Visual tracking: a research roadmap

Andrew Blake¹, Microsoft Research.

ABSTRACT. A research roadmap to many of the best known, and most used, contributions to visual tracking is set out. The scope includes simple appearance models, active contours, spatiotemporal filtering and briefly points to important further topics in tracking.

1 Introduction

Visual tracking is the repeated localisation of instances of a particular object, or class of objects, in successive frames of a video sequence. Video analysis may be causal or non-causal, but tracking is usually taken to be an online process, and therefore causal with some emphasis on efficient algorithms. The question of automatic initialisation, though sometimes important, is not addressed here. This is sensible in that there are plentiful applications where initialisation is not an issue, such as tracking vehicles on a highway, or indoor surveillance, in which initialisation can be effected by a simple motion trigger. The aim is to achieve location estimates at least as good as independent, exhaustive examinations of each frame [29]. Exploitation of object dynamics offers improved computational efficiency and more refined motion estimates. Perhaps most important of all, it offers extended capability to resolve ambiguity, as with a person in a crowd or a leaf on a bush (figure 1).

2 Simple appearance models

2.1 Simple patches

The most basic tracker consists of matching a template patch \( T(r), r \in \mathcal{T} \) onto an image \( I(r) \) under translation [40] by cross correlation. The aim is to minimise the misregistration error

\[
\rho = \sum_{r \in \mathcal{T}} [I(r) - T(r + u)]^2
\]  

and this can be done to subpixel resolution using an estimate of the gradient \( g(r) = \nabla I(r) \), computed using a suitable filter (such as a gradient of

¹www.research.microsoft.com/~ablake
Gaussian filter). Then the iterative registration algorithm alternates two steps, to convergence:

1. Newton step on $\rho$

   $$\dot{\rho} = \sum_i \left( g_i \cdot g_i^T \right)^{-1} \sum_n g_i \delta_i$$

2. Recompute template offset

   $$u \rightarrow u - \dot{\rho}$$

More generally, the class of transformations can be generalised from translation $x \rightarrow x + u$ to a larger class $x \rightarrow W_\mu(x)$ in which $\gamma$ are the parameters of, for example, an affine transformation or a non-rigid spline mapping [10] — see later for more details of these transformations. Taking $\mu = \mu_0 + \delta \mu$ and linearising gives

$$I(x) \approx T(W(x, \mu_0)) + \delta \mu \cdot \frac{\partial W}{\partial \mu} \nabla T,$$

which can be solved iteratively for $\mu$, to perform generalised registration [40, 3, 24].

### 2.2 Blobs

An alternative approach to localising regions is to model only the gross properties of a region, modelling it as a “blob” [56], a Gaussian mixture model (GMM) in a joint ($r, I$) position and colour space. Thus a pixel $I(r)$ is modelled probabilistically as belonging to a model $M$ with probability

FIGURE 1. Tracking in camouflage. The trail of tracked positions of a moving leaf, in heavy camouflage, at two different times in a sequence. For details of the method see section 4. Images reprinted from [8]. For related movies see robots.ox.ac.uk/~vdg/dynamics.html.
1. Visual tracking: a research roadmap

$p(r, I(r) \mid M)$ and in a new test image, each pixel is evaluated against each of a number of models $M \in M$. The model with the greatest likelihood is assigned to the pixel. The cluster of pixels with label $M$ is deemed to be the new position of object $M$, whose moments (mean etc.) can be computed to represent the location of object $M$, and the GMM for $M$ can also be updated periodically.

Recently a variation on the blob idea, “mean-shift” tracking [12] has been very influential because it allows progressive updating of object position without the obligation to visit all pixels of each and every frame. Successive approximations to the estimated locations of an object are obtained iteratively as:

$$\hat{f}_t = \frac{1}{C} \sum_{r \in \mathcal{T}} r \cdot w(r) g(||r - \hat{f}_{t-1}||^2)$$

(1.3)

where $C = \sum_{r \in \mathcal{T}} \sqrt{w(r)g(||r - \hat{f}_{t-1}||^2)}$, $g$ is the derivative of a particular kernel function used to build spatial density functions, and $w(r)$ is a weight measuring the degree of prevalence of the color of pixel $r$ in the template relative to its prevalence in the test object. The result, used over an image sequence, is a remarkably tenacious tracker (figure 2), despite its simplicity.

