OCCLUSIONS AND ACTIVE APPEARANCE MODELS

McElory Hoffmann, Ben Herbst and Karin Hunter
Applied Mathematics,University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa

Keywords: Model-based Object tracking in Image Sequences, Motion and Tracking, Active Appearance Models, Active Contours, Particle Filters.

Abstract: The deterministic active appearance model (AAM) tracker fails to track objects under occlusion. In this paper, we discuss two approaches to improve this tracker 's robustness and tracking results. The first approach initialises the AAM tracker with a shape estimate obtained from an active contour, incorporating shape history into the tracker. The second approach combines AAMs and the particle filter, consequently employing both shape and texture history into the tracker. For each approach, a simple occlusion detection method is suggested, enabling us to address occlusion.

1 INTRODUCTION

Active appearance models (AAMs) (Cootes et al., 1998) provide an elegant model-based framework for tracking objects. They incorporate both shape and texture (grayscale or colour information) into their formulation, and are therefore able to track the outline of an object and its appearance simultaneously. It is therefore easy to use the parameters provided by an AAM tracker in applications such as lipreading (Matthews et al., 2002). For this reason, we are interested in ways to robustly track with AAMs.

Stegmann (Stegmann, 2001) demonstrated that AAMs can be successfully applied to perform object tracking. In his deterministic approach, the AAM search algorithm is applied successively to each frame. However, this technique is not robust since the optimisation techniques employed in the AAM search algorithm only explore a small local region of interest. Also, no history of the object's movement and position is used to improve the optimisation. Therefore the tracker fails in certain situations, e.g. when the object moves fast or in the presence of occlusion, see Figure 1. This is because fast movements lead to a bad initialisation of the AAM optimisation routines and occlusion provides no local optima. One can modify the AAM itself to increase robustness, e.g.

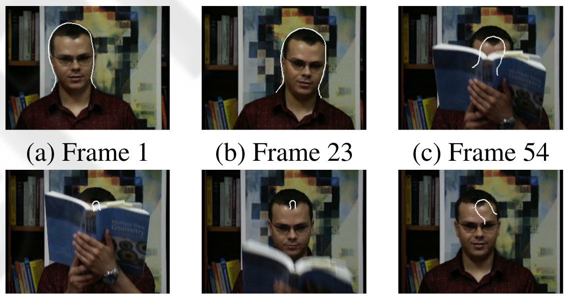

(a) Frame 1 (b) Frame 23 (c) Frame 54
(d) Frame 62 (e) Frame 67 (f) Frame 115

Figure 1: Selection of result frames illustrating the performance of the deterministic AAM tracker. In (c) and (d) the tracker loses its target due to the presence of occlusion and is not able to recover the target as shown in (e) and (f).

in (Gross et al., 2006) the AAM is modified to handle occlusions. In this paper, we describe easily implemented techniques as add-ons to the standard AAM tracker.

Building on work done in (Hoffmann et al., 2006), two improvements of the deterministic AAM tracker are discussed here, both intended to increase the robustness of the tracker. The first technique, presented in Section 3, initialises the AAM using the shape estimate obtained from active contours. In this way, the

AAM search algorithm is restricted to a local region of interest around the estimate from active contours. The second technique, presented in Section 4, uses a combination of a particle filter and an AAM to provide more robustness. Here temporal filtering predicts the parameters of the AAM so that the history of the object's movement and position enhances the AAM searches. Simple adaptions to handle occlusions are suggested for each combined tracker.

Since AMMs and particle filters are central to both techniques, we summarise these concepts in the next section, before presenting the two tracking techniques in Sections 3 and 4. Experimental results of both techniques are shown in Section 5 and we conclude in Section 6.

2 BACKGROUND

Here we give a brief overview of AAMs and particle filters.

2.1 Active Appearance Models (AAMs)

Active appearance models (Cootes et al., 1998; Stegmann, 2000) are template based and employ both shape and texture (colour information). Setting up the model involves training the model parameters, as discussed next.

