University of Liege
Faculty of Applied Science
Department of Computer Science

On-line Simultaneous Learning and Tracking of Visual Feature Graphs

Arnaud Declercq

August 2007

DEA thesis submitted in partial fulfillment of the requirements for the degree of Master of Applied Science in Computer Science.

Abstract

Model learning and tracking are two important topics in computer vision. While there are many applications where one of them is used to support the other, there are currently only few where both aid each other simultaneously. In this work, we seek to incrementally learn a graphical model from tracking and to simultaneously use whatever has been learned to improve the tracking in the next frames. The main problem encountered in this situation is that the current intermediate model may be inconsistent with future observations, creating a bias in the tracking results. We propose an *uncertain model* that explicitly accounts for such uncertainties by representing relations by an appropriately weighted sum of informative (parametric) and uninformative (uniform) components. The method is completely unsupervised and operates in real time.

Acknowledgements

I would like to thank all the people who made this work possible.
First, I would like to thank my supervisor, Justus H. Piater, for his precious guidance through my research and his constant availability.
I would like to thank my fiancée for his support and his love that makes everything look easier.
I would also like to thank my parents and friends for all those marvellous moments that help me to relax and for the numerous discussions that inspired my work.
Finally, I would like to thank my colleagues Renaud and Bertrand for the good work atmosphere they create.

Contents

List of Figures

Chapter 1

Introduction

Graphical models are popularly used to represent objects in terms of local appearance and spatial relations for detection, classification and tracking applications [4, 15, 18, 19]. Unsupervised learning of such feature graphs from images is difficult because of the correspondence problem of features between images. If learning is based on a video sequence, tracking can be used to solve this problem. On the other hand, tracking a set of features requires a good model if we want to avoid problems due to occlusion and clutter. In this thesis, we address these two tasks simultaneously in a way that they help each other.

The idea is to incrementally learn models of relations that exist between local features, and to use what we have already learned to improve tracking. Since tracking is on-line, it is important to use models with a low computational cost. In practice, we are only interested in learning specific types of relations (rigid, articulated, ...). For this purpose, parametric models or mixtures of a small number of Gaussians are more suitable than nonparametric models.

In this thesis, we seek to identify, while tracking, spatial relations whose distributions are roughly a combination of Gaussians, and simultaneously use them to aid the tracking by biasing the observation likelihoods. This will be helpful if the learned model is accurate, but detrimental if it is not.

The three main sources of undue bias are observations that are far from Gaussian, inaccurate parameter estimates based on few observations early during learning, and parameters that change over time. For example, consider an object sitting on a table: The system will learn rigid relations between the object and the table that have to be unlearned when the object is moved.

To manage such situations, we develop in this thesis an *uncertain graphical model* that explicitly accounts for its predictive power by representing

each pairwise potential as a mixture of a parametric model (reliable relation) and a uniform density (ignorance).

After discussing some background in the next chapter, this thesis will be divided into three parts :

- The first one introduces the feature graph and the methods used for its tracking and learning. This first part is also useful to understand the need of an uncertain model to successfully combine tracking and learning.
- The second part explains the different sources of uncertainty and how the uncertain model accounts for them. In this part we limit the method to rigid relations so that we can concentrate on the way to deal with uncertainty.
- The last part extends the method to articulated relations. Two possible extensions are proposed : one using a mixture of a small number of Gaussians and one selecting a set of transformed spaces where the relation is rigid.

Chapter 2

Related Work

Appearance- and structure-based approaches have been widely used for object representation. Appearance-based models usually model an object as color histograms [3], image templates [2] or neighborhood features such as SIFT. Structural models typically represent an object using 2D edges or 3D geometric models [9]. A natural way to combine the shape and appearance of an object is to use a graphical model whose nodes and edges correspond to local features and spatial relations between them [16, 18, 19].

On-line tracking is often addressed by using a prior model of the object. This model can be learned from training examples during a learning phase [16]. Another solution is to design the prior model of the object by hand [18]. This means that the model is specific to a particular object and is difficult to reuse in other situations. The model can also be defined in starting frames [2, 12], taking the risk that it no longer corresponds to changed appearances of the object over time. If we are only concerned with tracking, we can also use a dynamic model that is updated to accommodate the object appearance and structural changes [19].

Unsupervised learning of object models is computationally demanding because it has to find feature correspondences between images [15]. The temporal information from video can be used to solve this problem. Leordeanu and Collins [10] use tracking to group features into objects by observing their co-occurrences. Ramanan *et al.* [14] use a video sequence to build models of animals from temporally coherent clusters that represent body parts. While the former work does not use the learned relationships between parts to refine the matching process, the latter does not allow corrections of the model once it as been learned.

Some recent approaches have been developed to simultaneously learn and track object models. Lim *et al.* [11] propose a method that incrementally learns and adapts a low-dimensional eigenspace representation to reflect ap-

pearance changes of the target, thereby facilitating the tracking task. The same kind of model is used by Dowson and Bowden [6] by their N-tiers model that represents both the appearance and the structure of the object. These models do not express the uncertainty inherent in the current model. Moreover, they suppose that the selected region of interest only contains the object to learn.

In this thesis, we develop an *uncertain model* of relations between features that explicitly accounts for their uncertainty. Thus, a relation will contribute stability to tracking without exerting overly strong bias that would hamper tracking. No assumptions are made regarding the number of objects in a video; in fact, features are not explicitly grouped into objects at all.

Chapter 3

Combining Tracking with Learning

3.1 Introduction

Assume we want to learn a visual feature graph that can model an object shown in a set of images. A major difficulty of this task is to find the correspondences of the features between images. If we learn the model from a video instead of a set of images, we can use the temporal coherence existing between the frames to solve this correspondence problem by tracking the features. In this case, the feature positions are known for each frame and learning the relations between them becomes easier.

On the other hand, tracking a set of independent features in long sequences is a very difficult task due to occlusions, clutter, blur and noise effects. A common way to improve the tracking is to use a model of the relations existing between those features. This model is often learned off-line from training examples or designed by hand.

Here we don't have this model since it is what we wanted to learn in the first place. So it's a chicken-and-egg problem: we need to track features to learn relations between them but we need those relations to contribute stability to the tracking. Instead of complaining about what we don't have, let's consider the information we can use. Assume, for example, we are trying to track the set of features in the frame corresponding to time t in the video; which means we have already tracked them up to time $t-1$. Even if we don't have a model of the relations learned from the whole sequence, we can already learn a model from the frames up to time $t-1$ and use it to improve the tracking at time t. So, what we propose here is to incrementally learn a model of the relations existing between the features and to simultaneously

use whatever has been learned to improve the tracking in the next frames.

In the first section of this part, we will present the choices we made about the graphical model and explain how to incrementally learn it. The second section focuses on the tracking of this graph with Sequential Belief Propagation. The last section of this chapter presents some experiments.

3.2 Learning the Graphical Model

To model the features and their relations, we use an undirected graph where nodes and edges represent the visual features and the relations between them, respectively. We denote the state of each feature at time t by $x_{i,t}$ and its associated image observation by $z_{i,t}$. The joint target states and the joint observations are respectively denoted by $X_t = \{x_{1,t}, \ldots, x_{N,t}\}$ and by $Z_t = \{z_{1,t}, \ldots, z_{N,t}\}$. We also denote $Z_{0:t} = \{Z_1, \ldots, Z_t\}$, the joint observations of the whole sequence until time t.

A spatial relation between two features is represented by a potential function $\psi_{i,j,t}(x_{i,t}, x_{j,t})$ that expresses the constraint on the relative position of the features i and j. This pairwise potential depends on time t because it is learned on-line from the observations up to time $t-1$.

For a Markov network, the probability of the posterior of the joint state X_t given the image measurements Z_t can be written [7]

$$P(X_t|Z_t) = \frac{1}{Z_Q} \prod_{(i,j)\in E} \psi_{i,j,t}(x_{i,t}, x_{j,t}) \prod_{i\in V} p_i(z_{i,t}|x_{i,t}), \tag{3.1}$$

where Z_Q is a normalization constant, E is the set of edges in the graph, and V is the set of nodes.

As you can notice, no assumptions are made regarding the number of objects in a video; in fact, features are not explicitly grouped into objects at all. More or less informative relations are simply learned between features; coherently moving, rigid objects might then be identified as rigid subgraphs. The learning of the graphical model, then, consists of detecting features, create connections between them with any triangulation scheme and keep only those that are of interest. Features that are unable to create useful connections with other might just be removed from the graph.

3.2.1 The Feature Set

Many kinds of visual features can be used to describe local appearance of objects: image patches, SIFT descriptor, edges, color histograms,.... Since

the method is on-line and the weakness of the descriptor can be alleviated by relations, the low computational cost of the descriptor is more important than its robustness. We, then, choose the images patches to describe the features. In the same idea, many feature detectors exist but we choose to use the well known Harris detector. When the understanding of the graphical model learning process is the more important, we even simply initialize the features' position by hand during test phases.

It is important to notice that the description of a feature is learned in the frame it is detected and is not update afterward. The main reason of this choice is that the features are very local elements so we can make the assumption that their appearance will not be modified by more than an affine transformation. If a local element can have multiple appearances like an open or closed eye, it will just be represented with different features; their connections in the graph making it implicitly clear that those features correspond to the same location.

As we can see, the only thing we've got to learn about a feature is whether we keep it in the model or not. This will be done through its relations with other features: if a feature is correlated with others than it worth to keep it otherwise we simply discard it.

3.2.2 The Relation Set

Learning the relations is much more complex since a relation cannot be be learned from a single frame (unless it is perfectly rigid, which you don't know before watching it for a long time). The possible ways to represent it can also be demanding in terms of computational cost and memory space. For example, a smart representation of the relations seems to be the non-parametric density estimation since it is able to model any king of distribution. Unfortunately its computational cost is too high for an on-line process. On the other hand parametric models are very fast but are not able to represent all possible relations. In practice, we are only interested in learning specific types of relations (rigid, articulated,...) so the parametric model will be sufficient in this case. All we have to do, is to remove relations from the graph if they become too complex to be modeled with the chosen parametric model.

In this first part, we seek to learn and identify, while tracking, approximately rigid spatial relations whose distributions are roughly Gaussian-shaped. Therefore, the potential functions are of the form

$$\psi_{i,j,t}(x_{i,t}, x_{j,t}) = \mathrm{e}^{-\frac{(r_t - \mu_t)^2}{2\sigma_t^2}}, \tag{3.2}$$

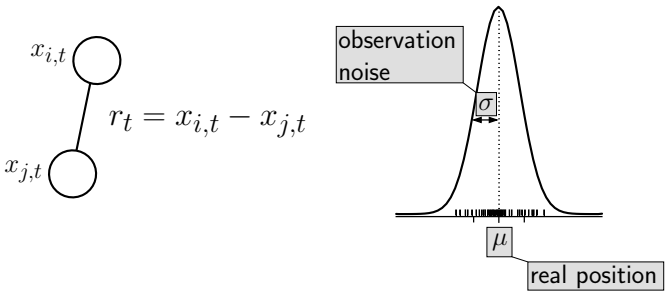


Figure 3.1: The potential function used to represent a relation between two features $x_{i,t}$ and $x_{j,t}$

where r_t corresponds to the observation of the relative position $x_{i,t} - x_{j,t}$ between two given features i and j at time t. The parameters μ_t and σ_t^2 are the value of the estimated rigid relation and the variance of the observations around this position.

The incremental learning of the Gaussian model parameters that maximize the likelihood of the relations observed in the video is given by the following equations:

$$\hat{\mu}_t = \frac{\pi_{t-1}\hat{\mu}_{t-1} + w_t r_t}{\pi_{t-1} + w_t}, \tag{3.3}$$

$$\hat{\sigma}_t^2 = \frac{\pi_{t-1}(\hat{\sigma}_{t-1}^2 + (\hat{\mu}_t - \hat{\mu}_{t-1})^2) + w_t(r_t - \hat{\mu}_t)^2}{\pi_{t-1} + w_t}, \tag{3.4}$$

$$\pi_t = \pi_{t-1} + w_t, \tag{3.5}$$

where $\hat{\mu}_t$ and $\hat{\sigma}_t^2$ are the mean and the variance of the Gaussian, and π_t is the cumulative weight of preview observations (with $\pi_0 = 0$). To account for more or less reliable observations, the parameter updates are weighted by their likelihood product $w_t = p(z_{i,t}|x_{i,t})p(z_{j,t}|x_{j,t})$. For example, in the case of an occlusion or a loss of tracking, nothing can be learned but the tracker will produce a (meaningless) observation nevertheless. We therefore discount such observations by using the observation likelihood as an indicator of reliability.