![Mean shift tracking](image.png)

**FIGURE 2. Mean shift tracking** A mean-shift tracker, (here in a particle filter form — see later) is used here to track player no 75 in a primitive form of sport. Image reprinted from [42].

2.2 Background maintenance

Blobs represent foreground objects as distributions over colour (and space) but modelling a background, assuming it is largely static, is also useful as a guide to what is not part of an object [41]. Just as blobs model the foreground as a mixture, so also modelling background pixels as mixture distributions is useful [46, 48]. If $M_b$ is the background model, then pixels could be tested for their likelihood of belonging to the background in general by evaluating $p(I \mid M_b)$, and high scoring pixels removed from consideration as possible parts of any foreground object. What is more powerful still, when the background is static, is to model each background pixel individually
by collecting statistics of colour over time from that pixel, and building a
mixture model for \( p(I \mid r, M_0) \). These form typically narrow distributions
which make powerful tests for background membership.

Having introduced some simple, though nonetheless very effective forms
of tracker, the next section looks at some elaborations on the basic theme of
matching shapes.

3 Active contours

An active contour is a parameterised curve \( r(s), 0 \leq s \leq 1 \) in the plane
that is set up to be attracted to features in an image \( I(r) \). A detailed
account of the development and mechanisms of active contours is given
globally [8], but here we summarise the main types. In section 4, explicitly
dynamical forms of active contour \( r(s, t), t \geq 0 \), attracted to an image
sequence \( I(t) \), are outlined. It focuses on the temporal filtering required to
extract information most effectively over a sequence, exploiting fully the
temporal coherence of the moving scene. This section is restricted to the
static case and follows the development of active contours from snakes to
parametric structures and affine contour models.

3.1 Snakes

"Snakes" [36] have been one of the most influential ideas in computer vision.
They were revolutionary in their time because they directed attention away
from bottom up edge detection, an enterprise which had become stuck in a
rut, towards top down, hypothesis driven search for object structures. The
main idea is that the active contour \( r(s) \) is dropped into a potential energy
field \( F(r) \) which is itself a function of the image intensity landscape. For
example \( F(r) = -[\nabla I] \) would generate an attraction of the snake towards
high image contrast. An equilibrium configuration of the snake satisfies an
(Euler-Lagrange) equation

\[
\left( \frac{\partial (w_1 r)}{\partial s} - \frac{\partial^2 (w_2 r)}{\partial s^2} \right)_{\text{internal forces}} + \nabla E_{\text{ext}} = 0. \tag{1.4}
\]

in which internal force parameters can be adjusted to give the curve a
tendency towards smooth shapes. Such a system can be converted to a
numerical scheme, for example using finite differences along a fine polygonal
approximation to the curve \( r(s) \), with typically hundreds of variables
corresponding to the polygon vertices \( q_i, i = 1, \ldots, M \). Equilibria are
then sought by iterative solving. Alternatively direct solution by dynamic
programming [1] is also possible, with the added attraction that hard con-
straints can be incorporated easily.
So far the snake is defined with respect to a single image \( I(r) \) but for shape tracking, its behavior over an image sequence \( I(r,t) \) must be defined. This can be expressed as a Lagrangian dynamical system [51,15] with distributed mass and viscosity, whose equations of motion could typically take the following form

\[
\rho \ddot{r}_t = - \left( \gamma r_t - \frac{\partial (w_1 r)}{\partial s} + \frac{\partial^2 (w_2 r)}{\partial s^2} \right) + \nabla F \tag{1.5}
\]

in which the additional parameters \( \gamma \) and \( \rho \) respectively govern viscosity of the medium and distributed mass along the contour.

Of course this leaves questions about how to choose parameters \( w_1, w_2, \gamma, \rho \), which may be spatial functions, not just constants, unanswered. This is a problem that can be addressed effectively in a rather different framework, that of probabilistic temporal filtering (see section 4). This idea was first cast [51] in a space of state vectors consisting of vertices of the snake polygon \( \{\lambda_i\} \). Practical implementation however, demands a much lower dimensional state space, not just for computational economy but for stability [7], and this is elaborated in section 4.

