figure 4.5: Part of Factor Graph model, the update of the H table

The message from factor node D' is based on the part of the factor graph shown in figure 4.5. To update the belief of $q(H_t)$ a message $m_{D' \to H_t}$ is sent. From the model definition, we take the following:

$$h_{p,t} = \begin{cases} C_{t-1} & \textbf{iff } L_{t-1} = p \\ h_{p,t-1} & \textbf{iff } L_{t-1} \neq p \end{cases}$$

When the message for person p and previous location c is computed, it should reflect the following: if location c is the previous location C_{t-1}, then the belief of person p, lastly seen at location c, should equal the belief of assigning L_{t-1} the value p. Otherwise, when location c differs from C_{t-1}, the belief of person p, seen lastly at location c, is the belief $h_{p,t-1}$ scaled by a factor which represents the belief person p was seen at time $t-1$. Thus $h_{p,t} = \alpha h_{p,t-1}$, where α is $1 - q(L_{t-1} = p)$. The message for person p and location c is:

$$m_{D' \to h_{p,t}}(h_{p,t} = c) = q(h_{p,t-1} = c)\ (1 - q(L_{t-1} = p)) + [c = C_{t-1}]q(L_{t-1} = p)$$

Where $[c = C_{t-1}]$ is 1 if $c = C_{t-1}$ and 0 otherwise.

The message from factor node F' is based on the part of the factor graph shown in figure 4.6. The derivation of the message is quite analogous to the derivation of the message from factor node C'.

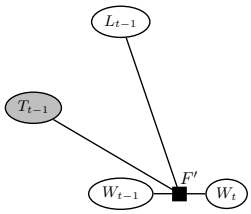


Figure 4.6: Part of Factor Graph model, the update of the W table

The message $m_{F' \to w_t}(W_t)$ is a probability table, containing all time slices and the associated probabilities that a person was last observed at that time. To calculate the message, we take from the model definition:

$$w_{t,p} = \begin{cases} T_{t-1} & \textbf{iff } L_{t-1} = p \\ w_{t-1,p} & \textbf{iff } L_{t-1} \neq p \end{cases}$$

Consider the message for a specific person p and specific time s. The belief that person p is last observed at time $T_{t-1} = s$, should equal the belief $L_{t-1} = p$. While the belief that person p is last observed at any other time, should equal the belief $W_{t-1=s,p}$ scaled by $1 - q(L_{t-1} = p)$. Thus the message for person p and time s is:

$$m_{F' \to w_{p,t}}(w_{p,t} = s) = \begin{cases} q(w_{p,t-1} = s)(1 - q(L_{t-1} = p)) & \textbf{iff } s \neq T_{t-1} \\ q(L_{t-1} = p) & \textbf{iff } s = T_{t-1} \end{cases} \tag{4.6}$$

Because this probability table will grow very fast (a new column for every person at any single time slice) a sparse representation will be used. For each person p only R

(e.g. 5) time slices are stored when the probability that person p was observed is above a certain threshold. All other time slices are considered to have zero probability and will therefore not be explicitly represented in the belief table.

So, table W is only R columns wide, the new time T_{t-1} is not included, but a message about $s = T_{t-1}$ is sent. This results in the fact that there are $R + 1$ messages for only R positions in the table. So the W table should temporarily be extended to $R + 1$ columns. After sending all messages, the column (time) with the lowest probability value should be dropped, and the table renormalized.

In the distributed setting, the bookkeeping variables are updated at the node where the observation is gathered. To use this latest information at other nodes, this information has to be shared. In our current implementation, the node updating the bookkeeping variables sent this information to all other nodes. The future research section at the end of this thesis includes some ideas on how this can be distributed using only local communication.

4.3 Summary

In this chapter two models for visual tracking in a wide area setting are described. The key difference between the two models is that the first model uses only an observation model from the appearance features, while the second model uses an observation model and a transition model, from appearance and from spatial-temporal features. Both models are implemented in a central and a distributed system, in the next chapter those systems are compared on performance using artificial generated data and real data.

Chapter 5

Multi-Person Tracking Experiments

We have performed a series of tests with the proposed models for visual tracking. The two models, one using only appearance features, and one using appearance- and spatial-temporal features, are implemented in a central system and in a distributed system. The distributed systems use the MON-EM Algorithm to learn the parameters of the Mixture of Gaussians. The distributed systems' performance will be compared to the central systems' performance, which uses a standard EM implementation.

In the text and tables below, the central tracking systems are referred to as *Central*, while the distributed systems are referred to as *Newscast*. The tracking model which uses only appearance based features is referred to as ABT, and the tracking system using appearance- and spatial-temporal features is called $ASTT$. The performance differences between both models and both implementations is examined. We evaluate the models on both artificially generated and real data.

Evaluation To compare the two different models quantitatively, we see visual tracking as an unsupervised classification problem, where observations of the same person have to be clustered together. A cluster represents a person's trajectory, we aim to recover the full trajectory of a person into a single cluster. So all observations of a person should be in one cluster, and this cluster should not have observations of other persons.

Analogously our evaluation criteria should reflect these two aspects of proper clustering: (i) all observations within a single reconstructed cluster belongs to a single person, and (ii) all observations of a single person are clustered in a single reconstructed cluster. These criteria are analogous to the *precision* and *recall* criteria which are often used in Information Retrieval settings.

$$Pr = \frac{1}{K}\sum_{k=1}^{K}\frac{\max_i |\hat{C}_k \cap C_i|}{|\hat{C}_k|} \tag{5.1}$$

$$Rc = \frac{1}{K}\sum_{i=1}^{K}\frac{\max_k |\hat{C}_k \cap C_i|}{|C_i|} \tag{5.2}$$

The precision of one cluster, is the percentage of how many observations of a proposed clustering ($\hat{C}_k$) are present in a single real cluster C_k. To deal with the unsupervised setting, the true clustering C_k that best matches a proposed clustering $\hat{C}_k$ is selected. The precision of all clusters (5.1) is defined as the average of the precisions of the single clusters.

The recall of one cluster is defined as the number of observations of the true clustering C_k are present in a single proposed cluster $\hat{C}_k$. It reflects the completeness of the recovery of the true clustering. Similar to the precision criteria, the best match is selected, and the recall of all clusters (5.2) is the average of the recall of the single clusters.

$$F1 = \frac{2 * Pr * Rc}{Pr + Rc} \tag{5.3}$$

It is trivial to gain a recall of 1, just by returning all observations into one single cluster. However, this will result in a very low precision. In order to evaluate both systems on only one parameter we use the F1-measure (5.3), which is the harmonic mean of precision and recall. It will penalize cases where a high recall and low precision (or vice versa) are returned.

5.1 Experiments on Artificial Data

Setup The artificial data is randomly generated according to the probabilistic model for appearance and spatial-temporal feature tracking, shown in figure 4.3. This allows us to evaluate both systems, ABT and ASTT, on the same dataset. Each observation consists of a 9-dimensional appearance vector O_n, a camera identifier C_n, and the wall clock time T_n. In the distributed implementations the camera identifier is also used to distribute the observations over the cameras. In the model for tracking where only appearance features are used, the appearance vector of the observations is used exclusively to learn the parameters.

Table 5.1: Experimental results on the standard set, the F1 performance and the number of EM iterations.

Normal Set		F1	EM-i
Newscast	ABT	.71 ± .13	7 ± 2
	ASTT	.77 ± .10	12 ± 7
Central	ABT	.71 ± .12	8 ± 2
	ASTT	.74 ± .10	11 ± 7

A set of 100 observations containing 5 persons, which are distributed over 25 cameras is used as a standard set. The performance of both probabilistic models (ABT & ASTT) and both implementations (Central & Newscast) on the standard set are shown in table 5.1. From the results it is clear that the extended ASTT model slightly increases the performance.

Several datasets are generated to investigate the performance of the models by variations in data difficulty, the number of persons, and the number of cameras. These experiments are discussed below.

Experiment 1 Data difficulty The difficulty of the generated data is measured by the c-separation value (Dasgupta 1999). An increasingly difficult recognition problem is indicated by decreasing c-separation values. A Mixture of Gaussians is c-separated if its components are pairwise c-separated. Components i and j are c-separated if:

$$\|\mu_i - \mu_j\|^2 \leq c \max(Tr\{\Sigma_i\}, Tr\{\Sigma_j\}).$$

A real world handwriting recognition problem described in (Dasgupta 2000), has a c-separation value of .67. While the real data (see section 5.2) used in this research have a c-separation value of .45.

Table 5.2: The F1 value of the models on different data difficulty

C-separation value		.25	.5	.75	1
Newscast	ABT	.51 ± .08	.71 ± .12	.88 ± .08	.91 ± .06
	ASTT	.58 ± .07	.81 ± .09	.93 ± .06	.96 ± .03
Central	ABT	.50 ± .05	.71 ± .13	.89 ± .08	.91 ± .05
	ASTT	.59 ± .07	.79 ± .09	.90 ± .06	.96 ± .04

We have generated different sets with c-separation values ranging from $\{.25, \ldots, 1\}$. Table 5.2 shows that the extended ASTT model always outperforms the ABT model. However, when the c-separation value increases the performance gain decreases. This is logical, because a high c-separation value indicates a simpler problem, where little can be gained by adding spatial-temporal features.