3.3 Tracking the Features

The Markov network presented in the previous section is a generative model at one time instant. To track it, we extend Eqn. 3.1 to account for a time dimension. Under the conventional Markov assumption of independent dynamic models,

$$P(X_t|X_{t-1}) = \prod_i p(x_{i,t}|x_{i,t-1}), \tag{3.6}$$

the posterior probability of the joint state X_t given the image measurements $Z_{0:t}$ can be expressed as

$$\begin{aligned} P(X_t|Z_{0:t}) = \frac{1}{Z_Q} \prod_{(i,j)\in E} \psi_{i,j,t}(x_{i,t}, x_{j,t}) \\ \times \prod_{i\in V} p_i(z_{i,t}|x_{i,t}) p(x_{i,t}|z_{i,0:t-1}), \end{aligned} \tag{3.7}$$

where

$$p(x_{i,t}|z_{i,0:t-1}) = \int p(x_{i,t}|x_{i,t-1}) p(x_{i,t-1}|z_{i,0:t-1}) \mathrm{d}x_{i,t-1}. \tag{3.8}$$

The marginal posterior distribution $p(x_{i,t}|Z_{0:t})$ of the feature i at time t can, then, be obtained with

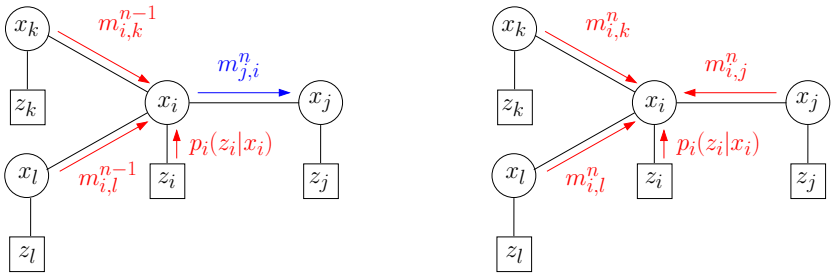
$$p(x_{i,t}|Z_{0:t}) = \int ... \int P(X_t|Z_{0:t}) \mathrm{d}x_{1,t} \mathrm{d}x_{i-1,t} \mathrm{d}x_{i+1,t} \mathrm{d}x_{N,t} \tag{3.9}$$

This kind of brute computation of marginal probabilities becomes quickly intractable because of the computation of multiple integrals. The (Loopy) Belief Propagation algorithm [13] is a more efficient algorithm that computes $p(x_{i,t}|Z_{0:t})$ for $i = 1, ..., N$ through a local message passing process.

3.3.1 Loopy Belief Propagation

To simplify matters, let's first consider the belief propagation without the time aspect. Belief propagation can be seen as an iterative message passing process where each node i sends to its neighboors j a message that contains its current belief about the state of node j. At any time during the execution of the algorithm, the current marginal distribution estimate for a node i is the normalized product of its local evidence $p_i(z_i|x_i)$ and of all incoming messages from the neighboring nodes:

$$p^n(x_i|Z) \propto p_i(z_i|x_i) \prod_{j\in N(x_i)} m^n_{i,j}(x_i) \tag{3.10}$$



(a) computation of $m_{j,i}^n(x_j)$ (b) computation of $p^n(x_i|Z)$

Figure 3.2: The belief propagation.

where $p^n(x_i|Z)$ represents the marginal distribution estimate at the n-th iteration, $N(x_i)$ is the set of nodes adjacent to i and $m_{i,j}^n(x_i)$ is a message sent from node j to node i at the n-th iteration.

To prepare a message for node j, the node i first compute an estimation of its own state based on its local evidence and the messages sent by all its neighbors but j. This product is then multiplied with the potential function related to i and j to produce the message from i to j:

$$m_{j,i}^n(x_j) \leftarrow \int_{x_i} \left[p_i(z_i|x_i)\psi_{i,j}(x_i, x_j) \times \prod_{k \in N(x_i)\setminus j} m_{i,k}^{n-1}(x_i) \right] \mathrm{d}x_i, \tag{3.11}$$

Figure 3.2 illustrates the method. In the case of tree structured graphical models, the algorithm is guaranteed to converge towards $p^n(x_i|Z)$. In graphical models with loops, convergence is not guaranteed but good empirical performance has been shown in the literature [20].

3.3.2 Sequential Belief Propagation

As shown by Hua and Wu [8], belief propagation can be adapted to account for a temporal information. The equations of this new version callled *Sequential Belief Propagation* correspond to equations 3.10 and 3.11 extended with a new factor in the product that express the time correlations between the feature states. The marginal distribution estimate for a node i at time t

is given by:

$$p^n(x_{i,t}|Z_{0:t}) \propto p_i(z_{i,t}|x_{i,t}) \prod_{j \in N(x_{i,t})} m^n_{i,j,t}(x_{i,t}) \times \int_{x_{i,t-1}} p(x_{i,t}|x_{i,t-1}) p(x_{i,t-1}|Z_{0:t-1}) \mathrm{d}x_{i,t-1}. \tag{3.12}$$

And the local message passed from node i to node j at time t and iteration n is given by

$$m^n_{j,i,t}(x_{j,t}) \leftarrow \int_{x_{i,t}} \Big[p_i(z_{i,t}|x_{i,t}) \psi_{i,j,t}(x_{i,t}, x_{j,t}) \times \int_{x_{i,t-1}} p(x_{i,t}|x_{i,t-1}) p(x_{i,t-1}|Z_{0:t-1}) \mathrm{d}x_{i,t-1} \times \prod_{k \in N(x_{i,t}) \setminus j} m^{n-1}_{i,k,t}(x_{i,t}) \Big] \mathrm{d}x_{i,t}, \tag{3.13}$$

Another contribution of Hua and Wu is the new way to compute the product of messages through particle filters. While *Nonparametric Belief Propagation* (NBP) [17], already used particles to represent the nodes' distribution, it models the messages in BP as Gaussian mixtures and uses elaborate MCMC samplers to sample the new Gaussian mixture kernels of the updated messages. This process is far too slow to be applied in real-time. Hua and Wu proposed to consider the messages as a set of weights for the particles of the node that receive the message. This way, the message product only consists of a product of scalars which is much more efficient than a product of Gaussian mixtures.

The particle of each node are propagated from time $t-1$ to time t and are used as the support on which the messages are propagated and the nodes' distribution are evaluated. Since the particle positions don't change during the belief propagation, we can evaluate the potential function between each pair of particles from two neighboring nodes before starting BP. The same applies to the evaluation of the image likelihood of the particles.

The different steps of the methods are shown below and in figure 3.3:

1. Sequential Monte Carlo
   (a) for each node, re-sample the particles
   (b) for each node, propagate the particles from time $t-1$ to time t
2. Initialization of Belief Propagation
   (a) for each node, compute the image likelihood of each particle.
   (b) for each relation, evaluate the potential function between the particles of the two nodes in the relation.
3. Iterations of Belief Propagation
   (a) for each relation between two nodes i and j, compute the messages $m^n_{j,i,t}(x_{j,t})$ and $m^n_{i,j,t}(x_{i,t})$
4. Inference result
   (a) for each node i, compute the marginal distribution $p^n(x_{i,t}|Z_{0:t})$

Notice that the temporal information is not computed like the other sources of information. While all of them are computed through a product of weights for the particles, the temporal information is multiplied with other sources of information through the distribution of the particles.

3.4 Experiments

In this section, we demonstate the performance of the method developed in this first part on a representative example of learning and tracking of 2D rigid relations. Figure 3.4 shows the result sequence taken with a webcam at a resolution of 320×240 and processed on-line. In order to create an example that is easy to follow, we decided to initialize the model by hand. Two features were selected from the head and two from the background. Since there are only four features, a fully connected graph of relations can be created without making the example unreadable. For each feature, two trackers were initialized: one independent (not connected) tracker (in blue) and one using the graphical model (in green). The variance of a relation is represented by the thickness of the line that connect the two features. This means that thick lines correspond to relations of low variance $\hat{\sigma}$ and vice versa.

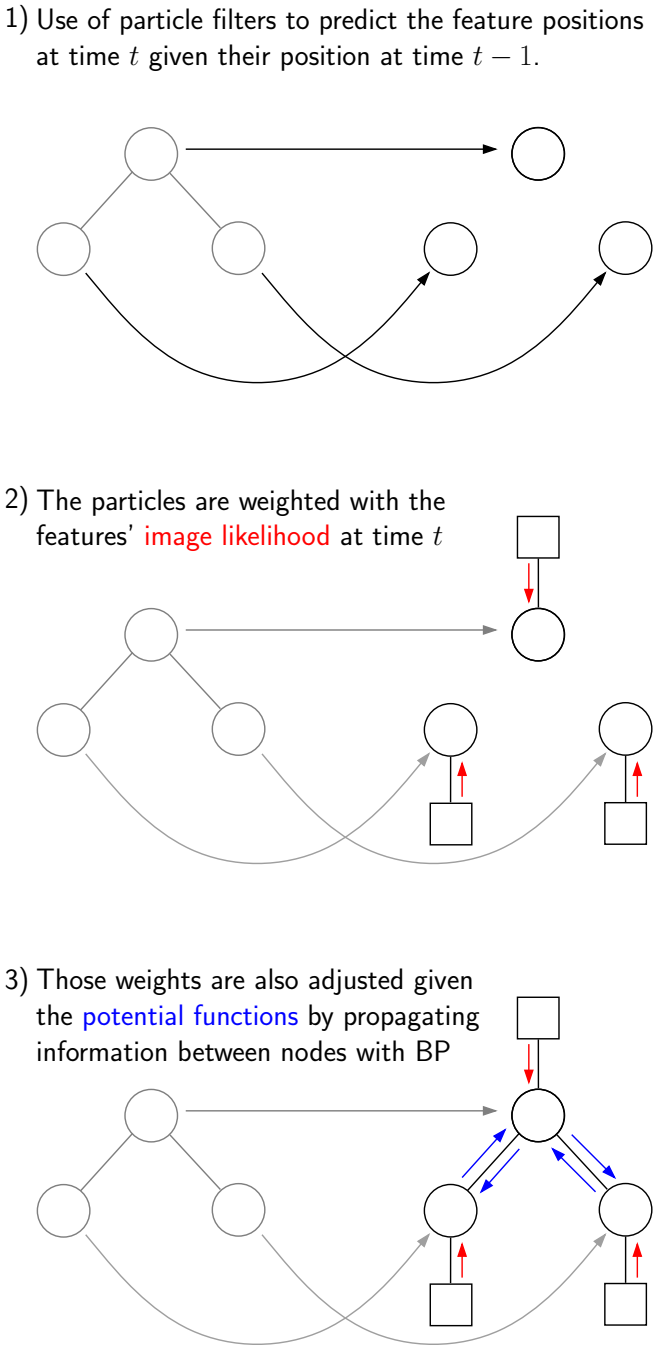


Figure 3.3: The 3 steps and 3 sources of information of SBP.

(a) frame 001 (b) frame 021 (c) frame 031 (d) frame 041

(e) frame 051 (f) frame 071 (g) frame 081 (h) frame 101

(i) frame 111 (j) frame 121 (k) frame 131

Figure 3.4: Representative results. Thick lines correspond to relations of low variance $\hat{\sigma}$ and vice versa. Each feature is tracked twice, with the help of the relations (green) and without (blue). Frame indices are given below each image.

In the beginning of the sequence, we can see that the relations are represented with very thick lines since the head is motionless Those relations prove to be helpful once the head is hidden. Indeed, we can see that even if the independent trackers (in blue) are completely lost, the linked trackers keep all the particles around the true position of the hidden feature. The learned relations are then very useful to avoid the failure of the tracking.