In the training phase of an AAM, the features on the outline of the object are recorded and then normalised with respect to translation, scale and rotation. This normalisation is done by setting up a common coordinate frame, the features are then aligned to this normalised frame, and the translation, scale and rotation parameters are recorded in a vector called the *pose* $\mathbf{p}$. Using the pose and the normalised coordinates, features in normalised coordinates can be translated, scaled and rotated to their original version in absolute coordinates. Then the *shape* is defined by the vector of features in normalised coordinates, $\mathbf{s}_i = [x_{i1}, \cdots, x_{in}, y_{i1}, \cdots, y_{in}]^T$, $i = 1, \ldots, l$ where l is the number of training images and n the number of features.

The *texture* of the object in question is described by a vector, $\mathbf{g}_i = [g_{i1}, g_{i2}, \ldots, g_{im}]^T$, $i = 1, \ldots, l$ with m the number of texture points. Typically a piecewise affine warp using a Delaunay triangulation of the shape vector is performed and underlying pixel values are then sampled, see for example (Stegmann, 2000).

Principal component analysis (PCA) is performed on the aligned shapes and textures and this yields

$$\mathbf{s} = \bar{\mathbf{s}} + \Phi_s \mathbf{b}_s \quad \mathbf{g} = \bar{\mathbf{g}} + \Phi_g \mathbf{b}_g \qquad (1)$$

where $\bar{\mathbf{s}}$ and $\bar{\mathbf{g}}$ are the average shape and texture vectors, Φ_s and Φ_g are the eigenvector matrices of the shape and texture covariance matrices, and $\mathbf{b}_s$ and $\mathbf{b}_g$ are the PCA projection coefficients, respectively.

A combined model parameter $\mathbf{c}$ is obtained by combining the PCA scores into $\mathbf{b} = [\Psi_s \mathbf{b}_s,\ \mathbf{b}_g]^T$ with Ψ_s a weighting matrix between pixel intensities and pixel distances. A third PCA is executed on the combined model parameter to obtain $\mathbf{b} = \Phi_c \mathbf{c}$, where Φ_c is the basis for the combined model parameter space. Writing $\Phi_c = \left[\Phi_{c,s}, \Phi_{c,g}\right]^T$, it is now possible to generate new shapes and texture instances by

$$\mathbf{s} = \bar{\mathbf{s}} + \Phi_s \Psi_s^{-1} \Phi_{c,s} \mathbf{c} \qquad (2)$$

$$\mathbf{g} = \bar{\mathbf{g}} + \Phi_g \Phi_{c,g} \mathbf{c}. \qquad (3)$$

This concludes the training phase of the AAM.

In the search phase of AAMs, the model parameter $\mathbf{c}$ and the pose $\mathbf{p}$ are sought that best represent an object in a new image not contained in the original training set. Notice that from (2) and (3) changing $\mathbf{c}$ varies both the shape $\mathbf{s}$ and the texture $\mathbf{g}$ of an object. The idea is to vary $\mathbf{c}$ (optimise over $\mathbf{c}$) so that the shape and texture generated by (2) and (3) fit the object in the image as well as possible. The objective function that is minimised is the difference

$$E = ||\mathbf{g}_{model} - \mathbf{g}_{image}||^2 = ||\delta\mathbf{g}||^2 \qquad (4)$$

between the texture values generated by $\mathbf{c}$ and (3), denoted as $\mathbf{g}_{model}$, and the texture values in the image, $\mathbf{g}_{image}$. Note that the image texture values $\mathbf{g}_{image}$ for a specific value of $\mathbf{c}$ are the values sampled from the shape generated by (2) and then translated, scaled and rotated using the pose $\mathbf{p}$. In summary, the optimisation over $\mathbf{c}$ and $\mathbf{p}$ minimises (4), i.e. produces the best fit of texture values.

