Contents

Chapter 1

Tracking: Fundamental Notions

In a tracking problem, one has some measurements that appear at each tick of a (notional) clock, and, from these measurements, one would like to determine the state of the world. There are two important sources of information. First, measurements constrain the possible state of the world. Second, there are dynamical constraints — the state of the world cannot change arbitrarily from time to time. Tracking problems are of great practical importance. There are very good reasons to want to, say, track aircraft using radar returns (good summary histories include [55, 57, 189]; comprehensive reviews of technique in this context include [34, 20, 414]).

Not all measurements are informative. For example, if one wishes to track an aircraft — where state might involve pose, velocity and acceleration variables, and measurements might be radar returns giving distance and angle to the aircraft from several radar aerials — some of the radar returns measured might not come from the aircraft. Instead, they might be the result of noise, of other aircraft, of strips of foil dropped to confuse radar apparatus (**chaff** or **window**; see [189]), or of other sources. The problem of determining which measurements are informative and which are not is known as **data association**.

Data association is the dominant difficulty in tracking objects in video. This is because so few of the very many pixels in each frame lie on objects of interest. It can be spectacularly difficult to tell which pixels in an image come from an object of interest and which do not. There are a very wide variety of methods for doing so, the details of which largely depend on the specifics of the application problem. Surprisingly, data association is not usually explicitly discussed in the computer vision tracking literature, which, as we shall see, tends to emphasize mechanisms of inference. However, whether a method is useful rests pretty directly on its success at data association — differences in other areas tend not to matter all that much in practice.

This review is intended to expose the main ideas of the subject, rather than to present citations of all human tracking papers (something that would probably not be possible). The ideas appear in a rough chronological order, reflecting what we see as the development of ideas in visual tracking. Initially, there was much emphasis on template and flow based tracking (this section); these methods proved unattractive in practice, for a variety of reasons. This phase was followed by a widespread interest in various methods of probabilistic inference, the hope being that dynamical models would offer helpful constraints on data association (section 2). However, dynamical models do not usefully constrain the data association problem when one is tracking humans; more recent methods have concentrated explicitly on building representations of image data that simplify tracking (section 3). We expect to see a unification of these methods with methods based in probabilistic inference in the near future.

`Pfinder`

1.1 General Observations

The literature on tracking people is immense. Furthermore, the problem has quite different properties depending on precisely what kind of representation one wishes to recover. The most important variable appears to be spatial scale. At a **coarse scale**, people are **blobs**. For example, we might view a plaza from the window of a building or a mall corridor from a camera suspended from the ceiling. Each person occupies a small block of pixels, perhaps 10-100 pixels in total. While we should be able to tell where a person is, there isn't much prospect of determining where the arms and legs are. At this scale, we can expect to recover representations of **occupancy** — where people spend time, for example [422] — or of **patterns of activity** — how people move from place to place, and at what time, for example [371].

At a **medium scale**, people can be thought of as blobs with attached **motion fields**. For example, a television program of a soccer match, where each individuals are usually 50-100 pixels high. In this case, one can tell where a person is. Arms and legs are still difficult to localize, because they cover relatively few pixels, and there is motion blur. However, the motion fields around the body yield some information as to how the person is moving. One could expect to be able to tell where a runner is in the phase of the run from this information — are the legs extended away from the body, or crossing?

At a **fine scale**, the arms and legs cover enough pixels to be detected, and one wants to report the configuration of the body. We usually refer to this case as **kinematic tracking**. At a fine spatial scale, one may be able to report such details as whether a person is picking up or handling an object. There are a variety of ways in which one could encode and report configuration, depending on the model adopted — is one to report the configuration of the arms? the legs? the fingers? — and on whether these reports should be represented in 2D or in 3D. We will discuss various representations in greater detail later. We do not regard the choice of a 2D representation or a 3D representation as significant to anything but detail in the tracking problem; in particular, as we shall show, passing from 2D to 3D representations is relatively straightforward under most conditions. In particular, the choice of 2D or 3D representation does not affect ambiguity in any way. If we track using a 2D representation and then lift to 3D, it may be difficult to recover 3D configuration unambiguously. These ambiguities are also present in the case where we track a 3D representation, though are often unremarked.

Each scale appears to be useful, but there are no reliable rules of thumb for determining what scale is most useful for what application. For example, one could see ways to tell whether people are picking up objects at a coarse scale. Equally, one could determine patterns of activity from a fine scale. Finally, as we shall see later, some quite complex determinations about activity can be made at a surprisingly coarse scale. Tracking tends to be much more difficult at the fine scale, because one must manage more degrees of freedom and because arms and legs can be small, and can move rather fast.

In this review, we focus almost entirely on the fine scale; even so, space will not allow detailed discussion of all that has been done. Our choice of scale is dictated by the intuition that good fine-scale tracking will be an essential component of any method that can give general reports on what people are doing in video. There are distinctive features of this problem that make fine scale tracking difficult:

- **State dimension:** One typically requires a high dimensional state vector to describe the configuration of the body in a frame. For example, assume we describe a person using a 2D representation. Each of ten body segments (torso, head, upper and lower arms and legs) will be represented by a rectangle of fixed size (that differs from segment to segment). This representation will use an absolute minimum of 12 state variables (position and orientation for one rectangle, and relative orientation for every other). A more practical version of the representation allows the rectangles to slide with respect to one another, and so needs 27 state variables. Considerably more variables are required for 3D models.

- **Nasty dynamics:** There is good evidence that such motions as walking have predictable, low-dimensional structure [345, 328]. However, the body can move extremely fast, with large accelerations. These large accelerations mean that one can stop moving predictably very quickly — for example, jumping in the air during a walk. For straightforward mechanical reasons, the body parts that move fastest tend to be small and on one end of a long lever which has big muscles at the other end (forearms, fingers and feet, for example). This means that the body segments that the dynamical model fails to predict are going to be hard to find because they are small. As a result, accurate tracking of forearms can be very difficult.