The last part of the sequence (from frame 111) is less pleasant: the learned potential functions create an overly strong bias is the tracking and keep the relation extremely rigid even if the head starts to move. This comes from the fact that we are too confident in the ability of the learned potential functions to predict the relations in the next frame.

3.5 Conclusion

In this first part, we introduced the methods used in the learning and the tracking of the feature graph. The simple combination of those methods has proved its usefulness in the case of occlusions but is still insufficient if the relations evolve along the video. In this case, the bias created by the overly restrictive potential function prevents the model from evolving. To overcome this problem, we have to find a way to reduce the influence of the current learned potential function if it is likely to create bias in the tracking. In the next chapter of this thesis, we will propose an *uncertain model* that accounts explicitly for uncertainty about the quality of the current model.

Chapter 4

Use of an Uncertain Rigid Model

4.1 Introduction

In this second part, we will develop an uncertain model that accounts for the uncertainty in the capability of the learned model to predict the position of the features. The sources of uncertainty can be divided into two groups:

Uncertainty in the tracking. This source of uncertainty comes from the assumption we implicitly made that the observed relations come from a stationary distribution which may not be true. For example, a feature may remain static up to time $t-1$ but start moving at time t. In this case, the learned model is no longer predictive of future observations, and its uncertainty is increased.

Uncertainty in the learning. The distribution of observations may correspond to a lesser or greater extent to the parametric model of interest, giving rise to higher or lower predictive uncertainty. Moreover, the uncertainty of a learned model decreases with the number of observations; even observations drawn from a parametric distribution of interest are of little predictive value at early stages of learning.

The uncertainty in the learning will evolve with time and, after a while, it will be possible to decide with high confidence if the relation can be modeled with the parametric model or not. On the other hand, the uncertainty in the tracking will remain constant since the relation can be stationary for an undetermined period of time.

In this chapter we will develop an uncertain model for rigid relations only. Like in the first part of the thesis, the informative part of the potential function will be represented with a Gaussian model.

4.2 Uncertainty in the Learning

Let's first start with the uncertainty in the learning. This uncertainty is related to the correspondence between the learned potential function and the observed relations used to learn it. The learned uncertain model will then be useful in two different situations.

Recognition. It's only when you accumulate observations of an object that you can gain confidence in recognizing it later. It's also only when you accumulate observations of a relation that you become able to determine if it is discriminative or not for the recognition. So the uncertain model is a representation of our current knowledge and confidence in the learned relation.

Tracking. This is clear that if a model is unable to represent the relations observed up to time $t-1$, it will neither be useful to predict the relation at time t. So the uncertainty about a potential function can also be used to balance the influence between all the relations used to track the features.

So, it seems that we can deduce the uncertainty in the learning from the observations used to compute the potential function, but how? The first thing to do is to separate the differant cases. Figure 4.1 show the three different situations that can occur:

1. The relation has been observed for a long time and was always rigid. There is a high chance that this relation is very reliable and thus that we must trust it more that others.

2. The relation has been observed only a few times but was always rigid. Even if it's a good start, it's too soon to be able to give an accurate estimation of the parameters of the model (in this case, of the mean and the variance of the Gaussian). This potential function can then be used but must have less influence than the old reliable ones.

3. The observations do not correspond exactly to the parametric model. In this case, it means that the relation is not 100% rigid. Depending on the level of matching between the observations and the model, the learned potential function must have more or less influence.

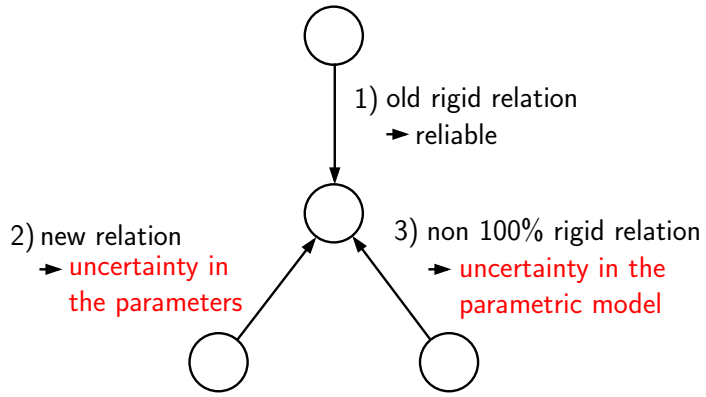


Figure 4.1: The reliable relations should have more influence than other.

While for the first case we can simply use the maximum likelihood potential function, the two others need to be modified in a way to account for the corresponding uncertainty. We will start with the uncertainty in the parameters in section 4.2.1 and complete the obtained model with the uncertainty in the parametric model in section 4.2.2

4.2.1 Uncertainty in the Parameters

When we have only a few observations, the Gaussian parameters produced by the maximum-likelihood estimation are uncertain. This uncertainty will decrease with the number of observations. We thus need to augment the variance of the informative part of the potential ψ (Eqn. 3.2) as a function of the number of observations. It turns out that the influence of the uncertainty in the mean is insignificant compared to the influence of the uncertainty in the variance; we therefore neglect the former. We consequently choose a variance $\tilde{\sigma}^2$ in a way that it bounds the risk of underestimating the true variance, i.e., $P(\tilde{\sigma}^2 \leq \sigma^2) = \alpha$, where conventionally $\alpha = 0.95$ (see figure 4.2). Since empirical estimates of variance follow a χ^2 distribution,

$$\tilde{\sigma}_t^2 = \frac{\pi_t}{\chi^2_{\pi_t - 1}(\alpha)} \hat{\sigma}_t^2, \tag{4.1}$$

where $\chi^2_{\pi_t - 1}(\alpha)$ is the inverse of the cumulative density function of the χ^2 distribution evaluated at probability α. Notice that the weight π_t is used

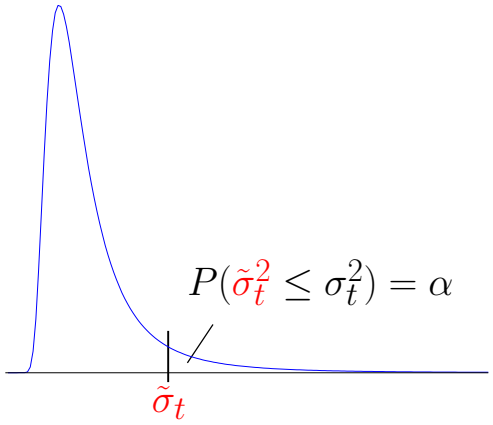


Figure 4.2: We choose a variance $\tilde{\sigma}^2$ in a way that bounds the risk of underestimating the true variance.

instead of the number of observations, following the same reasoning as in the maximum likelihood learning case: We do not want to excessively decrease the uncertainty in the model due to unreliable observations.

We can then simply define a new Gaussian model that takes the uncertainty in the parameters into account:

$$\psi_{i,j,t}(x_{i,t}, x_{j,t}) = \mathrm{e}^{-\frac{(r_t - \hat{\mu}_t)^2}{2\tilde{\sigma}_t^2}}. \tag{4.2}$$

4.2.2 Uncertainty in the Parametric Model

Since we use a parametric model, the range of relations that can be represented with this model is limited. It is obvious that not all the observed relations will correspond to the chosen model. As motivated earlier, we are mainly interested in learning those relations that fit the chosen parametric model. Those that do not correspond will simply be discarded. The problem is that we may need a lot of observations to be able to conclude that the model is not appropriate. During this time, the model is still used to track the features which may create a strong bias in the tracking, causing it to fail.

As, here, the parametric model corresponds to a Gaussian around a rigid position, the questions are "How to estimate the probability that the observed relation is rigid?" and "How to modify the potential function so that it doesn't create bias in the tracking?"

A Potential Function with Limited Bias

Let's first assume that we are able to estimate the probability that the relation corresponds to a Gaussian model. We call it λ_t for the probability at a given time t. Consider now the two extreme situations $\lambda_t = 1$ and $\lambda_t = 0$.

If $\lambda_t = 1$, then we can simply use the potential function of equation 4.2 without risk of creating bias.

If $\lambda_t = 0$, then the learned potential function is completely wrong and must be replaced with something else. But since the observations are not kept in memory, what can be used to compute this 'something else'? The answer is obvious: 'Nothing !'. In this case, all we can do is to say that we know nothing about the distribution of the relation. Then, the only thing we can do is to represent this lack of knowledge with a uninformative uniform potential.

In practice, nothing is black or white and a model may be more or less appropriate for the relation. Therefore, we represent the potential function by a weighted sum of the learned model and a uniform potential. The probability of observing a relation $r_{i,j,t} = x_{i,t} - x_{j,t}$ between features i and j at time t is then given by

$$\psi^{+}_{i,j,t}(x_{i,t}, x_{j,t}) = \lambda_t \mathrm{e}^{-\frac{(r_t - \hat{\mu}_t)^2}{2\tilde{\sigma}_t^2}} + (1 - \lambda_t) * 1, \tag{4.3}$$

Figure 4.3 shows an example of an uncertain potential function. Notice that the value of the uniform distribution is equal to one since it is the only value that does not affect a product which is what we want for an uninformative relation.

The only question left is "How to estimate the probability that the observed relation is rigid?"

The Probability of the Parametric Model

To estimate λ_t, we introduce a method inspired from the Kolmogorov-Smirnov test. Recall that the Kolmogorov-Smirnov distance is given by

$$K_n = \sqrt{n} \max_{-\infty<x<\infty} \left| \hat{F}(x) - F_n(x) \right|, \tag{4.4}$$

where n is the number of observations, $F_n(x)$ is the empirical cumulative distribution function of the n observations, and $\hat{F}(x)$ is the cumulative maximum-likelihood distribution. This distance is compared to a threshold, say, to classify a sample as Gaussian or non-Gaussian.

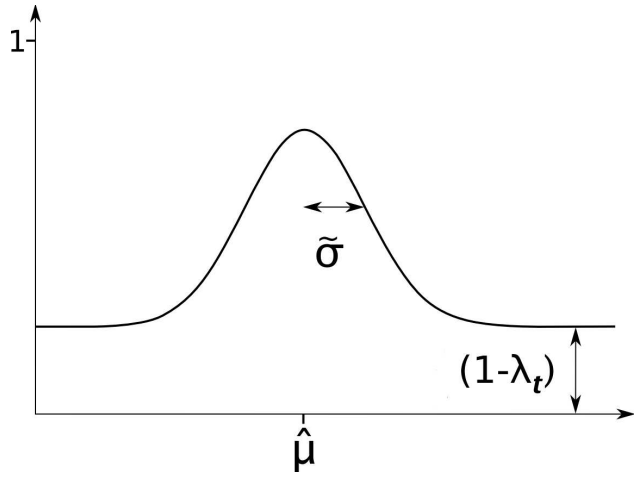


Figure 4.3: Example of an uncertain potential function.

Our context is different: We are not interested in precise Gaussianity but in relations that are about as *predictive* as Gaussians. Therefore, the original Kolmogorov-Smirnov test is unsuitable in that its sensitivity grows without bounds with n. Instead of Eqn. 4.4, we use an expression independent of the number of observations,

$$D = \frac{1}{|I|} \int_I \left| \hat{F}(x) - F_n(x) \right| \mathrm{d}x, \tag{4.5}$$

where I is the interval within which the two functions are compared. Notice that we use an integral instead of the maximum as it renders the measure both more robust and more discriminative in our context: Since it considers more than a single value it produces smoother curves, and outliers are more robustly detected because their cumulative effect is picked up by the integral.

To simplify matters, we assume that this distance D has a Gaussian distribution, which leads to the pseudo-probabilistic weighting function

$$\lambda_t = \mathrm{e}^{\frac{-D^2}{T_D^2}}, \tag{4.6}$$

where T_D is a user-settable parameter that represents the allowed deviation of observed relations from Gaussianity.

