Robust Infants Face Tracking with Active Appearance Model: a Mixed State Condensation Approach

Luigi Bagnato

24 November 2006

Preface

This thesis has been prepared over six months at the Image Analysis and Computer Vision laboratory within the Signal Processing Institute, ITS, at the École Polytechnique Féderale de Lausanne, EPFL, in fulfillment of the requirements for the degree of Master of Science in "Ingegneria Informatica e delle Telecomunicazioni" at University of Perugia.
During this period (from January to July 2006) Matteo Sorci, a PhD student of the research group lead by Prof. Jean-Philippe Thiran, assisted me for the preparation of this work.
My stay at EPFL concluded with a final oral presentation of my research to the members of research group of Prof. Jean-Philippe Thiran.

Perugia, November 2006
Luigi Bagnato

Contents

Chapter 1

Introduction

This work addresses the problem of tracking, in real video sequences, the global motion of an infant face as well as the local motion of its inner features. This is a challenging task in Computer Vision field, because of the variability of facial appearance within a video sequence, most notably due to changes in head pose, expressions, lighting or occlusions. Thus, much research has been devoted to the problem of face tracking, as a specially difficult case of non-rigid object tracking. This task requires, by definition, the use of a model which describes the expected structure of the face. The advantage of explaining image data in terms of model parameters is to provide a natural basis for further interpretation, and it can be exploited by Human-Machine Interaction applications. (Actually this work is currently used in the facial expression recognition project at EPFL[1]).

In figure 1.1, some example images show inherent difficulties when dealing with real video sequences of infants. There are two major elements, which add complexity to the tracking task:

1. Infants move continuously and in an unpredictable way, producing face self occlusions as most undesirable effect.

2. External objects (an hand in figure 1.1) may occlude the target face either partially or totally.

[1] `http://itswww.epfl.ch/~sorci`

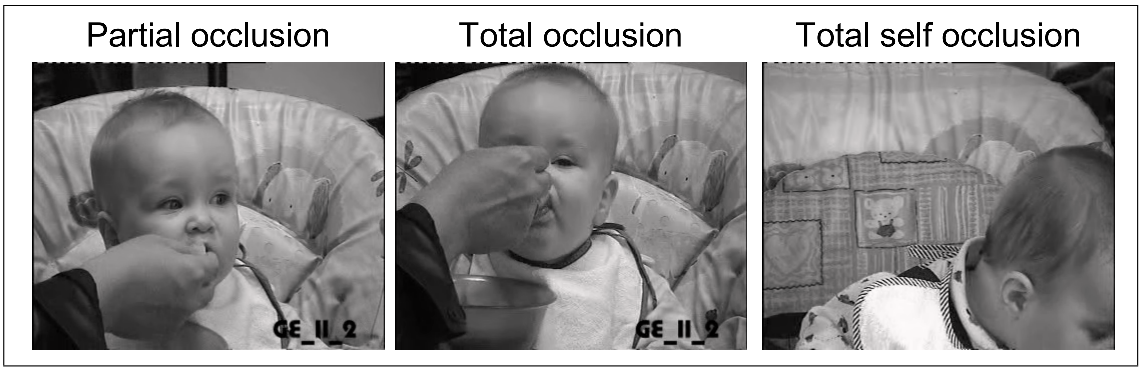


Figure 1.1: Example images from input video sequences.

The main objective of this thesis work is then to provide a theoretical framework and a practical implementation for a robust tracker, with the following characteristics:

1. Tracking of a single face in a video sequence.
2. High responsiveness: the tracker must follow the target face even if sudden movements occur.
3. Robustness to partial occlusions: the tracker must recognize a face in presence of limited partial occlusions.
4. Robustness to total occlusions: the tracker must recover the face after distraction by a total occlusion.

1.1 Approach

In this thesis, a model-based tracking approach is taken. Active Appearance Models (AAMs) were proposed by Cootes et al. [1] in 1998, as one of the more sophisticated *deformable template models*, a family of methods to represent real objects characterized by high variability. Since its introduction it has been successfully applied to face interpretation. When compared to other models, it has the main property to provide a complete and compact target representation.

Complete means that AAMs are photo-realistic generative models of appearance; thus, for instance, an AAM-face can generate convincing images of any individual, changing their expression, and so on. The compactness is due to an effective combination of statistical techniques used to learn plausible structure variations of the target. Furthermore, it comes with a fast and effective algorithm, named *AAM search*, to match the model to the image data.

Thus, an AAM has been a natural choice in this work for face representation. A more detailed description of AAMs is given in section 2.2.

Tracking can be formulated as a state estimation problem, based on observations in image sequences. Bayesian methods are attractive, as they provide a principled means of encoding uncertainty about parameter estimates. In probabilistic formulation, the system dynamics is described by a state transition distribution, and an observation model specifies the likelihood of an hypothesised state, i.e. the probability that the considered state might generate the observed data (a review on Bayesian filtering is given in Chapter 3).

Particle Filtering is a technique for the implementation of a recursive temporal Bayesian filter, by means of Monte Carlo simulations: compared to the well-known parametric filters, such as the Kalman filter, it does not assume a functional form of the posterior state distribution, and it is particularly suited for visual tracking, where occlusions and cluttered background encourage the distribution to be multi-modal.

For visual tracking, the Condensation is a well known particle filtering algorithm, which consists in approximating the posterior state distribution by a set of random weighted samples (particles), and in propagating this sample set through time using the system dynamics.

The state transition model, which dominates the propagation of particles, plays a very important role in the particle filter implementation. For our tracking problem, a single motion model results insufficient to predict accurately the position of the target. The stochastic framework offers rich modeling possibilities, and so we propose a mixed state Condensation scheme which includes multiple models to better describe the target dynamics and to manage the occurrence of large occlusions.

1.2 Thesis Overview

Chapter 2 This chapter presents a literature review of visual tracking, where different approaches are presented and discussed. Then, Active Appearance Models are described together with the AAM search.

Chapter 3 This chapter first formulates the task of tracking as a Bayesian inference problem. Particle Filters methods are then introduced as a recursive solution to implement a Bayesian Filter. The final stress is upon Condensation, a general particle filtering algorithm used for visual tracking.

Chapter 4 This chapter describes the main contributions of this work. A new Condensation based algorithm is obtained by integrating Active Appearance Models in a Bayesian framework and using a mixed-state model representation of the target dynamics.

Chapter 5 The proposed algorithm is compared with other common approaches found in literature, to assess its performance.

Chapter 6 Conclusion and future work.

Chapter 2

Related Work

In this chapter, we first give an overview of the the current state-of-the-art in visual tracking. We will focus on face tracking with Active Appearance Models, highlighting strengths and limits of existing approaches.

Then, we describe how to construct a face Active Appearance Model and how to use it to detect a face in an image.

2.1 Visual tracking

In the object tracking literature, the following formulation of the tracking problem is conveniently used: at each time step k, the goal is to infer the unobserved state of the object, denoted $\mathbf{x}_k$, given all the observed data until time k, denoted $\mathbf{y}_{1:k} = \{\mathbf{y}_1, .., \mathbf{y}_k\}$. When tracking a face, the unobserved state $\mathbf{x}_k$ includes motion or pose parameters like the position, scale and orientation of the face; when facial features are also tracked, the unobserved state should contain parameters describing the face inner motion. The observed data $\mathbf{y}_k$ consists of measurements derived from the current video frame, such as greylevel patches, edges, or color histograms [2][3]. In order to evaluate a hypothesized state, the measurements are actually only considered in the image area corresponding to the hypothesized location. For instance, the most natural measurement consists of the pixel greylevel values themselves.

The tracking task, then, essentially consists in searching the current state $\mathbf{x}_k$, which matches at best the measurements $\mathbf{y}_k$ in the current image. The tracking history $\hat{\mathbf{x}}_{1:k-1}$ is mainly used as a prior knowledge in order to search only a small subset of the state space. More specifically, for a hypothesized state $\mathbf{x}_k$, measurements $\mathbf{g}(\mathbf{x}_k)$ are extracted from the image patch at the hypothesized location, and those measurements are matched against some model of the object. The various tracking methods can be categorized according to the considered class of measurements, joint model of state and measurements, induced matching criterion (error functional to minimize or probability to maximize), and inference technique (deterministic or stochastic). According to this classification, a brief and non-exhaustive survey of tracking approaches is given below.

Roughly speaking, previous work on visual tracking can be divided into two groups: deterministic tracking and stochastic tracking.

Deterministic approaches usually reduce to an optimization problem, where the state $\mathbf{x}_k$ is usually seeked so as to minimize an error functional $d[\mathbf{g}(\mathbf{x}_k); \mathbf{g}_{model}]$, where $\mathbf{g}_{model}$ indicates a greylevel template face patch; the error function can be, for example, an euclidean distance. In a tracking setting, the state is supposed to evolve slowly between consecutive time steps. The solution is thus searched as a small displacement from the previous frame estimation: $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \widehat{\Delta\mathbf{x}_k}$. The optimal displacement is then typically obtained by a gradient-like descent method. The well-known Lucas-Kanade algorithm [4] is a particular case of such a formulation. Instead of being specified by a single face greylevel template $\mathbf{g}_{model}$, the face model can span a subspace of greylevel patches, learnt by principal component analysis [5] from a face training set. The error functional is then a distance from the image patch $\mathbf{g}(\mathbf{x}_k)$ to the face subspace, usually taken to be the distance to the projection in face subspace. The subspace modeling allows to account for some variability of the global face appearance. Using principal component analysis, the Active Appearance Models encode the variations of face appearance by learning the shape and texture variations (see section 2.2). Thus, they enable the tracking of both global motion and inner features. The Active

Appearance Algorithm, being essentially a deterministic search, gives a fast and effective way to match the model to the current image. However it seems to work well only when lighting conditions remain stable and only small occlusions are present.

Differently, stochastic tracking approaches often reduce to an estimation problem. The hidden state and the observations are linked by a joint distribution. A Markovian dynamic model describes how the state evolves through time. An observation model specifies the likelihood of each hypothesised state, i.e. the probability that the considered state may generate the observed data. Stochastic tracking improves robustness over its deterministic counterpart, by its capability for escaping local minimum, since the search directions are for the most part random even though they are governed by a deterministic state transition model. Bayesian filtering methods recursively evaluate the posterior density of the target state at each time step, conditionally to the history of observations until the current time. Stochastic implementations of Bayesian filtering are generally based on sequential Monte Carlo estimation, also known as particle filtering (see chapter 3). When compared with the analytical solution provided by the well-known Kalman filter, particle filtering has two advantages: it is not restricted to the case of linear and Gaussian models, and it can maintain multiple hypotheses of the current state, a desirable property when the background is complex and contains distracting clutter.

For video tracking, the Condensation algorithm (see section 3.3) was first proposed in conjunction with edge measurements produced by an edge detector [6]. Since then, this algorithm has attracted much interest, and other kinds of measurements have given valuable variants. For instance, tracking based on color histograms has gained recent interest, due to the development of efficient search techniques [7]. In the case of particle filtering, motion is used as an additional information in order to remove ambiguities due to the color used alone. However, since color histograms are global, they do not allow to track the motion of internal facial features (which is one of the goals in this work).

Regarding existing works on face tracking with AAM, in most cases [8] [3][9] the tracking itself uses the AAM search frame-by-frame, with no temporal dynamics and without provision for occlusion handling.

In [10] S. Hamlaoui and F. Davoine combine AAM with temporal dynamics and use a filtering scheme based on Condensation, where the dynamics are guided by an AAM search. Although the stochastic approach permits to augment robustness, they rely too much on the deterministic search and the resulting algorithm performs poorly in case of heavy occlusions.

2.2 Active Appearance Models

This section provides a brief treatment of how an Active Appearance Model build its statistical models of shape and texture, and how these are combined into one unified model.

We first start by giving a definition of shape and texture [11]:

- **Shape** is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object. The term shape is, in other words, invariant to Euclidean transformations.
- **Texture** is the pixel intensities across the object in question (if necessary after a suitable normalization).

The construction of the model involves three major steps. The first step is the data acquisition. Then a suitable normalization follows, after which the data are ready to be analyzed and described in terms of statistical models. The process setup is given as a flow chart, in figure 2.1.

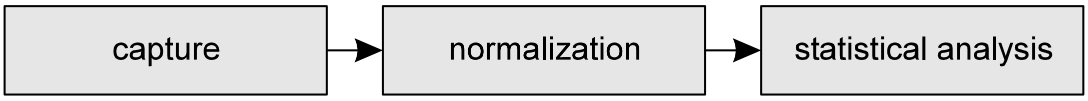


Figure 2.1: The three steps of handling shape and texture in AAMs.

To stress the coherence between shape and texture handling, the steps are specified below.

Capture

- **Shape** Captured by defining a finite number of points on the contour of the object in question.
- **Texture** Captured by sampling in a suitable image warping function (e.g. a piece-wise affine, thin-plate or another warp function).

Normalization

- **Shape** Brought into a normalized frame by aligning shapes with respect to position, scale and orientation using a Procrustes analysis.
- **Texture** Removing global linear illumination effects by standardization.

Statistical Analysis

- **Shape and Texture** Principal Component Analysis is performed to achieve a constrained and compact description.

2.2.1 Statistical Shape Model

One way to describe a shape is by locating a finite number of points, called *landmarks*, on the outline (see fig 2.2).

A mathematical representation of an n-point shape in k dimensions could concatenate each dimension into a kn-vector. In the following only 2D shapes are considered, hence k = 2. The vector representation for planar shapes would then be:

$$\mathbf{s} = [x_1, x_2, ..., x_n, y_1, y_2, ..., y_n]^T$$

Although the concept of landmarks conceptually is very useful, their acquisition can be very cumbersome and usually involves manually placing hundreds

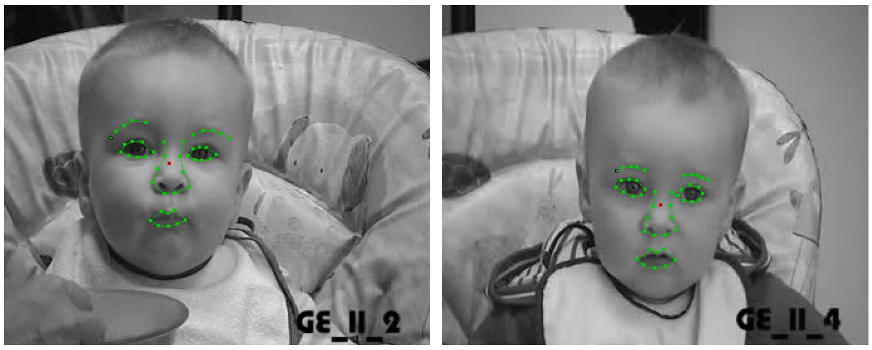


Figure 2.2: Two faces annotated using 44 landmarks.

of points, including constantly comparison to other annotations to ensure correspondence.

To obtain a true shape representation, according to our definition, location, scale and rotational effects need to be filtered out. This is carried out by establishing a coordinate reference, w.r.t. position, scale and rotation, commonly known as *pose*, to which all shapes are aligned.

An alignment procedure for obtaining such a coordinate reference is commonly known as Procrustes analysis [12], but it is not treated here.

Now we show how intra-class shape variation can be described consistently and efficiently. The fact alone that equivalence classes of shapes can be established, hint us in the direction that there must exist a sort of inter-point correlation. This is the only degree of freedom left to constitute the perception of a shape, since, according to the definition of shape, all position, scale and rotational effects are filtered out. A classical statistical method of dealing with such redundancy in multivariate data is the linear orthogonal transformation called *Principal Component Analysis* (PCA).

Conceptually, the PCA performs a variance maximizing rotation of the original variable space. Furthermore, it delivers the new axes ordered according to their variance. This is most easily understood graphically. In figure 2.3, the two principal axes of a two dimensional data set are plotted and scaled according to the amount of variation that each axis explains.

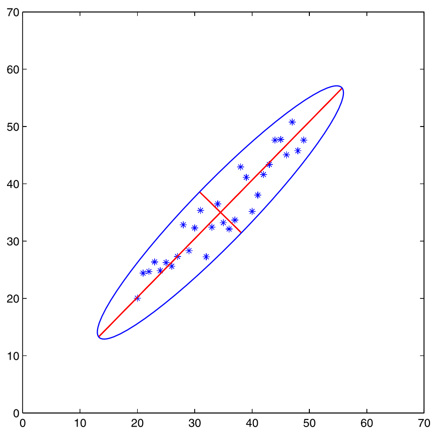


Figure 2.3: Principal axis. 2D example.

Hence, the PCA can be used as a dimensionality reduction method, by producing a projection of a set of multivariate samples into a subspace constrained to explain a certain amount of the variation in the original samples. One application of this is the visualization of multidimensional data. In connection to the example in figure 2.3, one could choose to discard the second principal axis, and visualize the samples by the orthogonal projection of the point upon the first (and largest) axis.

To describe shape variation, PCA is performed as an eigenanalysis of the covariance matrix of the aligned shapes. It is assumed that the set of shapes constitute some ellipsoid structure of which the centroid, that we call *mean shape*, can be estimated:

$$\bar{\mathbf{s}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{s}_i \tag{2.1}$$

The maximum likelihood estimate of the covariance matrix can thus be given as:

$$\Sigma_s = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T \tag{2.2}$$

The principal axes of the $2n^{th}$ dimensional shape ellipsoid are now given as the eigenvectors, Φ_s, of the covariance matrix:

$$\Sigma_s\Phi_s = \Phi_s\Lambda_s \tag{2.3}$$

where Λ_s denotes a diagonal matrix of eigenvalues $\Lambda_s = diag(\lambda_1, ...\lambda_{2n})$.

A shape instance can then be generated by deforming the mean shape by a linear combination of eigenvectors:

$$s = \bar{\mathbf{s}} + \Phi_s \mathbf{b}_s \tag{2.4}$$

where b_s are the shape model parameters. Essentially, the point or *nodal representation* of shape has now been transformed into a *modal representation*, where modes are ordered according to their *deformation energy*, i.e. the percentage of variation that they explain.

We need to determine how many modes to retain. This leads to a trade-off between the accuracy and the compactness of the model. However, it is safe to consider small-scale variations as noise. It can be shown that the variance along the axis corresponding to the i^{th} eigenvalue equals the eigenvalue itself, λ_i. Thus, to retain p percent of the variation in the training set, t modes can be chosen, satisfying the relationship:

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{2n} \lambda_i \tag{2.5}$$

In our face model, 98% of the shape variation can be modeled using 28 parameters. A rather substantial reduction since the shape space originally had a dimensionality of $2n = 2 \times 44 = 88$. To give an idea of the decay rate of the eigenvalues a percentage plot is shown in figure 2.4.

We conclude this section by remarking that the use of PCA as a statistical reparameterisation of the shape space, provides a compact and convenient way to deform a mean shape in a controlled manner, similar to what is observed in a set of training shapes. Hence, the shape variation has been modeled by obtaining a compact shape representation.