- **Complex appearance phenomena:** In most applications one is tracking clothed people. Clothing can change appearance dramatically as it moves, because the forces the body applies to the clothing change, and so the pattern of folds, caused by buckling, changes. There are two important results. First, the pattern of occlusions of texture changes, meaning that the apparent texture of the body segment can change. Second, each fold will have a typical shading pattern attached, and these patterns move in the image as the folds move on the surface. Again, the result is that the apparent texture of the body segment changes. These effects can be seen in figure 1.8.

- **Data association:** There is usually no distinctive color or texture that identifies a person (which is why people are notoriously difficult to find in static images). One possible cue is that many body segments appear at a distinctive scale as extended regions with rather roughly parallel sides. This isn't too helpful, as there are many other sources of such regions (for example, the spines of books on a shelf). Textured backgrounds are a particularly rich source of false structures in edge maps. Much of what follows is about methods to handle data association problems for people tracking.

1.2 Tracking by Detection

Assume we have some form of template that can detect objects reasonably reliably — a particularly important, useful and reliable example is a face template (a huge literature, this; there is some information in [120], otherwise start at [293, 323, 324, 325, 326, 375, 174, 401, 424, 332]). Assume that faces don't move all that fast, and there aren't too many in any given frame. Furthermore, the relationship between our representation of the state of a face and the image is uncomplicated. This occurs, for example, when the faces we view are always frontal or close to frontal. In this case, we can represent the state of the face by what it looks like (which, in principle, doesn't change because the face is frontal) and where it is.

Under these circumstances, we can build a tracker quite simply. We maintain a pool of tracks. We detect all faces in each incoming frame. We match faces to tracks, perhaps using an appearance model built from previous instances and also — at least implicitly — a dynamical model. This is where our assumptions are important; we would like faces to be sufficiently well-spaced with respect to the kinds of velocities we expect that there is seldom any ambiguity in this matching procedure. This matching procedure should not require one-one matches, meaning that some tracks may not receive a face, and some faces may not be allocated a track. For every face that is not attached to a track, we create a new track. Any track that has not received a face for several frames is declared to have ended (algorithm 1 breaks out this approach; figure **??** shows an example track).

This basic recipe for tracking by detection is worth remembering. In many situations, nothing more complex is required, and the recipe is used without comment in a variety of papers. As a simple example,

Assumptions: We have a detector which is reasonably reliable for all aspects that matter. Objects move relatively slowly with respect to the spacing of detector responses. As a result, a detector response caused either by another object or by a false positive tends to be far from the next true position of our object.

First frame:
Create a track for each detector response.

N'th frame:
Link tracks and detector responses. Typically, each track gets the closest detector response if it is not further away than some threshold. If the detector is capable of reporting some distinguishing feature (colour, texture, size, etc.), this can be used too.
Spawn a new track for each detector response not allocated to a track.
Reap any track that has not received a measurement for some number of frames.

Cleanup: We now have trajectories in space time. Link any where this is justified (perhaps by a more sophisticated dynamical or appearance model, derived from the candidates for linking).

Algorithm 1: *The simplest tracking by detection*

at coarse scales and from the right view, background subtraction and looking for dark blobs of the right size is sufficient to identify human heads. Yan and Forsyth use this observation in a simple track-by-detection scheme, where heads are linked across frames using a greedy algorithm [422]. The method is effective for obtaining estimates of where people go in public spaces.

The method will need some minor improvements and significant technical machinery as the relationship between state and image measurements grows more obscure. However, in this simple form, the method gives some insight into general tracking problems. The trick of creating tracks promiscuously and then pruning any track that has not received a measurement for some time is a quite general and extremely effective trick. The process of linking measurements to tracks is the aspect of tracking that will cause us the most difficulty (the other aspect, inferring states from measurements, is straightforward though technically involved). This process is made easier if measurements have features that distinctively identify the track from which they come. This can occur because, for example, a face will not change gender from frame to frame, or because tracks are widely spaced with respect to the largest practical speed (so that allocating a measurement to the closest track is effective).

1.2.1 Background Subtraction

The simplest detection procedure is to have a good model of the background. In this case, everything that doesn't look like the background is worth tracking. The simplest background subtraction algorithm is to take an image of the background and then subtract it from each frame, thresholding the magnitude of the difference (there is a brief introduction to this area in [120]). Changes in illumination will defeat this approach. A natural improvement is to build a moving average estimate of the background, to keep track of illumination changes (e.g. see [416, 337]; gradients can be incorporated [240]). In outdoor scenes, this approach is defeated by such phenomena as leaves moving in the wind. More sophisti-

Figure 1.1: **Fig 1 of surveillance/00868683** *Background subtraction identifies groups of pixels that differ significantly from a background model. The method is most useful for some some cases of surveillance, where one is guaranteed a fixed viewpoint and a static background changing slowly in appearance. On the* **left***, a background model; in the* **center***, a frame; and on the* **right***, the resulting image blobs. The figure is taken from Haritaoglu* et al. *[148]; in this paper, authors use an elaborate method involving a combination of thresholds to obtain good blobs. Figure* **??** *illustrates a method due to these authors that obtains a kinematic configuration estimate by parsing the blob.*

cated background models keep track of maximal and minimal values at each pixel [148], or build local statistical models at each pixel [369, 143, 370, 64, 125, 177, 178].