Notice that the cumulative density function of the observations $F_n(x)$ seems to be a simpler choice to model the relation than the parametric model. You may, then, wonder why we don't use it since it is able to represent any

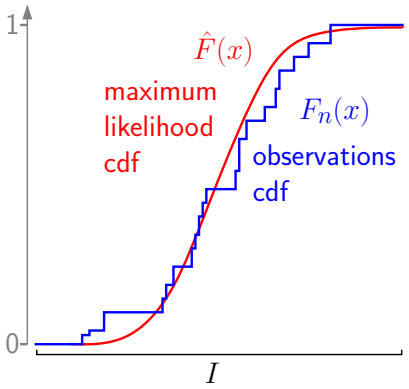


Figure 4.4: Method inspired from the Kolmogorov-Smirnov test.

king of distribution. The problem comes from the fact that $F_n(x)$ is only calculated on a limited interval I. Since the uncertainty on the model implies that the potential function can be greater than zero in an infinite interval, $F_n(x)$ cannot be used without creating bias in the tracking. On the other hand, it is enough to compare $F_n(x)$ and $\hat{F}(x)$ to a limited interval to get a good approximation of the correspondence between them.

4.3 Uncertainty in the Tracking

In our scenario, the potential functions are learned incrementally, which sometimes leads to situations where an observation at the current time t is in fact not well predicted by the relations learned up to time $t-1$. This may happen if, for example, the two features connected through a relation were motionless until time $t-1$, and one of them starts to move at time t. In this case, it is clear that the rigid relation learned from previous frames is no longer appropriate to track these features. To account for this *uncertainty in the tracking*, we need to augment accordingly the variance of the learned relations. Therefore, in the following we distinguish between the potential $\psi^{+}_{i,j,t-1}$ learned up to and including time $t-1$, and its variance-augmented counterpart $\psi^{-}_{i,j,t}$ that replaces $\psi_{i,j,t}$ in Eqn. 3.13.

A given relation is only used for tracking in the single next frame. Given that the observations in a video are spatially correlated over time, the next observations will not be too far from the current models. If we assume that the (application-dependent and fixed) likelihood of making an observation a distance Δ away from the learned model follows a Gaussian distribution of variance σ^2_Δ, a suitably augmented potential $\psi^{-}_{i,j,t}$ can be obtained by con-

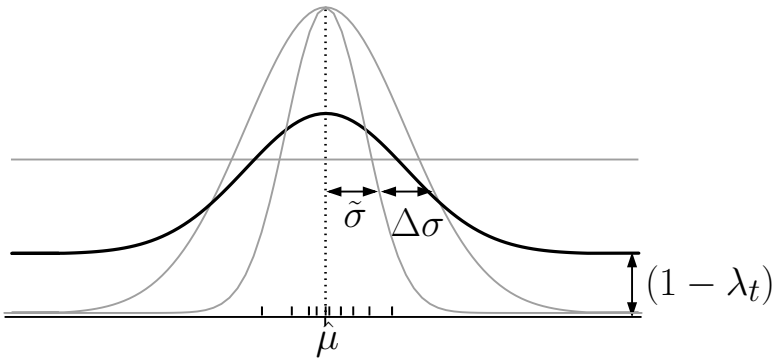


Figure 4.5: Inclusion of the uncertainty in the tracking in the model.

volving the learned model $\psi^+_{i,j,t-1}$ with a zero-mean Gaussian with a variance of this order of magnitude:

$$\psi^-_{i,j,t} = \psi^+_{i,j,t-1} \circledast N(0, \sigma_\Delta) \tag{4.7}$$

Putting together equations 4.7 and 4.3, we obtain the complete uncertain potential function used for the feature tracking:

$$\psi^-_{i,j,t} = \lambda_t \mathrm{e}^{-\frac{(r_t - \hat{\mu}_t)^2}{2(\tilde{\sigma}_t^2 + \sigma_\Delta^2)}} + (1 - \lambda_t) * 1. \tag{4.8}$$

Figure 4.5 illustrates the inclusion of the uncertainty in the tracking in the model and figure 4.6 represents the whole method that combines all the sources of uncertainty.

4.4 Experiments

4.4.1 The Complete Uncertain Model

In this section, we demonstate the performance of our method on a representative example of learning and tracking of 2D rigid relations. To emphasize the contribution of the relations we chose to use a very simple feature descriptor. Features are represented by fixed image templates extracted from the first frame, and their likelihoods are computed using the sum of squared pixel differences. Their 2D coordinates are tracked with particle filters; no orientation or scale changes are considered. The informative part of the relations is represented by a Gaussian model for each 2D coordinate. We use

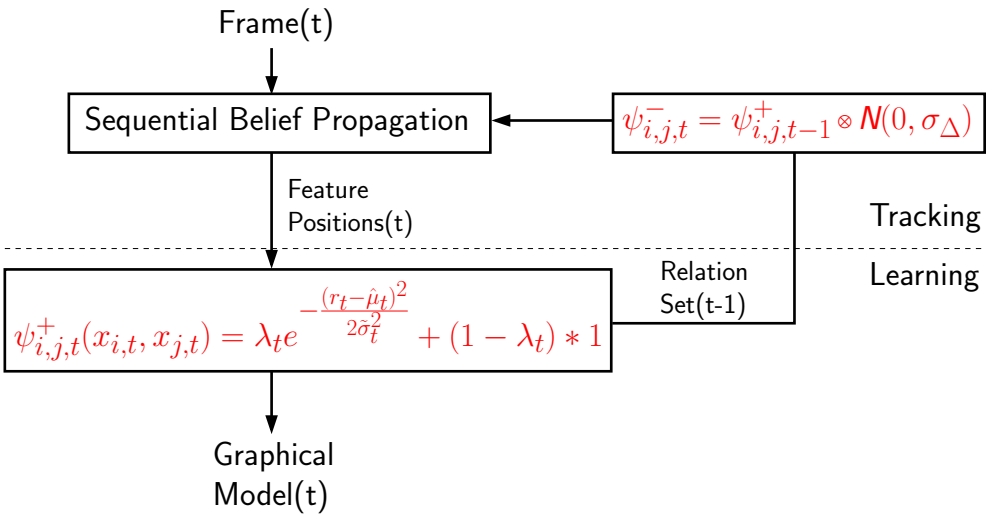
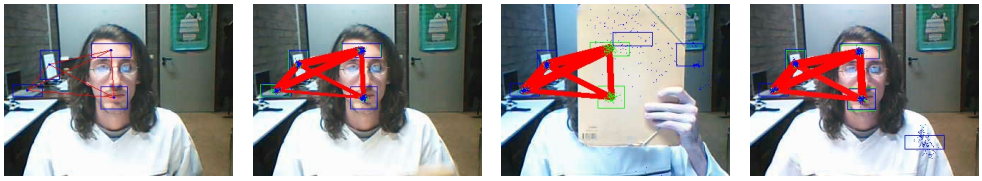


Figure 4.6: The whole algorithm.

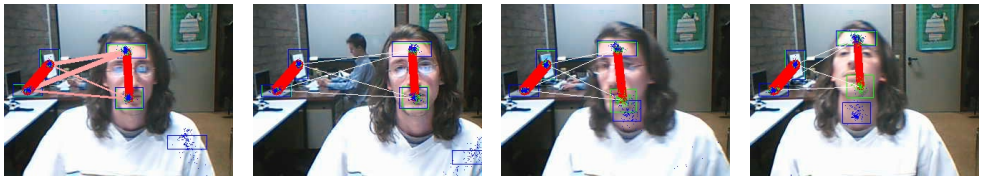
$T_D = 0.04$ and $\sigma_\Delta = 5$ pixels for equations 4.6 and 4.7. These are the only user-settable parameters of our method.

Figure 4.7 shows the result sequence taken with a webcam at a resolution of 320×240 and processed on-line. Four features were selected by hand, two from the face and two from the background. Relations were also selected by hand. In this example, the relations are first learned as rigid because the scene is initially motionless. As seen in Figs. 4.7(b) and 4.8(a), the relations related to the mouth are learned more slowly than the others due to their lower likelihood in the image. Once the head starts to move, the rigid relations connecting it with the background are rapidly unlearned. The probabilities λ_t of these relations become insignificant, and their variances increase. This clearly separates the graph into two subgraphs, one for the face and another for the background. Over the following frames, the face is successfully tracked despite the occlusions and its out-of-plane motions. Once the face–background relations have been detected as non-rigid, they do not influence the tracking anymore.

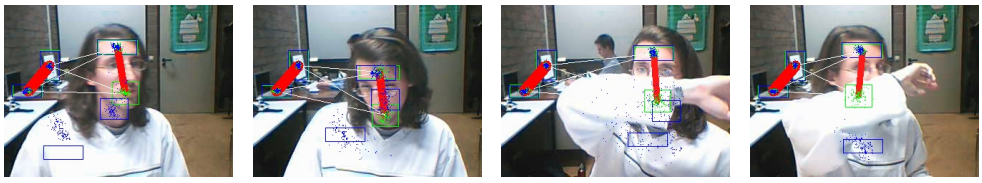
To illustrate the effect of the uncertain models on tracking, we track duplicate versions of the features without relations, represented in blue in Fig. 4.7. As the figure reveals, these features are tracked very poorly and have to be reinitialized many times during the sequence. It is thus clear that it would have been very difficult to learn a relational model from them without exploiting the – albeit uncertain – partially-learned relations from the start.



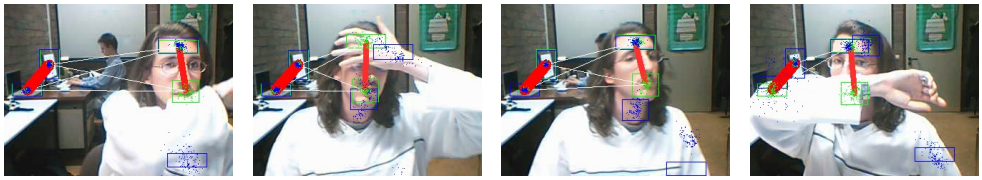
(a) frame 001 (b) frame 020 (c) frame 050 (d) frame 100



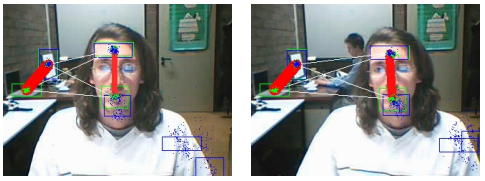
(e) frame 113 (f) frame 130 (g) frame 142 (h) frame 200



(i) frame 227 (j) frame 246 (k) frame 305 (l) frame 338

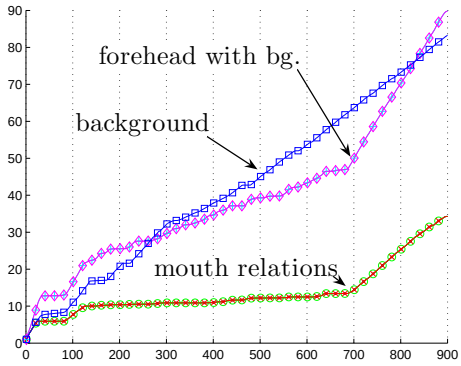


(m) frame 378 (n) frame 457 (o) frame 509 (p) frame 580

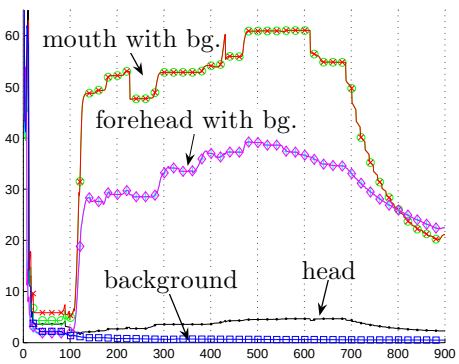


(q) frame 700 (r) frame 900

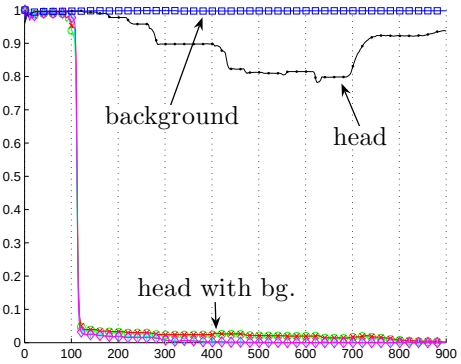
Figure 4.7: Representative results. Thick lines correspond to relations of low variance $\tilde{\sigma}$ and vice versa; red saturation is proportional to the probability λ_t. Each feature is tracked twice, with relations (green) and without (blue). Frame indices are given below each image.