2.2.2 Statistical Texture Model

To form a complete model of appearance, one must not only consider shape. To stress this point, we observe that shape is well defined only by inferring from

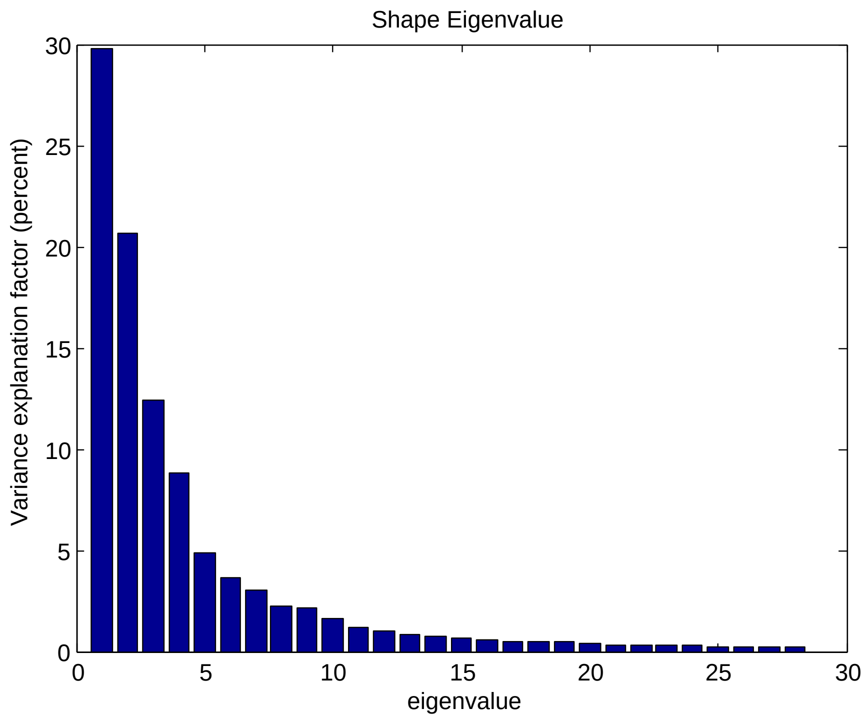


Figure 2.4: Shape eigenvalues in descending order.

knowledge of the pixel neighborhood.

In the shape case, the data acquisition is straightforward, because the landmarks in the shape vector constitute the data itself. In the texture case, one needs a consistent method for collecting the texture information between the landmarks, i.e. an image warping function needs to be established. This can be done in several ways. The most common choice is a piece-wise affine warp, based on the Delaunay triangulation of the mean shape. Thus to obtain texture information from the training set, each shape is warped to a reference shape (usually the mean shape) and sampled. Hereafter, a photometric normalization of the obtained textures is done to remove influence from global linear changes in pixel intensities. Then, the analysis is identical to that of the shapes. A compact PCA representation is derived to deform the texture in a manner similar to what is observed in the training set.

For further details on image warping and photometric normalization we refer to [11]. In the following we will only show the statistical analysis.

As a starting point for the texture variation modeling, the term “digital image” needs a discussion. The core of digital images is a set of spatial samples. In this thesis, raster images are considered. This merely means that the samples are arranged in a uniformly spaced spatial grid. The term spatial outlines that the ordering of the samples is crucial. This suggests some sort of spatial correlation between samples, leading to data redundancy as in the case of shapes. Hence, it is natural to adapt the PCA approach for the texture variation also.

The maximum-likelihood estimate of the mean texture of N normalized texture vectors is given as:

$$\mathbf{g}_s = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i \tag{2.6}$$

The maximum-likelihood estimate of the covariance matrix can then be written as:

$$\Sigma_g = \frac{1}{N}\sum_{i=1}^{N}(g_i - \bar{g})(g_i - \bar{g})^T. \tag{2.7}$$

The principal axes of the m-dimensional point cloud of textures are now given as the eigenvectors, Φ_g, of the covariance matrix:

$$\Sigma_g\Phi_g = \Phi_g\Lambda_g \tag{2.8}$$

where Λ_g is a diagonal matrix of eigenvalues. A texture instance can then be generated by deforming the mean texture by a linear combination of eigenvectors:

$$\mathbf{g} = \bar{\mathbf{g}} + \Phi_g\mathbf{b}_g, \tag{2.9}$$

where $\mathbf{b}_g$ are the texture model parameters.

The effect of varying the first three texture modes can be seen in figure 2.5.

2.2.3 Combined Model Formulation

We show now how to unify the presented shape and texture models into one complete, compact appearance model.

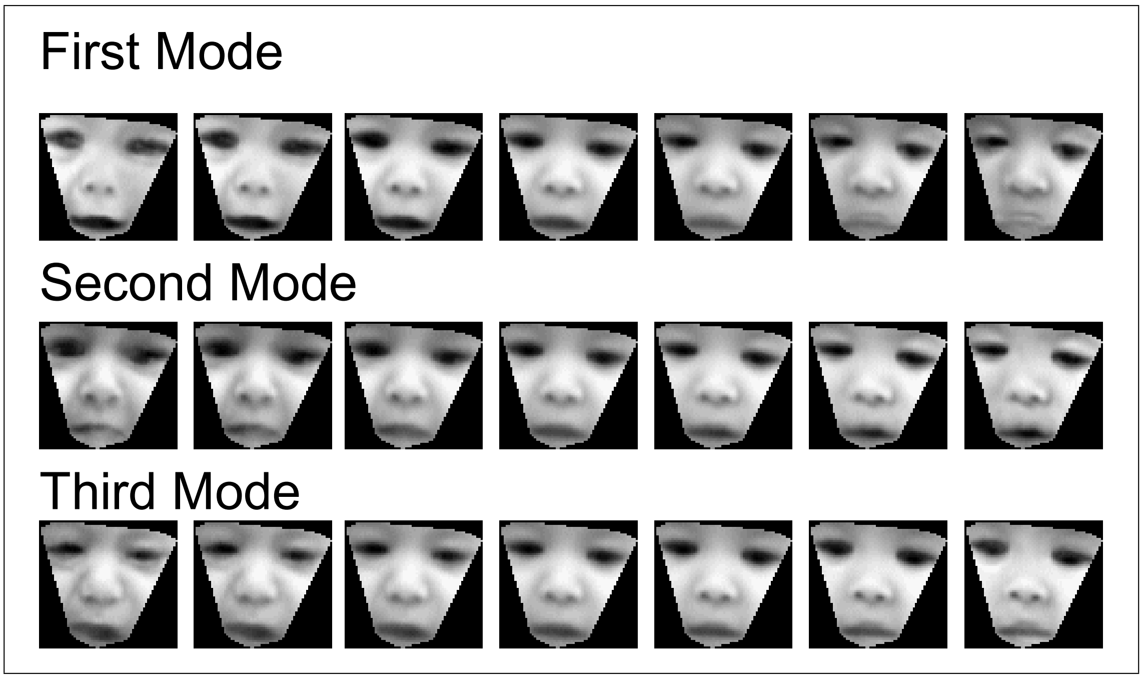


Figure 2.5: Mean texture deformation using 1st, 2nd and 3rd principal mode.

It was shown that an object instance can be constructed using the two set of model parameters of shape, $\mathbf{b}_s$, and texture, $\mathbf{b}_g$.

To remove correlation between shape and texture model parameters, and to make the model representation more compact, a 3rd PCA is performed on the concatenated shape and texture parameters, $\mathbf{b}$, of the training set, to obtain the combined model parameters, $\mathbf{c}$:

$$\mathbf{b} = \Phi_c \mathbf{c} \tag{2.10}$$

where Φ_c denotes a set of eigenvectors. The concatenated shape and texture parameters are easily obtained due to the linear nature of the model:

$$b = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \Phi_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \Phi_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \tag{2.11}$$

where W_s is a diagonal matrix providing a suitable weighting between pixel distances and pixel intensities.

Now, a complete model instance including shape $\mathbf{s}$ and texture $\mathbf{g}$, can be generated using the model parameters **c**.

$$\mathbf{s} = \bar{\mathbf{s}} + \Phi_s \mathbf{W}_s^{-1} \Phi_{c,s} \mathbf{c} \tag{2.12}$$

$$\mathbf{g} = \bar{\mathbf{g}} + \Phi_g \Phi_{c,g} \mathbf{c} \tag{2.13}$$

where

$$\Phi_c = \begin{pmatrix} \Phi_{c,s} \\ \Phi_{c,g} \end{pmatrix} \tag{2.14}$$

Regarding the compression of the model parameters, one should notice that the rank of Φ_c will never exceed the number of examples in the training set.

As in the shape PCA case, we can compress the combined model representation further, by removing the smallest eigenmodes. Again, it is safe to consider small-scale variation as noise. Thus, to retain p percent of the combined variation in the training set, t modes can be chosen satisfying:

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{N_c} \lambda_i \tag{2.15}$$

where N_c s the number of original modes.

2.2.4 AAM search

One of the attractive properties of the statistical models of shape and texture, is that it is possible to use these ones to search images for new instances of the class of objects that they represent, by means of an optimized algorithm called *AAM search*. Actually the combined model together with the AAM Search is what is called the Active Appearance Models.

The foundation of the AAM search is to treat the search as an optimization problem, in which the difference between the synthesized object delivered by the AAM and an actual image, is to be minimized. Formally this can be written as the difference vector $\delta\mathbf{I}$:

$$\delta\mathbf{I} = \mathbf{I}_{image} - \mathbf{I}_{model} \tag{2.16}$$

In this way, the fitting can be enhanced by adjusting the model and pose parameters to fit the image in the best possible way. Since the following discussion will be based on normalized texture vectors, $\delta\mathbf{I}$ will be denoted as $\delta\mathbf{g}$.

Though we have seen that the parameterization of the object class to be analyzed (a face in our case) can be dramatically compacted by the principal component analysis, the resulting system is still high-dimensional, and thus optimization is far from being an easy task. This is not only computationally cumbersome but also theoretically challenging since we are by no means guaranteed that the hyperspace sought in is smooth.

Methods for solving the optimization problem of deformable models, include general optimization techniques such as gradient based methods, like steepest descent.

AAMs circumvent these potential problems in a rather untraditional fashion.

It is proposed that the spatial pattern in $\delta\mathbf{g}$ can predict the needed adjustments in the model and pose parameters to minimize $\delta\mathbf{g}$. The simplest model one can use constitutes a linear relationship:

$$\delta\mathbf{c} = \mathbf{R}\delta\mathbf{g} \tag{2.17}$$

Cootes et al. show that this crude approximation suffices to produce good results in their work with AAMs [13][1][8][14][12].

To determine a suitable $\mathbf{R}$ in equation (2.17), a set of experiments are conducted, each experiment consisting of displacing both pose parameters $\mathbf{p}$ and model parameters $\mathbf{c}$ by a known amount, and measuring the difference between the model and the part of the image below the model. All results are eventually fed into a multivariate linear regression framework, to learn the linear relationship [11].

Given a method for predicting the correction which needs to be made in the model parameters, we can construct the AAM search as an iterative method for solving the optimization problem.

Given the current estimate of model parameters $\mathbf{c}_0$, and the normalized image sample at the current estimate, $\mathbf{g}_i$, one step of the iterative procedure is as follows:

- Evaluate the error vector $\delta\mathbf{g}_0 = \mathbf{g}_i - \mathbf{g}_m$.
- Evaluate the error $E_0 = |\delta\mathbf{g}_0|^2$.
- Compute the predicted displacement, $\delta\mathbf{c} = \mathbf{R}\delta\mathbf{g}_0$.
- Set $k = 1$.
- Let $\mathbf{c}_1 = \mathbf{c}_0 - k\delta\mathbf{c}$
- Sample the image at this new prediction, and calculate a new error vector, $\delta\mathbf{g}_1$.
- If $|\delta\mathbf{g}_1|^2 < E_0$ then accept the new estimate, $\mathbf{c}_1$, otherwise try for different value of k.

This procedure is repeated until no furhter reduction is observed in the error, $|\delta\mathbf{g}|^2$, and convergence is declared.

An example of an AAM optimization is given in figure 2.6. In this example, the model converged after 12 iterations.

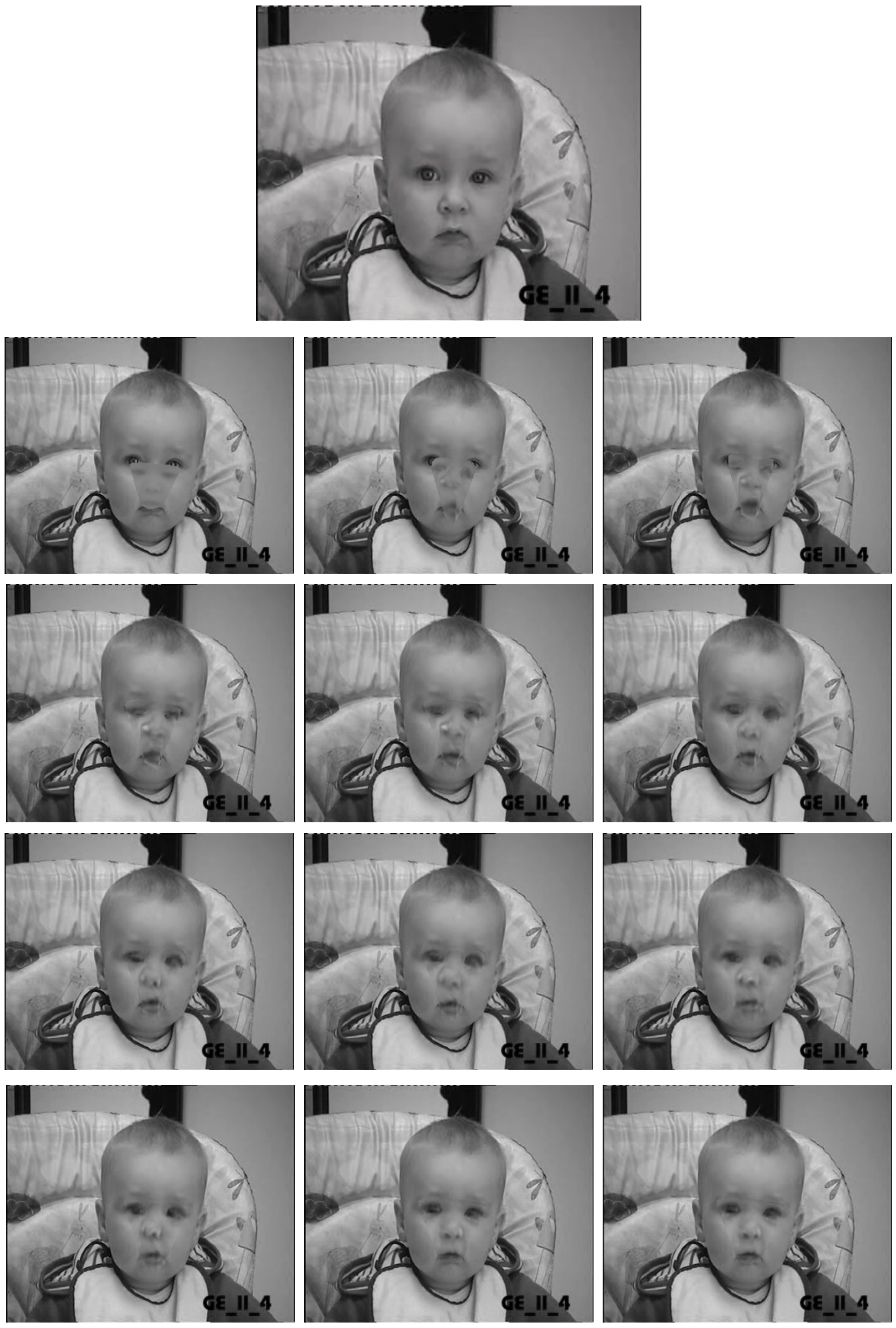


Figure 2.6: An AAM search example. On the top the original image. Below, from the upper left to the lower right, the optimization sequence (the algorithm converges after 12 iterations).

Chapter 3

Bayesian Filtering

Many problems in science require estimation of the state of a system that changes over time, using a sequence of noisy measurements made on the system. In order to analyze and make inference about a dynamic system we require to define and describe two models:

- A **transition model** describing the evolution of the state $\{\mathbf{x}_k, k \in \mathbb{N}\}$ with time.
- An **observation model** relating the noisy measurements $\{\mathbf{y}_k, k \in \mathbb{N}\}$ to the state.

Within a Bayesian framework, all relevant information about $\{\mathbf{x}_0, \mathbf{x}_1, .., \mathbf{x}_k\}$ given observations up to and including time k, can be obtained from the posterior distribution $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$. In many applications we are interested in estimating recursively in time this distribution, and particularly one of its marginals, the so-called filtering distribution $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. This problem is known as the Bayesian filtering problem or the optimal filtering problem [15]. Except for few special cases, including linear Gaussian state space models (Kalman filter) and hidden finite-state space Markov chains, it is impossible to evaluate these distributions analytically.

A recursive filtering approach means that received data can be processed sequentially rather than as a batch, so that it is neither necessary to store the

complete data set nor to reprocess existing data if a new measurement becomes available. Such a filter consists of essentially two stages: **prediction** and **update**. The prediction stage uses the transition model to predict the state distribution forward from one measurement time to the next. Since the state is usually subject to unknown disturbances (modeled as random noise), the prediction generally translates, deforms, and spreads the state distribution. The update operation uses the latest measurement to modify the prediction distribution. This is achieved using Bayes theorem, which is the mechanism for updating knowledge about the target state in the light of extra information from new data.
In the following we first give a description of the nonlinear tracking problem and its optimal solution. When certain constraints hold, this optimal solution is tractable.

More often the solution is intractable so we describe particle filters, which constitute an approximation strategy to the optimal solution.

3.1 Nonlinear Bayesian tracking

To define the problem of tracking, consider the evolution of the state sequence $\{\mathbf{x}_k, k \in \mathbb{N}\}$ of a target given by

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}), \tag{3.1}$$

where $\mathbf{f}(.)$ is a possibly nonlinear function of the state $\mathbf{x}_{k-1}$ and $\{\mathbf{v}_k, k \in \mathbb{N}\}$ is an i.i.d. process noise sequence. The objective of tracking is to recursively estimate the posterior distribution from measurements:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k), \tag{3.2}$$

where $\mathbf{h}(.)$ is a possibly nonlinear function, $\{\mathbf{v}_k, k \in \mathbb{N}\}$ is an i.i.d. measurement noise sequence. In particular, we seek filtered estimates of the posterior based on the set of all available measurements $\mathbf{y}_{1:k} = \{\mathbf{y}_i, i = 1, ..., k\}$, up to time k. From a Bayesian perspective, the tracking problem is that of recursively calculating some degree of belief in the state $\mathbf{x}_k$ at time k, given the data $\mathbf{y}_{1:k}$ up to time k. Thus, it is required to determine the probability density function (pdf) $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. It is assumed that the initial pdf $p(\mathbf{x}_0|\mathbf{y}_0) \equiv p(\mathbf{x}_0)$ of the state vector, which is also known as the prior, is available. Then, in principle, the pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ may be obtained recursively in two stages: prediction and update. Suppose that the required pdf $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ at time $k-1$ is available. The prediction stage involves using the transition model (3.1) to obtain the prior pdf of the state at time k via the Chapman-Kolmogorov equation [16]:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1}. \tag{3.3}$$

The reader should note that in the previous equation the first order marcovian hypothesis:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$$

has been used.

At time step k, a measurement $\mathbf{y}_k$ becomes available, and this may be used to update the prior (update stage) via Bayes rule

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \tag{3.4}$$

where the normalizing constant

$$p(\mathbf{y}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \tag{3.5}$$

depends on the likelihood function $p(\mathbf{y}_k|\mathbf{x}_k)$ defined by the measurement model (3.2) and the known statistics of $\mathbf{n}_k$. In the update stage, the measurement is used to modify the prior density to obtain the required posterior density of the current state. The recurrence relations (3.3) and (3.4) form the basis for the optimal Bayesian solution. This recursive propagation of the posterior density is only a conceptual solution in that, generally, it cannot be analytically determined. However, solutions do exist for a limited number of cases, including the Kalman filter.

3.1.1 Optimal Algorithms - Kalman Filter

The Kalman filter assumes that the posterior density at every time step is Gaussian distributed and, hence, represented by a mean and covariance. If $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ is Gaussian, it can be shown that $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ is also Gaussian, provided that some assumptions hold:

- $\mathbf{v}_{k-1}$ and $\mathbf{n}_k$ are drawn from Gaussian distributions of known parameters.
- $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1})$ is known, and is a linear function of $\mathbf{x}_{k-1}$ and $\mathbf{v}_{k-1}$.
- $\mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$ is a known linear function of $\mathbf{x}_k$ and $\mathbf{n}_k$.