Under some circumstances, background subtraction is sufficient to track people and perform a degree of kinematic inference. Haritaoglu *et al.* describe a system called W4, which uses background subtraction to segment people from an outdoor view [148, **?**]. Foreground regions are then linked in time by applying a second order dynamic model (velocity and acceleration) to propagate median coordinates (a robust estimate of the centroid) forward in time. Sufficiently close matches trigger a search process that matches the relevant foreground component in the previous frame to that in the current frame. Because people can pass one another or form groups, foreground regions can merge, split or appear. Regions appearing, splitting or merging are dealt with by creating (resp. fusing) tracks. Good new tracks can be distinguished from bad new tracks by looking forward in the sequence: a good track continues over time. Allowing a tracker to create new tracks fairly freely, and then telling good from bad by looking at the future in this way is a traditional, and highly useful, trick in the radar tracking community (e.g. see the comprehensive book by Blackman and Popoli [34]). The background subtraction scheme is fairly elaborate, using a range of thresholds to obtain a good blob (figure 1.1). The resulting blobs are sufficiently good that the contour can be parsed to yield a decomposition into body segments. They segment the contours using convexity criteria, then tag the segments using: distance to the head — which is at the top of the contour; distance to the feet — which are at the bottom of the contour; and distance to the median — which is reasonably stable. All this works because, for most configurations of the body, one will encounter body segments in the same order as one walks around the contour (figure **??**). Shadows are a perennial nuisance for background subtraction, but this can be dealt with using a stereoscopic reconstruction, as Haritaoglu *et al.* show ([147]; see also [179]).

1.2.2 Deformable Templates

Image appearance or **appearance** is a flexible term used to refer to aspects of an image that are being encoded and should be matched. Appearance models might encode such matters as: Edge position; edge orientation; the distribution of color at some scale (perhaps as a histogram, perhaps as histograms for each of some set of spatially localized buckets); or texture (usually in terms of statistics of filter

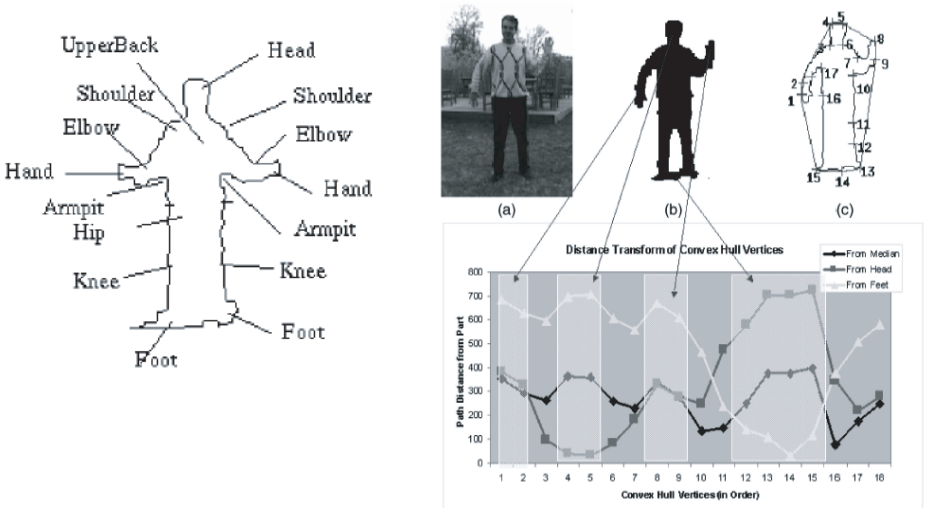


Figure 1.2: *For a given view of the body, body segments appear in the outline in a predictable manner. An example for a frontal view appears on the* **left**. *Haritaoglu* et al *identify vertices on the outline of a blob using a form of convexity reasoning (***right (b)** *and* **right (c)***), and then infer labels for these vertices by measuring the distance to head (at the top), feet (at the bottom) and median (***right (d)***). These distances give possibly ambiguous labels for each vertex; by applying a set of topological rules obtained using examples of multiple views like that on the* **left***, they obtain an unambiguous labelling.* **Fig 12, 15 of surveillance/00868683**

outputs.

A **deformable template** or **snake** is a parametric model of image appearance usually used to localize structures. For example, one might have a template that models the outline of a squash [191, 192] or the outline of a person [27], place the template on the image in about the right place, and let a fitting procedure figure out the best position, orientation and parameters.

We can write this out formally as follows. Assume we have some form of template that specifies image appearance as a function of some parameters. We write this template — which gives (say) image brightness (or color, or texture, and so on) as a function of space $\mathbf{x}$ and some parameters θ — as $T(\mathbf{x}|\theta)$. We score a comparison between the image at frame n, which we write as $I(\mathbf{x}, t_n)$, and this template using the a scoring function ρ

$$\rho(T(\mathbf{x}|\theta), I(\mathbf{x}, t_n))$$

Space will allow discussion of only some properties of the very large variety of possible scores. We will always assume that small values of the score are good and large values of the score are bad matches (meaning our discussion may differ from an author's by a minus sign on occasion).

1.2.2.1 Point Templates

One could build a template using a set of **active sites** within a model coordinate frame; write the matrix whose columns are vectors containing coordinates of a site as $\mathcal{X}$. The model allows these sites to appear in different places in the model coordinate frame. The position of the sites is typically a known affine function of some unknown shape parameters b_i, giving $\mathcal{X} = \overline{\mathcal{X}} + \sum_i \oplus_i b_i$. The affine function can be fit using principal coordinate analysis (section **??**) to a set of examples.