Experiment 2 Number of Cameras We want to investigate the performance of the algorithms by variations in the number of cameras. In table 5.3, the results of tracking 5 persons in an area with different number of cameras are shown. The total number of observations is the same for all experiments.

Table 5.3: The F1 value by increasing number of cameras

Number of Cameras		7	10	25	50
Newscast	ABT	.72 ± .16	.71 ± .14	.80 ± .06	.73 ± .12
	ASTT	.79 ± .14	.78 ± .10	.88 ± .05	.84 ± .08
Central	ABT	.71 ± .15	.71 ± .14	.81 ± .08	.72 ± .10
	ASTT	.81 ± .12	.80 ± .09	.86 ± .05	.84 ± .09

The performance comparison between the Central and Newscast implementation, shows that both implementations score equally well on both models. The performance gain of the ASTT model is large when there are many cameras. This is largely explained by

the fact, that when the same number of persons are tracked in a larger environment, the motion constraints weigh heavier. The persons could be more spread over the cameras. So the observation at time slice t can only be from a person from a rather small subset of all persons monitored.

Experiment 3 Number of Persons In table 5.4 we show the results of the systems when they monitor more persons. The total number of observations is 10 times the number of persons tracked. Though the number of observations per person is not fixed, but random according a uniform distribution.

The performance of the ABT model in both implementations is comparable, and it is difficult to say if it is significant influenced by the number of persons. The performance of both tracking systems increases significantly by using the ASTT model.

Table 5.4: The F1 measure of the systems by different number of persons

Number of Persons		5	10	15	20
Newscast	ABT	.78 ± .17	.71 ± .15	.78 ± .10	.69 ± .13
	ASTT	.87 ± .10	.80 ± .13	.88 ± .04	.79 ± .11
Central	ABT	.76 ± .15	.70 ± .15	.78 ± .10	.71 ± .14
	ASTT	.83 ± .09	.78 ± .13	.89 ± .04	.79 ± .10

5.2 Results on Real data

Setup We tested our tracking models on real-world human observations which were collected at seven disjoint locations at the university building. The layout of the cameras and the transition model are shown in figure 5.1. In total we gathered 70 observations of 5 persons, with an equal number of observations per person. For this set the data association is manually resolved to have a ground truth clustering. This data set is also used in (Zajdel 2006).

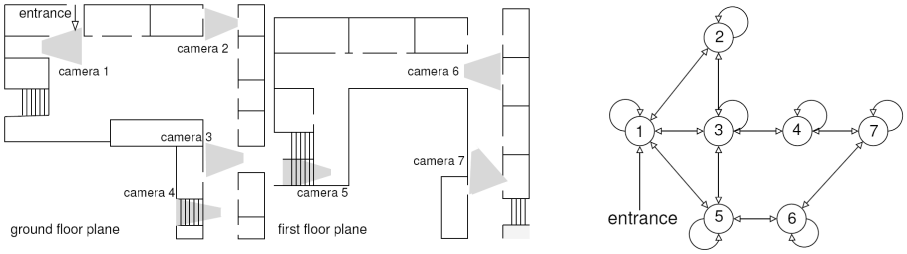


Figure 5.1: A sample layout with seven camera's (left) and the corresponding transition model (right)

The assumptions of Gaussian distributed noise in appearance features (artifacts due to illumination and pose) will most likely not hold without suitable preprocessing of the images. To minimize the effects of variable illumination at various cameras (intensity and colour), we use a so-called channel-normalized colour space (Drew et al. 1998). To minimize non-Gaussian artifacts due to pose, we use geometric colour mean (Zajdel 2006), which is shown in figure 5.2. The geometric colour mean is the mean colour of three separate regions (R_1, R_2, R_3) of the observed image. This results in a 9-dimensional appearance vector. The image of an observed person is split into three horizontal regions with fixed height ratios. The skip area correspond to an image region that usually provides little information. The resulting features are to a certain extent invariant to variations in a person's pose.

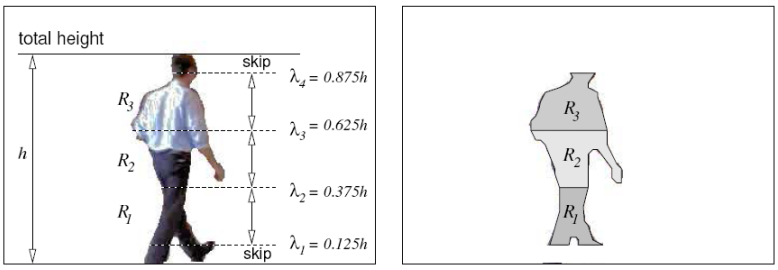


Figure 5.2: Computation of appearance features. The person detected in a single frame is divided in three regions with an equal height ratio. Within a region the average colour is computed in RGB colour-space, this results in a 3-dimensional vector. The complete appearance is described as a 9-dimensional feature vector.

The results of the Newscast implementation and the Central implementation on the real dataset are shown in table 5.5. The table shows the F1-measure and the number of EM iterations for both models (ABT and ASTT).

Table 5.5: Performance on Real data

Real data		F1	EM-i
Newscast	ABT	.64 ± .04	8 ± 2
	ASTT	.68 ± .06	10 ± 5
Central	ABT	.63 ± .03	8 ± 3
	ASTT	.68 ± .05	10 ± 4

The table shows that both implementations score almost identical on both performance measurements. It also shows that the extended ASTT model outperforms the ABT model. However the performance gain of the ASTT model is rather small.

To illustrate the tracking capabilities of both algorithms, we show in figures 5.3 and 5.4 the resulting trajectories from a run of the Newscast ABT and Newscast ASTT algorithms, and the ground truth belonging to the dataset. Each row in the figures represents an estimated trajectory of one person.

The tracking problem is seen as a unsupervised data association problem. Thus the different ordering of the the trajectories will not influence the performance. The F1 measure for the shown ABT clustering is .65, and the F1 measure for the shown ASTT clustering it is .74.

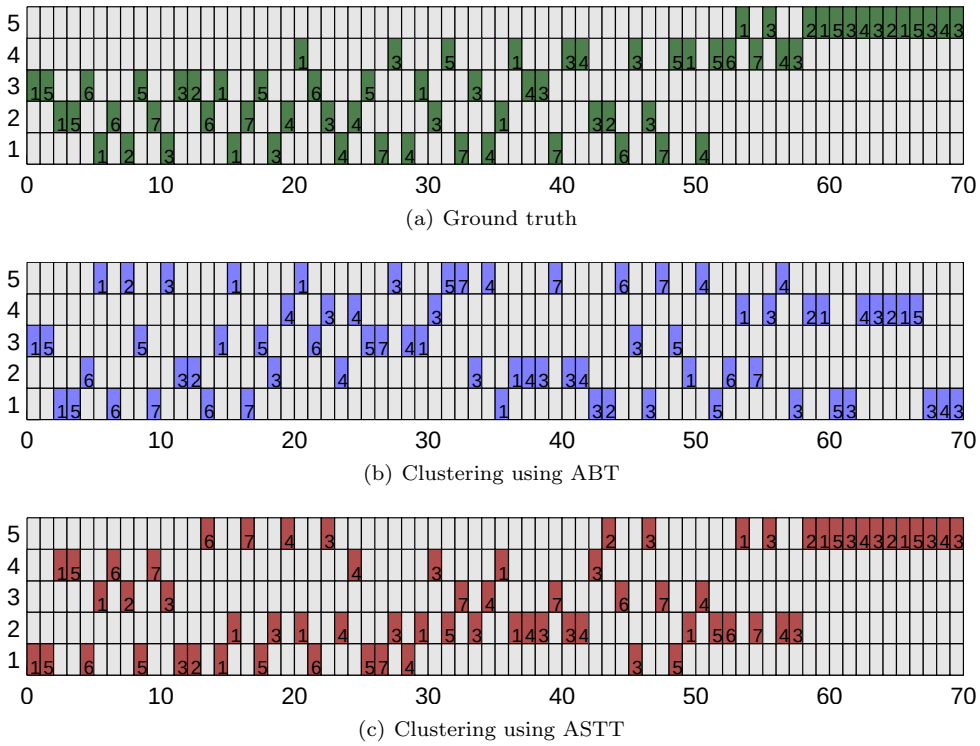
Figure 5.3 shows concisely the estimated trajectories of all persons. In the figure each observation $(1, \ldots, 70)$ is represented by a box in the row of the (estimated) label $(1, \ldots, 5)$. The number in the box of an observation, is the camera index $(1, \ldots, 7)$ where the observation was gathered.