That is, system equations can be rewritten as:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{3.6a}$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{n}_k \tag{3.6b}$$

where $\mathbf{F}$ and $\mathbf{H}$ are the matrices that define the linear functions.

Thus, an optimal analytical solution is provided to the tracking problem, if the (highly restrictive) assumptions hold. The implication is that no algorithm can ever perform better than a Kalman filter in this linear Gaussian environment.

3.1.2 Suboptimal algorithms - Extended Kalman Filter (EKF)

If (3.1) and (3.2) cannot be rewritten in the form of (3.6), because of the nonlinear functions, then a local linearization of the equations may be a sufficient description of the nonlinearity. The EKF is based on this approximation. The EKF utilizes the first term in a Taylor series expansion of the nonlinear function. A higher order EKF that retains further terms in the Taylor expansion exists, but the additional complexity has prohibited its widespread use. Recently, the unscented transform [17] has been used in an EKF framework [18]. The resulting filter, which is known as the unscented Kalman filter, considers a set of points that are deterministically selected from the Gaussian approximation to $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. All these points are propagated through the true nonlinearity, and the parameters of the Gaussian approximation are then re-estimated. For some problems, this filter has been shown to give better performance than a standard EKF, since it better approximates the nonlinearity; the parameters of the Gaussian approximation are improved. However, the EKF always approximates $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ to be Gaussian. If the true density is non Gaussian (e.g., if it is bimodal or heavily skewed), then a Gaussian curve can never describe it well. In such cases, particle filters will yield a performance improvement when compared to that of an EKF, as we will see in the next paragraph.

3.2 Particle Filtering methods

3.2.1 Sequential Importance Sampling (SIS)

The Sequential Importance Sampling (SIS) particle filter algorithm is a Monte Carlo (MC) method that forms the basis for most sequential MC filters developed over the past decades [15]. This sequential MC (SMC) is a technique for implementing a recursive Bayesian filter by mean of MC simulations. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the posterior pdf, and the SIS filter approaches the optimal Bayesian estimate. In order to develop the details of the algorithm, let $\{\mathbf{x}^i_{0:k}, w^i_k\}$ denote a random measure that characterizes the posterior pdf $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$, where $\{\mathbf{x}^i_{0:k}, i = 1, ..., N_s\}$ is a set of support points with associated weights $\{w^i_k, i = 1, ..., N_s\}$, and $\mathbf{x}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\}$, is the set of all states up to time k. The weights are normalized such that $\sum_{i=1}^{N_s} w^i_k = 1$. Then, the posterior density at time k can be approximated as:

$$p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i). \tag{3.7}$$

Therefore, we have a discrete weighted approximation to the true posterior, $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$.

The weights are chosen using the principle of **Importance Sampling** [15]. This principle relies on the following assumptions. Suppose that $p(\mathbf{x}) \propto \pi(\mathbf{x})$ is a probability density from which it is difficult to draw samples, but for which $\pi(\mathbf{x})$ can be evaluated (as well as $p(\mathbf{x})$ up to proportionality). In addition, let $\mathbf{x}^i$ be samples that are easily generated from a proposal $q(.)$ called an *importance density.* Then, a weighted approximation to the density $p(.)$ is given by

$$p(\mathbf{x}) \approx \sum_{i=1}^{N_s} w^i \delta(\mathbf{x} - \mathbf{x}^i) \tag{3.8}$$

where

$$w^i \propto \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{3.9}$$

is the normalized weight of the i^{th} particle.

Therefore, if the samples $\mathbf{x}_{0:k}^i$ were drawn from an importance density $q(\mathbf{x}_{0:k}|\mathbf{y}_{0:k})$, then the weights are defined to be:

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i|\mathbf{y}_{1:k})}{q(\mathbf{x}_{0:k}^i|\mathbf{y}_{0:k})}. \tag{3.10}$$

Returning to the sequential case, at each iteration, one could have samples constituting an approximation to $p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$ and it is required to approximate $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$ with a new set of samples. If the importance density is chosen to factorize such way that

$$q(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})q(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1}), \tag{3.11}$$

then one can obtain samples $\mathbf{x}_{0:k}^i$, drawing from $q(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$, by augmenting each of the existing samples $\mathbf{x}_{0:k-1}^i$ with the new sample $\mathbf{x}_k^i$. To derive the weight update equation, $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$ is first expressed in terms of $p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$, $p(\mathbf{y}_k|\mathbf{x}_k)$, and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$:

$$p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1}) \propto p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1}). \tag{3.12}$$

By substituting, the weight update equation becomes

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{y}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i, \mathbf{y}_{1:k})}. \tag{3.13}$$

Furthermore, if $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k)$, then the importance density becomes only dependent on $\mathbf{x}_{k-1}$ and $\mathbf{y}_k$. This is useful in the common case when only a filtered estimate of the posterior distribution is required at each time step. In such scenarios, only $\mathbf{x}_k^i$ need to be stored; therefore, one can discard the path $\mathbf{x}_{0:k}^i$ and the history of observations. The modified weight becomes then

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{y}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{y}_k)}, \tag{3.14}$$

and the posterior filtered density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ can be approximated as:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{3.15}$$

where the weights are defined in (3.14). It can be shown that as $N_s \longrightarrow \infty$ the approximation approaches the true posterior density, $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ [19]. The SIS algorithm thus consists of recursive propagation of the weights and support points, as each measurement is sequentially received.

Degeneracy Problem

A common problem with the SIS particle filter is the degeneracy phenomenon, where after a few iterations, all but one particle have negligible weight. It has been shown that the variance of the importance weights can only increase over time and, thus, it is impossible to avoid the degeneracy phenomenon. This degeneracy implies that a large computational effort is devoted to update particles whose contribution to the approximation is almost zero. A suitable measure of degeneracy of the algorithm is the effective sample size N_{eff} defined as

$$N_{eff} = \frac{N_s}{1 + Var(w_k^{*i})} \tag{3.16}$$

where $w_k^{*i} = p(\mathbf{x}_k^i|\mathbf{y}_{1:k})/q(\mathbf{x}_k^i|\mathbf{x}_{1:k-1}^i, \mathbf{y}_k)$, is referred to as the "true weight". This cannot be evaluated exactly, but an estimate $\hat{N_{eff}}$ of N_{eff} can be obtained by

$$\hat{N_{eff}} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \tag{3.17}$$

where w_k^i is the normalized weight obtained using (3.14). Notice that $N_{eff} \leq N_s$, and a small N_{eff} indicates severe degeneracy. Clearly, the degeneracy problem is an undesirable effect in particle filters. The brute force approach to reducing its effect, is to use a very large value for N_s. Since this is often impractical, usually one relies on a technique called *Resampling.*

3.2.2 Resampling

The basic idea of resampling is to eliminate particles that have small weights and to concentrate on particles with large weights. The resampling step involves generating a new set $\{x_k^{i*}\}_{i=1}^{N_s}$ by resampling (with replacement) N_s times from an approximate discrete representation of $p(x_k|y_{1:k})$ given by

$$p(x_k|y_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i).$$

The resulting sample is in fact an i.i.d. sample from the discrete density above; therefore, the weights are now reset to $w_k^i = 1/N_s$. Although the resampling step reduces the effects of the degeneracy problem, it introduces other practical problems. First, it limits the opportunity to parallelize since all the particles must be combined. Second, the particles that have high weights are, statistically, selected many times. This leads to a loss of diversity among the particles as the resultant sample will contain many repeated points. This problem, which is known as sample impoverishment, is severe in the case of small process noise. In fact, for the case of very small process noise, all particles will collapse to a single point within a few iterations.

The sequential importance sampling algorithm forms the basis for most particle filters that have been developed so far. The various versions of particle filters proposed in the literature can be regarded as special cases of this general SIS algorithm. These special cases can be derived from the SIS algorithm by an appropriate choice of importance sampling density and/or modification of the resampling step. Below, we present one of these approach, proposed in the literature, and mostly used in visual tracking.

3.2.3 Sampling Importance Resampling

The Sampling Importance Resampling (SIR) Filter is an MC method that can be applied to recursive Bayesian filtering problems. The assumptions required to use the SIR filter are very weak. The state dynamics $\mathbf{f}(.,.)$ and measurement functions $\mathbf{h}(.,.)$ need to be known, and it is required to be able to sample realizations from the process noise distribution of $\mathbf{v}_{k-1}$ and from the prior distribution $p(\mathbf{x}_k|\mathbf{y}_{1_k-1})$. Finally, the likelihood function needs to be available for pointwise evaluation (at least up to proportionality). The SIR algorithm can be easily derived from the SIS algorithm by an appropriate choice of

i) the importance density, where $q(\mathbf{x}_k|\mathbf{x}_{k-1} = \mathbf{x}^i_{k-1}, \mathbf{y}_k)$ is chosen to be the density $p(\mathbf{x}_k|\mathbf{x}_{k-1} = \mathbf{x}^i_{k-1})$,

ii) the resampling step, which is to be applied at every time index.

The above choice of importance density implies that we need samples from $p(\mathbf{x}_k|\mathbf{x}_{k-1} = \mathbf{x}^i_{k-1})$. A sample $\mathbf{x}^i_k$ from $p(\mathbf{x}_k|\mathbf{x}_{k-1} = \mathbf{x}^i_{k-1})$ can be created by first generating a process noise sample $\mathbf{v}^i_{k-1}$, and setting $\mathbf{x}^i_k = \mathbf{f}(\mathbf{x}^i_{k-1}, \mathbf{v}^i_{k-1})$. For this particular choice of importance density, it is evident that the weights are given by:

$$w_k^i \propto w_{k-1}^i p(\mathbf{y}_k|\mathbf{x}_k^i). \tag{3.18}$$

However, noting that resampling is applied at every time index, we have $w_{k-1}^i = 1/N \ \forall\, i$; therefore

$$w_k^i \propto p(\mathbf{y}_k|\mathbf{x}_k^i). \tag{3.19}$$

As the importance sampling density for the SIR filter is independent of measurement $\mathbf{y}_k$, the state space is explored without any knowledge of the observations. Therefore, this filter can be inefficient and is sensitive to outliers[1]. Furthermore, as resampling is applied at every iteration, this can result in rapid loss of diversity in particles, since the resultant samples will contain many repeated points. This problem, which is known as *sample impoverishment*, is severe in the case of small process noise. In fact, for the case of very small process noise, all particles will collapse to a single point within a few iterations. However, the SIR method does have the advantage that the importance weights are easily evaluated, and that the importance density can be easily sampled.

3.3 Condensation

Sequential Monte Carlo algorithms have gained prevalence in the visual tracking literature due in part to the **Condensation** (**Con**ditional **Den**sity propag**ation**) algorithm [20], which belongs to the class of SIR filters.

Spatio-temporal estimation has been dealt with thoroughly by Kalman filtering in the relatively clutter-free case, in which $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ can be modeled as Gaussian. These solutions work poorly in clutter that causes the posterior density to be multi-modal and, therefore, non-Gaussian. The Kalman filter as a recursive linear estimator is a special case, applying only to Gaussian densities, of a more general probability density propagation process. In the simple Gaussian case, the density function evolves as a Gaussian pulse that translates, spreads and is

[1]In statistics, an outlier is a single observation "far away" from the rest of the data.

reinforced, remaining throughout Gaussian, as in figure 3.1a.

The random component of the dynamical model leads to spreading, increasing uncertainty, while the deterministic component causes the density function to drift bodily. The effect of an external observation $\mathbf{y}_k$ is to superimpose a reactive effect and, consequently, the density tends to peak in the vicinity of observations. In clutter there are typically several competing observations, and these tend to encourage a non-Gaussian state density (fig 3.1b).

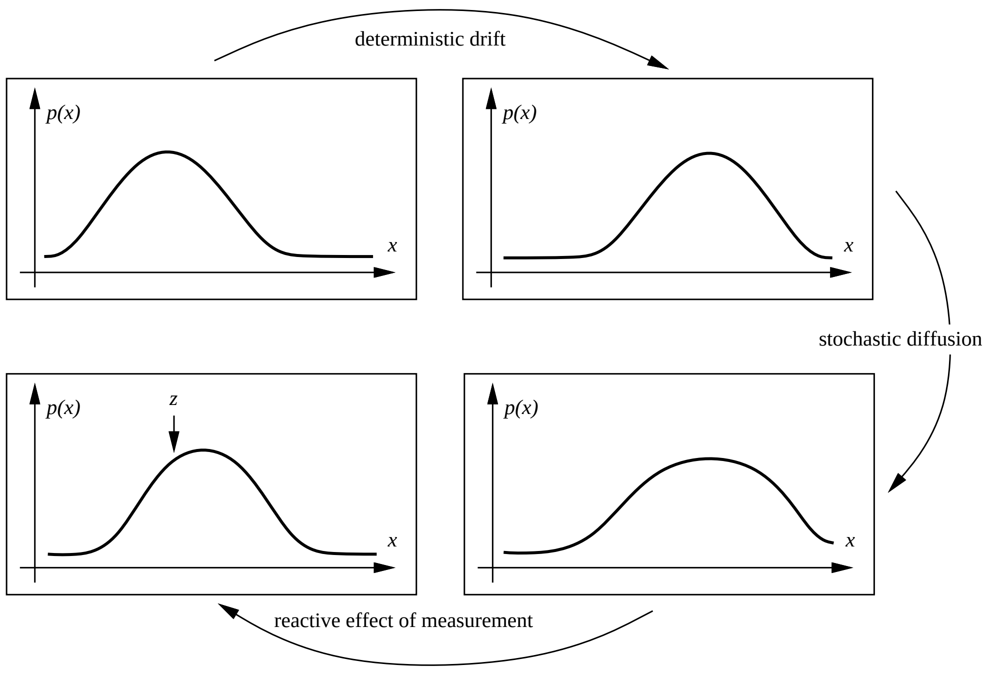
In Condensation the output of an iteration will be a weighted, time-stamped sample-set, denoted $\{\mathbf{s}_k^{(i)}, i = 1, .., N_s\}$, with weights w_k^i approximately representing the conditional state-density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ at time k. Clearly the process must begin with a prior density and the effective prior for time step k should be $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$. This prior is, of course, multi-modal in general and no functional representation of it is available. It is derived from the sample set representation $\{\mathbf{s}_{k-1}^{(i)}, \pi_{k-1}^{(i)}, i = 1, 2, .., N_s\}$ of $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ the output from the previous time step, to which the prediction

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}$$

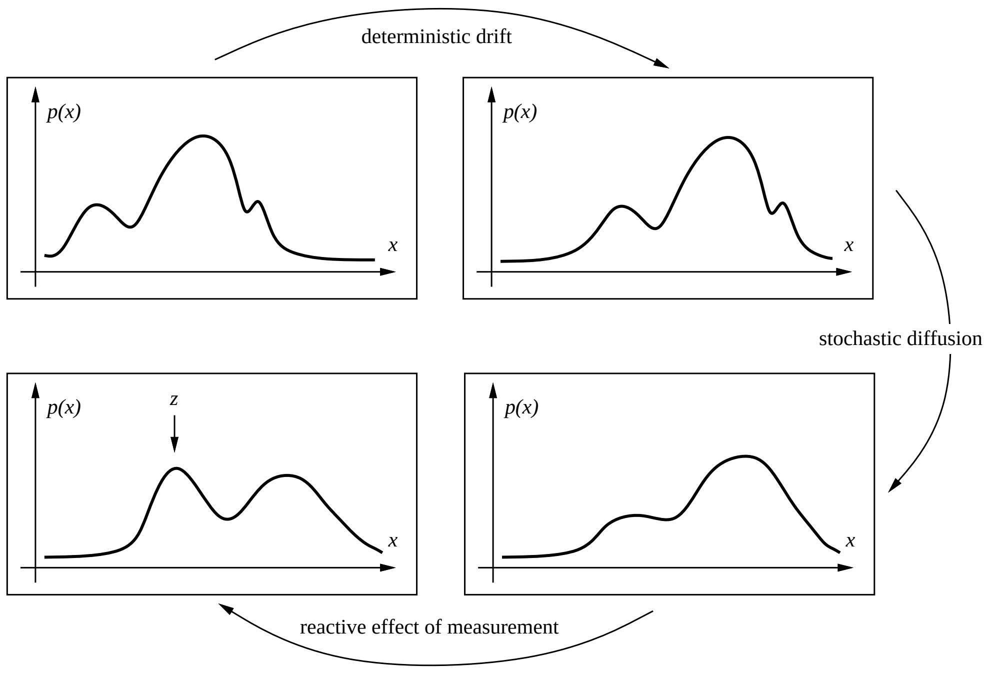
must then be applied. The iterative process, as applied to sample sets, is depicted in figure 3.2 and mirrors the continuous diffusion process in figure 3.1b. At the top of the diagram, the output from time step $k-1$ is the weighted sample set $\{\mathbf{s}_{k-1}^{(i)}, w_{k-1}^{(i)}, i = 1, .., N_s\}$.

The first operation, therefore, is to sample from the set, with replacement, N_S times, choosing a given element with probability w_{k-1}^i.

Some elements, especially those with high weights, may be chosen several times, leading to identical copies of elements in the new set. Others with relatively low weights, may not be chosen at all. Each element chosen from the new set is now subjected to the predictive steps. The predictive step is random and identical elements now split because each one undergoes its own independent Brownian motion step. At this stage, the sample set $\{\mathbf{s}_i^k\}$ for the new time step has been generated, but without its weights. Finally, the observation step is



(a) Gaussian case



(b) Non-Gaussian case

Figure 3.1: Probability density propagation.

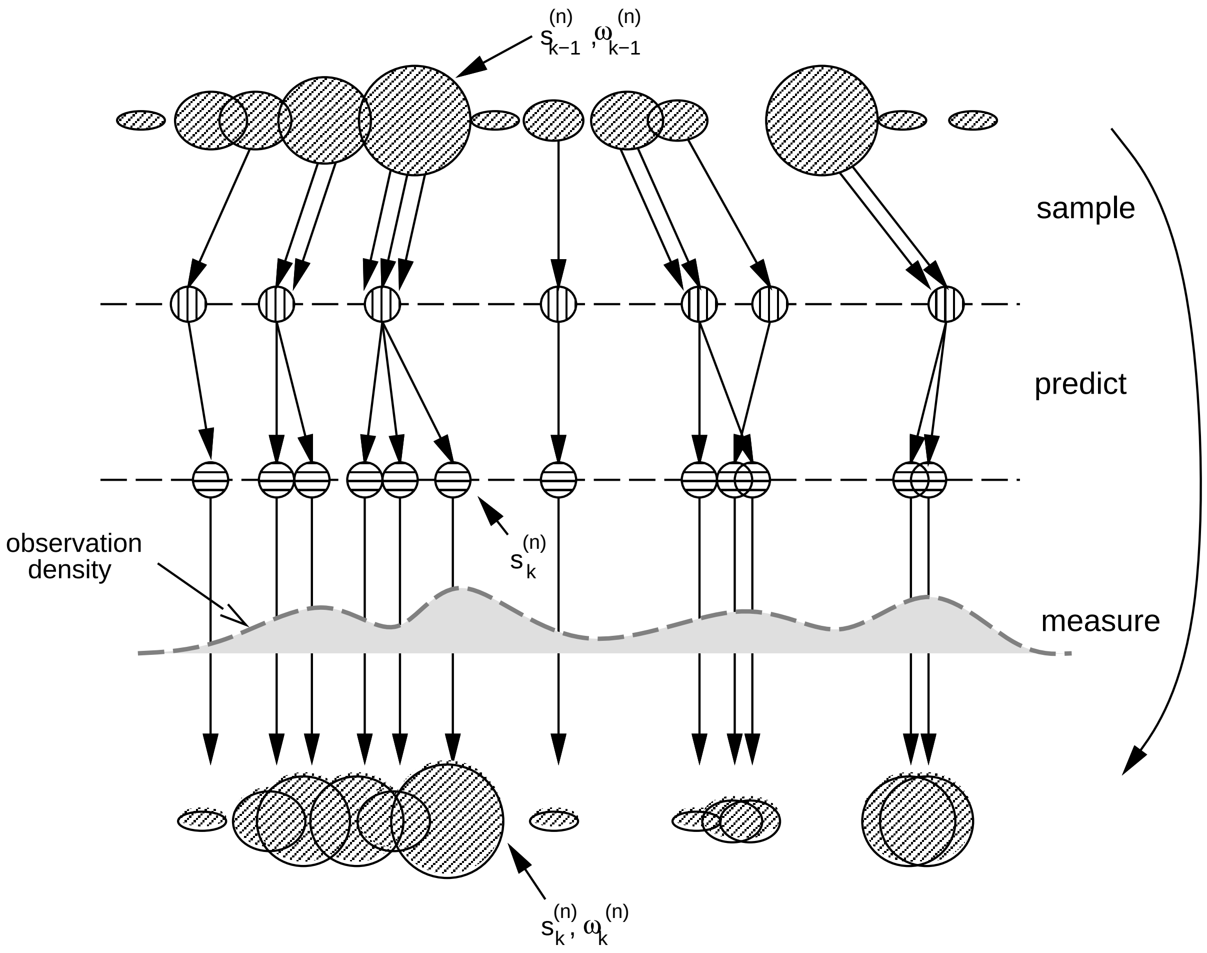

$s_{k-1}^{(n)}, \omega_{k-1}^{(n)}$
sample
predict
observation density
$s_k^{(n)}$
measure
$s_k^{(n)}, \omega_k^{(n)}$

Figure 3.2: Condensation Steps. One time-step in the Condensation algorithm.

applied, generating weights from the observation density to obtain the sample set representation of state density for time k.