The collection of sites is subject to some transformation before appearing in an image frame, as

$$\mathcal{T}(\theta)\mathcal{X}$$

where $\mathcal{T}$ might be translation, rotation and translation, rotation, translation and scale, or an affine transformation. We observe the resulting **keypoints** in an image; write the matrix whose columns are vectors containing coordinates of the image keypoint corresponding to a site as $\mathcal{Y}$. We can now formulate the goodness of fit as

$$\min_{b_i,\theta} \text{Trace}((\mathcal{Y} - \mathcal{T}(\theta)(\overline{\mathcal{X}} + \sum_i \oplus_i b_i))^T(\mathcal{Y} - \mathcal{T}(\theta)(\overline{\mathcal{X}} + \sum_i \oplus_i b_i))^T)$$

This method has had some popularity as an **active appearance model** One must be careful here, however. First, the minimization is not easy, because there is an implicit correspondence. We assumed that we could observe in the image which keypoint corresponded to which site, which isn't usually the case. The affine function that yields the sites can be derived in a variety of ways. Cootes *et al*.apply principal components to observed keypoints, which have been rectified to a canonical frame (one must be careful about correspondence here; see []) [].

1.2.2.2 Shape models by families of curves

It isn't particularly natural to think of an object as a set of points (though fashions change in this matter; for example, see [112, 361, 331, 330, 394, 186] for one view and [32, 54, 268, 207] for the other). One might instead encode objects using curves. The idea originates with **snakes** []. In this formalism, a snake is a curve whose cost is evaluated by integrating some function along its length. The cost is a sum of an image-based potential — usually computed from a feature map, often an edge map — and a geometric potential — which discourages sharp corners, excessive stretch, etc. Some curve is chosen as a start point, and the final encoding is obtained by allowing the curve to evolve in a way that minimizes the potential. For example, the curve might move to lie on the outline of an object, where there are many strong edge points. Inertia terms can be added to produce a snake that can track moving objects without excessive wiggle.

A large body of work modifies this idea to produce deformable templates based on curves. Blake and Isard give an excellent account of work in this area, with more information on the origins of the ideas than we can give here, in [35]). One important modification of this model is to drop the idea of computing the goodness of a curve by summing some potential along its length. Assume the curve $\mathbf{c}(s)$, parametrized by s. We will sum over a small set of points $s = s_i$ on the curve. Typically, we would like these points to be close to image points where there is a strong feature response — say an edge point. It can be inconvenient to find every edge point in the image (a matter of speed) and this class of template allows us to search for edges only along short sections normal to the curve (figure 1.3). We then fit a curve where the points on the curve given by $s = s_i$ have a minimum distance to observed features.

We now confine our discussion to B-spline curves, where $\mathbf{c}(s)$ is determined by a set of control points. Write the components of the control points stacked into a vector as $\mathbf{Q}$. Notice we do not need to encode transformations explictly, because B-splines are affine covariant. Write $\mathbf{C}$ for the vector with components $\mathbf{c}(s_i)$. Then there is some matrix $\oplus$ such that $\mathbf{C} = \oplus\mathbf{Q}$. Write $\mathbf{M}$ for the vector of measured locations at which we would like the points $\mathbf{c}(s_i)$ to lie. Then the distance from measurement points to features is

$$(\oplus\mathbf{Q} - \mathbf{M})^T(\oplus\mathbf{Q} - \mathbf{M})$$

Figure 1.3:

Figure 1.4: *An example of robustness problems*

This fitting distance is insufficient to give a match cost to a template, because it is unconstrained — it could most things (furthermore, parametrization can create issues; see [36] for details). The curve should be drawn from a **shape space** — a constrained family of curves. It is usual to employ a linear family of curves. Assume our curve has N control points, and we wish to encode a k-dimensional family of curves. Now write $\mathcal{W}$ for a $2N \times k$ matrix encoding the family of possible sets of control points, $\mathbf{Q}_0$ for a reference set of control points, and $\mathbf{X}$ for the k-vector of **shape parameters**. Our curve is represented by

$$\mathbf{Q} = \mathcal{W}\mathbf{X} + \mathbf{Q}_0$$

Blake and Isard give variety of constructions yield the form of $\mathcal{W}$ for curves that are translations, similarity transformations and affine transformations of a given curve (chapter ***, [36]). An alternative is to obtain a series of example curves, fit splines, and apply principal components to the coefficients (eg [27, **?**]).

We expect that a variety of curves is necessary to encode the outline of an object, and can regularize the fit by charging for substantial deviations. One then has a **reference shape** $\overline{\mathbf{X}}$ and a positive definite matrix $\mathcal{S}$. We bias the fitting cost by a term proportional to $(\mathbf{X}-\overline{\mathbf{X}})^T\mathcal{S}(\mathbf{X}-\overline{\mathbf{X}})$. The template matching cost can be obtained by computing

$$\min_{\mathbf{X}}(\oplus(\mathcal{W}\mathbf{X} + \mathbf{Q}_0) - \mathbf{M})^T(\oplus(\mathcal{W}\mathbf{X} + \mathbf{Q}_0) - \mathbf{M}) + \lambda(\mathbf{X} - \overline{\mathbf{X}})^T\mathcal{S}(\mathbf{X} - \overline{\mathbf{X}})$$

and the relevant shape is the minimizing value of $\mathbf{X}$.

1.2.2.3 Robustness

We have presented scoring a deformable template as a form of least squares fitting problem. There is a basic difficulty in such problems. Points that are dramatically in error, usually called **outliers** and traditionally blamed on typist error [322, 155], can be overweighted in determining the fit. Figure 1.4 shows an example of this very general problem, in the context of line fitting. Outliers in vision problems tend to be unavoidable, because nature is so generous with visual data that there is usually something seriously misleading in any signal. There are a variety of methods for managing difficulties created by outliers that are used in building deformable template trackers. An estimator is called **robust** if the estimate tends to be only weakly affected by outliers. For example, the average of a set of observations is not a robust estimate of the mean of their source (because if one observation is, say, mistyped, the average could be wildly incorrect). The median *is* a robust estimate, because it will not be much affected by the mistyped observation.