In the implementation of AAMs one assumes that there exists a linear relationship between the differences in texture values, $\delta\mathbf{g} = \mathbf{g}_{model} - \mathbf{g}_{image}$ and the optimum model parameters' updates, hence

$$\delta\mathbf{p} = R_p \delta\mathbf{g} \qquad (5)$$

$$\delta\mathbf{c} = R_c \delta\mathbf{g} \qquad (6)$$

where R_p and R_c are estimated during the training phase. The last step is to fine-tune the parameters $\mathbf{p}$ and $\mathbf{c}$ by gradient-descend optimisation strategies.

The optimisation strategy described above, requires a good initialisation for the following reasons:

- The shape generated by $\mathbf{c}$ and (2) is translated, scaled and rotated using $\mathbf{p}$—a large space to search.
- The assumption that there exists a linear relationship between the differences in texture values and the optimum model parameters' updates is only reasonable for small updates.

We conclude that AAMs provide a general framework to track or segment different types of objects. Furthermore, no parameters need to be specified by an expert to use them. On the downside, AAMs require objects to have distinct features/outlines and there is a training phase involved. Also, a good initialisation is required for the search algorithm.

2.2 The Particle Filter

Particle filters have become an important tool to track objects. Following (Isard and Blake, 1998; Ristic et al., 2004), we let the state vector $\mathbf{x}_t \in \mathbb{R}^{n_x}$ describe the object to be tracked at time step t, while the measurements are given by $\mathbf{z}_t \in \mathbb{R}^{n_z}$. We denote all the measurements up until the tth time step by $\mathbf{Z}_t \triangleq \{\mathbf{z}_i, i = 1, \ldots, t\}$. In the Bayesian framework, the goal is to find an estimate of $\mathbf{x}_t$ based on all the observations $\mathbf{Z}_t$. Thus the conceptual Bayes solution recursively updates the posterior pdf

$$p\left(\mathbf{x}_t|\mathbf{Z}_t\right) = \frac{p\left(\mathbf{z}_t|\mathbf{x}_t, \mathbf{Z}_{t-1}\right) p\left(\mathbf{x}_t|\mathbf{Z}_{t-1}\right)}{p\left(\mathbf{z}_t|\mathbf{Z}_{t-1}\right)} \tag{7}$$

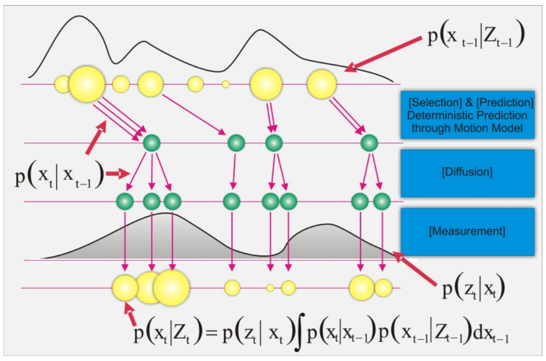
as the measurements become available. The essential idea of the particle filter is to approximate the posterior pdf in (7) by a set of samples $\{\mathbf{x}_t^i, i = 1 \ldots N\}$ and associated weights $\{\pi_t^i, i = 1 \ldots N\}$. We call these samples *particles*.

The particle filter algorithm consists of three phases that are repeated at each time step and are illustrated in Figure 2.

- The *selection* phase chooses the next set of particles according to the input set's relative probabilities, i.e. particles with larger weights will be chosen several times, and those with smaller weights will be selected fewer times or discarded.
- During the *prediction* phase the new set of particles is put through the process model to generate a prior pdf $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. To account for process noise, the particles are *diffused* by adding noise to them.
- The *measurement* phase updates the weights using the new measurement $\mathbf{z}_t$

$$\pi_t^i = \frac{p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{x}_t^i)}{\sum_{n=1}^{N} p(\mathbf{z}_t|\mathbf{x}_t = x_t^n)}. \tag{8}$$

Note that no assumption is made on the linearity of the system and the underlying model is not assumed to be Gaussian; hence particle filters can be viewed as a generalisation of the Kalman filter.