### 3.2 Parametric structures

If a lower dimensional state space is essential for stable tracking, one way to construct such a state space is in terms of a state vector \( X = (\lambda_1, \ldots, \lambda_K) \) whose components are physical degrees of freedom in the underlying object, representing a contour (or set of contours) \( r(s, X) \), \( s \in [0,1] \). For example \( X \) could encode the position and orientation of a rigid object. Then the image locations \( r(s_i, X) \), \( i = 1, \ldots, M \) of \( M \) distinguished features on
the curve (for example vertices of a polyhedral object) can be predicted, and compared with observed locations $r_f(s_i)$. In principle $X$ can then be estimated by minimising an error measure such as

$$E = \sum_{i=1}^{M} ||r(s_i, X) - r_f(s_i)||^2.$$  

(1.6)

To include the possibility that the model contains vertices or multiple disconnected segments, $r(s; X)$, $s \in [0, 1]$ need not be everywhere smooth, and may be discontinuous at a finite set of points along $s \in [0, 1]$.

A simple and highly effective example applies to the view of a road from a camera mounted forward-looking on a car, for navigation purposes [16]. In that case $X$ encodes the offset and orientation of the car on the road, and the observations are the road edges. Such a system resulted in the first autonomous, vision guided automobile to travel at realistic speeds on the open road. Other prominent examples of the parametric approach include real-time tracking of complex 3D wire-frame structures [27] and a hinged box [39], in which the prediction function $r(s; X)$ applies perspective projection to map a canonical structure, in state $X$, onto the image plane. The state vector $X$ can also incorporate further parameters which allow adjustment of the underlying canonical structure, in addition to position and orientation, allowing tracking of any object from a given family of objects. This was successful for example with tracking automobiles in overhead views of the highway [37], in which the pose of the vehicle and also variations in automobile shape were encoded together in the state vector $X$.

3.3 Affine contours

Another natural way to construct a low-dimensional state space for tracking is to specify parameters relating directly to image-based shape of the active contour. This is especially appealing because, as we will see, the contour $r(s; X)$ can then often be expressed as a linear function of $X$ and this considerably simplifies the task of curve fitting and (later) of temporal filtering [7]. One natural choice is the planar affine space in which $r(s; X)$ sweeps out the space of 2D affine transformations of a base shape $r(s)$:

$$r(s; X) = AF(s) + u$$

(1.7)

where $A$ is a $2 \times 2$ matrix and $u$ is a $2 \times 1$ vector. It is natural because it is known to span the space of outlines of a planar shape, in an arbitrary 3D pose, and viewed under affine projection (the approximation to image projection that holds when perspective effects are not too strong). It is linear because we can choose $X = (A, u)$ so that $r(s; X)$ is linear in $X$, and this linear relation is denoted

$$r(s; X) = H(s)X,$$

(1.8)
where \( H(s) \) is a simple (linear) function of \( \mathbf{r}(s) \). For non-planar 3D outlines, still under affine projection, there is a linear parameterisation of the form \( X = (A, u, v) \) (see \cite{8} for details) where \( v \) is another vector, so the dimensionality of \( X \) increases from 6 to 8. Of course the underlying dimensionality of the space is still 6 — three parameters for 3D translation and 3 for rotation — and the additional 2 are the price of insisting on a linear parameterisation.

Having defined the linear parameterisation \( \mathbf{r}(s; X) \) of image curves, a curve can now be fitted to a particular set of image data. Suppose the data itself is a curve \( \mathbf{r}_f(s) \), then the least squares fit, the curve \( \mathbf{r}(s; \hat{X}) \) minimising

\[
\int |\mathbf{r}(s; X) - \mathbf{r}_f(s)|^2 \, ds,
\]

is given simply by

\[
\hat{X} = \mathcal{H}^{-1} \int H^\top(s) \mathbf{r}_f(s) \, ds \quad \text{where} \quad \mathcal{H} = \int H^\top(s) H(s) \, ds,
\]

provided the solution is unique. For better stability, regularisation on \( \mathbf{r}(s; X) \) can also be introduced. The integrals in (1.10) have to be computed finitely in practice, and this can be achieved by a using finite parameterisation of the base curve \( \mathbf{r}(s) \) (and therefore also of \( H(s) \)): for example \( \mathbf{r}(s) \) can be modelled as a B-spline \cite{7, 8} or simply as a polygon \cite{14}.