Figure 3.3 gives a synopsis of the algorithm in its original formulation [6].

3.3.1 ICondensation

Condensation can be extended to incorporate the statistical technique of Importance Sampling. Importance sampling offers a mathematically principled way of directing search combining prediction information based on the previous object position and motion with any additional knowledge available. In the standard formulation of the Condensation algorithm (see fig 3.3) positions of samples $\mathbf{s}_k^{(n)}$ are fixed in the prediction stage using only the previous approximation of the state density $\{\mathbf{s}_k^{(n)}, w_k^{(n)}\}$ and the motion model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. The portions of state space which are to be examined in the measurement stage are therefore determined before any measurements are made. This is appropriate when the sample set approximation to the state density is accurate. In principle as the state density evolves over time the random nature of the motion model induces some non-zero probability everywhere in state space that the object is present at that point. With a sufficiently good sample set approximation this would tend to cause all areas of state space to lie near some samples so even motions which were extremely unlikely given the model would be detected and could therefore be tracked. In practice however the finite nature of the sample set approximation means that all of the samples will be concentrated near the most likely object positions. There may be several such clusters corresponding to multiple hypotheses but in general each cluster will be fairly localised, which is precisely the behaviour which permits an efficient discrete representation of high dimensional state spaces. The result is that large areas of state space contain no samples at all. In order to robustly track sudden movements the process noise of the motion model must be artificially high thus increasing the extent of each predicted cluster in state space. To populate these larger clusters with enough samples to permit effective

Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}'^{(n)}_t$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}'^{(n)}_t = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from

$$p(\mathbf{x}_t|\mathbf{x}_{t-1} = \mathbf{s}'^{(n)}_t)$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}'^{(n)}_t + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.
3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 3.3: Condensation Algorithm in its original formulation.

tracking the sample set size must be increased and the algorithm therefore runs more slowly. Various techniques have been proposed to improve the efficiency of the representation in random sampling filters [21]. Importance Sampling applies when auxiliary knowledge is available in the form of an importance function $q(\mathbf{x})$ describing which areas of state-space contain most information about the posterior. Importance sampling can be applied in the context of Condensation sampling and this extension is called Icondensation [22]. The idea is to concentrate samples in those areas of state space by generating sample positions $\mathbf{s}_k^{(n)}$ from an importance function $q(\mathbf{x}_k)$. The desired effect is to avoid as far as possible generating any samples which have low weights, since they are "wasted" as they provide a negligible contribution to the posterior. A correction term f/q must be added to the sample weights giving:

$$w_k^{(n)} = \frac{f(\mathbf{s}_k^{(n)})}{q(\mathbf{x}_k^{(n)})} p(\mathbf{y}_k|\mathbf{x}_k = \mathbf{s}_k^{(n)}) \quad \textit{where} \quad f(\mathbf{s}_k^{(n)}) = p(\mathbf{x}_k = \mathbf{s}_k^{(n)}|\mathbf{y}_{1_k-1}) \tag{3.20}$$

to compensate for the uneven distribution of sample positions. This correction term ensures that, for large N_S, importance sampling has no effect on the consistency of the approximation.The effect of the correction ratio is to preserve the information about motion coherence which is present in the dynamical model. Although the samples are positioned according to $q(\mathbf{x}_k)$ the distribution approximated by $\{\mathbf{s}_k^{(n)}, w_k^{(n)}\}$ still generates $p(\mathbf{x}_k|\mathbf{y}_k)$. Importance sampling is again intended to improve the efficiency of the sample set representation but does not change the probabilistic model.

Chapter 4

Tracking Baby Faces

The problem of tracking faces in dense visual clutter and in presence of heavy occlusions is a challenging task.

As mentioned in chapter 2, in our work we have decided to describe faces by means of the Active Appearance Models (AAMs). By construction the AAMs encode the variations of face appearance by learning shape and texture variations, enabling the tracking of both global motion and inner features. In practice, tracking using the deterministic AAM search, appears to work well while the lighting conditions remain stable and only small occlusions are present. However, large occlusions often make the AAM search converge to incorrect positions and loose track of the face.

It is common in literature to deal with these kind of tracking problems adopting a probabilistic formulation, since it permits to model motion uncertainty.

In this chapter we propose a solution for the tracking problem integrating AAMs in the Bayesian framework described in chapter 3.

Kalman filter approaches rely on the strong assumptions of linearity of the transition model, and on the gaussian nature of the posterior density (see section 3.1). The hypotheses are too restrictive and do not hold in presence of a complex and cluttered background. A stochastic implementation of the filter seems to be the right choice since it does not rely on the restrictive Kalman filter assumptions and it can maintain multiple hypotheses of the current state, thus improving the robustness of the tracker.

A number of different approaches, found in literature, for the tracking are presented and discussed before proposing our solution. The final algorithm, based on Condensation, integrates in a Bayesian mixed-state framework multiple motion models, allowing to cope with the limitations of previous approaches.

In chapter 5 all the algorithms will be compared in terms of performance and accuracy.

4.1 State Space Formulation

Using Active Appearance Models, we can represent the shape and texture of a face in terms of a vector $\mathbf{c}$ of parameters, obtained by projecting the image in a subspace learned by PCA in a initial training phase. To complete the description of the face, the four pose parameters are also needed, namely $\mathbf{p} = (\alpha, \vartheta, t_x, t_y)$, representing scale, orientation and position, respectively. The state vector $\mathbf{x}$, which contains the parameters used to infer about the object (the face) is thus composed by the concatenation of the vector $\mathbf{c}$ of combined parameters and vector $\mathbf{p}$ of pose. It should be noted that $\mathbf{x}$ is a high dimensional state vector (about 60-dimensional), and some considerations are necessary, therefore, in order to stochastically span such a huge state space.

For a Bayesian formulation of the tracking problem, accordingly on what we said in chapter 3, we need to specify a state transition model and an observation model.

The observed data $\mathbf{y}_k$ consist of measurements derived from the current video frame and, specifically, pixel graylevel patches. In order to evaluate a hypothesized state, actually the measurements are only considered in the image area corresponding to the hypothesized location. Basically, a given state $\mathbf{x}_k$ (motion parameters) is then evaluated by comparing the motion-compensated graylevel image patch $\mathbf{g}(\mathbf{x}_k)$ with a graylevel template face patch $\mathbf{g}_{model}$. In this context, a proper choice of the texture distance is a crucial point. Partial occlusion can be managed by choosing an appropriate error function, robust to outliers. This problem has already been tackled by [11]; here, some other considerations will be made.

4.1.1 Choosing a Metric Space

Both the AAM search and the observation model in the probabilistic framework are driven by a texture difference, which is used to compute the distance between the face model and the true face image. In order to guarantee a *robust texture distance* is necessary to improve the insensitivity to outliers.

To formalize the model fitting problem, a set of parameters, $\mathbf{c} = [c_1, .., c_p]^T$, are adjusted to fit a set of measurements (e.g. an image), $\mathbf{g} = [g_1, ..., g_m]^T$. This is done by minimizing the residuals [23]:

$$E = \sum_{i=1}^{m} \rho(g_i - u(i, \mathbf{c}), \sigma_s) = \sum_{i=1}^{m} \rho(e_i, \sigma_s) \tag{4.1}$$

where u is a function that returns the model reconstruction of the i^{th} measurement and σ_s is the scale parameter that determines what should be deemed outliers. The ρ-function determines the weighting of the residuals, and it is also called the *error norm*. Basic AAMs, as seen in section 2.2.4, use the quadratic norm (or simply the 2-norm) without any normalization:

$$E = \sum_{i=1}^{m} (g^i_{model} - g^i_{image})^2. \tag{4.2}$$

However, that the quadratic norm is notoriously sensitive to outliers, since these will highly contribute to the overall solution, due the rapid growth of the x^2 function. In the following work, two other norms will be considered that give experimentally better results.

Lorenzian Norm

It is a smooth norm which falls off quickly:

$$\rho(e_i, \sigma_s) = \log(1 + \frac{e_i^2}{2\sigma_s^2}) \tag{4.3}$$

The logarithmic function compresses the error dynamic range, improving robustness but at the same time reducing the accuracy.

Mahalanobis Distance

This similarity measure is closely related to the Mahalanobis distance. They both take into account the variance of the training set, resulting in similarity measure less sensitive to large residuals in areas of high variance (as observed in the training set):

$$\rho(e_i) = \frac{e_i^2}{2\sigma_i^2} \tag{4.4}$$

where σ_i is the maximum likelihood estimate of the variance of the i^{th} pixel. Notice however, that this is not the Mahalanobis distance since in that case σ_i should be the variance of the difference. This norm can be slightly modified to gain robustness [10]:

$$\rho(e_i) = \begin{cases} \frac{e_i^2}{2\sigma_i^2} & if\, |\frac{e_i}{\sigma_i}| \leq h \\ h|\frac{e_i}{\sigma_i}| - h^2 & if\, |\frac{e_i}{\sigma_i}| > h \end{cases} \tag{4.5}$$

where h is a fixed threshold, above which $|\frac{e_i}{\sigma_i}|$ is considered to be an outlier.

This error function behave well in most cases and is the one used during the experiments.
To prove the utility of the robust error norm, some experiments have been done. In figure 4.1 are shown the results of an AAM search on still images: in the upper part the euclidean norm is used while in the lower part the robust norm described above. The examples reported are typical cases where the classical

AAM norm does not work, while the robust norm performs well. This improvement is crucial for tracking purpose since all the approaches presented rely on a good initialization.

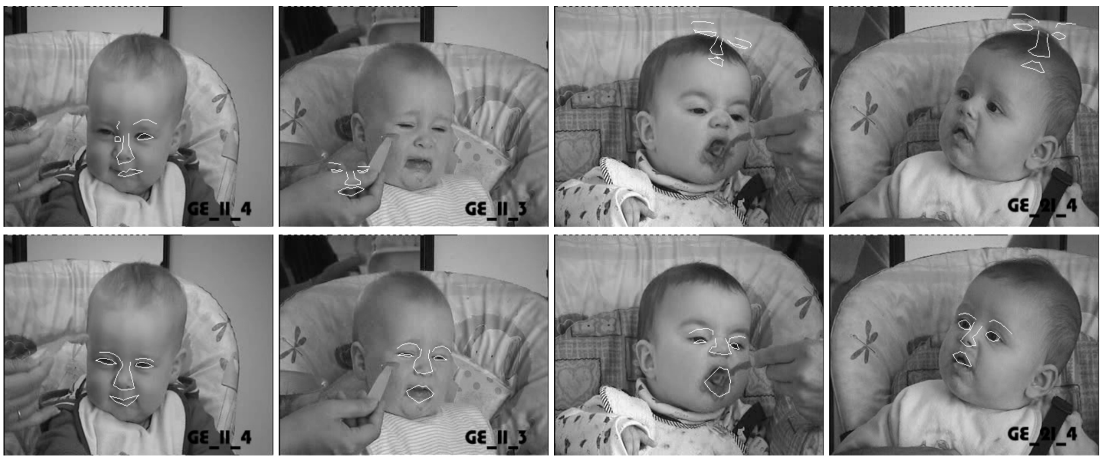


Figure 4.1: Comparison Between Euclidean and Robust Norm. Four still images are searched with AAM: a)in the upper part the euclidean norm is used b)in the lower part the robust norm is used.

4.1.2 Observation Model

The Observation Model is based on the difference between sampled pixel gray level patch of the current image and that generated by the AAM built model. The likelihood function $p(\mathbf{y}_k|\mathbf{x}_k)$ indicates the probability that a hypothesized state $\mathbf{x}_k = (\mathbf{c}_k, \mathbf{p}_k)$ gives rise to the observed data. Since the observed data consist of pixel graylevel values, it is straightforward to look for a function with the following form:

$$p(\mathbf{y}_k|\mathbf{x}_k) = p(\mathbf{y}_k|\mathbf{c}_k, \mathbf{p}_k) = C \exp -d[\mathbf{g}_{model}(\mathbf{c}_k), \mathbf{g}_{image}(\mathbf{c}_k, \mathbf{p}_k)] \qquad (4.6)$$

where $\mathbf{g}_{image}(\mathbf{c}_k, \mathbf{p}_k)$ is the image patch sampled at the hypothesized pose and shape, $\mathbf{g}_{model}(\mathbf{c}_k)$ is the model texture representing the hypothesized appearance of the face, and C is a normalizing constant. The texture distance $d[;]$ is an error measure, summed over all L pixels of both textures:

$$d[\mathbf{g}_{model}, \mathbf{g}_{image}] = \sum_{l=1}^{L} \rho(\frac{g^l_{model} - g^l_{image}}{\sigma_l}) \tag{4.7}$$

and can be chosen among those presented in the previous section.

4.1.3 State Transition Model

The state transition model characterizes the dynamics between frames. In a visual tracking problem, it is ideal to have an exact motion model governing the kinematics of the object. Motion models are used as predictors to increase the robustness and accuracy of visual trackers. As already seen in section 2.2.4, AAM search can predict motion accurately if some condition hold. To add robustness to the deterministic search, we add a noise contribution modeling uncertainty in prediction.

The state space equation can therefore be written in its general form as :

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{S}_k\mathbf{u}. \tag{4.8}$$

In this equation, $\mathbf{f}(\mathbf{x}_{k-1})$ represent a deterministic function of the previous state vector; the second term is an additive random contribution:

- $\mathbf{u}$ is a vector of independent normal random variates with zero mean and unit variance;
- $\mathbf{S}_k$ is a diagonal matrix

  $$\mathbf{S}_k = diag(\sigma_k^{c_1}, ..., \sigma_k^{c_m}, \sigma_k^{\alpha}, .., \sigma_k^{t_y})$$

  which specifies the standard deviation of the random draw for each appearance/pose parameter.

Whereas the choice of gaussian noise is expected when accurate model uncertainty cannot be provided, choosing an appropriate function $\mathbf{f}(.)$ is not an easy task, and depends on the particular situation. The role of the deterministic function is that to predict the pose and appearance parameters using data available from the previous time step. Even if deterministic AAM search should dominate the system dynamics, in some situation a fixed constant-velocity model with fixed noise variance can give better results: this is the case when total occlusions occur, for example.

Summarizing, there are two possible choices for the state equation that can be used:

1. a fixed constant-velocity model with fixed noise variance

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{S}_k\mathbf{u}; \tag{4.9}$$

2. an adaptive dynamical model, guided by a deterministic AAM search:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \partial\mathbf{x}_k + \mathbf{S}_k\mathbf{u} \tag{4.10}$$

   where $\partial\mathbf{x}_k$ is the predicted shift of the tracked parameters.

The dynamical model can then be expressed, in probabilistic form, as follows:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) \propto \exp -\frac{1}{2}\left\|\mathbf{S}_k^{-1}[\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1})]\right\|^2. \tag{4.11}$$

The (4.11) clearly shows the Markov nature of the dynamical model.

To complete the description of the model, we need to specify the initial distribution $p(\mathbf{x}_0|\mathbf{y}_0) = p(\mathbf{x}_0)$ ($\mathbf{y}_0$ being the set of no measurements). At time $k = 0$ we do not know about the object to be tracked. The problem of find a proper function thus corresponds to the problem of initializing the tracking.

A Hierarchical Search Algorithm as been proposed in [25] as an efficient solution for initialization with AAMs, which basically consists in an exhaustive search of the target in the image, obtained performing normal AAM searches sparsely over the image, using perturbations of the pose parameters. A solution to provide the initial distribution could be, therefore, to use the result of such AAM initialization to be the mean of a gaussian distribution with covariance $\mathbf{S}_0$. In this case, however, a wrong initialization would cause the tracking to fail since the resulting distribution only covers a limited portion of the state space. If the target is too far from the estimated initial position (the quantification of far being related to the value of covariance matrix $\mathbf{S}_0$) the tracker will hardly find it again.

Since we are in a Bayesian framework we can exploit the advantages of the stochastic approach to give robustness to the initialization. A substantial improvement is thus gained formulating Active Appearance Models in a Bayesian setting as in [11], specifying a prior and a likelihood probability distribution:

- **Prior**. $p(\mathbf{x}|\theta)$ where $\mathbf{x}$ denotes the model parameters $\mathbf{c}$ and the pose parameters $\mathbf{p}$, and θ denotes the mean shape, mean texture and their corresponding covariance structure. θ is obtained from the training set.
- **Likelihood**. $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{y}$ denotes the image being searched in, so the first frame in the video sequence. This likelihood is the same as in (4.6).

Regarding the prior distribution, since Active Appearance Models do not have an explicit prior model, it is chosen to be uniform. The limits of this distribution are implicit in the AAM since in the training phase, as said in section 2.2, shape and texture are constrained by PCA. The initial distribution thus is chosen as the prior distribution deriving from posing AAMs in the bayesian framework:

$$p(\mathbf{x}_0) = p(\mathbf{x}|\theta).$$

Adaptive Dynamics

According to the state transition model, pose/appearance parameters are drawn around the predicted state $\mathbf{f}(\mathbf{x}_{k-1})$ with dispersions (standard deviations) given by $\mathbf{S}_k$. Following the idea from Zhou et al [24], we consider adaptive dispersion given by:

$$(\sigma_k^{c_1}, ..., \sigma_k^{c_m}, \sigma_k^{\alpha}, .., \sigma_k^{t_y}) = R_k(\sigma_0^{c_1}, ..., \sigma_0^{c_m}, \sigma_0^{\alpha}, .., \sigma_0^{t_y}) \tag{4.12}$$

where $diag(\sigma_0^{c_1}, ..., \sigma_0^{t_y})$ are the fixed reference standard deviations, and the scaling factor R_k is proportional to the square root of ϵ_k, which is a measure of variance corresponding to a texture error averaged over the L pixels of the texture:

$$\epsilon_k = \frac{2}{L} \sum_{i=1}^{L} \rho(e_i, \sigma_s). \tag{4.13}$$

The value of R_k can then be bounded using a maximum and a minimum value to constrain it in the interval $[R_{min}, R_{max}]$:

$$R_k = \max(\min(R_{min}, \sqrt{\epsilon_k}), R_{max}). \tag{4.14}$$

The meaning of such a choice is easily explained: if the error augments, the variance increase proportionally in order to explore a larger area of the state space subregion, covered by the predicted distribution. This must be followed, however, by a corresponding increase in the number of particles used to explore it; thus, an adaptive number of particle is proposed as well:

$$N_k = N_0 R_k, \tag{4.15}$$

where N_0 is a fixed number of particles. The use of adaptive dispersion permits to improve temporal performance with respect to the fixed variance/number of particles, a case that implies an overestimate of these quantities.