$p(x_{t-1}|Z_{t-1})$
[Selection] & [Prediction] Deterministic Prediction through Motion Model
$p(x_t|x_{t-1})$
[Diffusion]
[Measurement]
$p(z_t|x_t)$
$p(x_t|Z_t) = p(z_t|x_t)\int p(x_t|x_{t-1})p(x_{t-1}|Z_{t-1})dx_{t-1}$
Figure 2: A graphical illustration of one iteration of the particle filter. (Image courtesy of S.Fleck (Fleck et al., 2005)).

3 ACTIVE CONTOURS AND ACTIVE APPEARANCE MODELS

The combined active contour and active appearance model (AC-AAM) proposed by (Sung and Kim, 2006) finds a shape to initialise the AAM algorithm using active contours. This method rectifies the situation in which the AAM only works well locally. A summary of this technique is presented below.

3.1 Contours and Shape Space

Here B-splines are used to model the outline of the tracked object (Blake and Isard, 1998).

Given a set of coordinates of control points $(x_1, y_1), \ldots, (x_n, y_n)$, a B-spline is the curve $\mathbf{r}(s) = [x(s),\ y(s)]^T$ formed by a parametrisation with parameter s on the real line,

$$\mathbf{r}(s) = \begin{bmatrix} \mathbf{B}(s)^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}(s)^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}^x \\ \mathbf{Q}^y \end{bmatrix} \tag{9}$$

where $\mathbf{B}(s)$ is the $n \times 1$ vector of B-spline basis functions, and $\mathbf{Q}^x$, $\mathbf{Q}^y$ are the vectors of control points consisting of the x-coordinates and y-coordinates respectively. We refer to a curve $\mathbf{r}(s)$ in (9) as a *contour*.

The dimension of the vector space spanned by $\mathbf{r}(s)$ is $N_Q = 2N_B$, where N_B is the number of control points of the B-spline. This implies N_Q degrees of freedom and it means that the object can deform in N_Q different ways if we track it over successive frames. This amount of deformation leads to many tracking errors, therefore the deformation is restricted to a lower dimensional space, known as the shape space.

The shape space is defined as a linear mapping from a shape vector $\mathbf{X} \in R^{N_x}$ to a spline vector $\mathbf{Q} = \left[\mathbf{Q}^x,\ \mathbf{Q}^y\right]^T \in \mathbb{R}^{N_Q}$ and the mapping is given by

$$\mathbf{Q} = W\mathbf{X} + \mathbf{Q}_0 \tag{10}$$

where W is a matrix with rank $N_X \ll N_Q$, describing the allowed transformations. By restricting $\mathbf{X}$, $\mathbf{Q}$ is essentially a deformation of the template $\mathbf{Q}_0$, and the type of deformation allowed is determined by W.

3.2 Active Contours (ACs)

Following (Isard and Blake, 1998), we summarise the contour based particle filter. We refer to this combination as *active contours* (ACs). Adapting a particle filter for a particular implementation, requires the specification of a state vector, process model and measurement model.

3.2.1 The State Vector

The state vector at each time step t is given by the shape space vector, thus $\mathbf{x}_t = \mathbf{X}_t$. This allow us to generate a vector of B-spline control points $\mathbf{Q}$ for each particle using (10).

3.2.2 The Process Model

States evolve according to a simple random walk given by

$$\mathbf{x}_t^{(i)} = \mathbf{x}_{t-1}^{(i)} + S_t^{(i)}\mathbf{u}_t^{(i)} \qquad (11)$$

where $S_t^{(i)}$ is the process noise covariance and $\mathbf{u}_t^{(i)}$ is a vector of normal distributed random variables. One can use more sophisticated process models and the reader is referred to (Blake and Isard, 1998) for a detailed discussion.