From figure 5.3 we also can compute the number of transition errors. We consider a transition error to be a person's movement (according to an estimated trajectory) from one camera to another camera which is not a valid transition, according to the transition model. For example, according to the transition model (see figure 5.1) the movement from camera 5 to camera 7 is not directly possible. In the ABT model, when the transition model is not taken into account, the number of transition errors is 13. In the ASTT model, which includes the transition model, the number of transition errors is reduced to 6.

The observations are shown in figure 5.4, ordered according to the estimated trajectories. In the ground truth figure, we see that the colours of a single person in the observations truly differ.

Neither the ABT nor ASTT model, has estimated a complete trajectory. While most trajectories of the ABT model are messed up, the estimated trajectories of the ASTT models shows at least significant fragments of the true trajectories.

The ASTT model, has estimated two trajectories (3 & 4) with a precision of 1, which means that all observations in the trajectory belongs to the same person. Also, the ASTT model has estimated one trajectory (1) with a recall of 1, which means that all observations of a single person are returned into one trajectory.



(a) Ground truth

(b) Clustering using ABT

(c) Clustering using ASTT

Figure 5.3: Example Clustering of the persons from the real-world data set. Each label $(1, \ldots, 5)$ is represented as a row. For each observation $(1, \ldots, 70)$ a box is drawn at the row of the (estimated) label, inside the box the camera index $(1, \ldots, 7)$ where the observation is gathered is given.

(a) Ground truth

(b) Clustering using ABT

(c) Clustering using ASTT

Figure 5.4: Example Clustering of the persons from the real-world data set. Each row represents an (estimated) trajectory of the ground truth, the Newscast ABT and the Newscast ASTT systems respectively.

5.3 Conclusions

The results of the Newscast implementation and the central implementation on artificial data shows that both implementations perform equally well. even in different settings, with a different number of cameras, or a different number of persons. We can conclude that the performances of the Newscast implementation using MON-EM, and the central implementation using standard EM are equal to each other.

The performance is comparable for both the F1-measurement as well as for the number of EM iterations needed to convergence. Also, in many other different test settings we have run, both algorithms perform the same.

The use of the more elaborative ASTT model has advantages above the simple ABT model. These advantages becomes certainly clear when more persons are tracked, or when more cameras are used.

The results on real data show also that standard EM and Multi-Observations Newscast EM perform almost the same. The performance increase of the ASTT model, compared to the ABT models is relatively small. The c-separation and eccentricity values of the real data differs from the artificial data, which explains why the F1-measure is lower than on the standard set of the artificially generated data.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis we have considered distributed tracking of persons with multiple cameras. For this tracking the parameters of a Mixture of Gaussians have to be learned. We have developed Multi-Observations Newscast EM for learning these parameters from a set of observations which are distributed over the cameras. Conclusions with respect to Multi-Observations Newscast EM are given in section 6.1.1, and conclusions with respect to Distributed Multi-Camera tracking are given in section 6.1.2.

6.1.1 Multi-Observations Newscast EM

Multi-Observations Newscast EM (MON-EM), is a gossip-based distributed algorithm, to learn the parameters of a Mixture of Gaussians (MoG). It relies on the MON-Averaging algorithm for calculating the mean of a set of distributed values, as presented in this thesis. To improve the performance of MON-EM, we have introduced MON-KM as an initialization algorithm.

MON-Averaging is a gossip-based protocol to calculate the mean of a set of distributed values, it runs in parallel on all nodes of a network. Each node starts with a local estimate of the mean followed by a number of communication cycles until convergence, in which pairs of nodes repeatedly exchange their local estimates and combine them by weighed averaging. We have theoretically proven that, using MON-Averaging, the local estimates of each node will converge exponentially fast to the correct mean. An experiment has revealed that the convergence rate of MON-Averaging is not influenced by the distribution of the observations over the nodes.

We have presented a distributed K-Means algorithm MON-KM as initialization method for MON-EM. MON-KM uses MON-Averaging to update the mean of each cluster.

The performance of MON-KM is equal to a standard implementation of the K-Means algorithm, where all observations are at a central repository.

MON-EM is a gossip-based EM algorithm, which learns the parameters of an MoG from distributed observations. It is a generalisation of Newscast EM presented in (Kowalczyk and Vlassis 2005), the original algorithm assumes that every node has exactly one observation. Our developed algorithm learns from the more realistic situation where a node could have any number of observations.

The main difference between standard EM and MON-EM, is that the M-step is implemented as a sequence of gossip-based cycles. In the E-step, the responsibilities are computed at each node for the locally available data, given the current parameters. In the M-step, the updates of the parameters, which are averages over all data, are calculated using MON-Averaging. After the M-step the local parameters of all nodes are equal to each other. Also, the local parameters after each M-step are equal to the standard EM parameters for the current EM iteration.

Experiments have shown that MON-EM performs almost identically to standard EM. For example, on a set of 100 observations, distributed over 25 nodes, and generated from a 5 component MoG, the conditional entropy of standard EM was $.42 \pm .15$, and the conditional entropy for MON-EM was $.48 \pm .15$. The performance of MON-EM compared to standard EM, with respect to conditional entropy and number of EM iterations, is in many settings equal and independent of the number of cameras, number of observations, number of components and the data difficulty.

6.1.2 Distributed Multi-Camera Tracking

The MON-EM algorithm is used for distributed multi-camera tracking. In this setting each camera has a non-overlapping field-of-view with the other cameras. The tracking task is to cluster the observations of people, gathered at the different cameras, into trajectories. In our distributed system each camera is a stand-alone tracking unit, which stores its observations locally. The cameras solve the tracking by considering their own observations and through communication with each other.

A Dynamic Bayesian Network (DBN) is defined to encode the probabilistic model of the tracking problem. The DBN consists of an appearance model and a motion model. Generated observations consist of appearance features and spatial-temporal features. The appearance features of a person are assumed to be samples drawn from an MoG, with for each person a kernel with unique parameters. The spatial-temporal features, camera index and wall clock time, are assumed to be noise free.

For tracking, we have implemented two probabilistic models. The first model comprises the appearance model only. While the second model consist of the appearance model and the motion model. Both models are implemented in a central system and in a distributed system. The central system uses a standard EM algorithm to learn the parameters of the appearance model. While the distributed system uses the MON-EM algorithm to learn these parameters.

We have run several experiments with both models and both implementations. The results show that both implementations perform equally well. While the more elaborate

model, which includes the appearance model and motion model, outperforms the simple model. This becomes even more clear in settings with many persons to track, or where many cameras are used. In a setting where 100 observations are artificially generated for 5 persons, with 25 cameras, the simple appearance model has a F1 measure of $.80 \pm .06$, while the appearance- and motion-model has a F1 measure of $.88 \pm .05$.

Both probabilistic models and both implementations are also tested on real world data. The real data consists of 70 observations of 5 persons, gathered at the university building. The performance of the distributed implementation and the centralised implementation on the real data is equally well. However, there is little performance gain using the elaborate probabilistic model.

To conclude, this thesis revealed that MON-EM performs equally well to standard EM, and we have shown its use in Distributed Multi-Camera Tracking. We have theoretically proven that learning from distributed data does not have to be an approximation of learning from central data. Learning from distributed data with MON-EM is equal to learning from central data with standard EM, this will hold for any kind of data. The performance of MON-EM is independent of the distribution of the observations over the nodes, and the number of nodes.

6.2 Future Research

During the research into MON-EMseveral possible improvements contrived. For example, MON-EM could be altered into a *greedy* EM method, which gradually yields more kernels. Instead of starting with an estimation of the parameters for all kernels, the Mixture of Gaussians (MoG) is build kernel-wise. The algorithm starts with the optimal one-kernel MoG and starts repeating two steps: first adding a new kernel and second appling the EM algorithm (Verbeek 2004). The motivation for the greedy approach is that it is expected to be easier than finding the optimal starting estimation of the parameters. Greedy Multi-Observations Newscast EM could be implemented according to ideas presented in (Verbeek et al. 2003; Vlassis et al. 2005). This could be useful for tracking problems where the number of persons is not known in advance.

The bandwidth usage of each M-update cycle of MON-EM is rather high. To reduce the bandwidth usage, we can consider to lower the number of M-update steps. This could possibly be done by running several EM iterations locally before communicating the parameters to the other nodes. A distributed EM algorithm based on a fixed routing scheme, with multiple local EM iterations before communicating is presented in (Nowak 2003).

In this research MON-EM for MoG parameter estimation is used. The MON-EM algorithm relies on the use of Multi-Observations Newscast Averaging (MON-Averaging), to update the parameters. However, the ideas from MON-Averaging could be used to calculate almost any kind of statistics from a set of distributed values, for example the mean value, the maximum or minimum value, and the variance. This allows the use of any kind of statistical learning method, in a distributed system. Also, instead of the MoG model, any other statistical model of data could be used.

The gossip-based Newscast algorithm assumes a flat hierarchy of nodes, in a network

where arbitrary point-to-point communication is possible. In this thesis we have shown, with an experiment, that MON-EM also performs well when each node has only a restricted number of other nodes to communicate with. This possibly enables to use a more structured hierarchy of nodes, where some nodes only communicate to nearby nodes, and some other nodes communicate over greater distance. This could be useful when MON-EM is used in a wireless sensor system.