There remains one important issue. The fitting scheme above is correct only if correspondence between the curves is known — that is, for any given value of \( s \), the point \( \mathbf{r}(s; X) \) in the plane is supposed to correspond to the point \( \mathbf{r}_f(s) \) on the data curve. In practice, of course, this is not the case: \( \mathbf{r}_f(s) \) may be parameterised quite differently from \( \mathbf{r}(s; X) \) so that in principle one should fit \( \mathbf{r}(s; X) \) to \( \mathbf{r}_f(g(s)) \), for some unknown reparameterisation function \( g \). In the case that the reparameterisation is not too severe, this is dealt with approximately by replacing total displacement in (1.9) by normal displacement \cite{8, Ch. 6}, as in figure 4. Normal displacement is commonly used, for this reason, in tracking systems \cite{14, 14}.

For full details on curve fitting, regularisation, recursive fitting and normal displacement see \cite{8, Ch. 6}.

3.4 Nonrigidity

Nonrigid motions fall outside the affine families described above, but may still be captured by a suitable space of shapes. The widely used “Active Shape Model” (ASM) \cite{14} does this by analysing a training set of contours, and constructing an eigen-space of shape by Principal Components Analysis (PCA). Initially the high-dimensional parameterisation \( X = (q_i, i = 1, \ldots, V) \) of polygon vertices is chosen. Then the training set \( \{\mathbf{r}_1(s), \ldots, \mathbf{r}_{N_v}(s)\} \) of curves is encoded in terms of its polygon-vertex
representation $X_1,\ldots,X_{N_x}$. Now the sample covariance matrix $\Sigma$ of the $X_1,\ldots,X_{N_x}$ is computed and, as usual in PCA, its dominant eigenvectors are retained, and form a compact basis for curve shape. Components in this basis form a new, low-dimensional curve parameter $X$ which captures our rigidity. Finally it is possible to combine the rigid and the nonrigid approach by explicitly projecting out the affine variations in the training set $\{r_1(s),\ldots,r_{N_y}(s)\}$ of curves, and using PCA to account only for the remaining nonrigid variability. In this way the curve parameter $X$ contains both affine components and, separately, components for nonrigid deformation as in figure 5.

FIGURE 5. ASM components The dominant eigenvectors from PCA analysis of a training set of lip shapes, describing the main non-rigid components of motion. Images reprinted from [8].
3.5 Robust curve distances

Simple least squares error measures like (1.9), and its modified counterpart for normal displacement, have no built in robustness to distortions of the data, in particular those caused by occlusion and clutter. The advantage of (1.9) is its tractability, in that it is quadratic and so can be minimised in closed form. "Chamfer matching", which has been used with notable success in pedestrian detection [19], exchanges some tractability for robustness. In place of summing squared-distance (1.9), summing a truncated distance \( \int d_{\epsilon}(r(s; X) - r_f(s')) ds \), where \( d_{\epsilon}(x) = \min(|x|, \epsilon) \), is more tolerant to outliers. Furthermore, the ideal of minimising over possible parameterisations, previously approximated by normal displacements, can be fully restored to give an asymmetric distance

\[
\rho = \int \min_{s'} d_{\epsilon}(r(s; X) - r_f(s')) ds, \tag{1.11}
\]

which can be expressed as

\[
\rho = \int D(r(s; X)) ds, \quad \text{where } D(r) = \min_{s'} d_{\epsilon}(r - r_f(s')). \tag{1.12}
\]

The image \( D(r) \) is the "chamfer image" which can be precomputed for a given observed data curve \( r_f(.) \). In this way, much of the computational load of computing \( \rho \) is compiled, once for all, into the computation of \( D(r) \). Then the marginal cost of multiple evaluations of \( \rho \) for numerous different values of \( X \) is very low, consisting simply of a summation along the curve \( r(s; X) \). This low marginal cost makes up considerably for the lack of closed form minimisation, and can be used to search efficiently over both pose and shape. Further organisation of shapes into a tree structure based on similarity makes matching even more efficient by reducing the number of evaluations of \( \rho \) required, and this has been very successful in matching even articulated shapes [19, 49].

A related distance measure [30], mentioned briefly here as a relative of the chamfer distance, is the Hausdorff distance \( \min_s \max_{s'} |r(s; X) - r_f(s')| \) which is also asymmetric and, in its pure form, not robust. Robustness is dealt with in practice by replacing \( \min_s \), which is frail in that it makes the Hausdorff distance dependent on the distance between two particular points on each of the curves, by a quantile over \( s \).