3.2.3 The Measurement Model

The binary edge map for the current frame is passed to the algorithm to estimate the weight associated with each particle. To calculate the weight for the ith particle, the following procedure is followed:

- Using (10) and the state vector, calculate the vector of control points $\mathbf{Q}^i$.
- Calculate the normal lines for each control point.
- For each control point, search along the normal line until an edge is found. Let the distance from the control points to the edge be $d_j, j = 1, \ldots, n$ where n is the number of control points. If an edge is not found, set the distance equal to the length of the normal line. Calculate the total length $d^i = \sum_{j=1}^{n} d_j$.
- The weight for the ith particle is then given by

$$\pi_i = \exp\left(-\frac{(d^i)^2}{\sigma^2}\right). \qquad (12)$$

The weights are normalised afterwards to sum to unity. From equation (12), we see that a small value of d^i will result in a larger value of π_i and vice versa. The variance σ determines how much preference we give to particles with a lower distance d^i.

We need to find a representative estimate $\mathbf{x}_t^e$ from the many hypothesis curves, typically the particle with the highest weight or a weighted average is chosen.

Furthermore, every time an edge is not found for a particular particle, it is recorded and denoted by n_d^i. If $\sum_{i=1}^{N} n_d^i$ is larger than a certain pre-set threshold, occlusion is declared. This method for the detection of occlusion provides adequate results, for more sophisticated techniques see e.g. (MacCormick, 2002).

3.3 Initialisation of AAM with AC

The AC-AAM tracker consists of two parts. At time step t, the first part performs standard AC tracking with the estimate from the tracker denoted by $\mathbf{x}_t^e$. In the second part, (10) is used together with $\mathbf{x}_t^e$ to generate a shape estimate

$$\mathbf{Q}_t^e = W\mathbf{x}_t^e + \mathbf{Q}_0. \qquad (13)$$

Note that $\mathbf{Q}_t^e$ and the AAM shape representation $\mathbf{s}$ (not normalised with respect to the pose) are equivalent. This shape $\mathbf{Q}_t^e$ is then used to initialise standard AAM optimisation and the result is output as the best fitted AAM. The result of the AAM is then used to initialise the AC tracker again.

In the presence of occlusion the AAM optimisation fails since there is no optimum. Therefore, when occlusion is detected by the AC part, the tracker will output no estimate for the resulting AAM, i.e. the AAM is "switched off".

Note that this technique uses the particle filter indirectly. The particle filter is an integral part of the AC tracker, but the AAM part of the AC-AAM tracker does not utilise the particle filter.

4 AN ACTIVE APPEARANCE MODEL BASED PARTICLE FILTER

This section details the second approach of increasing the robustness of the AAM tracker. It differs from the AC-AAM technique in the sense that the AAM is not initialised by a secondary technique, instead temporal filtering predicts the parameters of the AAM. Consequently, the history of the object is used to improve the overall robustness of the AAM. The technique is an extension of the direct combination of an AAM with a particle filter as was introduced by (Hamlaoui and Davoine, 2005).

As in the case of the AC tracker, the adaption of the particle filter to work in conjunction with an AAM requires the specification of a state vector, a process model and a measurement model.

4.1 The State Vector

It is clear, from (2) and (3) that one can synthesise a shape $\mathbf{s}$ and texture $\mathbf{g}$ for a particular image from the model parameters $\mathbf{c}$. Therefore, the state vector is a combination of the model parameters $\mathbf{c}$ and the pose $\mathbf{p}$ and at time step t it is given by $\mathbf{x}_t = \begin{bmatrix}\mathbf{p}, & \mathbf{c}\end{bmatrix}^T$.