Other possible directions of future research are more specific to the use of a distributed system for multi-camera tracking. In the described probabilistic models several hidden variables are used, which denote the previous location and previous time of an observation. These variables are used in the motion model. In our implementation these variables are shared by broadcasting them, while it should be possible to use only neighbour communication.

A possible update scheme would be, when a camera receives a new observation, it pulls the current hidden states from all its neighbours. The camera merges these hidden states into a consistent one, and assigns the observation to the most likely track given this state. Next, the camera sends his updated hidden state to its neighbours. The hidden state at each camera is not necessary complete, it is not guaranteed that all information is included. However it should contain enough information about the persons seen at the neighbouring cameras.

The described probabilistic model which uses an appearance model and a motion model requires that the number of persons tracked in the area is known in advance. That is quite an assumption for any real-life application. The model described in this thesis and the model presented in (Zajdel 2006) are quite similar. For example, both models assume that the trajectory of a single person is a first-order Markov Model. However, the model from Zajdel is more general, it does not require to know the number of people.

Bibliography

Bandyopadhyay, S., Giannella, C., Maulik, U., Kargupta, H., Liu, K., and Datta, S. (2006). Clustering Distributed Data Streams in Peer-to-Peer Environments. *Information Sciences*, volume 176(14), pages 1952–1985.

Bilmes, J. (1997). Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. *Technical report*, University of Berkely.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blatt, D. and Hero, A. O. (2004). Distributed Maximum Likelihood Estimation in Sensor Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Cucchiara, R., Grana, C., Prati, A., and Vezzani, R. (2005). Computer Vision System for In-House Video Surveillance. *Proceedings of the IEE Vision, Image, and Signal Processing*, volume 152(2), pages 242–249.

Dasgupta, S. (1999). Learning Mixtures of Gaussians. In *Proceedings of the Symposium on Foundations of Computer Science*, page 634. IEEE Computer Society, Washington, DC, USA.

Dasgupta, S. (2000). Experiments with Random Projection. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 143–151. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Dekkers, S. and Homburg, G. (2006). Evaluatie Cameratoezicht op Openbare Plaatsen. By order of the Dutch Department of Home Affairs.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the *EM* Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, volume 39(1), pages 1–38.

Drew, M. S., Wei, J., and Li, Z.-N. (1998). Illumination-Invariant Color Object Recognition via Compressed Chromaticity Histograms of Color-Channel-Normalized Images. In *Proceedings of the International Conference on Computer Vision*, pages 533–540.

Flight, S. and Hulshof, P. (2006). Evaluatie cameratoezicht Wallen en Nieuwendijk Amsterdam - Verslag van vier metingen: 2003, 2004, 2005 en 2006. By order of the city counsel of Amsterdam.

Forman, G. and Zhang, B. (2000). Distributed Data Clustering can be Efficient and Exact. *SIGKDD Explor. Newsl.*, volume 2(2), pages 34–38.

Gavrila, D. M. (1999). The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, volume 73(1), pages 82–98.

Jelasiy, M., Kowalczyk, W., and van Steen, M. (2003). Newscast Computing. *Technical report*, Vrije Universiteit Amsterdam.

Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-Based Computation of Aggregate Information. *Proceedings of the IEEE symposium on Foundations of Computer Science*, page 482.

Kettnaker, V. and Zabih, R. (1999). Bayesian Multi-Camera Surveillance. *Proceedings of the IEEE Computer Society Conference on Conmputer Vision and Pattern Recognition*, volume 2, pages 2253–2261.

Kowalczyk, W. and Vlassis, N. (2005). Newscast EM. *Advances in Neural Information Processing Systems*, volume 17.

Kröse, B., Vlassis, N., and Zajdel, W. (2004). Bayesian Methods for Tracking and Localization. In *Proceedings of the Philips Symposium On Intelligent Algorithms*, pages 27–38.

Kschischang, F., Frey, B., and Loeliger, H.-A. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, volume 47, pages 498–519.

Lin, X., Clifton, C., and Zhu, M. (2005). Privacy-Preserving Clustering with Distributed EM Mixture Modeling. *Knowledge and Information Systems*, volume 8(1), pages 68–81.

Masoud, O. and Papanikolopoulos, N. (2003). A Method for Human Action Recognition. *Image and Vision Computing*, volume 21(8), pages 729–743.

Mensink, T., Zajdel, W., and Krose, B. (2007). Distributed EM Learning for Appearance Based Multi-Camera Tracking. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*. To appear.

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

Nakazawa, A., Kato, H., and Inokuchi, S. (1998). Human Tracking Using Distributed Vision Systems. *Proceedings of the International Conference on Pattern Recognition*, volume 01, page 593.

Nguyen, N., Bui, H., Venkatesh, S., and West, G. (2003). Recognising and Monitoring Highlevel Behaviours in Complex Spatial Environments. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

Nikseresht, A. and Gelgon, M. (2006). Fast Decentralized Learning of a Gaussian Mixture Model for Large-Scale Multimedia Retrieval. *Proceedings of the Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, volume 0, pages 373–379.

Nowak, R. (2003). Distributed EM Algorithms for Density Estimation and Clustering in Sensor Networks. *IEEE Transactions on Signal Processing*, volume 51, pages 2245–2253.

Remagnino, P., Shihab, A., and Jones, G. (2004). Distributed Intelligence for Multi-Camera Visual Surveillance. *Pattern Recognition*, volume 37(4), pages 675–689. Special Issue on Agent-based Computer Vision.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.

Stewart, J. (2001). *Calculus Concepts and Contexts*. Brooks - Cole.

Sycara, K. P. (1998). Multiagent Systems. *AI Magazine*, volume 19(2), pages 79–92.

Tasoulis, D. and Vrahatis, M. (2004). Unsupervised Distributed Clustering. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*.

Valera, M. and Velastin, S. (2005). Intelligent Distributed Surveillance Systems: a review. *Proceedings of the IEE Vision, Image, and Signal Processing*, volume 152(2), pages 192–204.

Verbeek, J. (2004). *Mixture Models for Clustering and Dimension Reduction*. Ph.D. thesis, Universiteit van Amsterdam.

Verbeek, J., Vlassis, N., and Kröse, B. (2003). Efficient Greedy Learning of Gaussian Mixture Models. *Neural Computation*, volume 15, pages 469–485.

Vlassis, N., Sfakianakis, Y., and Kowalczyk, W. (2005). Gossip-Based Greedy Gaussian Mixture Learning. In *Proceedings of the Panhellenic Conference on Informatics*.

Webb, A. R. (2002). *Statistical Pattern Recognition*. John Wiley and Sons Ltd.

Weitenberg, A., Jansen, E., van Leiden, I., and Ferwerda, H. (2003). Cameratoezicht: De menselijke factor. By order of the Programme of Police and Science.

Zajdel, W. (2006). *Bayesian Visual Surveillance - From Object Detection to Distributed Cameras*. Ph.D. thesis, Universiteit van Amsterdam.

Zajdel, W., Krijnders, J., Andringa, T., and Gavrila, D. (2007). CASSANDRA: Audio-Video Sensor Fusion for Aggression Detection. In *Proceedings of the IEEE International Conference on Advanced Video and Signal-based Surveillance*. London UK. To appear.

Zivkovic, Z. and Krose, B. (2004). An EM-like Algorithm for Color-Histogram-Based Object Tracking. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 798–803.

Appendix A

Derivation of EM-Algorithm for Mixture of Gaussians

The derivation in this section was established by combining parts from (Bishop 2006) and (Webb 2002). The goal is to present a more elaborative derivation of the EM algorithm for Mixtures of Gaussians, than was provided in section 2.3.

A Mixture of Gaussians (MoG) can be described as a linear combination of K Gaussian distributions, in the form:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{A.1}$$

$$\mathcal{N}(x_n|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{A.2}$$

Where x is a data point in $\mathcal{R}^d$, and where π_k is the mixing coefficient, μ_k the mean, and Σ_k the covariance matrix of kernel k. The parameters are also denoted as the vector $\theta = \{pi_k, \mu_k, \Sigma_k\}_{k=1}^K$.

Suppose we have a set of N observations $\{x_1, \ldots, x_N\}$, then the log of the likelihood is given by:

$$\ln\{p(X|\theta)\} = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)\right\} \tag{A.3}$$

The maximization of this function is more complex than for a single Gaussian, because of the summation over K. If we set the derivative of the log-likelihood to zero, we will not obtain a closed form solution.

The Expectation-Maximization algorithm (Dempster et al. 1977) is a method to estimate the parameters of the MoG with maximum-likelihood. It alternates between two steps, the Expectation step and the Maximization step.

The basic procedure is as follows. We suppose that we have a set of 'incomplete' data vectors $\{x_1, \ldots, x_N\}$ and we wish to maximize the likelihood (A.3). Let us denote $\{y_1, \ldots, y_N\}$ a typical 'complete' version of $\{x\}$, that is, each vector x_n is augmented by the missing class label z_n. Usually a binary vector $z_n = (z_{1n}, \ldots, z_{Kn})$ is used, where $z_{kn} = 1$ if x_n belongs to the kth component and zero otherwise.