(a) weight



(b) variance



(c) probability

Figure 4.8: Evolution of the relations. The two relations between mouth and background are superimposed, as are those between the forehead and the background. In (a), all three mouth relations are superimposed.

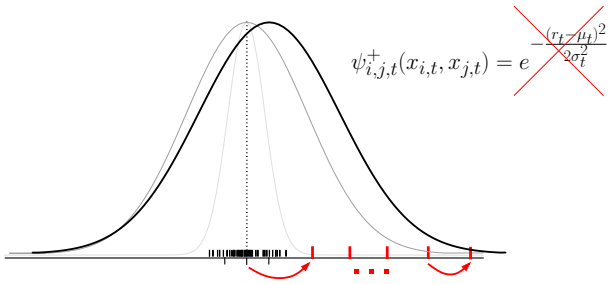


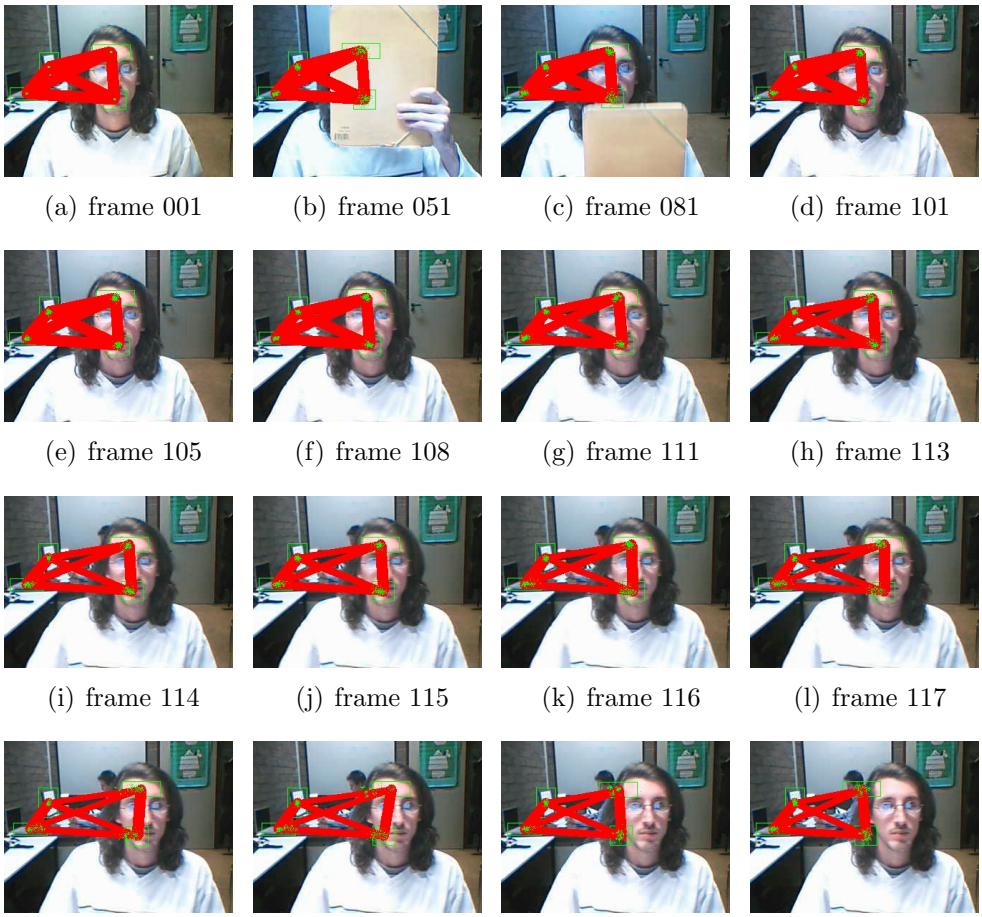
Figure 4.9: The relation is not rigid anymore

The end of the sequence (frames 700–900) is mostly motionless. Figure 4.8 shows that the probability of the forehead–mouth relation slowly increases and that all variances decrease. The probabilities of the relations between the facial features and the background do not increase because they were clearly non-rigid during the major part of the sequence. It will thus take much more time for their observation distributions to return to a Gaussian shape.

We implemented our algorithm on a Pentium Core 2 Duo 2 × 2 GHz. For a number of features between 4 and 10 with 3 relations each, it runs at between 8 and 20 frames per second.

4.4.2 The Uncertain Model without the Uniform part

We can see in figure 4.8 that the variance and the probability of the model have similar reactions. We might then wonder if the uniform distribution is really useful and if the increase in the variance isn't enough to avoid the strong counterproductive bias. The example in figure 4.10 is based on the same video and the same initialization than the previous section but the potential functions are learned without the uniform part. As we can see, even if the variance increases once the head moves, allowing to track it successfully the beginning of its motion, the tracking fail after a few frames. The reason is that the observations are not well represented by the parametric model any more and the variance values become meaningless as we show in figure 4.9.



(a) frame 001 (b) frame 051 (c) frame 081 (d) frame 101

(e) frame 105 (f) frame 108 (g) frame 111 (h) frame 113

(i) frame 114 (j) frame 115 (k) frame 116 (l) frame 117

(m) frame 118 (n) frame 119 (o) frame 120 (p) frame 121

Figure 4.10: Even if the variance increases once the head moves allowing to track it successfully the beginning of its motion, the tracking fails after a few frames.

4.5 Conclusion

In this part we presented a new framework for on-line learning of feature graphs. This method is completely unsupervised and uses tracking to find correspondences between features. At the same time, information extracted from previous frames is immediately used to aid the tracking in the new frame. For representing this information, we proposed an uncertain model of relations based on a parametric model that incurs only a low computational cost.

Several sources of uncertainty were identified and incorporated into the representation of the relations. The resulting uncertain model contributes stability to tracking without exerting overly strong, counterproductive bias.

The experiment demonstrates the ability of the uncertain model to assist tracking without biasing it, and – conversely – that tracking was essential for learning the uncertain model. The algorithm performed successfully under various difficulties such as occlusions, clutter and spurious connections between uncorrelated features.

Until now, we presented the theory for the case of rigid, Gaussian relational models, but similar developments are possible for other parametric distributions. We will explore some of the possible ways to extend this theory to articulated relations in the third part of this thesis.

Chapter 5

Use of an Uncertain Articulated Model

5.1 Introduction

In this last part, we propose to extend the uncertain model to articulated relations. The easiest way to do it seems to replace the rigid model with a Gaussian distribution around some parametric curve (a segment, an arc,...). If we do that, the only thing we need to change is the incremental learning of the maximum likelihood model. Unfortunately, incremental learning of parametric models is a surprisingly difficult task. Indeed the maximum likelihood curve is the one that minimize the square distance between the observations and their projections on the curve. As we can see in figure 5.1, the projections evolve with the parametric model. Since we need to recompute the projections each time the curve changes, it is impossible to estimate properly the new model and its probability only from the previous model and the new observation. The sequential learning is then an important limitation to the possible extentions of the uncertain model.

The case of the Gaussian model was simple because the projections cannot change. We then propose two possible combinations of Gaussian models that strongly reduce or completely avoid the projections problem: one with a Gaussian sum and one with a Gaussian product.

The Gaussian mixture model. Here, the observations are splitted between the Gaussians of the mixture. Even if the projections may switch from one Gaussian to an other, Arandjelovic *et al.* [1] showed that it is possible to incrementally learn a Gaussian mixture while keeping the projections mostly unchanged. We will adapt their method to our uncertain Gaussian model.

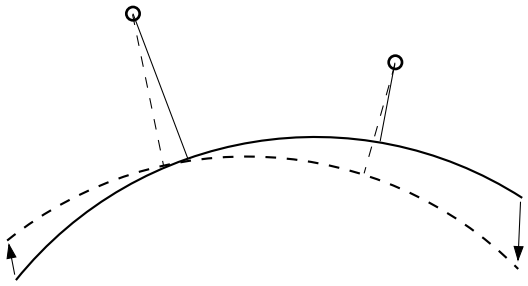


Figure 5.1: The projections evolve with the parametric model.

Product of space transformations. Even if the relation is not rigid in the image space, it can still be Gaussian distributed along some function of the image coordinates. For example, if the relation is distributed along a circle centered at one of the features, the distance between them is a constant. If the relation is a short arc, it can also be approximated with a line segment. If we now combine the information from the distance function and the line function, we can pretty well limit the potential function to this arc. in general, we can describe a relation with a set of space transformations where this relation is observed as rigid. The combination of those weak descriptors will allow to express articulated relations as a combination of Gaussian models; avoiding the problem of projections.

In this chapter we will explain these two approaches and explore their possibilities.

5.2 Uncertain Mixture of Gaussians

The most common way to fit a Gaussian mixture to a set of observations is with the Expectation-Maximization (EM) algorithm [5]. For a proper initialization, the algorithm will give the following Maximum Likelihood solution on the Gaussian parameters:

$$\pi^i = \sum_{j=1}^{N} p(i|r_j)w_j, \qquad \hat{\mu}^i = \frac{\sum_{j=1}^{N} r_j p(i|r_j)w_j}{\pi^i}, \tag{5.1}$$

$$\hat{C}^i = \frac{\sum_{j=1}^{N} (r_j - \hat{\mu}^i)(r_j - \hat{\mu}^i)^T p(i|r_j)w_j}{\pi^i}, \tag{5.2}$$

where $\hat{\mu}^i$ and $\hat{C}^i$ are the mean and the variance of the Gaussian i, and π^i is its cumulative weight. r_j is the observation j of the relation and w_j its weight. $p(i|r_j)$ is the probability of the i-th component conditioned on relation observation r_j.

Since the learning is incremental, the Gaussians can only be updated with the model in the previous time and the new observation. Rewriting the previous equations for an update with the new observation r_t, we get

$$\pi_t^i = \sum_{j=1}^{t-1} p_t(i|r_j)w_j + p_t(i|r_t)w_t, \tag{5.3}$$

$$\hat{\mu}_t^i = \frac{\sum_{j=1}^{t-1} r_j p_t(i|r_j)w_j + r_t p_t(i|r_t)w_t}{\pi_t^i}, \tag{5.4}$$

$$\hat{C}_t^i = \frac{\sum_{j=1}^{t-1}(r_j - \hat{\mu}_t^i)(r_j - \hat{\mu}_t^i)^T p_t(i|r_j)w_j + (r_t - \hat{\mu}_t^i)(r_t - \hat{\mu}_t^i)^T p_t(i|r_t)w_t}{\pi_t^i} \tag{5.5}$$

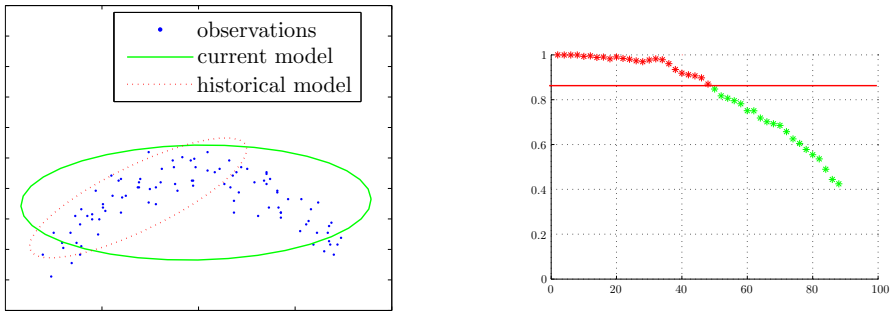
As we can see, we can only forget the previous observations if we make the assumption

$$p(i|r_t) \backsimeq p(i|r_{t-1}), \qquad \forall t \tag{5.6}$$

which can be false for two reasons:

- The position and variance of the Gaussians change with time causing the component likelihoods $p(i|r_t)$ to change too.
- The number of Gaussians in the mixture is not constant. When a new Gaussian is added to the mixture, the component likelihoods related to the observations closed to its mean are strongly affected.