In figure 4.2 is shown how the number of particles, N_k, varies in a video sequence: N_k remains approximatively constant in normal condition; when an external hand occludes the face (upper image) it increase proportionally to the error.

4.2 Condensation based tracking

Once the elements for bayesian filtering are given, an algorithm for sequential estimation of the filtered posterior density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ must be specified.

Condensation is a very general algorithm and has been successfully applied in many visual tracking problems. This is the reason for which we decided to use it as a basis for our solution.

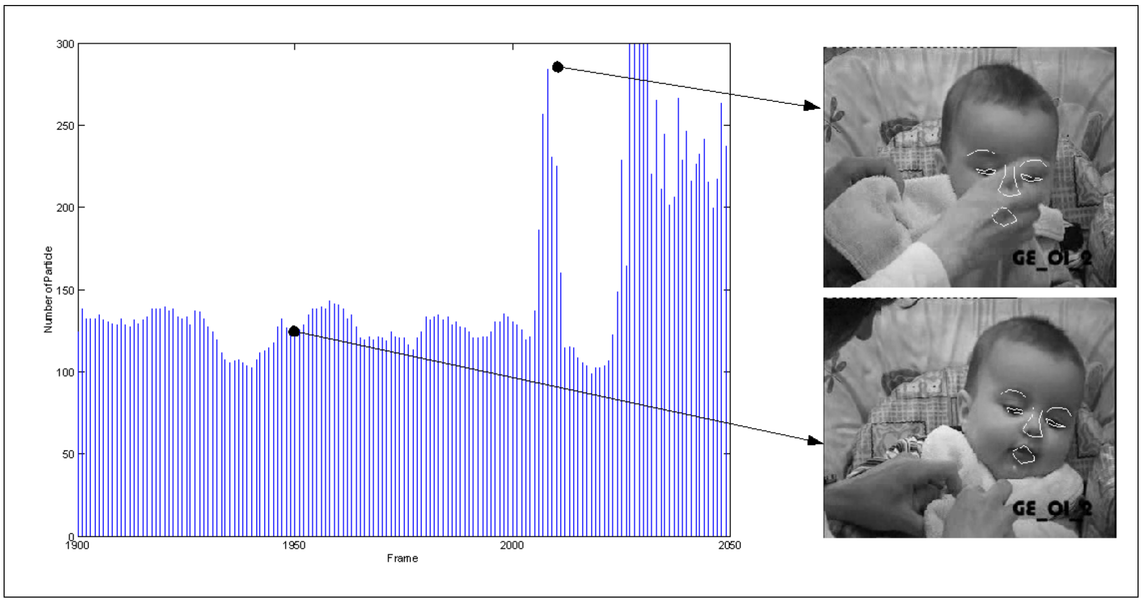


Figure 4.2: Adaptive Number of Particles.

The main aim of Condensation (and of particle filters in general) is to provide an accurate, discrete representation of the posterior density using the minimum possible number of particles, to keep computational costs low. Although asymptotic convergence of particle filters has been demonstrated [19], it is difficult to prove any general result for a finite number of particles. Likewise, it is difficult to make any precise, provable statement on the crucial question of how many particles are required to give a satisfactory representation of the densities for filter operation. What we can state is that the required number of particles depends on:

- the dimension of the state vector: in [26] is shown how “curse of dimensionality” affect in general particle filtering;
- the precision of the deterministic dynamics: when the deterministic prediction gives a solution far from the real system state, the random part of the motion model must be capable of correcting the error by extensively exploring the state space. This results in a larger samples set or , in other words, in an increased number of particles.

An AAM-Condensation approach, using either (4.9) or (4.10) as motion model, has three main problems:

1. AAMs describe faces with high-dimensional vectors;
2. the deterministic AAM search is highly sensitive to large occlusion so robustness is achieved only by increasing the noise variance;
3. accuracy is only accomplished at the cost of unacceptable time performance.

Common solutions found in literature, in order to solve those problems, trade off speed with robustness, following two kind of approaches:

1. A zero velocity model is used with a reduced state vector, obtained, for example, by retaining only the most representative components.
2. The search is carried out only in those regions of the image where the object is predicted to be. This can be achieved combining Importance Sampling and the Condensation algorithm (see section 3.3.1).

In figure 4.3 a general flow chart of the Condensation based tracker is presented. In every approach presented in the following the filtering scheme consists in propagating a Sample Set $\{\mathbf{s}_k^{(n)}, w_k^{(n)}\}$ through time. At each time step k, a new Sample Set is generated *resampling* the previous sample set and *propagating* each particle according to the dynamical model. Each particle is weighted then by the likelihood function and the state estimate at time k, $\hat{\mathbf{x}}_k$, is chosen as the Maximum a Posteriori (MAP) of the approximated filtered posterior distribution:

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{y}_{1:k}). \tag{4.16}$$

In the following we propose two tracking solutions: in the first we track only head pose with a classic Condensation scheme. In the second we use ICondensation (see section [2]) to improve time performance.

In the next section 4.3 we will show how to integrate them to improve overall performances.

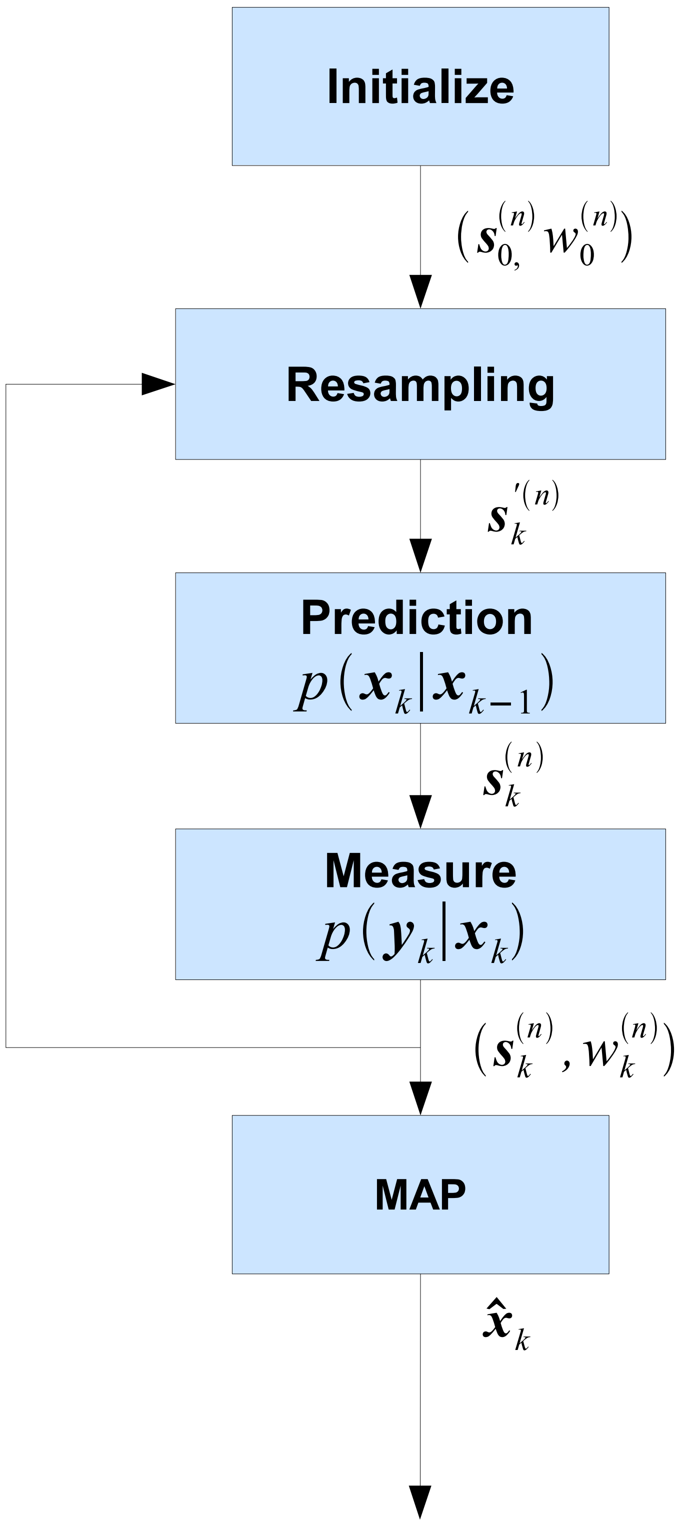


Figure 4.3: Block Diagram of the generic Condensation-based tracker.

4.2.1 Condensation Based Pose Tracking

Condensation can be used to track global motion of face or, in other words, pose parameters, as in [27].

When tracking the 4 pose parameters, (4.9) is a convenient form for the dynamical model. This low-level approach permits to overcome the aforementioned difficulties, but as counterpart the vector of combined parameters $\mathbf{c}$ is not tracked since we artificially pose it to zero: $\mathbf{c} = \mathbf{0}$.

The state transition model can be written as:

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{S}_k \mathbf{u} \tag{4.17}$$

basically a random walk[1] on the pose parameters $\mathbf{p} = (\sigma, \vartheta, t_x, t_y)$. The predicted state is simply the previous state, to which a random noise with fixed variance is added. If the variance is very small, it is difficult to model rapid movements; if the variance is large, it is computationally inefficient, since many more particles are needed to accommodate for such noise variance. At each time step a Sample Set is found as in 3.3 with likelihood function (4.6) rewritten as:

$$p(\mathbf{y}_k|\mathbf{p}_k) = p(\mathbf{y}_k|\mathbf{p}_k) = C \exp -d[\mathbf{g}_{model}(\mathbf{c}), \mathbf{g}_{image}(\mathbf{c}, \mathbf{p}_k)] \quad , where \quad \mathbf{c} = \mathbf{0} \tag{4.18}$$

This approach offers a great robustness, thanks to the ability of Condensation to recover from heavy occlusions. Obviously, the accuracy of the algorithm, intended as the ability to generate a photo realistic synthetic replica of the face, cannot be guaranteed since the appearance parameters ($\mathbf{c}$) are not tracked.

4.2.2 Directing Search Using Importance Sampling

To track the entire state vector $\mathbf{x} = (\mathbf{c}, \mathbf{p})$ and keep computational time low, we use the AAM search as deterministic state predictor and an importance function in the ICondensation framework to restrict the width of the search.

[1]In Mathematics a random walk is a series of sequential movements in which the direction and size of each move is randomly determined.

The dynamics are modelled as:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \partial\mathbf{x}_k + \mathbf{S}_k\mathbf{u}. \tag{4.19}$$

In the above equation, $\partial\mathbf{x}_k = (\partial\mathbf{c}_k, \partial\mathbf{p}_k)$ indicates the predicted shift in pose and appearance parameters, obtained by an automatic AAM search in the current frame.

Assuming a successful tracking at time step $k-1$, we know that the face at step k must lye in the "neighborhood" of the previous state estimate $\hat{\mathbf{x}}_{k-1}$. Thus the importance function is chosen to be:

$$q(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1} = \hat{\mathbf{x}}_{k-1}). \tag{4.20}$$

Each particle, constituting the sample set, is drawn from the importance function, and can be obtained by randomly perturbing a shifted version of the previous state estimate (the shift being determined by an AAM search). Then, each weight is calculated using (3.20). In figure 4.4 the flow chart has been modified to describe this approach.

It should be clear that the main feature of Condensation (maintaining multiple hypothesis) is lost: we just consider the best candidate of the previous sample set in the prediction stage, leaving out all the others.

The accuracy of this algorithm strictly depends, therefore, on the validity of the AAM search.

Exploration of a small region, that most likely contains good candidates, permits a better refining of the deterministic search, with respect to random sampling the state space thoroughly. On the other hand, in case of occlusions, an erroneous prediction will cause the tracking to fail. When we can rely on a good estimate at previous time step, the algorithm works well and permits a fast tracking. But if heavy or total occlusions occur the tracker will loose the position and hardly will recover it.

To summarize we expect the best results, both in terms of speed and accuracy, from this approach if only partial occlusions occur.

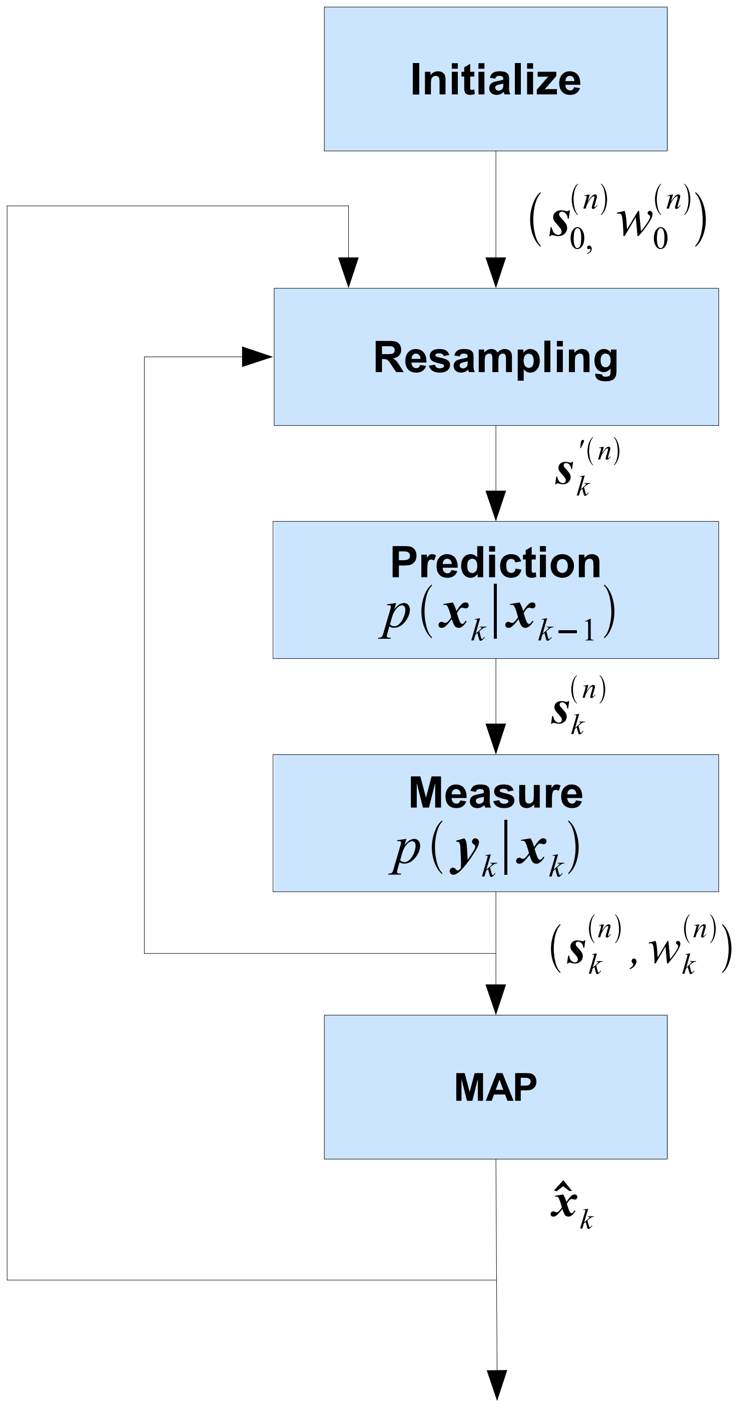


Figure 4.4: Block Diagram for the ICondensation-based tracker

4.3 A Mixed-State Condensation Approach

Two kind of approaches have been presented in the previous section, which seem to be compensatory in terms of accuracy and robustness.

In occlusion-free situations, the ICondensation with dynamics guided by AAM search behaves well, allowing fast and accurate tracking. The use of a robust error function in the observation model allows, furthermore, to improve the robustness to partial occlusions. However, when heavy or total occlusion occur the tracker is easily distracted.

When the pose is tracked with a classic Condensation scheme and zero order dynamics, accuracy is sacrificed for robustness. In particular the tracker allows for automatic reinitialization, as intrinsic property of Condensation, after total occlusion.

As should be clear, what emerges from this qualitative analysis of the problem, is that more then one dynamical model should be integrated into the tracker, to allow a wider range of motion to be supported without losing the advantages of an accurate prediction.

To summarize we can say that in case of limited occlusion the deterministic AAM search should be reasonably trusted, leaving to the importance sampling the task to improve the accuracy of the detection. In the second scenario, scenes with strong occlusions, a tracker less accurate, but more robust to occlusions should be preferred. In our framework the Condensation based tracker will accomplish this task.

Our solution then consists in merging the solutions presented in the above section by means of an automatic model switching procedure.

The tracker diagram is shown in figure 4.5. With respect to the general scheme of figure 4.3 a new block has been added, whose output is used in the resampling stage: a weighted average of the state estimates is used for state prediction, constituting in all respects a third motion model, which results particularly useful for tracking reinitialization.

In the following we will first show how to manage a mixed continuous-discrete representation of the state that supports multiple motion models. Then we will describe the new motion model and, finally, formulate our final algorithm.

4.3.1 Mixed-State Condensation Framework

The purpose in this section is to extend the Condensation framework to permit a mixed-state object representation, combining continuous-valued shape parameters with a discrete label encoding which one, among a discrete set of motion models, is in force. A joint pdf for the mixed-state model is derived, and due to the structure of the algorithm, model switching is performed jointly with tracking. The fact that the process density $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ can have a somewhat general form can be exploited to allow the Condensation algorithm to support, and automatically switch between, multiple motion models. The extended state is defined as

$$\mathbf{X} = (\mathbf{x}, \theta), \ \ \theta \in 1, .., N_m \tag{4.21}$$

where θ is a discrete variable labelling the current model. The process density can then be decomposed as follows [28]:

$$\begin{aligned} &p(\mathbf{X}_k|\mathbf{X}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}, \theta_k)P(\theta_k|\theta_{k-1}, \mathbf{x}_{k-1}) \\ &P(\theta_k|\theta_{k-1}, \mathbf{x}_{k-1}) : P(\theta_k = j|\theta_{k-1} = i, \mathbf{x}_{k-1}) = T_{ij}(\mathbf{x}_{k-1}) \end{aligned}$$

where the T_{ij} are *state transition probabilities.* The continuous motion models for each transition are given by the *sub-process densities* $p_{\theta_j}(x_k|x_{k-1})$

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \theta_k) : \ p(\mathbf{x}_k|\mathbf{x}_{k-1}, \theta_k = j) = p_{\theta_j}(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{4.22}$$

The algorithm for the mixed-state Condensation is shown in table 4.1. When tracking with a model of this form, discrete state transitions can be expected to occur automatically when appropriate. Each discrete state transition with non zero probability, contributes some samples to the state distribution. As soon as one model predicts significantly more accurately than the others, that model will dominate.