In the E-step we calculate the the expectation of the class labels given the data and the current parameters:

$$q_n = E(z_n|x_n, \theta)$$

Where $q_n(i)$ is the probability that x_n belongs to class i is given by (A.4). The $q_n(i)$ is also called the responsibility, because it represents the responsibility of cluster i for generating observation n.

$$q_n(k) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{A.4}$$

In the M-step we maximize the parameters given the current responsibilities q. We set the derivative of (A.3) to zero with respect to the different parameters.

$$0 = \frac{d}{d\theta} \ln \{p(X|\theta)\} \tag{A.5}$$

Updating the Mean
To obtain the derivative with respect to the mean μ_k, we use the following mathematical rules

$$\begin{aligned} \frac{d}{dx} ln\{f(x)\} &= \frac{f'(x)}{f(x)} \\ \frac{d}{dx} exp^{f(x)} &= f'(x) * exp^{f(x)} \\ \frac{d}{dx} x^T A x &= (A + A^T)x \\ \Sigma_k^{-1} &= (\Sigma_k^{-1})^T \end{aligned}$$

The first three are standard derivation rules. The latter comes from the fact that the covariance matrix holds $\Sigma_{ij} = \Sigma_{ji}$. With those rules, the derivative of (A.3) is acquired as follows:

$$\frac{d}{d\mu} \ln \{p(X|\theta)\} = \sum_{n=1}^{N} \left\{ \frac{\frac{d}{d\mu} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \right\} \tag{A.6}$$

$$\frac{d}{d\mu} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) = \pi_k \frac{d}{d\mu} \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp^{-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)} \tag{A.7}$$

$$= \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) * \frac{d}{d\mu} \left\{ -\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k) \right\} \tag{A.8}$$

$$\frac{d}{d\mu}\left\{-\frac{1}{2}(x_n-\mu_k)^T\Sigma_k^{-1}(x_n-\mu_k)\right\} = -\frac{1}{2}(\Sigma_k^{-1}+(\Sigma_k^{-1})^T)(x_n-\mu_k) \tag{A.9}$$

$$= -\Sigma_k^{-1}(x_n-\mu_k) \tag{A.10}$$

Putting all this together, and setting the derivation to zero results in:

$$0 = -\sum_{n=1}^{N}\left(\frac{\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}\Sigma_k^{-1}(x_n-\mu_k)\right) \tag{A.11}$$

The first part equals to the probabilities calculated in the E-step (A.4). Multiplying (A.11) with Σ_k and rearranging we obtain the update formula for the mean μ_k of kernel k (A.12).

$$\mu_k = \frac{1}{\sum_{n=1}^{N}q_n(k)}\sum_{n=1}^{N}q_n(k)x_n \tag{A.12}$$

Updating the Covariance
To obtain the derivative (A.5) with respect to the covariance matrix Σ_k, we use the following differentiation rules for matrices:

$$\frac{d}{dA}|A^{-1}| = -|A^{-1}|(A^{-1})^T$$
$$\frac{d}{dA}x^TA^{-1}y = -(A^{-1})^Txy^T(A^{-1})^T$$

With those rules, the derivative of (A.3) with respect to Σ_k is acquired as follows:

$$\frac{d}{d\Sigma}\ln\{p(X|\theta)\} = \sum_{n=1}^{N}\left\{\frac{\frac{d}{d\Sigma}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}\right\}$$

$$\begin{aligned}\frac{d}{d\Sigma}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k) &= \pi_k\frac{d}{d\Sigma}\frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}}\exp^{-\frac{1}{2}(x_n-\mu_k)^T\Sigma_k^{-1}(x_n-\mu_k)}\\ &= \pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)*\frac{1}{2}(\Sigma_k^{-1})^T\\ &\quad+\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)*\frac{1}{2}(\Sigma_k^{-1})^T(x_n-\mu_k)(x_n-\mu_k)^T(\Sigma_k^{-1})^T\\ &= \pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)*\frac{1}{2}\left\{(\Sigma_k^{-1})^T+(\Sigma_k^{-1})^T(x_n-\mu_k)(x_n-\mu_k)^T(\Sigma_k^{-1})^T\right\}\end{aligned}$$

Putting this together and using the responsibilities q obtained from the E-step we get:

$$\frac{d}{d\Sigma}\ln\{p(X|\theta)\} = \sum_{n=1}^{N}q_n(k)\,\frac{1}{2}\left((\Sigma_k^{-1})^T+(\Sigma_k^{-1})^T(x_n-\mu_k)(x_n-\mu_k)^T(\Sigma_k^{-1})^T\right) \tag{A.13}$$

Which results in the following update rule for covariance matrix Σ_k for the kth kernel:

$$\Sigma_k = \frac{1}{\sum_{n=1}^{N} q_n(k)} \sum_{n=1}^{N} q_n(k)(x_n - \mu_k)(x_n - \mu_k)^T \tag{A.14}$$

Updating the Mixing Coefficient
Finally, the log-likelihood with respect to the mixing coefficients π_k has to be maximized. This maximization is subjected to the constraint $\sum_k \pi_k = 1$. This can be achieved by using a Lagrange multiplier.

Maximizing (A.3) with respect to π_k leads to the equation:

$$0 = \sum_{n=1}^{N} q_n(k) \frac{1}{\pi_k} - \gamma$$

We find that the Lagrange multiplier $\gamma = \sum_{k=1}^{K} \sum_{n=1}^{N} q_n(k) = N$. Thus the update function for pi_k is:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} q_n(k) \tag{A.15}$$

This completes the maximization step of the EM algorithm. The E-step and M-step are repeated until no further improvement is made.

Appendix B

Newscast Averaging Variance Reduction

In this appendix the variance reduction proof given in section 3.2 is explained in more detail.

In Multi-Observations Newscast Averaging each node i stores a number of n_i observations $\{x_{i,1}, \ldots, x_{i,n_i}\}$, and has a local estimate $\hat{\mu}_i$ of the global mean μ. The weighed mean of the local estimates, $\sum_i n_i \hat{\mu}_i$ always equals the global mean μ. The variance of the local estimates represents how large the differences between the estimates $\hat{\mu}_i$ and the mean of the local estimates are. When the variance tends to zero, the local estimations are converged to the correct mean.

We define the un-normalized variance of the local estimates $\hat{\mu}_i$, at cycle t as: $\Phi_t = \sum_{i=1}^{c} (\hat{\mu}_i - \mu)^2$.

Lemma In each cycle of Multi-Observations Newscast Averaging the variance of the local estimates decrease on average by γ, with $\gamma \leq \frac{1}{2\sqrt{e}}$.

Proof Suppose that the mean of the observations is μ, and that within cycle t, the nodes initiate contact in the order $c_1, c_2, \ldots, c_c$. The current local estimate at node c_i is $\hat{\mu}_i$. To calculate the variance $\Phi_{t,1}$ after node c_1 contacts node c_i, we take $\Phi_{t,0}$ subtract the variance of local estimates $\hat{\mu}_1$ and local estimate $\hat{\mu}_i$ before communication, and add the variance of the local estimates after the communication $\hat{\mu}'_1$ and $\hat{\mu}'_i$:

$$\Phi_{t,1} = \Phi_{t,0} - (\hat{\mu}_1 - \mu)^2 - (\mu_i - \mu)^2 + 2\left(\frac{w_1\mu_1 + w_i\mu_i}{w_1 + w_i} - \mu\right)^2 \tag{B.1}$$

This last part can be rewritten to

$$
\begin{aligned}
2\left(\frac{w_1\hat{\mu}_1 + w_i\hat{\mu}_i}{w_1 + w_i} - u\right)^2 &= 2\left(\frac{w_1\hat{\mu}_1}{w_1 + w_i} + \frac{w_i\hat{\mu}_i}{w_1 + w_i} - u\right)^2 \\
&= \frac{2w_1^2}{(w_1 + w_i)^2}\hat{\mu}_1^2 + \frac{4w_1\, w_i}{(w_1 + w_i)^2}\hat{\mu}_1\hat{\mu}_i - \frac{4w_1}{(w_1 + w_i)}\hat{\mu}_1\mu \\
&\quad + \frac{2w_i^2}{(w_1 + w_i)^2}\hat{\mu}_i^2 - \frac{4w_i}{(w_1 + w_i)}\hat{\mu}_2\mu + 2\mu^2 \\
&= \frac{2w_1^2}{(w_1 + w_i)^2}\hat{\mu}_1^2 + \frac{4w_1\, w_i}{(w_1 + w_i)^2}\hat{\mu}_1\hat{\mu}_i - \frac{4(w_1^2 + w_1w_i)}{(w_1 + w_i)^2}\hat{\mu}_1\mu \\
&\quad + \frac{2w_i^2}{(w_1 + w_i)^2}\hat{\mu}_i^2 - \frac{4(w_i^2 + w_iw_1)}{(w_1 + w_i)^2}\hat{\mu}_2\mu + \frac{2(w_1^2 + w_i^2 + 2w_1w_i)}{(w_1 + w_i)^2}\mu^2
\end{aligned}
$$