Arandjelovic *et al.* [1] proposed to use an *Historical Gaussian Mixture Model* (HGMM) that corresponds to the Gaussian Mixture at the moment a new Gaussian was added. At each time, they compare the current mixture



(a) The current and historical models. (b) Evolution of the probability.

Figure 5.2: Splitting of an uncertain Gaussian.

with the HGMM and if the difference in the parameters of a Gaussian is too big, they split it into its historical part and a new Gaussian built with the remaining observations. The use of the Historical Model combined with the temporal coherence of the observations allows the assumption of equation 5.6 to be true. In this section, we propose to adapt this idea to our uncertain model.

5.2.1 Splitting of an Uncertain Gaussian

Let's first consider the simple case where the mixture is composed of a single Gaussian uncertain model. We are then in the same situation as in the rigid-model case. Assume now that the probability of this Gaussian model become very low meaning that the relation is not rigid and cannot be modeled with a Gaussian. We then have two possible choices: to remove the relation from the graph or to switch to a more complex model. Here we choose the second solution by adding a second Gaussian in the mixture, increasing the range of relations the mixture can model.

The question is how and when to split the Gaussian. To answer the 'when?', we use a threshold on the probability of the uncertain model: it specifies the minimum probability that is allowed in the mixture model. This way we guarantee that the quality of the model will always be higher than this limit. The 'how?' is solved by using a historical model that corresponds to the last occurrence of the uncertain model with a probability higher than a second threshold as shown in figure 5.2. When a Gaussian has to be split, we

replace it by its historical model and add a new Gaussian learned from the the difference between this Gaussian and its historical model; whose parameters are given by:

$$\pi^{(n)} = \pi^{(o)} - \pi^{(h)}, \qquad \hat{\mu}^{(n)} = \frac{\pi^{(o)}\hat{\mu}^{(o)} - \pi^{(h)}\hat{\mu}^{(h)}}{\pi^{(n)}}, \tag{5.7}$$

$$\begin{aligned} \hat{C}^{(n)} &= \frac{(C^{(o)} - \mu^{(o)}\mu^{(o)^T})\pi^{(o)} - (C^{(h)} + \mu^{(h)}\mu^{(h)^T} - \mu^{(h)}\mu^{(o)^T} - \mu^{(o)}\mu^{(h)^T})\pi^{(h)}}{\pi^{(n)}} \\ &+ \mu^{(n)}\mu^{(o)^T} + \mu^{(o)}\mu^{(n)^T} - \mu^{(n)}\mu^{(n)^T}, \end{aligned} \tag{5.8}$$

where (n), (o) and (h) refer respectively to the new, the old and the historical Gaussian. Arandjelovic *et al.* [1] argue that, thanks to the temporal coherence between the observations, the split of the old Gaussian gives good results. This is often correct for the case where the relation evolves in one direction creating new observations only on one side of the Gaussian. In the case of a cyclic motion it is not correct anymore. This situation is easy to detect since the Gaussian created by the splitting will have a low Gaussianity probability. In that specific situation, no useful information about the distribution of the observation exists anymore and we then simply learn the new Gaussian from the last observation. Like that, the complexity of the mixture is still increased. Some observations are lost in the process but they were not useless since they allow to detect the need to increase the mixture complexity. Moreover, the goal is to converge to a good representation of the relation not to fit exactly the observations during the learning phase.

5.2.2 Learning the Uncertain Gaussian Mixture

Let's finally consider the whole mixture. The most important thing about the mixture is that each observation is assign to only one Gaussian, the nearest. Which means that the component likelihoods $p(i|r_t)$ are always equal to 0 or to 1. Several reasons motivate that choice:

- The computation cost is reduce since we only have to update one Gaussian's parameters at a time.
- It pushes the Gaussians to separate from each other.

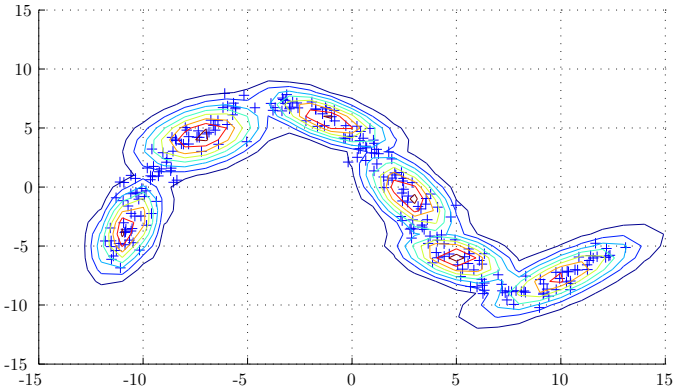


Figure 5.3: Example of a learned Gaussian Mixture.

Since some of the Gaussians may be created by mistake, it is interesting to have a process that remove the useless Gaussians from the mixture. So we compute, for each Gaussian, its frequency of update and delete those that have a frequency considerably lower than others (for example 10 times lower).

We also increase with time the splitting threshold on the Gaussian probability so that the model is very tolerant at the beginning and become more and more strict on the quality of the model.

The complexity of the model is limited by a limit on the maximal number of Gaussians in the mixture. This way, we can limit the connections in the graph to the relations that are most restrictive.

Figure 5.3 shows an example of a learned Gaussian Mixture for a complex relation.

5.2.3 Experiments

Influence of the order of Observations

Since the Gaussians are added on by one, it is obvious that the observations may be not equally distributed among them. We can then wonder about the observations' order dependency of the model. Fortunately, thanks to the increasing value of the splitting threshold and the deletion of the low-frequency Gaussians, the model often converges to similar solutions. To illustrate this behaviour, we generated observations of a circular motion at different angular speeds. In order to simulate the observation noise, we added some Gaussian noise on the position of the observations. As we can see in figure 5.4, the number of generated Gaussians is always equal to 7. In some

rare cases, we got only 6 Gaussians but it seems to be more dependent on the observation noise than on the angular speed.

Influence of the Probability Parameter T_D

The Gaussianity probability of equation 4.6 has, here, a slightly different purpose. Instead of balancing the influence of the different relations, it allows to detect if the complexity of the mixture model needs to be increased. This means that the parameter T_D on the allowed standard deviation of the observations from Gaussianity will influence the number of Gaussians in the mixture. Figure 5.5 shows the resulting mixtures for different values of T_D. As expected, a low value of T_D results in a more complex mixture than a higher value.

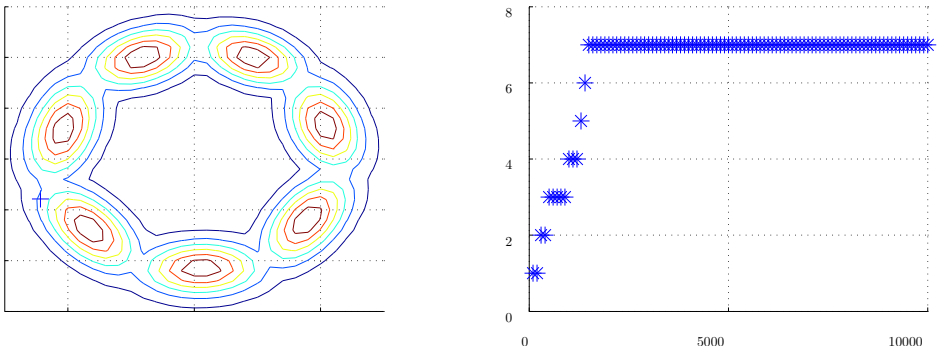
5.3 Uncertain Combination of Weak Parametric Models

While, in the previous section we presented a potential function that corresponds to a sum of models, here we propose a product of weak models; each limiting the relation. The idea is to say that, even if the relation is not rigid in the image space, it can still be rigid for some function of the image coordinates. For example, if the relation observations are distributed along a circle centered at zero, the function $f = r_x^2 + r_y^2$, where r_x and r_y are the relative cartesian coordinates of the features, will correspond to a rigid distribution.

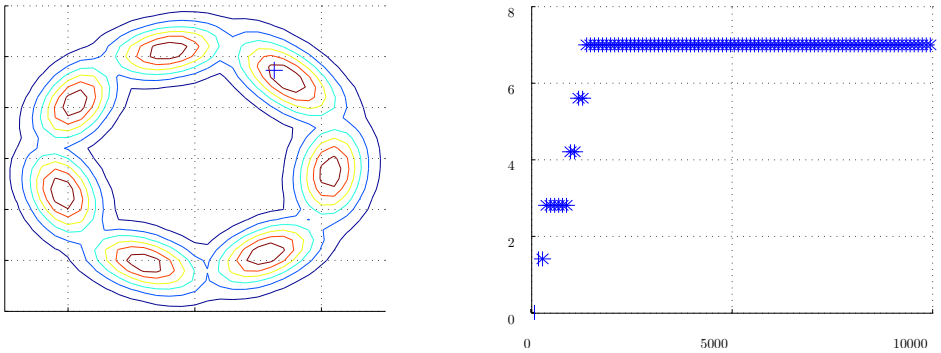
In this section we will first introduce a learning method based on a set of preselected functions. After that, we will explain how to automatically select those functions. We will finish the section with some experiments of the method.

5.3.1 Combination of Redundant Variables

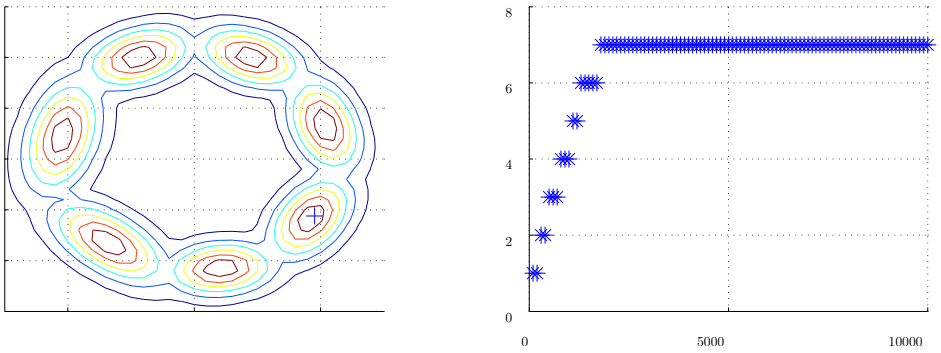
The uncertain rigid model of the second part can already be seen as a combination of two weak models: one in each cartesian coordinate. This means that if, for example, the relative position of two features is along a straight horizontal line, only their vertical relative position r_y will be rigid and informative. If now, the relation is distributed along an oblique line of 45 degrees, r_x and r_y will not be rigid anymore but the variable $r_x - r_y$ will. So, by using a set of redundant variables, we increase our chances to express a articulated relation with a Gaussian model in one of the variable space.



(a) angular speed = 1 rad

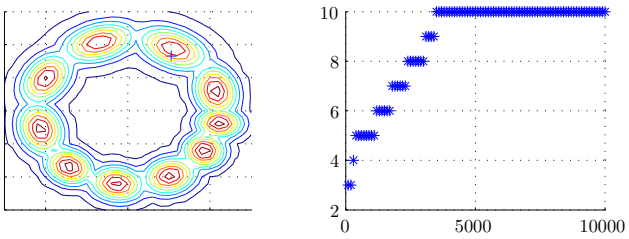


(b) angular speed = 0.1 rad

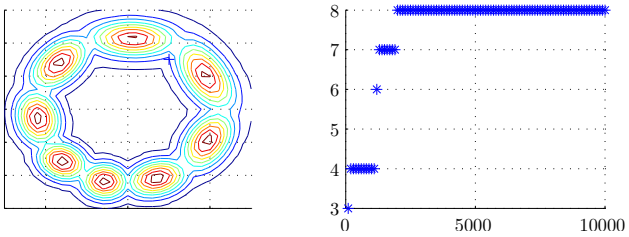


(c) angular speed = 0.01 rad

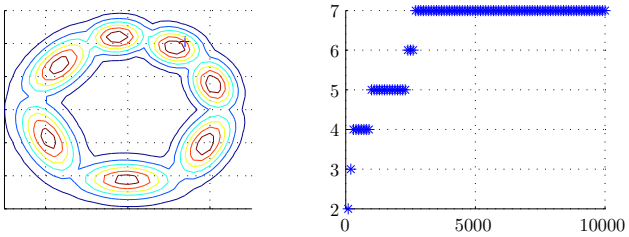
Figure 5.4: Resulting model and evolution of the number of Gaussians with the number of observations for different angular speeds.