4 Spatio-temporal filtering

The difference between tracking and localisation is that tracking exploits object dynamics, both for efficiency and for effectiveness.
4.1 Dynamical models

Dynamical models can be more or less elaborate, according to the nature of the motion being modelled. Some motions, for example of vehicles, talking lips or human gait are often quite predictable and it makes sense to model them in some detail [4, 9]. In any case it is natural to think of a class of motions, and a probability distributions over that class, which is very naturally represented as an AutoRegressive process (ARP) on the state vector $X$ at time $t$ (denoted $X_t$). A simple ARP on $X_t$, expressed in terms of a “driving” vector $w_t$ of independent Gaussian noise variables, and constant square matrix $B$, takes the form (first order AR process)

$$X_t = F(X_{t-1}, w_t),$$

with $F$ linear, and some examples follow.

**Tethered:** $X_t = Bw_t$

**Brownian:** $X_t = X_{t-1} + Bw_t$

**Constant velocity:** $X_t = X_{t-1} + Bw_t + v$

**Constrained Brownian:** $X_t = aX_{t-1} + Bw_t$ with $|a| < 1$

**Damped oscillation:** $X_t = a_1X_{t-1} + a_2X_{t-2} + Bw_t$ with appropriate $a_1, a_2$.

The last is, of course, not a first-order AR process, but is 2nd order, of the form $X_t = F(X_{t-1}, X_{t-2}) + w_t$. Details of the expressive power of various AR models, the roles of the various constants, and algorithms for learning them from training data are detailed in [8, Ch. 9]. Of course these are just a few of the possible linear dynamical models. More elaborate models may also be appropriate, and nonlinearity is also powerful for allowing switching between different kinds of motions [33] — effectively mixtures of AR models.

4.2 Kalman filter for point features

Classically, the Kalman filter is the exact computational mechanism for incorporating predictions from an AR model of dynamics into a stream of observations, and in due course this important idea was introduced into machine vision [25, 21, 17]. The most straightforward setting is the tracking of point features, such as polyhedral vertices, used with an affinely deforming image structure [45] (recall section 3.3) or a 3D rigid body structure [26] (as section 3.2). In either case, it is essential to represent explicitly the uncertainty in the observation $r_f(s_i)$ of each point, in terms of independent, two-dimensional standard Gaussian noise vectors $\nu_i$:

$$r_f(s_i) = r(s_i, X) + \sigma_i \nu_i \ i = 1, \ldots, M$$

(1.14)
where $\sigma_i$ is the magnitude of the positional uncertainty associated with the measured image location $r_f(s_i)$ of the $i^{th}$ feature. Measurement uncertainty can then be traded off with uncertainty in the (noise driven) AR predictions to achieve a natural and automatic balance between the influence of observations and of prediction. The result is that an estimate $\hat{X}_t$ of state $X_t$ is propagated in the following manner.

\[
\hat{X}_t = F(\hat{X}_{t-1}, 0).
\] (1.15)

- the ARP prediction equation (1.13) with zero noise.

Each measurement $r_f(s_1, t), \ldots, r_f(s_M, t)$ is assimilated as:

\[
\hat{X}_t \leftarrow \hat{X}_t + K_{i,t} (r_f(s_i, t) - r(s_i, \hat{X}_t)).
\] (1.16)

The “Kalman gains” $K_{i,t}$ are computed by an associated recursion whose details are omitted here, but see e.g. [16].

### 4.3 Kalman filter for contours

Kalman filtering for contour tracking [7] proceeds in a similar fashion as for point-features, but using the idea of normal displacement, introduced in section 3.3 and illustrated here in fig 6. Only the normal component of feature displacement is assimilated, so that step (1.16) above takes instead the form:

\[
\hat{X}_t \leftarrow \hat{X}_t + K_{i,t} [n(s_i, t) \cdot (r_f(s_i, t) - r(s_i, \hat{X}_t))],
\] (1.17)

where $n(s_i, t)$ is the normal to the curve $r(s, \hat{X}_t)$ at the $i^{th}$ sample point $s = s_i$. Unlike the case of point features, where the locations $s = s_i$ are locations...
on the contour of distinguished point features, here the $s = s_i$ are simply a convenient sampling pattern along the length of the contour, implementing a numerical approximation of the mean-square normal displacement.