4.2 The Process Model

Using the update of the model parameters (5) and (6), the states evolves according to

$$\mathbf{x}_t^{(i)} = \hat{\mathbf{x}}_{t-1} + \begin{bmatrix} R_p \\ R_c \end{bmatrix} \delta\mathbf{g}_{t-1} + S_t^{(i)}\mathbf{u}^{(i)} \qquad (14)$$

where $\hat{\mathbf{x}}_{t-1}$ is the estimate of the state vector at time step $t-1$, $S_t^{(i)}$ is the process noise covariance and $\mathbf{u}^{(i)}$ is a vector of normally distributed random variables. However, if occlusion occurs, (14) is not a good representation for the process model due to the inclusion of the AAM optimisation given by $\begin{bmatrix}R_p, & R_c\end{bmatrix}^T \delta\mathbf{g}_{t-1}$. Hence, if occlusion is detected (discussed in the next section), the simpler process model

$$\mathbf{x}_t^{(i)} = \mathbf{x}_{t-1}^{(i)} + S_t^{(i)}\mathbf{u}^{(i)} \qquad (15)$$

is used, where $\mathbf{x}_{t-1}^{(i)}$ is the ith particle at the time $t-1$ and the other parameters remain unchanged.

4.3 The Measurement Model

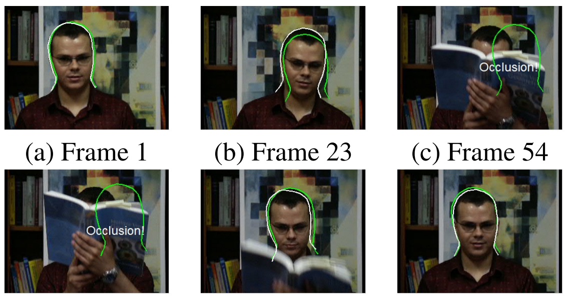
Since the purpose of the measurement model is to classify how well a particular particle fits the underlying image, the error (4) is measured. However, to account for the outliers we use the Lorentzian norm

$$\rho_i = \rho\left(E_i, \sigma_s\right) = \log\left(1 + \frac{E_i}{2\sigma_s^2}\right) \qquad (16)$$

where σ_s is the scale parameter that discards outliers. Then the pre-normalised weights are given by

$$\pi_t^i = \exp\left(-\frac{\rho_i}{\sigma^2}\right) \qquad (17)$$

where σ plays the same role as in the case with the AC tracker. Furthermore, the number of times n_ρ the error measurement ρ_i exceeds a pre-set threshold is recorded. When n_ρ is larger than half the number of particles, occlusion is declared.



(a) Frame 1 (b) Frame 23 (c) Frame 54
(d) Frame 62 (e) Frame 67 (f) Frame 115

Figure 3: Selection of result frames to indicate the performance of the AC-AAM tracker. The green and white shapes are the output of active contours and AC-AAM respectively. Note that the AC-AAM provides no AAM when the object undergoes occlusion in (c) and (d).

5 EXPERIMENTAL RESULTS

We implemented the AAM-based particle filter tracker in C++ using the open-source AAM-API (Stegmann et al., 2003), while the AC implementation is in MATLAB. To illustrate the effectiveness of these trackers, they are used to track a head and shoulders moving against a cluttered background. The videos containing the full movement are available for download from (Hoffmann, 2006). The appearance model consists of 14 feature points and the AAM was trained using 6 images.

In Figure 3 the results obtained with the AC-AAM tracker are shown, while the corresponding results obtained with the AAM-based particle filter are illustrated in Figure 4. Both trackers are able to track the movement of the object accurately.

When the AC-AAM approach is used, a simple implementation for the AC tracker suffices, i.e. a basic dynamic model and a simplified shape space. This can be seen in Figure 3 where the output of the AC tracker is not as accurate as it could be, but the resulting AAM fit is good. This illustrates the principal idea of the AC-AAM approach: the robustness of the AC tracker is used to initialise the AAM which in turn, adjusts well to the underlying image. The AC-AAM tracker detects occlusion as can be seen in Figure 3(c) and (d), and the AAM optimisation is disabled for these frames.

The AAM-based particle filter successfully tracks the head and shoulders in the presence of occlusion as illustrasted in Figure 4(c) and (d). It could be argued that this tracker handles occlusion more successfully than the AC-AAM tracker, since it provides an estimate for all frames (also those with occlusion) and the estimate is consistent with our intuition.