The scheme of finding edge points by searching out some distance along the normal from a curve is a **gating** strategy (figure 1.5). In this case, one limits the distance searched. Ideally, there is only one edge point in the search window, but if there are more one takes the closest (strongest, *mutatis mutandis* depending on application details). If there is nothing, one accepts some fixed score, chosen to make the cost continuous. This means that the cost function, while strictly not differentiable, is not dominated by

Figure 1.5: *Gating*

very distant edge points. These are not seen in the gate, and there is an upper bound on the error any one site can contribute. Again, this strategy confers robustness.

An alternative is to use an **m-estimator**. One would like to score the template with a function of squared distance between site and measured point. This function should be close to the identity for small values (so that it behaves like the square) and close to some constant for large values (so that large values don't contribute large biases). A natural form is

$$\rho(u) = \frac{u}{u+\sigma}$$

so that, for d^2 small with respect to σ, we have $\rho(d^2) \approx d^2$ and for d^2 large with respect to σ we have $\rho(d^2) \approx 1$. The advantage of this approach is that nearby edgepoints dominate the fit; the disadvantage is that even fitting problems that are originally convex are no longer convex when the strategy is applied. Numerical methods are consequently more complex, and one must use multiple start points. There is little hope of having a convex problem, because different start points correspond to different splits of the data set into "important" points and outliers; there is usually more than one such split. Again, large errors no longer dominate the estimation process, and the method is almost universally applied for flow templates (section **??**).

1.2.2.4 The Hausdorff Distance

The **Hausdorff distance** is a method to measure similarity between binary images (for example, edge maps; the method originates in Minkowski's work in convex analysis [], where it takes a somewhat different form). Assume we have two sets of points P and Q; typically, each point is an edge point in an image. We define the Hausdorff distance between the two sets to be

$$H(P,Q) = max(h(P,Q), h(Q,P))$$

where

$$h(P,Q) = \max_{p \in P} \min_{q \in Q} \parallel p - q \parallel$$

The distance is small if there is a point in Q close to each point in P and a point in P close to each point in P. There is a difficulty with robustness, as the Hausdorff distance is large if there are points with no good matches. In practice, one uses a variant of the Hausdorff distance (the **generalized Hausdorff distance**) where the distance used is the k-th ranked of the available distances rather than the largest. Define F_k^{th} to be the operator that orders the elements of its input largest to smallest, then takes the k'th largest. We now have

$$H_k(P,Q) = max(h_k(P,Q), h_k(Q,P))$$

where

$$h_k(P,Q) = F_k^{th}(\min_{q \in Q} \parallel p - q \parallel)$$

(for example, if there are $2n$ points in P, then $h_n(P,Q)$ will give the median of the minimum distances). The advantage of all this is that some large distances get ignored.

Now we can compare a template P with an image Q by determining some family of transformations $\mathcal{T}(\theta)$ and then choosing the set of parameters $\hat{\theta}$ that minimizes

$$H_k(\mathcal{T}(\theta) \circ P, Q)$$

This will involve some form of search over θ. The search is likely to be simplified if — as applies in the case of tracking — we have a fair estimate of $\hat{\theta}$ to hand.

1.2.2.5 Tracking by Detection with a Deformable Template

Deformable templates have not been widely used as object detectors, because finding a satisfactory minimum — one that lies on the object of interest, most likely a global minimum — can be hard. The search is hard to initialize because one must identify the feature points that should lie within the gate of the template. However, in tracking problems this difficulty is mitigated if one has a dynamical model of some form. For example, the object might move slowly, meaning that the minimum for frame n will be a good start point for frame $n + 1$. As another example, the object might move with a large, but near constant, velocity. This means that we can *predict* a good start point from frame $n + 1$ given frame n. A significant part of the difficulty is caused by image features that don't lie on the object, meaning that another useful case occurs in the near absence of clutter — perhaps background subtraction, or the imaging conditions, ensures that there are few or no extra features to confuse the fitting process.

Baumberg and Hogg track people with a deformable template built using a B-spline as above, with principal components used to determine $\mathcal{W}$ [**?**]. Figure 1.6 gives a sense of the wide variations in shape that can be obtained with such models. They use background subtraction to obtain an outline for the figure, then sample the outline. For this kind of template, correspondence is generally a nuisance, but in some practical applications, this information can be supplied from quite simple considerations. For example, Baumberg and Hogg work with background subtracted data of pedestrians at fairly coarse scales from fixed views [**?**]. In this case, sampling the outline at fixed fractions of length, and starting at the lowest point on the principal axis yields perfectly acceptable correspondence information (figure 1.6).

Huttenlocher *et al.*track using the Hausdorff distance [167]. The template, which consists of a set of edge points, is itself allowed to deform. Images are represented by edge points. They identify the instance of the latest template in the next frame by searching over translations θ of the template to obtain the smallest value of $H_k(\mathcal{T}(\theta) \circ P, Q)$. They then translate the template to that location, and identify all edge points that are within some distance of the current template's edge points. The resulting points form the template for the next frame. This process allows the template to deform to take into account, say, the deformation of the body as a person moves. Performance in heavily textured video must depend on the extent to which the edge detection process suppresses edges and the setting of this distance parameter (a large distance and lots of texture is likely to lead to catastrophe).