4.4 Particle filter

The Kalman filter has two limitations that can prove very restrictive in relatively unconstrained tracking problems.

1. **Clutter**: it is limited to one observation $r_f(s_i, t)$ for each contour location $r(s_i, t)$. Clutter in the image tends to generate multiple observations at each location, as figure 7 shows.

![Figure 7. Image clutter disrupts observations](image)

*Active contour and normals are shown. Crosses mark observations of high contrast features, some of which are triggered by the true object outline while others are responding to clutter, both inside and outside the object. Image reprinted from [31].*

2. **Dynamics**: the Kalman filter is limited to ARP models of dynamics. Mild non-linearities can be dealt with, in practice, by local linearisation. Hybrid dynamical models that switch between ARPs (e.g. flight/bouncing/rolling) demand a more powerful mechanism for temporal filtering.

Particle filters are a class of Monte-Carlo temporal filters that are more powerful than the Kalman filter in that they escape both from the restrictions of clutter [31] and dynamics [33], but at the cost of being only approximate. The idea of sampling shapes in cluttered observations derives originally from static studies [23]. The earliest form of the particle filter was the “bootstrap filter” [22]. The more powerful form described here is based [32, 38] on importance sampling.
**Temporal update** for time step $t - 1 \rightarrow t$

From the sample-set $\{X_{t-1}^{(n)}, \pi_{t-1}^{(n)}, n = 1, \ldots, N_S\}$ at time $t - 1$, construct a new sample-set $\{X_t^{(n)}, \pi_t^{(n)}, n = 1, \ldots, N_S\}$ for time $t$, as follows.

1. **Select** samples $X_i^{(n)}$ by sampling from the “proposal distribution” $q_t(X_t | X_{t-1}^{(n)})$.

2. **Weight** the new particles in terms of the vector of measured features $z_t = \{r_f(s_1, t), \ldots, r_f(s_M, t)\}$:

   $$\pi_t^{(n)} = \pi_{t-1}^{(n)} \frac{p(z_t | X_t = X_t^{(n)}) p(X_t = X_t^{(n)} | X_{t-1} = X_{t-1}^{(n)})}{q_t(X_t = X_t^{(n)} | X_{t-1} = X_{t-1}^{(n)})}$$

3. **Resample** at occasional time-steps, to avoid the distribution of weights becoming too uneven:

   (a) Sample, with replacement, from $\{X_t^{(n)}, n = 1, \ldots, N_S\}$, selecting $X_i^{(n)}$ with probability proportional to $\pi_t^{(n)}$, to form a new, resampled set $\{X_t^{(n)}, n = 1, \ldots, N_S\}$.

   (b) Reset all weights to $\pi_t^{(n)} = 1$.

**FIGURE 8 A Particle filter. Standard form of particle filter, following [49]**

The essence of the particle filter is summarised in figure 8. In place of the single estimate $X_t$ in the Kalman filter, particle filters maintain an entire set $\{X_{t-1}^{(n)}, n = 1, \ldots, N_S\}$ of possible estimated values of the state $X_t$. This is a robust approach that allows the explicit representation of ambiguity in a way that a Kalman filter simply cannot. For example in clutter, the ambiguity is generated by uncertainty as to which of many visible features is actually generated by the true object. With hybrid dynamics, the ambiguity reflects uncertainty as to which ARP model currently explains the observed motion; typically ambiguity is heightened around the time that the model switches. The particle set for time $t$ consists of the set of possible values $\{X_t^{(n)}\}$ along with a set of positive weights $\{\pi_t^{(n)}\}$.

The algorithm description explains how the particle set evolves from one timestep to the next. First new values $X_t^{(n)}$ are generated by sampling from a proposal distribution $q_t$. In the simplest **Condensation** [31] or bootstrap [22] forms of the filter,

$$q_t(X_t | X_{t-1}^{(n)}) = p(X_t | X_{t-1} = X_{t-1}^{(n)})$$

— the proposal is simply a simulation of the dynamical model itself. In other words, particles are generated by predicting the change of state from
time-step \( t - 1 \) to timestep \( t \). In the case of ARP dynamics (1.13) this gives

\[ X_t^n = F(X_{t-1}^n, w_t^n), \tag{1.18} \]

where the \( w_t^n, n = 1, \ldots N \) are independent draws of a standard normal variable, thus using the ARP to make noisy predictions of object position. In this way, particles \( X_t^n \) sweep out a set of \textit{a priori} probably values for \( X_t \).