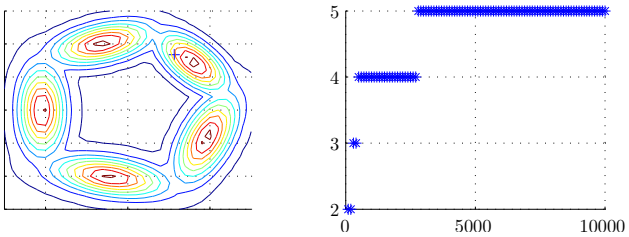
(a) $T_D = 0.02$



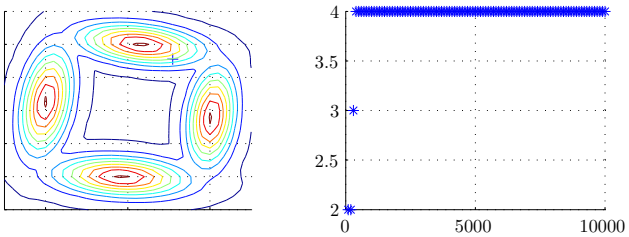
(b) $T_D = 0.025$



(c) $T_D = 0.03$



(d) $T_D = 0.035$



(e) $T_D = 0.04$

Figure 5.5: Resulting model and evolution of the number of Gaussians with the number of observations for different angular speeds.

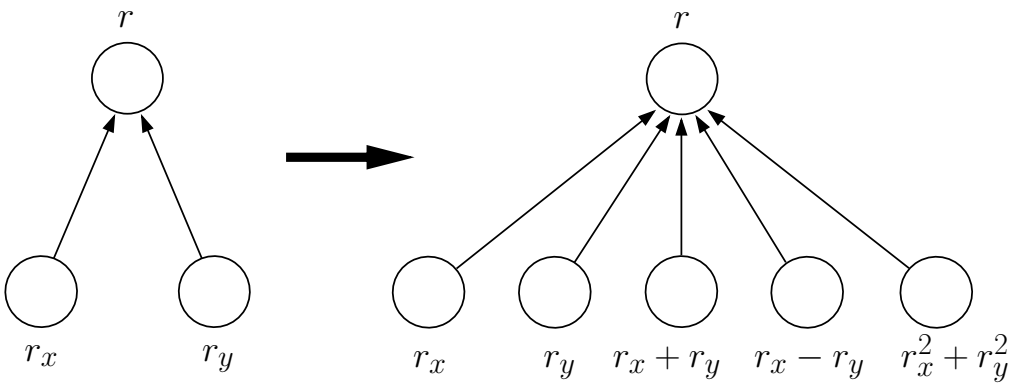


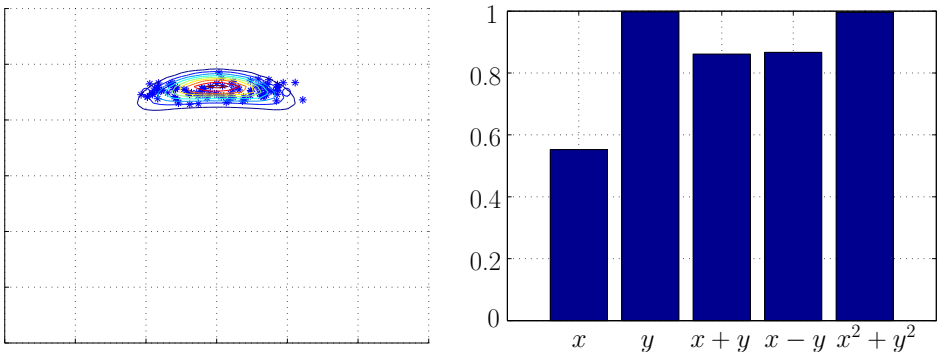
Figure 5.6: We can make the model more complex by adding redundant variables.

The set of variables we choose determines the set of relations we want to detect and represent. For example, if we use the variable set $V_S = \{r_x, r_y, r_x + r_y, r_x - r_y, r_x^2 + r_y^2\}$, we will be able to model the lines with an orientation of 0, 45, 90 and 135 degrees and the circles centered at zero. As we an see in figure 5.7, combinations of them can also strongly limit the relation potential function. Even if those results seem promissing, we have to keep in mind that each variable is limited to very short range of distributions. For example, $r_x - r_y$ is only capable to detect line with an orientation around 45 degrees; not any kind of line. This means that, if we want to be able to detect and represent all possible lines, we have to provide a lot of variables to be able to cover all the possibilities with sufficient precision. The case of the circle is even worse since we have to cover all possible centers. Figure 5.8 shows some of those situations. Notice that, if the observation are not well fitted by the model, none of the variables' probability is high; making the uniform part of the uncertain model significant.

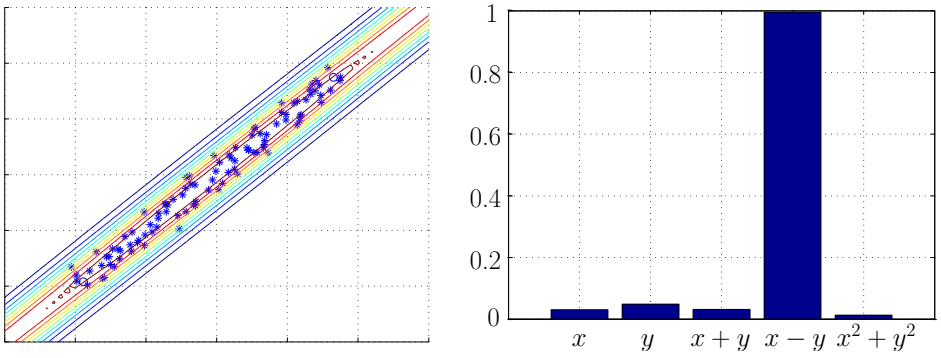
5.3.2 Selection of Appropriate Models

It would be great if we could explicitly express that we are only interested in, say, lines and circles and let the algorithm find the proper variables. This brings us back to the problem of incremental learning of a parametric model which is too difficult to deal with. So, somehow, we have to make sacrifices in order to go further. Let's begin with a tiny one: what is possible if we keep the last K observations?

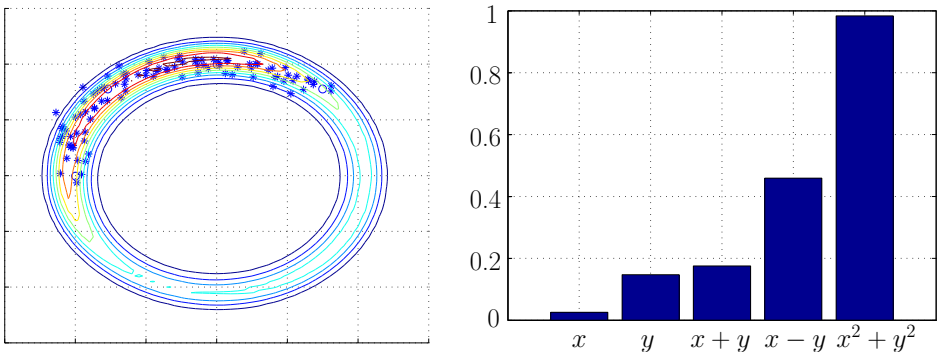
In this case, we can learn a parametric function of the cartesian coordi-



(a) nearly rigid

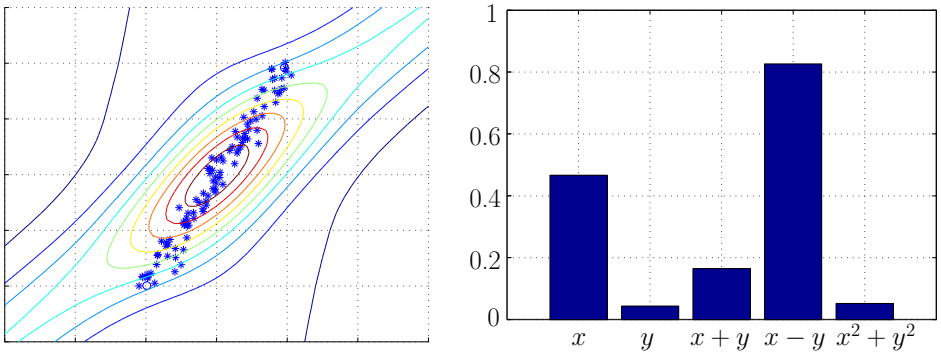


(b) oblique line of 45 degrees

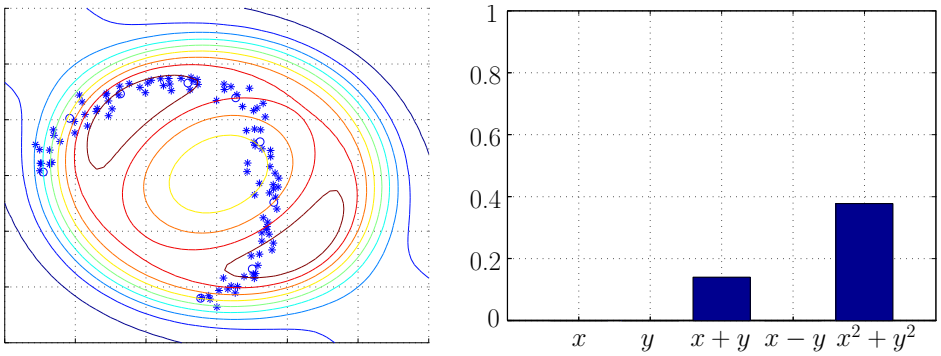


(c) arc with center at zero

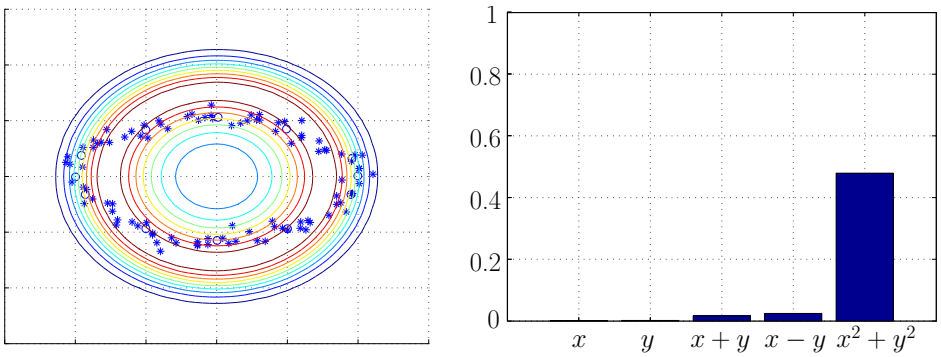
Figure 5.7: Resulting model and probability of each variable for appropriate observations.



(a) oblique line of 70 degrees



(b) arc with non-zero center



(c) ellipse

Figure 5.8: Resulting model and probability of each variable for less-appropriate observations.

nates that corresponds to a rigid Gaussian distribution for those observations. There is no guarantee that this function will prove to be useful but, at least, its chances of fitting the relation are higher than an arbitrarily selected function. Assume, for example, we want to learn functions that are able to model a line and a circle. In that case, the function must be like:

$$f_i = w_1 * x + w_2 * y + w_3 * x^2 + w_4 * y^2, \tag{5.9}$$

This function can be learned from the K observations by solving the following system of equations (with SVD, for example):

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & y_2^2 \\ : & : & : & : & : \\ 1 & x_K & y_K & x_K^2 & y_K^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ : \\ 0 \end{bmatrix} \tag{5.10}$$

Like for the Gaussian mixture model, a new variable is added each time the current model is not able to estimate the relation properly anymore. Contrary to the mixture model, we only need that a least one of the function fits the observations. Indeed, here, we multiply the information brought by each function while, in the case of the mixture, we add it. So, we just create a new function each time none of the existing functions has a higher probability than a given fitting threshold. We can also decide to remove functions with a low probability since they are not informative anymore.