`Mean shift tracking`

1.3 Tracking using Flow

The difficulty with tracking by detection is that one might not have a deformable template that fully specifies the appearance of an object. It is quite common to have a template that specifies the shape of the domain spanned by the object and the type of its transformation, but not what lies within. Typically, we don't know the pattern, but we do know how it moves. There are several important examples:

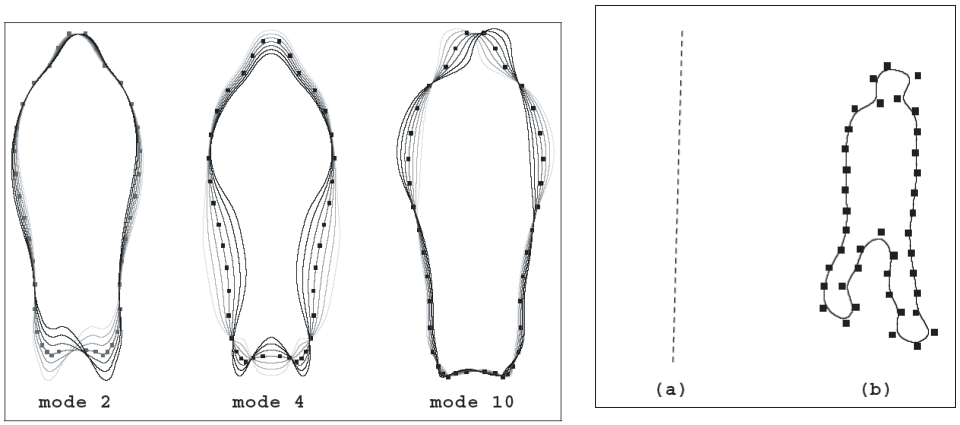


Figure 1.6: **Baumberg+Hogg93 paper** *Baumberg and Hogg use a point template where the sites are points of fixed parameter value on a closed B-spline and the shape parameters are given by the choice of control points for this spline [**?**]. Templates are fit to a training set, and modes of variation extracted by a principal component analysis on the fitted control points. The three examples on the* **left** *show the range of variation in outline shape and keypoint position obtainable by varying the coefficients of some principal components. Their application involves tracking pedestrians at a relatively coarse scale, using background subtraction to identify the pedestrian. In this application, keypoints are obtained by sampling the outline at evenly spaced intervals, starting at the lowest of the two points where the outline crosses the principal axis (***right***). This strategy imposes a correspondence which appears to be satisfactory in practice.*

- **Human body segments** tend to look like a rectangle in any frame, and the motion of this rectangle is likely to be either Euclidean or affine, depending on imaging circumstances.

- **A face in a webcam** tends to fill a blob-like domain and undergo mainly Euclidean transformations. This is useful for those building user interfaces where the camera on the monitor views the user, and there are numerous papers dealing with this. The face is not necessarily frontal — computer users occasionally look away from their monitors — but tends to be large, blobby and centered.

- **Edge templates**, particularly those specifying outlines, are usually used because we don't know what the interior of the region looks like. Quite often, as we have seen, we know how the template can deform and move. However, we cannot score the interior of the domain because we don't know (say) the pattern of clothing being worn.

In each of these cases, we cannot do tracking by detection as above because we do not posess an appropriate template. As a matter of experience, objects don't change appearance much from frame to frame (alternatively, we should use the term appearance to apply to properties that don't change from frame to frame). All this implies that parts of the previous image could serve as a template if we have a motion model and domain model. We could use a correspondence model to link pixels in the domain in frame n with those in the domain in frame $n + 1$. A "good" linking should pair pixels that have similar appearances. Such considerations as camera properties, the motion of rigid objects, and computational expense suggest choosing the correspondence model from a small parametric family.

All this gives a formal framework. Write a pixel position in the n'th frame as $\mathbf{x}_n$, the domain in the n'th frame as $\mathcal{D}_n$, and the transformation from the n'th frame to the $n+1$'th frame as $\mathcal{T}_{n\rightarrow n+1}(\cdot;\theta_n)$. In this notation θ_n represent parameters for the transformation from the n'th frame to the $n+1$'th frame, and we have that $\mathbf{x}_{n+1} = \mathcal{T}_{n\rightarrow n+1}(\mathbf{x}_n;\theta_n)$.

We assume we know $\mathcal{D}_n$. We can obtain $\mathcal{D}_{n+1}$ from $\mathcal{D}_n$ as $\mathcal{T}_{n\rightarrow n+1}(\mathcal{D}_n;\theta_n)$. Now we can score the parameters θ_n representing the *change* in state between frames $n+1$ and n by comparing $\mathcal{D}_n$ with $\mathcal{D}_{n+1}$ (which is a function of θ_n). We compute some representation of image information $\mathbf{R}(\mathbf{x})$, and, within the domain $\mathcal{D}_{n+1}$ compare $\mathbf{R}(\mathbf{x}_{n+1})$ with $\mathbf{R}(\mathcal{T}_{n\rightarrow n+1}(\mathbf{x}_n;\theta_n))$, where the transformation is applied to the domain $\mathcal{D}_n$.

1.3.1 Optic Flow

Generally, a frame-to-frame correspondence should be thought of as a **flow field** (or an **optic flow field**) — a vector field in the image giving local image motion at each pixel. A flow field is fairly clearly a correspondence, and a correspondence gives rise to a flow field (put the tail of the vector at the pixel position in frame n, and the head at the position in frame $n+1$). The notion of optic flow originates with *****.

A useful construction in the optic flow literature assumes that image intensity is a continuous function of position and time, $I(\mathbf{x},t)$. We then assume that the intensity of image patches does not change with movement. While this assumption may run into troubles with illumination models, specularities, etc., it is not outrageous for small movements. Furthermore, it underlies our willingness to compare pixel values in frames. Accepting this assumption, we have

$$\frac{dI}{dt} = \nabla I \cdot \frac{d\mathbf{x}}{dt} + \frac{\partial I}{\partial t} = 0$$

(known as the **optic flow equation**, e.g. see [161]). Flow is represented by $d\mathbf{x}/dt$. This is important, because if we confine our attention to an appropriate domain, comparing $I(\mathcal{T}(\mathbf{x};\theta_n),t_{n+1})$ with $I(\mathbf{x},t_n)$ involves, in essence, estimating the total derivative. In particular,

$$I(\mathcal{T}(\mathbf{x};\theta_n),t_{n+1}) - I(\mathbf{x},t_n) \approx \frac{dI}{dt}$$

Furthermore, the equivalence between correspondence and flow suggests a simpler form for the transformation of pixel values. We regard $\mathcal{T}(\mathbf{x};\theta_n)$ as taking $\mathbf{x}$ from the tail of a flow arrow to the head. This justifies the notation $\mathcal{T}(\mathbf{x};\theta_n) = \mathbf{x} + \delta\mathbf{x}(\theta_n)$.