Using these three equations:

$$
\begin{aligned}
\frac{2w_1^2}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)^2 &= \frac{2w_1^2}{(w_1 + w_i)^2}(\hat{\mu}_1^2 - 2\hat{\mu}_1\mu + \mu^2) \\
\frac{2w_i^2}{(w_1 + w_i)^2}(\hat{\mu}_i - \mu)^2 &= \frac{2w_i^2}{(w_1 + w_i)^2}(\hat{\mu}_i^2 - 2\hat{\mu}_i\mu + \mu^2) \\
\frac{4w_1w_i}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)(\hat{\mu}_i - \mu) &= \frac{4w_1w_i}{(w_1 + w_i)^2}(\hat{\mu}_1\hat{\mu}_i - \hat{\mu}_1\mu - \hat{\mu}_i\mu + \mu^2)
\end{aligned}
$$

this can be simplified to:

$$
\begin{aligned}
2\left(\frac{w_1\hat{\mu}_1 + w_i\hat{\mu}_i}{w_1 + w_i} - u\right)^2 &= \frac{2w_1^2}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)^2 + \frac{2w_i^2}{(w_1 + w_i)^2}(\hat{\mu}_i - \mu)^2 \\
&\quad + \frac{4w_1w_i}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)(\hat{\mu}_i - \mu)
\end{aligned}
$$

Using the equation above we can rewrite (B.1) to:

$$
\Phi_{t,1} = \Phi_{t,0} - \frac{2w_1^2}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)^2 - \frac{2w_i^2}{(w_1 + w_i)^2}(\hat{\mu}_i - \mu)^2 + 4\frac{w_1\, w_i}{(w_1 + w_i)^2}(\hat{\mu}_1 - \mu)(\hat{\mu}_i - \mu)
$$

From here we continue the same as in section 3.2. When we take the expectation of the variance, and use the facts that (1) $E[w_i] = E[w_j] = n/c = \hat{w}$ for all nodes c_i and c_j, thus $\frac{2w_1^2}{(w_1+w_i)^2} = \frac{2\hat{w}^2}{4\hat{w}^2} = \frac{1}{2}$; (2) all nodes are contacted with an equal probability of

$\frac{1}{c}$; and (3) for all nodes $E[\hat{\mu}_i] = \mu$, gives:

$$E[\Phi_{t,1}|\Phi_{t,0} = \phi] = \phi - \frac{1}{2}(\hat{\mu}_1 - \mu)^2 - \frac{1}{2c}\underbrace{\sum_{i=1}^{c}(\mu_i - \mu)^2}_{\phi}$$
$$= \left(1 - \frac{1}{2c}\right)\phi - \frac{1}{2}(\mu_1 - \mu)^2$$

After c such updates the complete communication cycle has passed. The variance of the next cycle Φ_{t+1} is on average:

$$E[\Phi_{t+1}|\Phi_t = \phi] = \left(1 - \frac{1}{2c}\right)^c \phi - \frac{1}{2}\sum_{i=1}^{c}\left(1 - \frac{1}{2c}\right)^{c-i}(\mu_i - \mu)^2$$

Bounding the term $\left(1 - \frac{1}{2c}\right)^{c-i}$ by $\left(1 - \frac{1}{2c}\right)^c$ and using the fact that $\left(1 + \frac{z}{n}\right)^n \leq e^z$ (Stewart 2001) finally results in:

$$E[\Phi_{t+1}|\Phi_t = \phi] \leq \frac{1}{2}\left(1 - \frac{1}{2c}\right)^c \phi \leq \frac{\phi}{2\sqrt{e}}.$$

Thus after t cycles of Multi-Observations Newscast Averaging, the original variance Φ_0 is reduced to $\Phi_t \leq \frac{\Phi_0}{\left(2\sqrt{(e)^t}\right)}$ on average. The variance reduction is at an exponential rate, which means that the nodes converge to the correct mean extremely fast. The derived variation reduction rate is equal to the variation reduction of Newscast EM (Kowalczyk and Vlassis 2005).

Appendix C

Messages in Appearance- and Spatial-Temporal Feature Tracking

In section 4.2 a Dynamic Bayesian Network is presented for tracking multiple persons using appearance features and spatial-temporal features. The model is shown in figure C.1, the inference in this model is quite complex. We use the sum-product algorithm (Kschischang et al. 2001) which is explained in section 2.2.

The sum-product algorithm sends messages from variables to factors and from factors to variables. The messages for our DBN were presented in the section 4.2, however not yet very elaborately. In this section we will describe the messages in more detail.

In the model, shown in figure C.1, the following variables are used:

- X is the collection of parameters of the Mixture of Gaussians, it describes the intrinsic parameters for each person which do not change over time.
- L_t is the unique label of the person observed at time slice t, it is a hidden variable.
- O_t is the appearance feature vector of the person observed in time slice t, it is an observed variable.
- C_t is the camera location where observation t is made, it is a discrete noise free observed variable, $C_t \in \{C_1, \ldots, C_N\}$.
- H_t is the vector of indicators $h_{t,i}$ for all persons, $H_t = (h_{t,1}, \ldots, h_{t,P})$. The camera location where person p was last observed before time slice t, is $h_{t,p}$. This is a distribution over all cameras: $h_{t,p} \in \{C_1, \ldots, C_N\}$.
- T_t is the Wall Clock Time of the current observation, it is a continuous noise free observed variable.

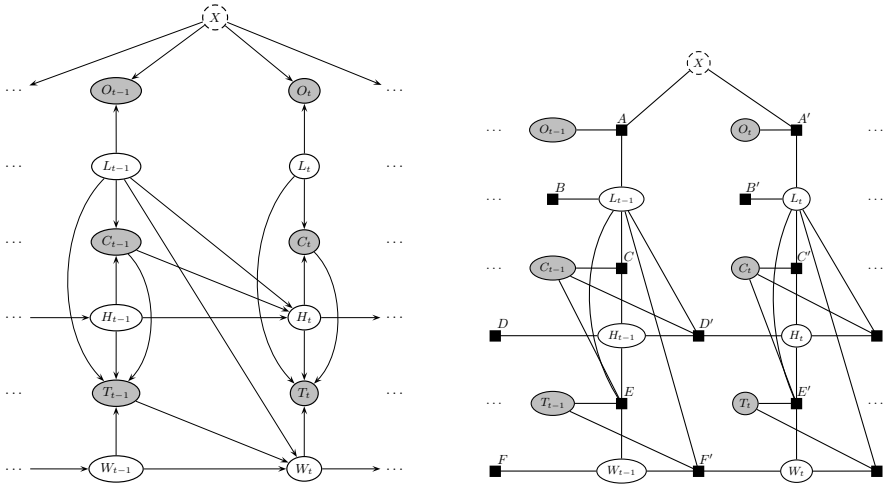


Figure C.1: Dynamic Bayesian Network and Factor Graph *left* the Bayesian Network for Appearance, Spatial and Temporal Tracking and *right* the accompanying Factor Graph Model

- W_t is the vector of time-indicators, $W_t = (w_{t,1}, \ldots, w_{t,P})$. The wall clock time that person p was last observed before time slice t, is $w_{t,p}$.

H_t and W_t are bookkeeping variables, they store information from previous observations. This enables the use of a first-order motion model for a person.

For clarity we provide again the probability density function (pdf) for every variable in the model conditioned on its parents, or prior pdf for the variables without parents.

- $P(L_t)$, the a-priori probability of person p to be observed, a uniform random distribution is used for this.
- $P(O_t|L_t, X)$, the distribution on the appearance O_t of a person observed at time slice t. The appearance features O_t depend on the intrinsic person parameters $X(L_t)$. However distortions are introduced by differences in pose or viewing angle, therefore the probability density function is modelled with a Gaussian distribution, $\mathcal{N}(O_t; X(L_t))$.
- $P(C_t|H_t, L_t)$, is the probability that person p is seen at camera c when he/she is last seen at camera $h_{t,p}$. We see the probability distribution of arriving at camera c, when departing from camera d, as a property of the environment. Therefore, in our implementation the pdf is given, and not learned from the observations. However, this distribution can also be learned from data in order to capture typical paths in a given environment. With the use of the hidden variable H_t, this is modelled with a first-order Markov motion model, $P(C_t|H_{t,L_t})$.
- $P(T_t|W_t, L_t, H_t, C_t)$, models the time of arriving at camera C_t, knowing that a person left camera $h_{(L_t)}$ at time $w_{(L_t)}$. For this we use a Gamma distribution, $P(T_t|w_{(L_t)}, h_{(L_t)}, C_t) = \Gamma(\Delta t|C_t, h_{(L_t)})$, where $\Delta t = T_t - w_{(L_t)}$, is the travel time. The travel time is the difference between the current wall clock time T_t and the

previous wall clock time $w_{(L_t)}$. The expected travel time between two locations is considered as a property of the environment and is given beforehand.