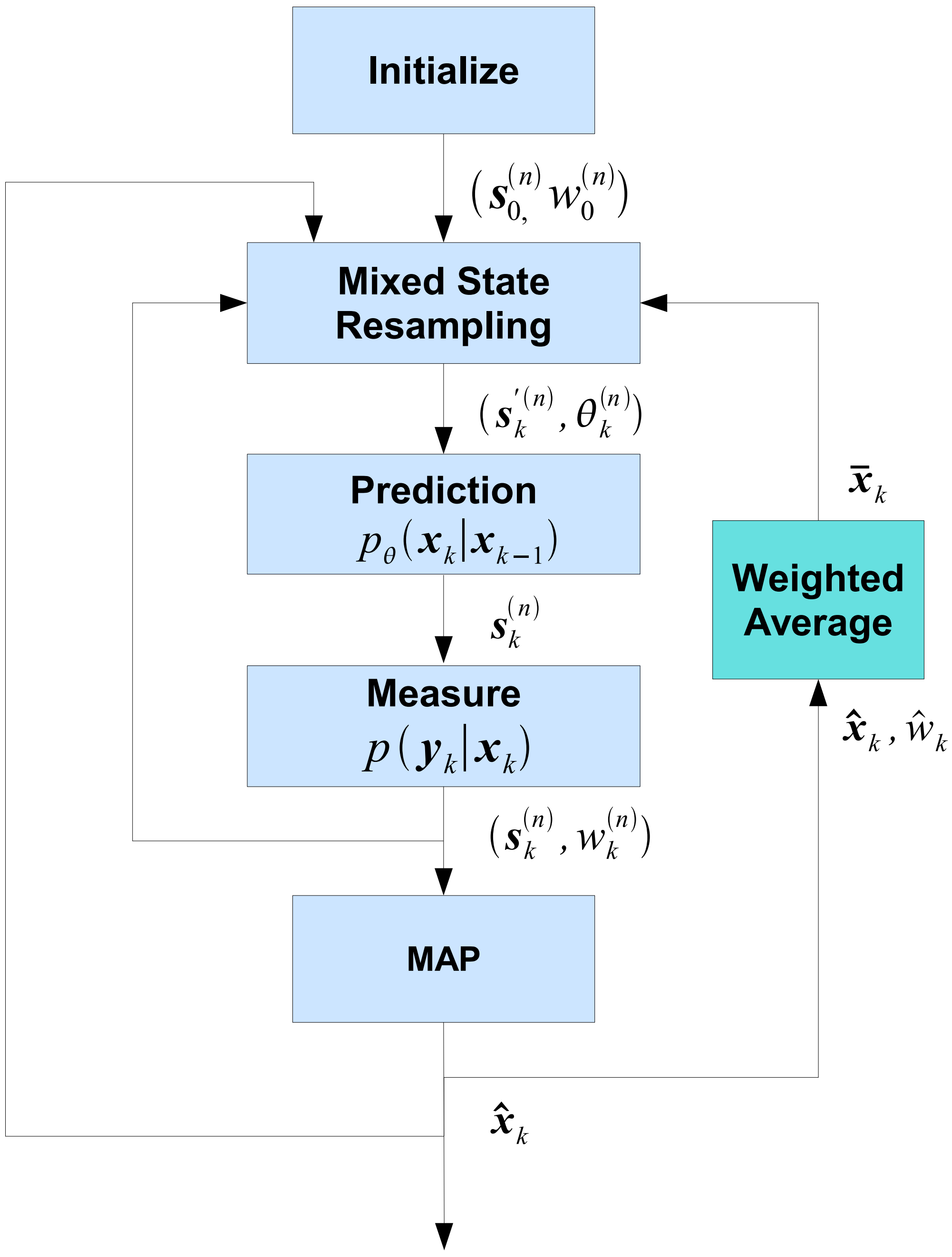

Initialize
$(\boldsymbol{s}_{0,}^{(n)} w_0^{(n)})$
Mixed State Resampling
$(\boldsymbol{s}_k^{'(n)}, \theta_k^{(n)})$
Prediction
$p_\theta(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$
$\boldsymbol{s}_k^{(n)}$
Measure
$p(\boldsymbol{y}_k|\boldsymbol{x}_k)$
$(\boldsymbol{s}_k^{(n)}, w_k^{(n)})$
MAP
$\hat{\boldsymbol{x}}_k$
Weighted Average
$\bar{\boldsymbol{x}}_k$
$\hat{\boldsymbol{x}}_k, \hat{w}_k$

Figure 4.5: Block Diagram for the proposed Mixed-State Condensation tracker

We will assume in the following that $T_{ij}(x_k) \equiv T_{ij}$, and these transition probability will be specified by hand in the form of a transition matrix $\mathbf{T}$.

From the "old" sample set $\{\mathbf{s}_{k-1}^{(n)}, w_{k-1}^{(n)}, n = 1, .., N_{k-1}\}$ at timestep $k-1$, construct a "new" sample set $\{\mathbf{s}_k^{(n)}, w_k^{(n)}, n = 1, .., N_k\}$ for time k. Construct the new sample set as follow:

1. **Resample.** Select a sample $\mathbf{s}_k^{*(n)} = (\mathbf{x}^{*(n)}, i)$ as follows:
   - Generate a random number j with probability proportional to $w_{k-1}^{(j)}$.This can be done by binary subdivision using cumulative probabilities.
   - Set $\mathbf{s}_k^{*(n)} = s_{k-1}^{(j)}$
2. **Predict.** Sample from $p(\mathbf{X}_k|\mathbf{X}_{k-1} = \mathbf{s}_k^{*(n)})$ to choose $\mathbf{s}_k^{(n)} = (\mathbf{x}_k^{(n)}, \theta_k^{(n)})$:
   - Sample transition probabilities:
     $P(\theta_k|\theta_{k-1} = i)$
     to find $\theta_k^{(n)}$.
   - Sample sub-process density $p_{\theta_k^{(n)}}(\mathbf{x}_k|\mathbf{x}_k^{*(n)})$ to find $\mathbf{x}_k^{(n)}$.
3. **Measure.** Assign a weight to the new position in terms of the image data $\mathbf{y}_k$: $w_k^{(n)} = p(\mathbf{y}_k|\mathbf{X}_k = \mathbf{s}_k^{(n)})$

Normalize the weights so that $\sum w_k^{(n)} = 1$

Table 4.1: The Mixed-State Condensation algorithm.

4.3.2 A Third Motion Model from data averaging

When considering the face motion in a real video sequence, there is not a clear trajectory of the face that one can model. In fact, we basically consider motion as random from frame to frame, and use the AAM search for prediction if certain conditions hold.

Here, we want to model a kind of a priori knowledge on the face motion, closely related to the considered sequences. We can describe it, informally, as follows: in each video sequence the infants are sitting, thus their movements are somehow constrained around a point in the scene. In other words, we can assume that there is a point in the scene, where is more probable to find the target; furthermore, we can assume that the face lies in a neighborhood of such point, with probability that decreases with the distance.

To formalize the problem, the above assumption is translated in the mathematical hypothesis of unimodality of the pdf $p(\mathbf{x}_k)$. If, for sake of simplicity, we consider a gaussian distribution, we can write:

$$p(\mathbf{x}_k) = N(\bar{\mathbf{x}}, \mathbf{C}_k) \tag{4.23}$$

where $\bar{\mathbf{x}}$ is the (fixed) *mean vector* and $\mathbf{C}_k$ is the covariance matrix of the distribution at time k; both quantities are, of course, unknown.

Then, the basic idea is to use the information from all the state vector trajectory in order to estimate the mean vector. If $\{(\hat{\mathbf{x}}_1, w_1), .., (\hat{\mathbf{x}}_k, w_k)\}$ is the estimated trajectory of the state vector up to time k, then a simple (unbiased) estimator $\bar{\mathbf{x}}_k$ of $\bar{\mathbf{x}}$, is the weighted sum [29]:

$$\bar{\mathbf{x}}_k = \frac{\sum_{i=1}^{k} w_i \mathbf{x}_i}{\sum_{i=1}^{k} w_i}. \tag{4.24}$$

In the above equation, weights are used to give more importance to plausible candidates, thus reducing the variance of the estimate: w_i are the unit normalized weights used to approximate the posterior. This means that w_i becomes larger as the posterior reveals a predominant peak, which should happen when measurements are unambiguous.

The mean vector has an interesting interpretation, when referred to the infant face tracking. It represents, in fact, both the mean spatial position of the target in the image, and the mean face in the sense of mean expression of the baby. Images taken from a video sequences and shown in figure 4.6, give visual feedback of the estimation process. As it can be seen, the *mean face* is always in a neighborhood of the true face.

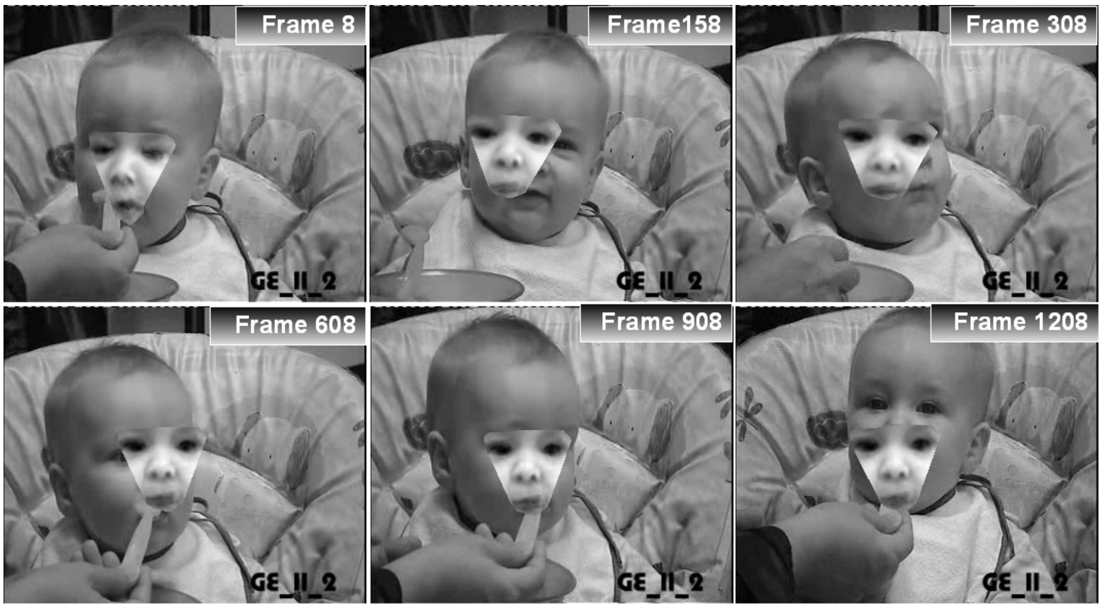


Figure 4.6: Position of the mean face $\bar{\mathbf{x}}$ in a long video sequence, represented by the (unnormalized) synthetic face.

Thus, a new motion model is defined by:

$$\mathbf{x}_k = \bar{\mathbf{x}}_k + \mathbf{S}_k\mathbf{u} \tag{4.25a}$$

$$p(\mathbf{x}_k) = N(\bar{\mathbf{x}}_{k-1}, \mathbf{S}_k) \tag{4.25b}$$

Note that the used covariance matrix for the distribution is $\mathbf{S}_k$, the same as in (4.8), and not $\mathbf{C}_k$, since we consider it as design parameter in the context of particle filtering.

The above equations can be used to include some probability of tracking reinitialization: at each time step k, some particles can be generated by sampling from $p(\mathbf{x}_k)$. Since this distribution is a kind of a priori, particles generated in such a way are specially useful after distraction due, for example, to total occlusions.

From an algorithmic point of view, it should be noted that this filtering operation does not alter the complexity of the algorithm, neither temporally nor spatially. In fact, an efficient recursive implementation can be used:

$$\begin{aligned}\mathbf{M}_k &= \mathbf{M}_{k-1} + \mathbf{x}_k w_k,\\ S_k &= S_{k-1} + w_k,\\ \bar{\mathbf{x}}_k &= \mathbf{M}_k / S_k,\end{aligned}$$

so that only the vector $\mathbf{M}_k$ and the scalar S_k must be stored at time k.

4.3.3 Final Algorithm

The Mixed-State Condensation algorithm given in table 4.1, allows to take advantage of the three motion models described by (4.17), (4.19) and (4.25). The integration of different motion models is governed by the state transition distribution $P(\theta_k|\theta_{k-1})$ specified in matrix form $\mathbf{T}$.

A useful, graphical way to represent the mixed state tracker is by means of the state transition diagram in figure 4.7. Such a representation suggests that the system is made up of two different interacting processes: a Markov chain (continuous state) $\{\mathbf{x}_0, \mathbf{x}_1, .., \mathbf{x}_k\}$ representing the target face, and a Markov chain $\{\theta_0, \theta_1, .., \theta_k\}$, with finite state space, describing the evolution of the motion model with time. The possible values of θ_k are: R (Reinitialization), I (Importance Sampling), P (Pose).

In other contexts, the Interacting Multiple Model algorithm [30] is used to address this situation [31][32].

Since a particle filter is used as the tracker, each particle can be simply expanded to contain the motion model estimate. The particle is propagated forward in time according to the dynamics implied by the motion model, and transitions between models happen according to the transition matrix $\mathbf{T}$. This matrix is a design parameter and is very important for tracking success.

We propose two different forms for the transition matrix:

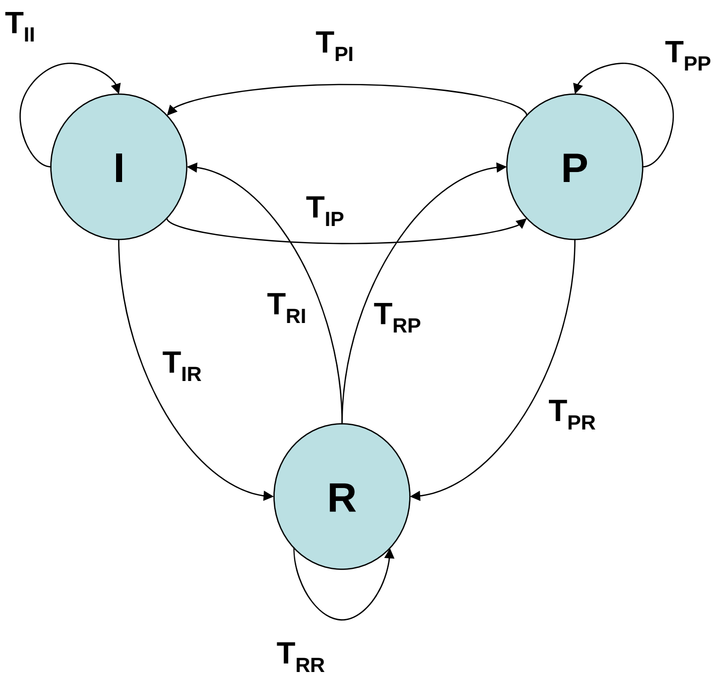


Figure 4.7: State Transition Diagram for the mixed-state Tracker.

1.

$$\mathbf{T} = \begin{pmatrix} T_{RR} & T_{RI} & T_{RP} \\ T_{IR} & T_{II} & T_{IP} \\ T_{PR} & T_{PI} & T_{PP} \end{pmatrix} = \begin{pmatrix} \alpha & 1-\alpha-\delta & \delta \\ \alpha & 1-\alpha-\delta & \delta \\ \alpha & \delta & 1-\alpha-\delta \end{pmatrix} \tag{4.26}$$

2.

$$\mathbf{T} = \begin{pmatrix} T_{RR} & T_{RI} & T_{RP} \\ T_{IR} & T_{II} & T_{IP} \\ T_{PR} & T_{PI} & T_{PP} \end{pmatrix} = \begin{pmatrix} \alpha & 1-\alpha-\delta & \delta \\ \alpha & 1-\alpha-\delta & \delta \\ \alpha & 1-\alpha-\delta & \delta \end{pmatrix} \tag{4.27}$$

The matrix $\mathbf{T}$ has been parameterized by α and δ to make its interpretation easier:

- α is, in both cases, a *reinitialization* parameter, since, at each time step, a number of particles proportional to α is generated from model (4.25).
- The meaning of δ changes, instead, between the first and the second representation [33]: in the first case it represents an *adaption speed* parameter,

that is controls how rapidly the probability flows from the model I to the model P. Thus, we can say that it trades off adaptation rate and steady state behavior. In the second case, we give it the meaning of *robustness* parameter, controlling how promptly the tracker switches into (robust) pose tracking.

In our experiments, the second form of the matrix has been chosen, since it gives an easy way to control the robustness of the tracker.

Summarizing, at each time step k the proposed algorithm chooses a particle from the previous sample set proportionally to its weight. The particle is then propagated through one of the three dynamical models in accordance with the current motion label. If, for example, a particle is chosen with label I, then with probability T_{II} a particle is drawn from the importance function $q(\hat{\mathbf{x}}_{k-1})$, with probability T_{IR} it is drawn from (4.25), and with probability T_{IP} it is propagated through (4.17). Finally, the particle is weighted in accordance to observation, and the multiplicative factor f/q is applied if it was generated with Importance Sampling. A detailed description of the algorithm is given in table 4.2.

From the "old" sample set $\{\mathbf{s}_{k-1}^{(n)}, w_{k-1}^{(n)}, n = 1, .., N_{k-1}\}$ at timestep $k-1$, construct a "new" sample set $\{\mathbf{s}_k^{(n)}, w_k^{(n)}, n = 1, .., N_k\}$ for time k. Construct the new sample set as follow:

1. **Resample.** Select a sample $\mathbf{s}_k^{*(n)} = (x^{*(n)}, i)$ as follows:
   - Generate a random number j with probability proportional to $w_{k-1}^{(j)}$.This can be done by binary subdivision using cumulative probabilities.
   - Set $\mathbf{s}_k^{*(n)} = s_{k-1}^{(j)}$
2. **Predict.** Sample from $p(\mathbf{X}_k|\mathbf{X}_{k-1} = s_k^{*(n)})$ to choose $\mathbf{s}^{(n)} = (\mathbf{x}_k^{(n)}, \theta_k^{(n)})$:
   - Sample transition probabilities $P(\theta_k|\theta_{k-1} = i)$ to find $\theta_k^{(n)}$.
   - Sample sub-process density $p_{\theta_k^{(n)}}(\mathbf{x}_k|\mathbf{x}_k^{*(n)})$ as follow:

     a) if $\theta_k^{(n)} = I$ choose $\mathbf{s}_k^{(n)}$ by sampling from $q(\hat{\mathbf{x}}_{k-1})$ and set the importance correction factor $\lambda_k^{(n)} = \frac{f(s_k^{(n)})}{q(s_k^{(n)})}$

     b) if $\theta_k^{(n)} = P$ choose $\mathbf{s}_k^{(n)}$ propagating the sample according to the equation $\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{S}_k\mathbf{u}$ and set the importance correction factor $\lambda_k^{(n)} = 1$

     c) if $\theta_k^{(n)} = R$ choose $s_k^{(n)}$ by sampling from $N(\bar{\mathbf{x}}_{k-1}, \mathbf{S}_k)$ and set the importance correction factor $\lambda_k^{(n)} = 1$
3. **Measure.** Assign a weight to the new position in terms of the image data $\mathbf{y}_k$ and the importance sampling correction term:

$$w_k^{(n)} = \lambda_k^{(n)} p(\mathbf{y}_k|\mathbf{X}_k = \mathbf{s}_k^{(n)})$$

Normalize the weights so that $\sum w_k^{(n)} = 1$

Table 4.2: The Final Algorithm.

Chapter 5

Test and Results

In the previous chapter we have presented our proposed approach for infant face tracking based on a mixed state Condensation technique. In the following chapter we show the application of our technique on several sequences and we report the comparative results between our approach and some other techniques described in previous chapters.

The chapter is organized as follows: first implementation details are given, such as the programming language adopted, the function library used, the description of the face model. Then, we describe how we decide to assess the performance of a tracking algorithm and, finally, we present the results in a comparative manner.