The learning of the potential function starts with only two functions: $f_1 = x$ and $f_2 = y$. If the observation distribution is too complex for the current model, we learn both a new linear and a new quadratic function of x and y. This way, a complex relation is always a combination of, at least, four weak parametric functions. It is important to notice that the chances of learning a good parametric function increase if the relation's motion is not too slow. Indeed, this way, the last K observations have higher chances to be representative of the relation and, then, to provide a good parametric function. To limit the effect of slow motions, we can sample observations only every n frames.

5.3.3 Experiments

We have tested the methods for different values of the parameters and the results were as expected: the influence of T_D is not as important as for the mixture of Gaussians; which is not really surprising. Indeed, a low value of T_D only implies the creation of more parametric function but those functions

are mostly similar. The changes in the resulting potential function are not as impressive as for the Gaussian mixture. Figure 5.9 shows some examples of the combination of weak learned model.

In general, the number of parametric functions kept by the algorithm is very low (between 1 and 4). Since the combination of weak descriptors is specially useful either if one of the descriptor fits exactly the observations or if many descriptor may be combined, it is interesting to see how the results evolve if we increase the number of descriptors. Notice that the goal in this work is not to perfectly fit the observations but to learn relations that, somehow, help the tracking. Anyway, let's try the method with a bigger set of functions learned each time we need a descriptor:

$$
\begin{aligned}
f_a &= w_1 * x + w_2 * y \\
f_b &= w_1 * x + w_4 * y^2 \\
f_c &= w_3 * x^2 + w_2 * y \\
f_d &= w_3 * x^2 + w_4 * y^2 \\
f_e &= w_1 * x + w_2 * y + w_3 * x^2 + w_4 * y^2
\end{aligned}
$$

As expected, the potential functions are more selective as we can see in figure 5.10. So, depending on the kind of relations we want to model and the relative importance between precision and speed, we can design a set of functions type that, combined, will allow to refine the description of the relations of interest.

5.4 Conclusion

In this chapter, we presented two extentions of the uncertain model to articulated relations. Even if the incremental learning limits the types of model we can use, the uncertain model is still useful through the combination of Gaussian models. The two proposed extentions were each related to a different approach of combination: one by addition of models and one by multiplication of them. For both method, the concept of probability gets a second use: the detection of the need to increase the complexity of the combination.

The computational time needed for each method is not more than a few time slower than the one required for the rigid uncertain model. Indeed, in the case of the Gaussian mixture, only one Gaussian is updated for each frame and, in the case of the weak descriptors, their number is very low and only a few SVD occur on small matrices.

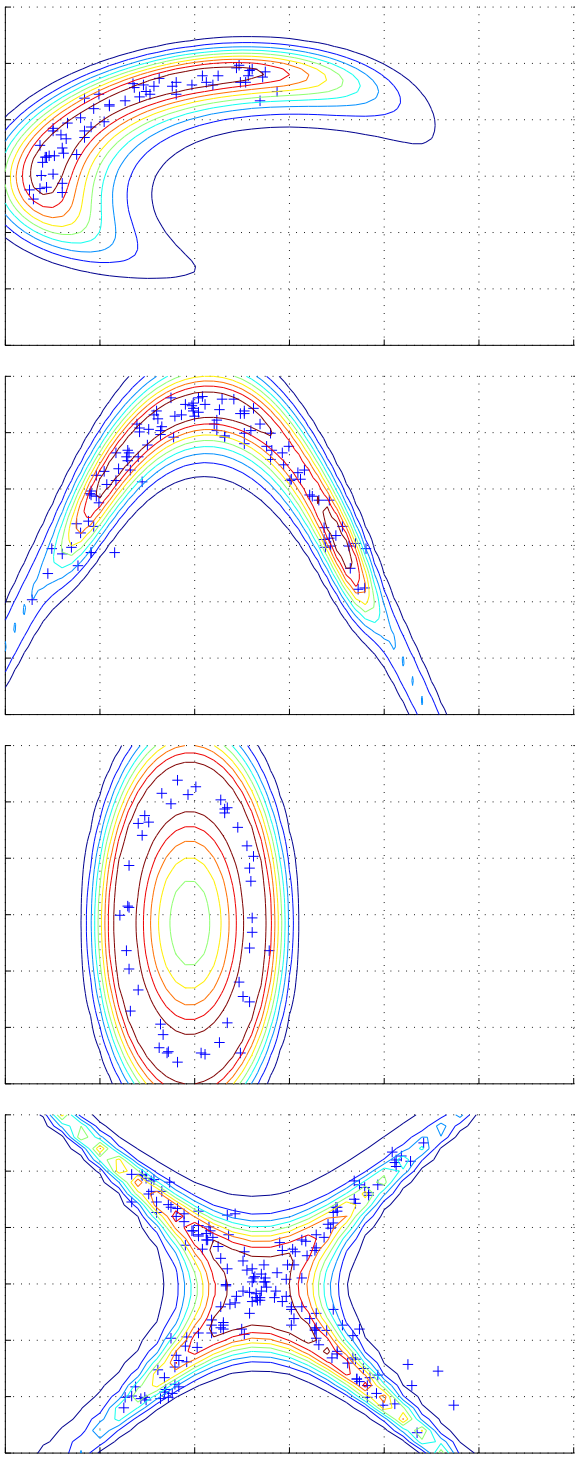


Figure 5.9: Some examples of the combination of weak learned model.

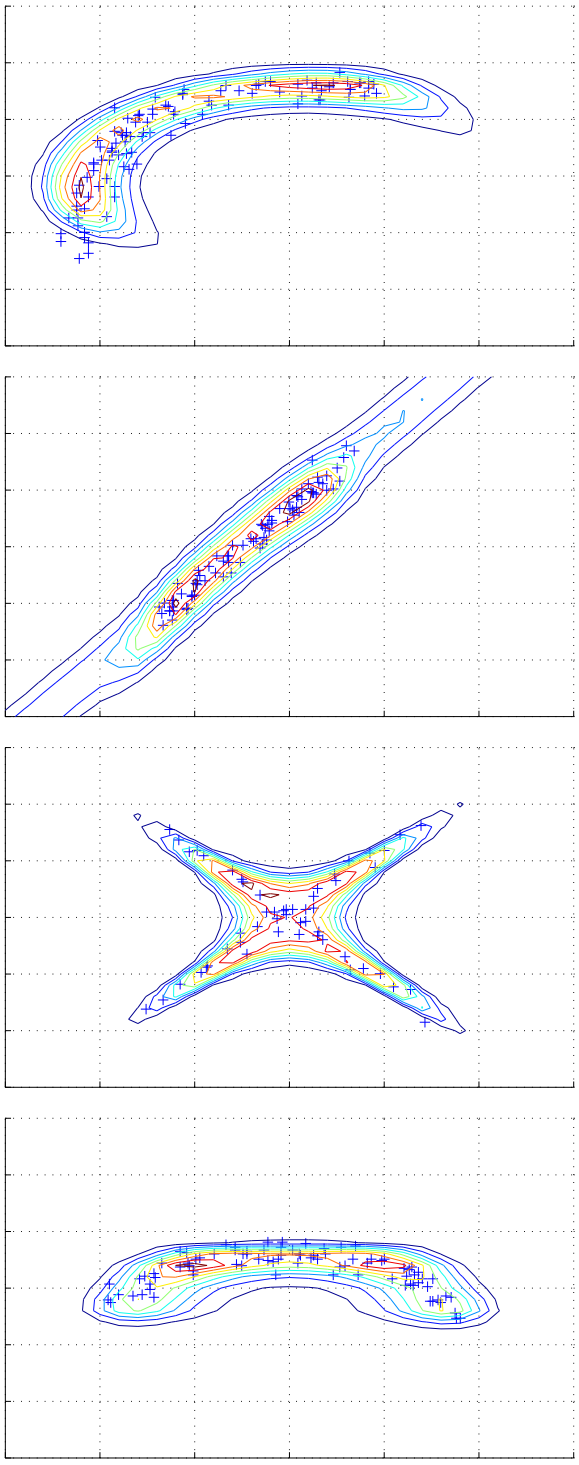


Figure 5.10: Some more examples of the combination of weak learned model. This time, with more quadratic function types

Chapter 6

Conclusion

In this thesis we presented a new framework for on-line learning of feature graphs. This method is completely unsupervised and uses tracking to find correspondences between features. At the same time, information extracted from previous frames is immediately used to aid the tracking in the new frame. For representing this information, we proposed an uncertain model of relations based on a parametric model that explicitly accounts for its predictive power.

Several sources of uncertainty were identified and incorporated into the representation of the relations. The resulting uncertain model contributes stability to tracking without exerting overly strong, counterproductive bias. The experiment demonstrates the ability of the uncertain model to assist tracking without biasing it, and – conversely – that tracking was essential for learning the uncertain model. The algorithm performed successfully under various difficulties such as occlusions, clutter and spurious connections between uncorrelated features.

The uncertain model was developed for rigid and articulated relations. For the case of articulated relations, two approaches were proposed to solve the problem of incremental learning of elaborate relations; both based on a combination of Gaussian models. The two complementary extentions were related to a different approach of combination: one by addition of models and one by multiplication of them. For both method, the concept of probability gets a second use: the detection of the need to increase the complexity of the combination. Each of the uncertain model is appropriate for an on-line process since they are learned incrementally and incur only a low computational cost.

Bibliography

[1] O. Arandjelovic and R. Cipolla. Incremental learning of temporally-coherent gaussian mixture models, 2006.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV'04*, 56(3):221–255, 2004.

[3] V. Comaniciu and P. Meer. Kernel-based object tracking. In *IEEE Trans. Pattern Anal. Machine Intell. vol 25*, pages 564–577, 2003.

[4] D. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher. Object recognition by combining appearance and geometry. In *Toward Category-Level Object Recognition*, pages 462–482, 2006.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likekihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[6] N. Dowson and R. Bowden. N-tier simultaneous modelling and tracking for arbitrary warps. In *BMVC'06*, volume 2, pages 569–578, 2006.

[7] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *IJCV'00*, 40(1):25–47, 2000.

[8] G. Hua and Y. Wu. Multi-scale visual tracking by sequential belief propagation. *CVPR'04*, 1:826–833, 2004.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *IJCV'98*, 29(1):5–28, 1998.

[10] M. Leordeanu and R. Collins. Unsupervised learning of object models from video sequences. *CVPR'05*, 1:1142–1149, 2005.

[11] J. Lim, D. A. Ross, R.-S. Lin, and M.-H. Yang. Incremental learning for visual tracking. In L. Saul, Y. Weiss, and L. Bottou, editors, *NIPS'05*, pages 801–808, 2005.

[12] T. Mathes and J. Piater. Robust non-rigid object tracking using point distribution models. In *BMVC'05*, pages 849–858, 2005.

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[14] D. Ramanan, S. M.-D. A. Forsyth, and M.-K. Barnard. Building models of animals from video. *IEEE Trans. Pattern Anal. Machine Intell.*, 28(8):1319–1334, 2006.

[15] F. Scalzo and J. H. Piater. Unsupervised learning of dense hierarchical appearance representations. In *ICPR'06*, volume 2, pages 395–398, 2006.

[16] L. Sigal, Y. Zhu, D. Comaniciu, and M. Black. Tracking complex objects using graphical object models. In *Proc. International Workshop on Complex Motion*, 2005.

[17] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *CVPR*, pages 605–612, 2003.

[18] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual hand tracking using nonparametric belief propagation. In *CVPRW'04*, volume 12, page 189, 2004.

[19] F. Tang and H. Tao. Object tracking with dynamic feature graph. In *VS-PETS'05*, pages 25–32, 2005.

[20] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Comput.*, 13:2173–2200, 2001.