`O'RourkeBadler? Hogg?`

1.3.2 Image stabilization

This form of tracking can be used to build boxes around moving objects, a practice known as **image stabilization**. One has a moving object on a fairly uniform background, and would like to build a domain such that the moving object is centered on the domain. This has the advantage that one can look at relative, rather than absolute, motion cues. For example, one might take a soccer player running around a field, and build a box around the player. If one then fixes the box and its contents in one place, the vast majority of motion cues within the box are cues to how the player's body configuration is changing (compare root and segment representations, section **??**). As another example, one might

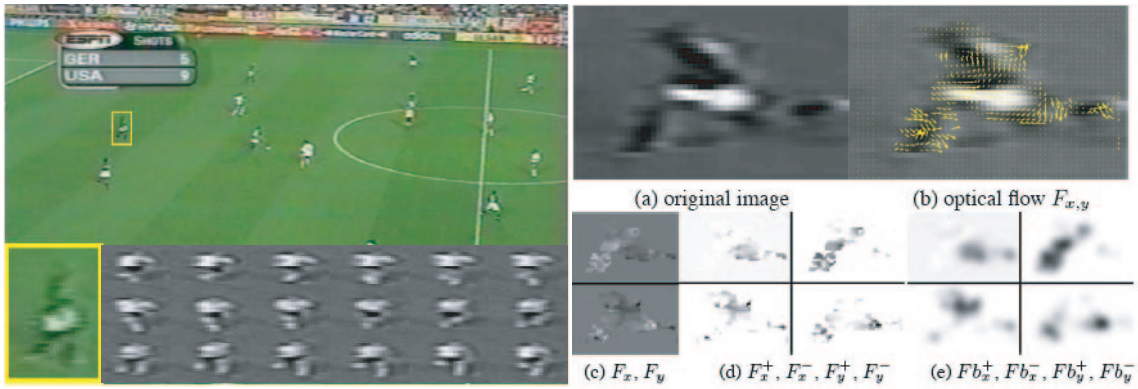


Figure 1.7: **Figure 1, part of 3 and 4 from Efros et al. Recognizing action at a distance.** *Flow based tracking can be useful for medium scale video. Efros* et al. *stabilize boxes around the torso of players in football video using a sum of squared differences (SSD) as a cost function and straightforward search to identify the best translation values. As the figure on the* **left** *shows, the resulting boxes are stable with respect to the torso. On the* **top right**, *larger versions of the boxes for some cases. Note that, because the video is at medium scale, it is difficult to resolve arms and legs, which are severely affected by motion blur. Nonetheless, one can make a useful estimate of what the body is doing by computing an estimate of optic flow (***bottom right**, F_x, F_y*), rectifying this estimate (***bottom right**, F_x^+, F_x^-, F_y^+, F_y^-*) and then smoothing the result (***bottom right**, Fb_x^+, *etc.). The result is a smoothed estimate of where particular velocity directions are distributed with respect to the torso, which can be used to match and label frames.*

stabilize a box around an aerial view of a moving vehicle; now the box contains all visual information about the vehicle's identity.

Efros *et al.* use a straightforward version of this method, where domains are rectangles and flow is pure translation, to stabilize boxes around people viewed at a medium scale (for example, in a soccer video). In some circumstances, good results can be obtained by matching a rectangle in frame n with the rectangle in frame $n+1$ that has smallest sum-of-squared differences — which might be found by blank search, assisted perhaps by velocity constraints. This is going to work best if the background is relatively simple — say, the constant green of a soccer field — as then the background isn't a source of noise, so the figure need not be segmented (figure 1.7). For more complex backgrounds, the approach may still work if one performs background subtraction before stabilization. At a medium scale it is very difficult to localize arms and legs, but they do leave traces in the flow field. The stabilization procedure means that the flow information can be computed with respect to a torso coordinate system, resulting in a representation that can be used to match at a kinematic level, without needing an explicit representation of arm and leg configurations (figure 1.7).

1.3.3 Cardboard people

Flow based tracking has the advantage that one doesn't need an explicit model of the appearance of the template. Ju et al build a model of legs in terms of a set of articulated rectangular patches ("cardboard people"). Assume we have a domain D in the n'th image $I(\mathbf{x}, t_n)$ and a flow field $\delta\mathbf{x}(\theta)$ parametrized by θ. Now this flow field takes D to some domain in the $n+1$'th image, and establishes a correspondence