5.1 Implementation

Implementation of the tracker is based on AAM-API, a C++ implementation of Active Appearance Model. This library has been developed by M. B. Stegmann for his diploma project [11], and provided as open source.

In order to build our AAM representation of face (see section 2.2), we have manually landmarked a set of 222 images using the facial model reported in figure 5.1. The texture vector is composed of 2420 pixels and the shape vector dimension is 44. The model is built using 28 shapes modes, 99 texture modes and

59 combined appearance modes thus retaining the 98% of the combined shape and texture variation.

Figure 5.3 shows the variance associate to each mode of variation of the combined eigenvalues in descending order, while 5.2 shows the effect of varying the first three combined model parameters.

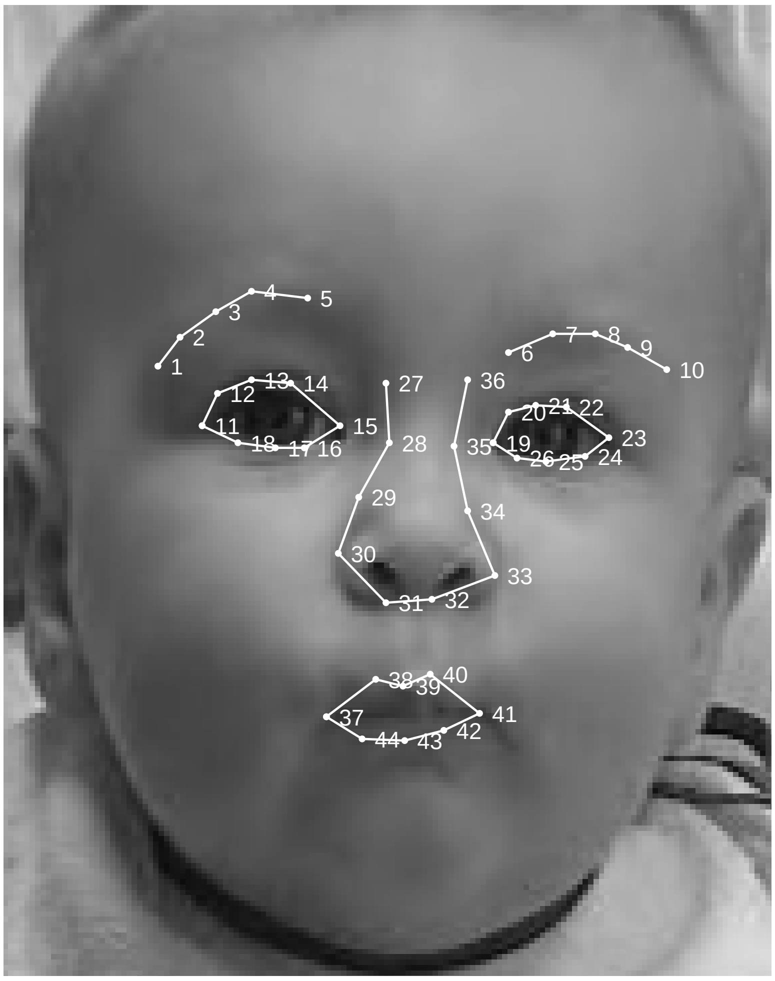


Figure 5.1: Hand annotated image. The 44 landmarks representation of the face.

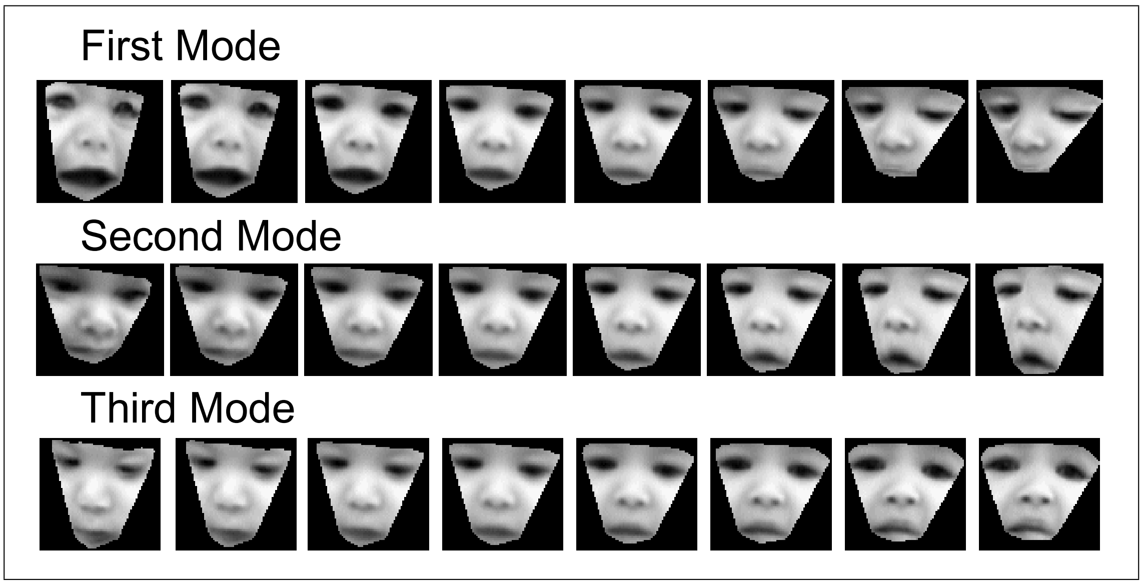


Figure 5.2: The effect of varing the first three model parameters between ±3 standard deviations.

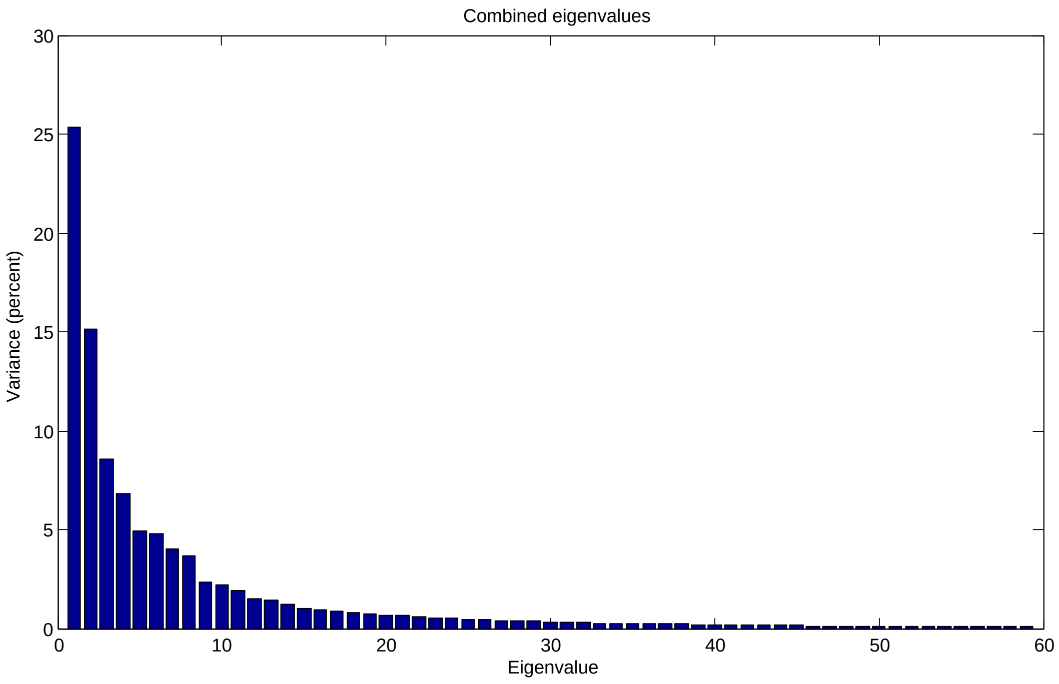


Figure 5.3: Combined eigenvalues. 59 combined appearance modes retain 98% of the combined shape and texture variation

5.2 Performance Assessment

Assessing tracker performance is not straightforward. In general, it is difficult to find a performance index that summarizes the "goodness" of a given approach, which also depends on the particular application the tracker is designed for. Since we have no real-time constraints, we consider accuracy and robustness the main components to evaluate and in the following we will define what accuracy and robustness represent in our context. With the term accuracy, we mean the ability to reproduce a synthetic face as "near" as possible to the true face. In order to measure the closeness between the true face and the tracked one we decide to use the texture distance introduced in 4.1.1.

Then to assess performance at least two approaches can be adopted [11]:

1. The first approach is that to compare the tracked face model with a known *ground truth* to produce a measure of the *true error.* This imply, however, that for each frame in the video sequence, manual landmarks must be placed on the face. This is extremely high time-consuming, hardly feasible for video sequences with an high number of frames.

2. As a second approach, the results could be validated directly by using the tracking error in (4.7), that is the distance between the texture generated from the tracked parameters **c**, and the image patch sampled in correspondence to the tracked pose and shape. In the following, we will refer to this procedure as *self-contained validation.*

If a total occlusion occurs in a video frame, both approaches cannot be performed, since the target face is not in the image and the the frame should be excluded from the analysis. Regarding the second approach, the self-validation is well defined only when the tracking succeeds or, at least, when the tracked shape is placed upon the true face. In figure 5.4, two situations are presented: on the left, the face is well tracked. In this case self validation process is well posed since the sampled image patch represents the true face. On the right, instead, the tracker fails. In this case the tracking error is a misleading measure. When using self validation, a preliminary classification is necessary in order to separate the tracked frames accordingly to the two described situations. In the following we will use the terms *in-track* and *target-lost* referring to the two cases reported in fig 5.4.

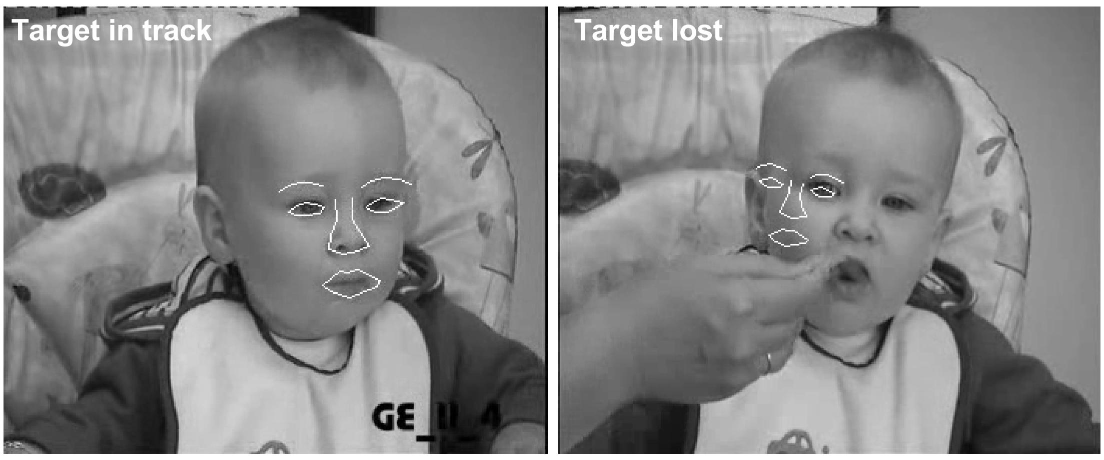


Figure 5.4: On the left: the tracker succeeds so the self validation is well posed. On the Right: the tracker fails so the tracking error is meaningless.

The definition of a robustness measure is more problematic. Robustness of a tracker refers to the property of succeeding in tracking even when distraction occurs, that is, when the face is occluded by other objects and/or when in the background some object mimics the appearance of a face. Even though the fitting error is partially related to robustness (the loss of a target implies increase of the fitting error), this error measure cannot be directly associated to the property of robustness. For our purposes we state that a tracker is as much robust as more frames are classified in-track. A simple index then consists in counting the number of tracking failures in a video sequence or, in other words, the number of target-lost frames.

Even if we have no time constraints, computational time must be reasonable. Furthermore under the same results in terms of accuracy and robustness, we clearly prefer an algorithm which performs better in time. To assess time performance, we have decided to use the mean frame rate defined as:

$$R_F = \frac{Total\,Number\,of\,frame}{Total\,Computational\,time\,(sec)} \tag{5.1}$$

5.3 Results

To test the Mixed State Condensation tracker we used 50 video sequences. For all of them a visual analysis has been done, resulting in good overall performance of the proposed tracker. To show the benefits of our approach we have chosen 3 video sequences that we consider representative for performance assessment. Simulation results for the algorithms discussed in chapter 4 are presented in a comparative manner.

For two of these sequences, we have used a self-contained validation. For the third sequence, we additionally performed an accurate analysis: first hand-placed landmarks were annotated in a subset of the video frames; then, we found an exact values for the tracking error comparing the tracked face model with the ground truth.

At the end, we compared the algorithms in terms of throughput (as defined in (5.1)) and showed that the mixed state algorithm uses a quite reasonable amount of time resources.

5.3.1 Self Validation

To perform a self validating analysis, we have chosen two long video sequences:

1. **Sequence 1** is about 2700 frames long. The sequence is characterized by heavy total occlusions, which often occur when an hand completely covers the face (see figure 5.5)

2. **Sequence 2** is about 2200 frames long. In this sequence the infant moves widely and even hides the face for a long time (about 200 frames) (figure 5.6).

In figures 5.5 and 5.6 we show some tracking results. In the image grid each column represents a tracked frame from the sequence, while each row is related to one of the 3 compared approaches: the Mixed State Condensation, the ICondensation and the Condensation-pose tracker.

For displaying purpose only 4 frames per sequence are shown but these are representative of the whole sequence. Below the image grid a plot shows the tracking error for the entire sequences. In both image sequences there is a central event: in the first sequence an hand occludes totally the face (frame 2029) while in the second a total self-occlusion occurs (frame 1512).

We can notice that although the Condensation-pose tracker is quite robust to the occlusion, the tracking error is still very high since it doesn't track the inner motion of the face.

The behavior of the ICondensation is exactly the opposite: it tracks well as long as the target is not occluded, but it is not able to recover tracks once distracted. The Mixed State Condensation shows a good trade-off behavior: it automatically switch model choosing the best for each situation. It reveals high robustness and the best accuracy (see 5.1) from this self validated analysis. This is confirmed by accurate results reported in the next section 5.3.2.

	ICondensation	**Condensation-pose**	**Mixed State Condensation**
Sequence 1	0.042	0.045	**0.019**
Sequence 2	0.047	0.069	**0.029**

Table 5.1: Mean tracking-error for the first two sequences.

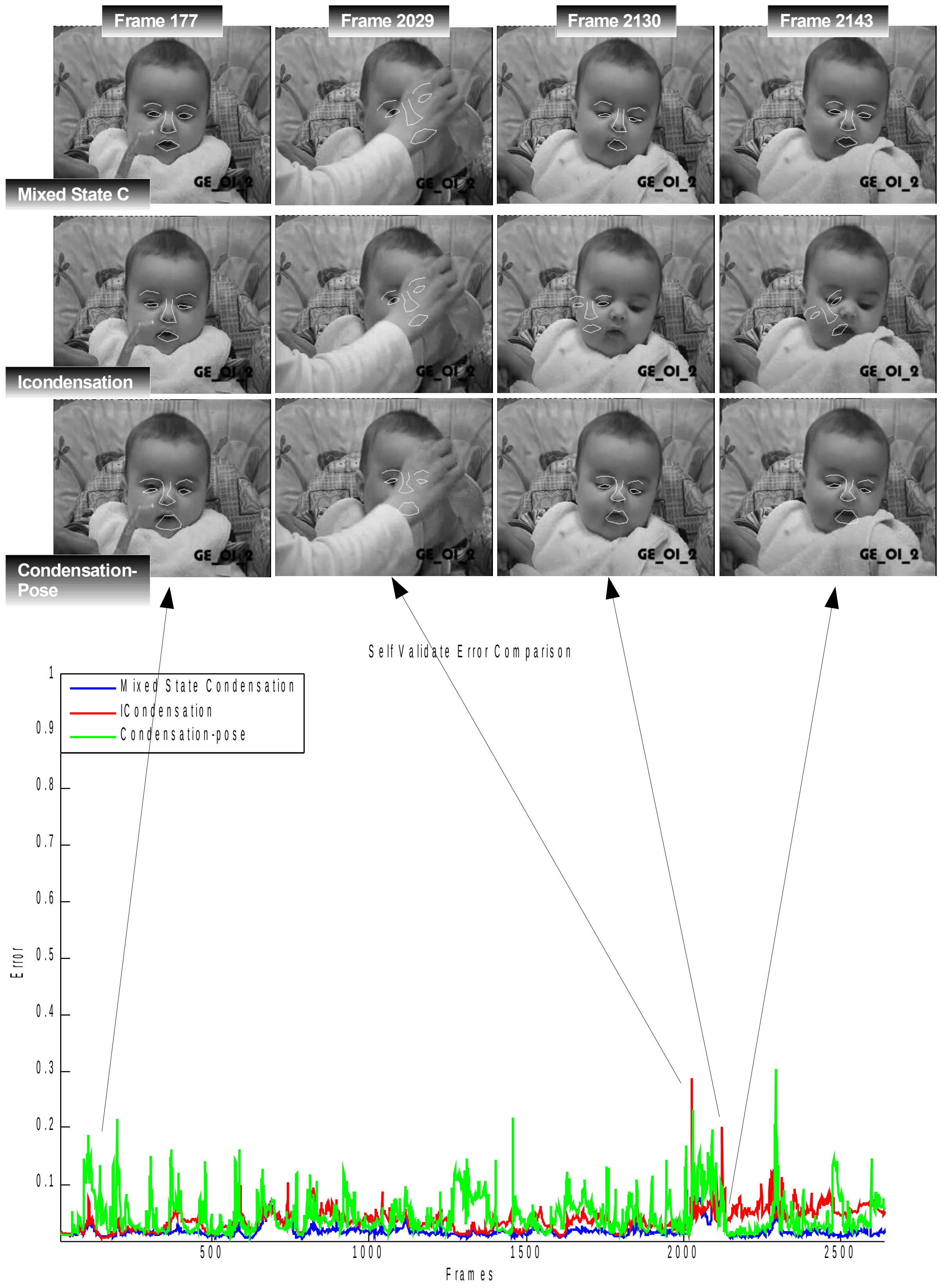


Figure 5.5: Results of self-contained validation analysis for the first sequence. The tracking errors (Lorenzian norm) are plotted.