A more adventurous form of proposal distribution uses hints from the image — "importance sampling" — at time \( t \) to generate probable values for \( X_t \).

For example, tracking hands or faces, a "pinkness" measure \( q_t^\text{pink}(X) \) can be used to generate states likely to coincide with skin colouration in the image.

The second step of the algorithm generates the weights \( \pi_t^n \) and in doing so achieves two things: i) it takes account of the new measurements \( r_f(s_t, t) \); and ii) it compensates for any bias in the proposal distribution \( q_t(\cdot) \). Again, the simplest case is the \textit{Condensation} filter, in which \( q_t(\cdot) \) is unbiased, and the formula for weights simplifies to

\[ \pi_t^{(n)} = \pi_{t-1}^{(n)} p(z_t|X_t = X_t^{(n)}). \tag{1.19} \]

A simple example of a measurement process was given earlier (1.14), and in that case the observation likelihood is the Gaussian

\[ p(z|X) \propto \exp - \sum_{i=1}^M \frac{1}{2\sigma_z^2} \| r_f(s_i) - r(s_i, X) \|^2. \tag{1.20} \]

Of course, part of the point of the particle filter is to be able to track in clutter, and then the simple likelihood (1.20) is replaced by something non-Gaussian with multiple modes [31].

The third step of the algorithm controls the efficacy of the particle set in representing the posterior distribution over \( X_t \) via occasional reweightings. Details of how exactly reweighting is triggered are omitted here, but see [31].

Results of particle filtering for an active contour was given in figure 1. This example uses simple \textit{Condensation} [31] to track a blowing leaf in severe clutter. The figure shows a trail of estimated mean states (??) over time.

5 Further topics

There are a number of further topics in tracking that build on the ideas already outlined, and go beyond them in various intriguing ways. There is no space here to explore them in the depth they deserve, so pointers and brief summaries will have to suffice.
Fusing contour and appearance Much of this roadmap has addressed contour tracking, and in section 2 we briefly outlined approaches to appearance tracking. More recently there have been breakthroughs in joint modelling and localisation of contour and appearance [13] and the related approach [47], without dynamics however. An alternative fusion of appearance and contour combines particle filtering of contours [42] with an observation model like the one used in mean-shift tracking.

Filter Banks Observations based around contours have drawbacks both from the point of view of the principles of good Bayesian inference and, as above, the need to fuse both contour and appearance information. A complementary approach is to model the observations as the joint output of a set of filter banks [20, 50], which harnesses both appearance from filters within the object contour, and contrast from those that straddle the contour. The approach becomes even more powerful when combined with background modelling [34]. Another impressively powerful variation models filter outputs as a hybrid [35], with each filter switching independently between models for static, steady motion, or random walk.

Articulated and deformable structures Modelling deformation has been discussed above, and there are numerous variations on the theme, for example “deformable templates” [18, 57]. Outright articulation — jointed assemblies of rigid bodies — can be dealt with effectively using greedy strategies [29, 44], though at considerable computational cost, which can be mitigated using observation-cost gradient information [11]. Alternatively, the ASM approach of section 3.4 can be used for articulation also [5]. Issues arising in image-based models when image topology changes as the body articulates have been addressed using several shape space models connected via “wormholes” [28], in a Markov network. Alternatively, cartoon-like catalogues of outline-exemplars with differing topologies [19, 52], also connected in a Markov network, and matched using chamfers, are a very effective memory-intensive approach.

Persistence Finally, there have been striking advances in trained recognisers for localising faces and walking figures, in a single frame [53, 54]. These are so powerful and efficient that, without any recourse to dynamical models, real-time performance can be achieved on a modern workstation. However, these too can benefit from a dynamical approach [2, 55], promising real-time tracking in the background of a desktop machine’s process load, and on portable devices, in the future.

All of these issues and others will be treated in more detail in a forthcoming, long version of this roadmap article [6].
6 REFERENCES


1. Visual tracking: a research roadmap