- $H_t|C_{t-1}, L_{t-1}, H_{t-1}$, models the update procedure of the H_t vector given the previous observation. For each person the last location where he is observed is stored in this vector. The value of H_t is determined by:

$$P(H_t|C_{t-1}, L_{t-1}, H_{t-1}) = \begin{cases} 1 & H_{t(p=L_{t-1})} = C_{t-1} \\ 1 & H_{t,(p\neq L_{t-1})} = H_{t-1,p} \\ 0 & otherwise \end{cases}$$

  This models, that H_t is equal to H_{t-1} for all persons, except for the person seen at time $t-1$. For that person H_t should be equal to C_{t-1}.

- $W_t|T_{t-1}, L_{t-1}, W_{t-1}$, models the update procedure of the W vector at time t given the previous observation. For each person the time he was last observed is stored and maintained in this vector, this vector has a great analogy with the H_t vector. The value of W_t is determined by:

$$P(W_t|T_{t-1}, L_{t-1}, W_{t-1}) = \begin{cases} 1 & W_{t,(p=L_{t-1})} = T_{t-1} \\ 1 & W_{t,(p\neq L_{t-1})} = W_{t-1,p} \\ 0 & otherwise \end{cases}$$

Posterior distribution of observation
To calculate the posterior probability of variable L_t, we use the sum-product algorithm (Kschischang et al. 2001), which is explained in section 2.2. These calculations use only a part of the complete factor graph. The complete factor graph is shown in figure C.1, while the used part of the factor graph is shown in figure C.2. The posterior probability of the hidden label L_t depends only on variables from the current time slice t. Therefore, in this section, the time slice index t is dropped.

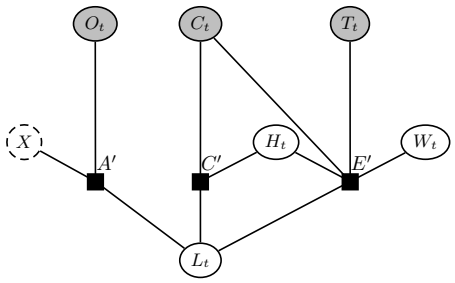


Figure C.2: Part of Factor Graph from the tracking model, which shows the factors within one time slice

In a factor graph there exist two kinds of nodes, variable nodes and factor nodes. Both nodes send messages (m), variable nodes to one (or more) factor nodes and factor nodes to one variable node. From section 2.2, we know that the following messages should be send:

$$m_{x\to f}(x) = \prod_{h\in n_x\setminus\{f\}} m_{h\to x}(x)$$

$$m_{f\to x}(x) = \sum_{\sim x} \left(f(x) \prod_{y\in n_f\setminus\{x\}} m_{y\to f}(y) \right)$$

The factor graph in figure (C.2) allows the following factorization of the global function:

$$g(O, C, T, X, H, W, L) = f_{A'}(O, X, L) f_{C'}(C, H, L) f_{E'}(T, C, W, H, L)$$

The posterior probability of L can be computed with the following function:

$$g_L(L) = \underbrace{\sum_X \sum_O f_{A'}(O, X, L)}_{m_{D^{prime} \to L}(L)} \underbrace{\sum_C \sum_H f_{C'}(C, H, L)}_{m_{C' \to L}(L)} \underbrace{\sum_T \sum_W \sum_C \sum_H f_{E'}(T, C, W, H, L)}_{m_{E' \to L}(L)}$$

According to this function the posterior probability of L depends only on variables from the current time slice.

The message from **factor node** A' is obtained as follows:

$$\begin{aligned} m_{D' \to L}(L) &= \sum_{\sim L} \left(f_{A'}(O, X, L) \prod_{y \in \{O, X\}} m_{y \to A'}(y) \right) \\ &= \sum_O \sum_X \{ f_{A'}(O, X, L)\ m_{O \to A'}(O)\ m_{X \to A'}(X) \} \\ &= \sum_O \sum_X \{ \mathcal{N}(O; X(L))\ m_{O \to A'}(O)\ m_{X \to A'}(X) \} \end{aligned}$$

From the generative model it is acquired that for a specific O and X, holds that $f_{A'}(O, X, L) = \mathcal{N}(O; X(L))$. The message from variable node O to the factor node A' is a special message, because it considers an observed variable. Therefore, the message should only exist for the observed O_t. This is modelled with a Dirac-delta probability function, which is only 1 if and only if $O = O_t$. The sum over different observations is equal to the specific case with this pdf where the Dirac-delta is 1.

The message from parameter node X to factor node A' holds almost the same. The parameters X are learned by the system, and therefore only the current parameters should be considered, this is also modelled with a Dirac-delta pdf.

The message from factor node A' becomes:

$$m_{D' \to L}(L) = \mathcal{N}(O; X(L)) \tag{C.1}$$

The message from **factor node** C' to variable node L is constructed as follows:

$$\begin{aligned} m_{C' \to L}(L) &= \sum_{\sim L} \left(f_{C'}(C, H, L) \prod_{y \in \{C, H\}} m_{y \to C'}(y) \right) \\ &= \sum_C \sum_H (f_{C'}(C, H, L)\ m_{C \to C'}(C)\ m_{H \to C'}(H)) \end{aligned}$$

The factor function $f_{C'}(C, H, L) = P(C|H(L))$, is the first order motion model, that is the probability of the current location given the previous location of a person.

Variable node C is an observed variable, and therefore modelled with the Dirac-delta pdf. The sum over different values for C equals to the equation where $C = C_t$. The message from variable node H is equal to the current belief of H, thus $m_{H \to C'}(H) = q(H)$

The message becomes:

$$m_{C' \to L}(L) = \sum_{h \in H} P(C|h_L))\ q(h_L)) \tag{C.2}$$

where h_L means, the entry of person L at location h.

The message from **factor node** E' is:

$$m_{E' \to L}(L) = \sum_{\sim L} \left(f_{E'}(T, C, W, H, L) \prod_{y \in \{T,C,W,H\}} m_{y \to f_{C'}}(y) \right)$$

This message should reflect the belief given the transition and travel time model. For the travel time model, the expected travel time for every pair of locations C_i, C_j is given at prior. The variables C and T are observed variables, therefore they are modelled with a Dirac-delta pdf.

This simplifies the message to:

$$m_{E' \to L}(L) = \sum_{h \in H} \sum_{w \in W} \left(f_{E'}(T, C, W, H, L)\ m_{H \to f_{E'}}(h)\ m_{W \to f_{C'}}(w) \right)$$

Assume that we want to send the message for person $p \in L$. From the generative model we obtain that for a specific value of $\{T, C, W, H, p\}$ holds that $f_{E'}(T, C, W, H, p) = P(T|W, p, H, C) = \Gamma(T - W|C, H(p))$. Next, the message ,$m_{H \to E'}(h)$, is the current belief state of H for person p at location h thus $q(h_p)$. Last, $m_{W \to E'}(w)$ is the current belief state of W for person p for time w thus $q(w_p)$).

This results in the following message for person p:

$$m_{E' \to L}(p) = \sum_{h \in H} \sum_{w \in W} (\Gamma(T - w_p|C, h_p)\ q(h_p)\ q(w_p)) \tag{C.3}$$

With those three described messages, $m_{D' \to L}(L)$, $m_{C' \to L}(L)$, and $m_{E' \to L}(L)$, we can compute the posterior on L. The current belief of L for person p is

$$q(L(p)) = m_{f_{A' \to L}}(p)\ m_{f_{C' \to L}}(p)\ m_{f_{E' \to L}}(p),$$

The used factor graph seems to contain cycles, and therefore a loopy belief method (Zajdel 2006; Kschischang et al. 2001) should be implemented. But when we take a closer look at the cycle connections, it seems reasonable to ignore the cycles. The messages from observed variables $\{O, C, W\}$ will not change, because they are observed. So the only node which causes a cycle is the node H. This is a latent variable, however, for a specific time slice t it has a given belief of the person's previous location. Therefore

the message sent from this variable will not change, and so the factor graph can be considered to be cycle free.

Update between two time slices
The connection between two different time slices comes from factor nodes $f_{D'}$ and $f_{F'}$. Those factor nodes are responsible for the update of the H and W vector.

To update the belief of $q(H_t)$ a message $m_{D' \to H_t}$ is sent. This message is based on the part of the factor graph shown in figure C.3.

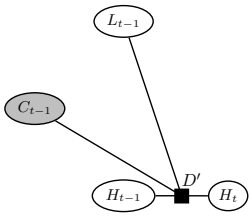


Figure C.3: Part of Factor Graph model, which shows the update of the H table

The message should reflect the following, if location c is last location C_{t-1}, then the belief of person p lastly seen at location c should equal to the belief of assigning L_{t-1} the value p. When this is not the case, thus location c is not equal to the last location, the belief of person p seen lastly at location c is only scaled by a factor which represents the belief person p was seen at time $t-1$. Which is $h_{p,t} = \alpha h_{p,t-1}$, where α is $1 - q(L_{t-1} = p)$. The message is computed as follows:

- $h_{t,p} = \begin{cases} C_{t-1} & \textbf{iff } L_{t-1} = p \\ h_{t-1,p} & \textbf{iff } L_{t-1} \neq p \end{cases}$ (This is taken from the model definition)
  $h_{t,p}$ does not depend on $h_{t-1,n}$ for $n \neq p$.