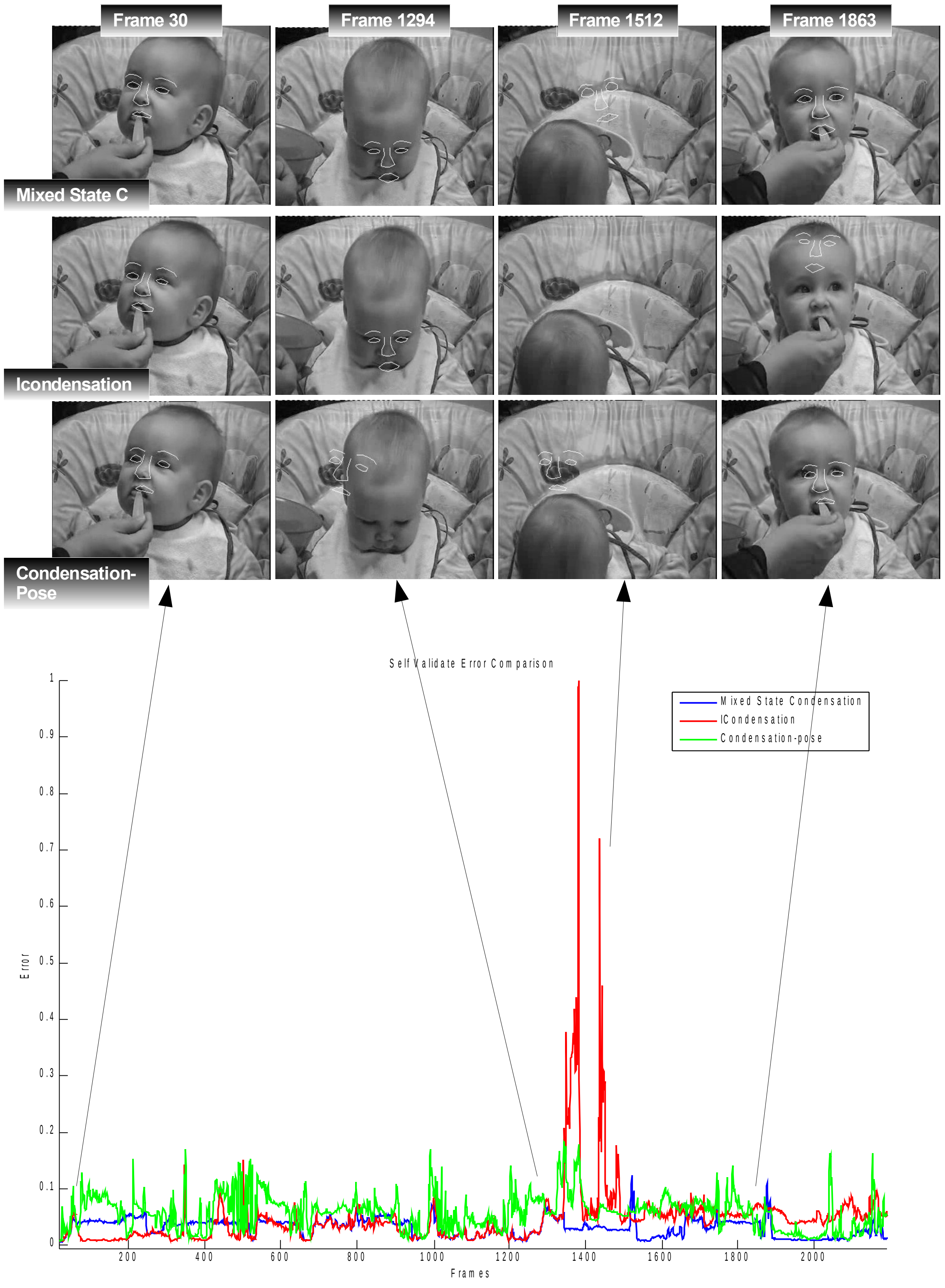


Figure 5.6: Results of self-contained validation analysis for the second sequence. The tracking errors (Lorenzian norm) are plotted.

5.3.2 Accurate analysis

In section 5.3.1 a self-contained validation of two long video sequences resulted in a outperforming of the Mixed State tracker when compared to the others. Our purpose is to confirm such results with a more accurate analysis, on the basis of the considerations made in section 5.2. The drawback of the self validation approach is that the tracking error is only an approximated measure of the texture distance between the tracked face and the true one. Even if we perform a self validation only on the target in-track frames, partial occlusion or extreme poses of the face will result in a tracked shape not perfectly aligned to the true face. This means that it can even happen that the tracked error is lower than the true error; this is the case, for example, when partial occlusions occur and the tracked face is found in a position of local minimum (respect to the texture distance) in a non-face region. The only solution, in such cases, is to manually annotate the image.

On the basis of the above considerations, we performed the following steps:

- We chose a shorter video sequence of 800 frames and manually annotated a part of it: we sampled the sequence at a fixed interval of 5 frames, annotating thus 800/5=160 frames.

- We excluded from analysis 14 total occluded frames, and we classified the remaining frames as in-track or target-lost on the basis of the considerations made in section 5.2.

  Then we calculated the tracking error for the in-track frames, and the true error for the in-track frames when a ground truth was available. The results are shown in figure 5.7 and 5.8.

- Moreover we found an *optimum model* of face for each ground truth. The optimum model can be defined as the face model that matches at best the true face in the image. When an hand-annotated image is given, the optimum model can be obtained projecting the true shape and the corresponding texture points into the subspace spanned by the **c** parameters, simply inverting the linear relationship $\mathbf{b} = \Phi_c \mathbf{c}$ (recall section 2.2), where **b** is the vector of concatenated shape and texture points.

- The optimum-model true-error has been used for displaying purposes, and it is the black line in the error plot. Thus, the black line represents the approximate lower bound for error, because it basically models the limits of AAM to model the face.

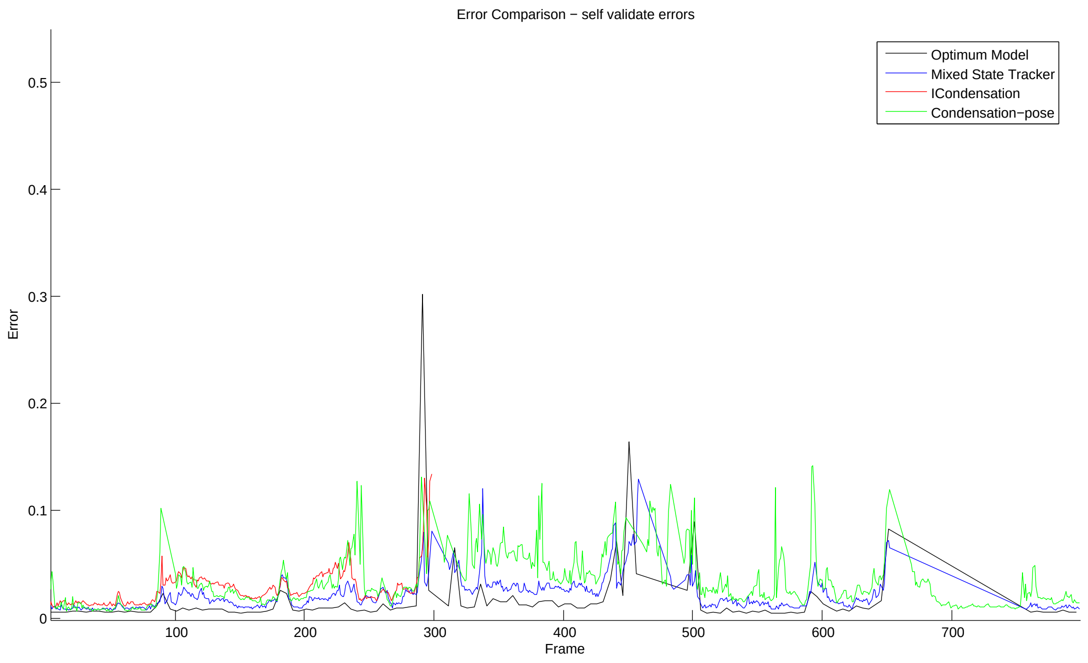


Figure 5.7: Tracking errors comparison for the third sequence. The black line represent the optimum-model error.

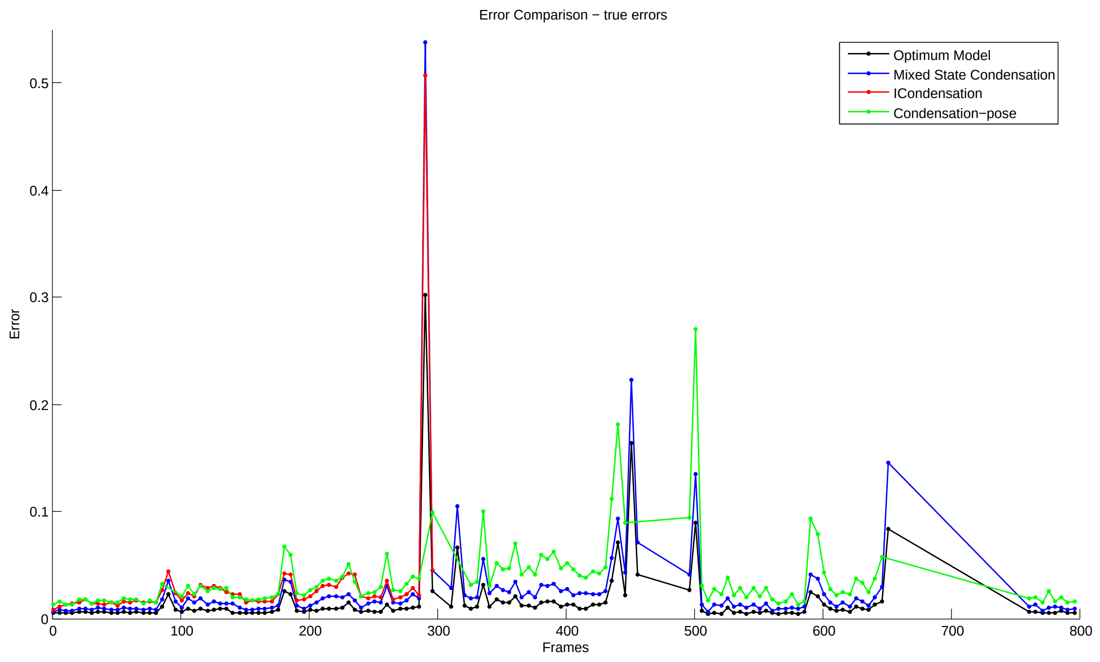


Figure 5.8: True errors comparison for the third sequence. The black line represent the optimum-model error.

We summarize the results for the accurate analysis in table 5.2. The mixed state tracker still outperforms in terms of accuracy, even when the true error is considered and it is comparable to Condensation-pose in terms of number of failures.

Comparing numerical results in table 5.2, one can note that, on average, tracking error is lower then the true error. On the whole, however, the results from the two different analysis are in accordance, showing that the self contained validation can be trusted.

	ICondensation	**Condensation-pose**	**Mixed State Condensation**
Mean (true) Error	0.03	0.036	**0.026**
Mean (tracking) Error	0.027	0.032	**0.021**
Number of failures	488	64	**136**

Table 5.2: Results of accurate analysis.

5.3.3 Time performance

The results given in table 5.3 are obtained with a P4 1.8 GHz processor, equipped with 512MB of RAM. A further algorithm has been used for benchmarking: “AAM Search”. This is a simple algorithm in which the AAM Search is applied frame-by-frame: it cannot be used for tracking in practical situations, since it is neither accurate nor robust, however it gives a kind of reference for time performance. From the table 5.3, it is clear that, despite the increased complexity of the tracker, the Mixed State Condensation outperforms in sequence 1, and has comparable performance for the other sequences. This is due to the adaptive dynamics described in section 4.1.3: basically, the number of particles used to represent the posterior distribution of the state vector, and therefore the computational time, is proportional to the accuracy of algorithm. Thus, the results in table 5.3 are in accordance with the results in table 5.2, where we have shown that the Mixed State tracker outperforms the other approaches in accuracy.

	ICondensation	**Condensation-pose**	**Mixed State Condensation**	**AAM Search**
Sequence 1	2.62	2.58	2.53	3.18
Sequence 2	3.13	2.57	2.59	2.5
Sequence 3	3.19	2.48	2.5	2.63

Table 5.3: Time performance comparison (values are given in frames/sec).

Chapter 6

Conclusions and Future Work

In this work we presented a stochastic framework for robust face tracking using complex models of face appearance. When compared to other approaches found in literature [2][10], the presented tracker not only succeeds in crucial cases of occlusion, but experiments show that it outperforms in accuracy and find an equilibrate trade-off between robustness and use of time resources. The resulting algorithm is an adapted mixed state Condensation combined with AAM. The stochastic Condensation search compensates for AAM limits in handling occlusions, and due to an appropriate choice of motion models, permits an efficient reinitialization of tracking, when an exhaustive search in the image is impractical. Furthermore the approach is general enough to be applied to other face tracking problems: an advantage of the probabilistic approach is that it is modular, in the sense that application-specific dynamical models or observation models can be included seamlessly.

6.1 Main Contributions

The main contributions of this thesis are:

- A new robust algorithm for face tracking, obtained combining the AAMs within the mixed-state Condensation framework. Three different state evolution models have been used to model the target dynamics. In particular, a new motion model has been proposed for tracker reinitialization, specified by a gaussian distribution with mean obtained from a weighted average of state estimates up to time k.

- An efficient implementation of the tracker in the C++ language, that can be used for further developments.

6.2 Propositions for further work

Here, we briefly discuss some ideas developed during the thesis work, that either were out of the scope or out of reach within the given time span.

- **Robust AAM Search.** In 4.1.3, it has been shown how, in (near) occlusion-free situations, AAM Search can help in the prediction stage to find a more accurate state estimate. A zero velocity model, with random noise, must be used instead when occlusions cause the AAM prediction to fail. Noise variance must be high enough to allow rapid movements to be tracked hence the algorithm runs more slowly. The efficiency of the tracking algorithm heavily depends on the accuracy of the state evolution model. A robust AAM search should be used, therefore, to improve the predictive step. The problem of constructing AAM with occlusions has been addressed in recent works on AAM extension [34][35][25][36], and could constitute an interesting development for tracking purposes.

- **Multi-View AAM.** For the addressed problem, a near frontal model has been considered sufficient to capture the expected variations of the face. This is not always true, and large rotations or profile views are primary sources of self occlusions. The mixed-state framework can be used to automatically support multiple motion models, together with a set of face models to represent the variations in appearance from different view-point.

Thus for example the left and right profile model can be used to add robustness to the tracker.

- **Framework for test.** Performance assessment is an extremely hard task. Reliability and reproducibility of results are strongly related to an accurate definition of a framework for testing. This requires, for example, to define: proper performance parameters related to the specific application, the way the measurement are done, the definition of suitable indexes of performance, and a ground truth data set.

- **Real time implementation.** Even though some effort was spent on optimizing the algorithm resulting in satisfactory time performance, there is still scope for improvement. In particular, real time implementation is an attractive issue.

Bibliography

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Lecture Notes in Computer Science*, vol. 1407, pp. 484–??, 1998.

[2] F. Dornaika and F. Davoine, "Head and facial animation tracking using appearance-adaptive models and particle filters," in *Real Time Vision for Human-Computer Interaction*, p. 153, 2004.

[3] J. Ahlberg, "An active model for facial feature tracking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, pp. 566–571, June 2002.

[4] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[5] M. A. Carreira-Perpinan, "A review of dimension reduction techniques," tech. rep., University of Sheffield, Jan. 1997.

[6] A. Blake and M. Isard, "The CONDENSATION algorithm - conditional density propagation and applications to visual tracking," in *NIPS* (M. Mozer, M. I. Jordan, and T. Petsche, eds.), pp. 361–367, MIT Press, 1996.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 25, no. 5, pp. 564–575, 2003.

[8] F. Bettinger, T. F. Cootes, and C. J. Taylor, "Modelling facial behaviours," in *BMVC* (P. L. Rosin and A. D. Marshall, eds.), British Machine Vision Association, 2002.

[9] J. Ahlberg, "Using the active appearance algorithm for face and facial feature tracking," in *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. xx–yy, 2001.

[10] S. Hamlaoui and F. Davoine, "Facial action tracking using an AAM-based condensation approach," Mar. 18 2005.

[11] M. B. Stegmann, "Active appearance models: Theory, extensions and cases," Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Aug. 2000.

[12] T. Cootes and C. Taylor, "Statistical models of appearance for computer vision," 1999.

[13] C. T. G.J. Edwards and T. Cootes, "Interpreting face images using active appearance models," Apr. 14 1998.

[14] T. F. Cootes and C. J. Taylor, "Constrained active appearance models," in *ICCV*, pp. 748–754, 2001.

[15] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," Mar. 05 2000.

[16] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussianbayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, pp. 174–188, Feb. 2002.

[17] R. van der Merwe, A. Doucet, N. de Freitas, and E. A. Wan, "The unscented particle filter," in *NIPS* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 584–590, MIT Press, 2000.

[18] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter," May 30 2001.

[19] D. Crisan and A. Doucet, “A survey of convergence results on particle filtering methods for practitioners,” Oct. 04 2002.

[20] M. Isard and A. Blake, “C-conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, pp. 5–28, Aug. 1998.

[21] D. S. A. Gordon, N.J. Salmond, “Novel approach to nonlinear/non-gaussian bayesian state estimation,” in *Radar and Signal Processing, IEE Proceedings F*, pp. 107–113, Apr. 1993.

[22] M. Isard and A. Blake, “ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework,” *Lecture Notes in Computer Science*, vol. 1406, pp. 893–??, 1998.

[23] M. J. Black and A. Rangarajan, “On the unification of line processes, outlier rejection, and robust statistics with applications in early vision,” *Int. J. Computer Vision*, vol. 19, pp. 57–92, July 1996.

[24] S. K. Zhou, R. Chellappa, and B. Moghaddam, “Visual tracking and recognition using appearance-adaptive models in particle filters,” *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1491–1506, 2004.

[25] G. Edwards, T. Cootes, and C. Taylor, “Advances in active appearance models,” in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, vol. I, (Los Alamitos, CA), pp. 137–142, IEEE, Sept. 20–27 1999.

[26] F. Daum and J. Huang, “Curse of dimensionality and particle filters,” *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 4, pp. 174–188, Mar. 2003.

[27] F. Davoine and F. Dornaika, *Head and facial animation tracking using appearance-adaptive models and particle filters.* Springer Verlag, 2005.

[28] M. Isard and A. Blake, “A mixed-state CONDENSATION tracker with automatic model-switching,” in *ICCV*, pp. 107–112, 1998.

[29] S. M. Kay, *Fundamentals of statistical signal processing: estimation theory.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[30] Y. Blom, H.A.P. Bar-Shalom, “The interacting multiple model algorithm for systems with markovian switching coefficients,” *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 33, pp. 780–783, Aug. 1988.

[31] J. Wang, D. Zhao, W. Gao, and S. Shan, “Interacting multiple model particle filter to adaptive visual tracking,” in *ICIG ’04: Proceedings of the Third International Conference on Image and Graphics (ICIG’04)*, (Washington, DC, USA), pp. 568–571, IEEE Computer Society, 2004.

[32] J. Boers, Y. Driessen, “Interacting multiple model particle filter,” in *(1990 - 1994) Radar and Signal Processing, IEE Proceedings F*, vol. 150, pp. 344–349, 2003.

[33] A. H. C. K. K. Kastella, “Multiple model particle filtering for multi-target tracking,”

[34] I. Matthews and S. Baker, “Active appearance models revisited,” *International Journal of Computer Vision*, vol. 60, pp. 135–164, Nov. 2004.

[35] R. Gross, I. Matthews, and S. Baker, “Constructing and fitting active appearance models with occlusion,” in *Face Processing in Video*, p. 72, 2004.

[36] F. D. la Torre and M. J. Black, “A framework for robust subspace learning,” *International Journal of Computer Vision*, vol. 54, no. 1-3, pp. 117–142, 2003.

[37] R. X. Li and V. P. Jilkov, “Survey of maneuvering target tracking: I. dynamic models,” vol. 4473, SPIE, 2001.

[38] F. Dornaika and F. Davoine, “Simultaneous facial action tracking and expression recognition using a particle filter,” in *ICCV*, pp. 1733–1738, IEEE Computer Society, 2005.

[39] Z. Zeng and S. Ma, “Head tracking by active particle filtering,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 82–87, 2002.

[40] Wikipedia, “Chapman-kolmogorov equation — wikipedia, the free encyclopedia,” 2006. [Online; accessed 14-November-2006].

[41] Wikipedia, “Markov chain — wikipedia, the free encyclopedia,” 2006. [Online; accessed 14-November-2006].

[42] Wikipedia, “Nondeterministic algorithm — wikipedia, the free encyclopedia,” 2006. [Online; accessed 14-November-2006].

[43] Wikipedia, “Particle filter — wikipedia, the free encyclopedia,” 2006. [Online; accessed 27-June-2006].

[44] Wikipedia, “Importance sampling — wikipedia, the free encyclopedia,” 2006. [Online; accessed 14-November-2006].