- Starting with the definition of a message, we derive the following equation:

$$m_{D' \to h_{t,p}}(h_{t,p}) = \sum_{h_{t-1,p}} \sum_{L_{t-1}} P(h_{t,p} | h_{t-1,p}, L_{t-1,p}) \frac{q(L_{t-1})}{m_{D' \to L_{t-1}}} \frac{q(h_{t-1,p})}{m_{D' \to h_{t-1,p}}}$$

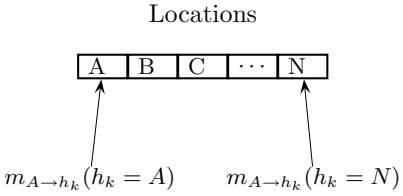
  Because, $m_{D' \to L_{t-1}}$ and $m_{D' \to h_{t-1,p}}$ are initialized as uniform potentials, we get:

$$m_{D' \to h_{t,p}}(h_{t,p}) = \sum_{h_{t-1,p}} \sum_{L_{t-1}} P(h_{t,p} | h_{t-1,p}, L_{t-1,p}) q(L_{t-1}) q(h_{t-1,p})$$

- We calculate the message for person p, and thus we drop the index p

$$m_{D' \to h_t}(h_t) = \sum_{h_{t-1}} \sum_{L_{t-1}} P(h_t | h_{t-1}, L_{t-1}) q(L_{t-1}) q(h_{t-1})$$

- The message $m_{D' \to h_t}(h_t)$ is a probability table, with for each location the probability that the person was last observed there, see the figure below.



- Suppose we update entry c in the table

$$
\begin{aligned}
m_{D'\to h_t}(h_t = c) &= \sum_{h_{t-1}} \sum_{L_{t-1}} P(h_t = c|h_{t-1}, L_{t-1}) q(L_{t-1}) q(h_{t-1}) \\
&= \sum_{L_{t-1}\neq p} \left[\sum_{h_{t-1}} P(h_t = c|h_{t-1}, L_{t-1} \neq p) q(L_{t-1} \neq p) q(h_{t-1}) \right] + \\
&\quad \sum_{h_{t-1}} P(h_t = c|h_{t-1}, L_{t-1} = p) q(L_{t-1} = p) q(h_{t-1}) \\
&= \left[\sum_{L_{t-1}\neq p} q(L_{t-1}) q(h_{t-1} = c) \right] + \begin{cases} 0 \cdot q(L_{t-1} = p) & \textbf{iff } c \neq C_{t-1} \\ 1 \cdot q(L_{t-1} = p) & \textbf{iff } c = C_{t-1} \end{cases} \\
&= q(h_{t-1} = c)(1 - q(L_{t-1} = p)) + [c = C_{t-1}] q(L_{t-1} = p)
\end{aligned}
$$

  The value of $[a = b]$ is one if and only if $a \equiv b$ and otherwise it is zero.

The final message (for person p and location c) is

$$m_{D'\to h_{p,t}}(h_{p,t} = c) = q(h_{p,t-1} = c)(1 - q(L_{t-1} = p)) + [c = C_{t-1}] q(L_{t-1} = p) \quad \text{(C.4)}$$

The message from **factor node** F' is based on the part of the factor graph shown in figure C.4. The derivation of the message is quite analogous to the derivation of the message from factor node D'.

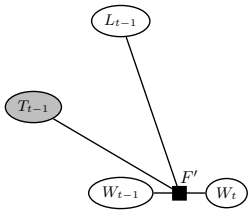


Figure C.4: Part of Factor Graph model, which shows the update of the W table

The message $m_{F'\to w_t}(W_t)$ is a probability table, with for each time the belief that a person was last observed there. Consider the message for person p and time s. The belief that person p is last observed at time $T_{t-1} = s$, should equal the belief $L_{t-1} = p$. The belief that person p is last observed at any other time, should equal the belief $W_{t-1=s,p}$ scaled by $1 - q(L_{t-1} = p)$. The message is computed as follows:

- $w_{t,p} = \begin{cases} T_{t-1} & \textbf{iff } L_{t-1} = p \\ w_{t-1,p} & \textbf{iff } L_{t-1} \neq p \end{cases}$ (This is taken from the model definition)
  $w_{t,p}$ does not depend on $w_{t-1,n}$ for $n \neq p$.

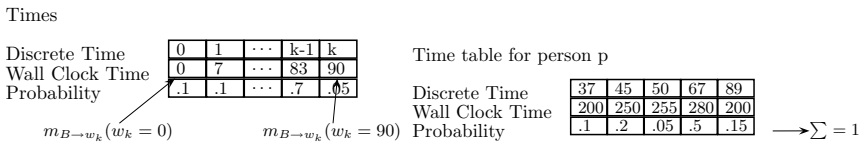
- The definition of a message gives us:

$$m_{F' \to w_{t,p}}(w_{t,p}) = \sum_{w_{t-1,p}} \sum_{L_{t-1}} P(w_{t,p}|w_{t-1,p}, L_{t-1,p}) \frac{q(L_{t-1})}{m_{F' \to L_{t-1}}} \frac{q(w_{t-1,p})}{m_{F' \to w_{t-1,p}}}$$

  Because, $m_{F' \to L_{t-1}}$ and $m_{F' \to w_{t-1,p}}$ are initialized with uniform potentials we can ignore those messages. We sent a message about person p and therefore we drop the index p.

$$m_{F' \to w_t}(w_t) = \sum_{w_{t-1}} \sum_{L_{t-1}} P(w_t|w_{t-1}, L_{t-1}) q(L_{t-1}) q(w_{t-1})$$

- The message $m_{F' \to w_t}(W_t)$ is a probability table, with for each time the probability the person was last observed there, see the figure below.



- Suppose we update entry $wct = 280$ in the table (see the right figure)

$$\begin{aligned} m_{F' \to w_t}(w_t = 280) &= \sum_{w_{t-1}} \sum_{L_{t-1}} P(w_t = 280|w_{t-1}, L_{t-1}) q(L_{t-1}) q(w_{t-1}) \\ &= \sum_{L_{t-1} \neq p} \left\{ \sum_{w_{t-1}} P(w_t = 280|w_{t-1}, L_{t-1} \neq p) q(L_{t-1} \neq p) q(w_{t-1}) \right\} + \\ &\quad \sum_{w_{t-1}} P(w_t = 280|w_{t-1}, L_{t-1} = p) q(L_{t-1} = p) q(h_{t-1}) \end{aligned}$$

  Note that the following holds

$$P(w_t = 280|w_{t-1}, L_{t-1} \neq p) = \begin{cases} 1 & \textbf{iff} w_{t-1} = w_t \\ 0 & \textbf{otherwise} \end{cases}$$

$$\sum_{L_{t-1}} q(L_{t-1}) = 1 \Rightarrow \sum_{L_{t-1} \neq p} = 1 - (L_{t-1} = p)$$

  Thus the message can be rewritten to

$$\begin{aligned} m_{F' \to w_t}(w_t = 280) &= \left\{ \sum_{L_{t-1} \neq p} q(L_{t-1}) q(w_{t-1} = 280) \right\} + \begin{cases} 0 & \textbf{iff } T_{t-1} \neq 280 \\ q(L_{t-1} = p) & \textbf{iff } T_{t-1} = 280 \end{cases} \\ &= q(w_{t-1} = 280)(1 - q(L_{t-1} = p)) + [T_{t-1} = 280] q(L_{t-1} = p) \end{aligned}$$

The final message (for person p and time s) is

$$m_{F' \to w_{p,t}}(w_{p,t} = s) = \begin{cases} q(w_{p,t-1} = s)(1 - q(L_{t-1} = p)) & \textbf{iff } s \neq T_{t-1} \\ q(L_{t-1} = p) & \textbf{iff } s = T_{t-1} \end{cases} \tag{C.5}$$

Because this table will grow very fast (a new column for every person at any single time slice), it will be approximated. For person p only X (e.g. 5) time slices are stored where the probability that person p was observed is above a certain threshold. All other time slices are considered to have zero probability and will therefore not be explicitly represented in the belief table. This is also called a sparse representation.

So, table W is only R columns wide, the new time T_{t-1} is not included, but a message about $s = T_{t-1}$ is sent. This results in the fact that there are $R + 1$ messages for only X positions in the table. So the W table should temporarily be extended to $R + 1$ columns. After sending all messages the column (time) with the lowest probability.

With the message from factor node F', we have completed the definition of the messages. In this section we have described, in detail, all messages in our tracking model. Experimental results with this tracking model are described in chapter 5.