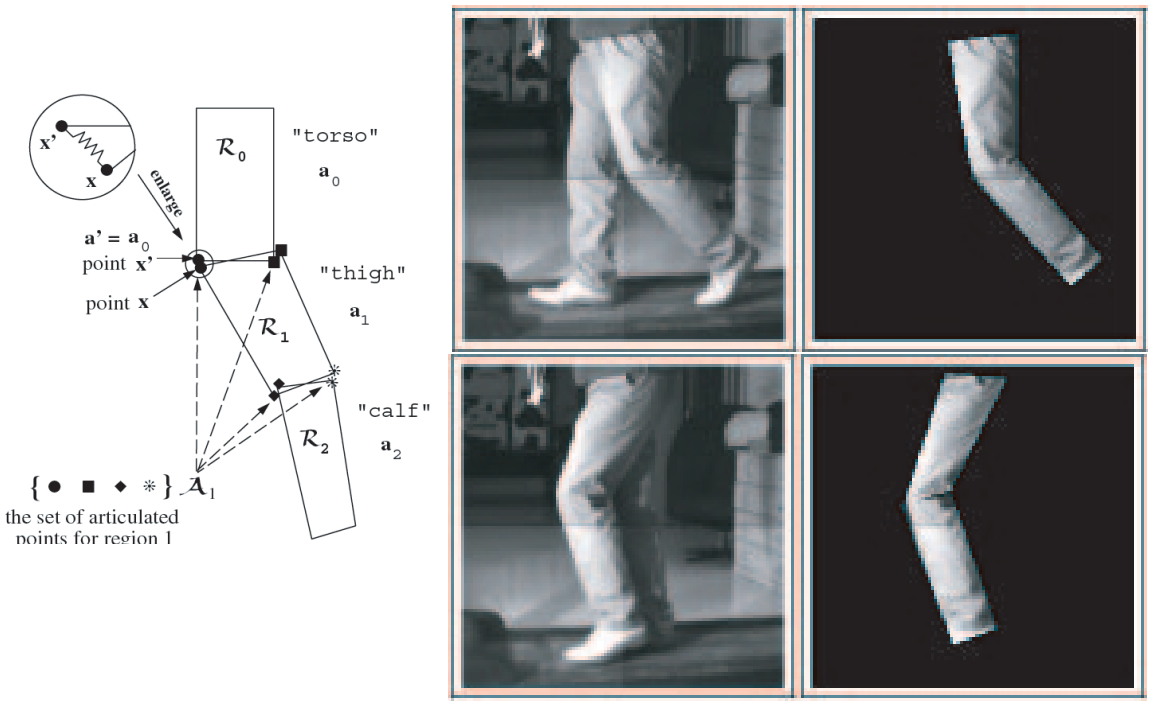


Figure 1.8: **Figure 3, 4 from Ju, Black and Yacoob, "Cardboard People", no permission yet** *On the* **left**, *a 2D flow based model of a leg, called a "cardboard people" model by Ju* et al *[190]; there is a lower leg, an upper leg and a torso. Each domain is roughly rectangular, and the domains are coupled with an energy term to ensure they do not drift apart. The model is tracked by finding the set of deformation parameters that carve out a domain in the $n+1$'th frame that is most like the known domain in the n'th frame. On the* **right**, *two frames from a track, with the* **left column** *showing the original frame and the* **right column** *showing the track. Notice how the pattern of buckling folds on the trouser leg changes as the leg bends; this leads to quite significant changes in the texture and shading signal in the domain. These changes can be a significant nuisance.*

between pixels in the n'th and the $n+1$'th image They score

$$\sum_{D} \rho(I_{n+1}(\mathbf{x} + \delta\mathbf{x}(\theta)) - I_n(\mathbf{x}))$$

where ρ is some measure of image error, which is small when the two compare well and large when they are different. One could use the sum of squared differences, but this is unwise because the score is not robust. It is usual to use

$$\rho(u) = \frac{u}{u + \sigma}$$

for σ some parameter (cf the discussion above, see chapter *** of [120], or [322, 155]). Notice that this is a very general approach to the tracking problem, with the difficulty that, unless one is careful about the flow model the problem of finding a minimum might be hard. To our knowledge, the image score is always applied to pixel values, and it seems interesting to wonder what would happen if one scored a difference in texture descriptors; we explain the reason to consider this below (section **??**).

Typically, the score is not minimized directly, but is approximated with the optic flow equation and

with a Taylor series to get

$$\sum_D \rho(I(\mathbf{x}+\delta\mathbf{x}(\theta), t_{n+1}) - I_n(\mathbf{x}, t_n)) \approx \sum_D \rho(\frac{dI}{dt}) = \sum_D \rho(\frac{\partial I}{\partial x}\delta x(\theta_n) + \frac{\partial I}{\partial y}\delta y(\theta_n) + \frac{\partial I}{\partial t})$$

(this works because $\Delta t = 1$). Now assume that a patch has been marked out in a frame; then one can determine its configuration in the next by minimizing this error summed over the domain. The error itself is easily evaluated using smoothed derivative estimates. As we show below, we can further simplify error evaluation by building a flow model with convenient form. To track an articulated figure, we attach a further term that encourages relevant vertices of each separate patch to stay close. Similarly, Black et al construct parametric families of flow fields and use them to track lips and legs, in the latter case yielding a satisfactory estimate of walk parameters [33]. In both cases, the flow model is view dependent. Yacoob and Davis build a view independent parametric flow field models to track views of walking humans [418]. As one would expect, this technique can be combined with others; for example, the W4S system of Haritaoglu *et al.* uses a "cardboard people" model to track torso configurations within the regions described above [147].

1.3.4 Building Flow Templates

We have seen how to construct tracks given parametric models of flow. But how is one to obtain good models? One strategy is to take a pool of examples of the types of flow one would like to track, and try to find a set of basis flows that explains most of the variation (for examples, see [190]). In this case, and writing θ_i for the i'th component of the parameter vector and $\mathbf{F}_i$ for the i'th flow basis vector field, one has

$$\delta\mathbf{x} = \sum_i \theta_i \mathbf{F}_i$$

Now write ∇I for the image gradient and exploit the optic flow equation and a Taylor series as above. We get

$$\rho(\sum_i \theta_i((\nabla I)^T \mathbf{F}_i) + \frac{\partial I}{\partial t})$$

This can be done with a singular value decomposition (and is equivalent to **principal components analysis** or **PCA**; see [120, **?**]). A second strategy is to assume that flows involve what are essentially 2D effects — this is particularly appropriate for lateral views of human limbs — so that a set of basis flows that encodes translation, rotation and some affine effects is probably sufficient. One can obtain such flows by writing

$$\delta\mathbf{x} = \begin{pmatrix} u(\mathbf{x}) \\ v(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} a_0 + a_1 x + a_2 y + a_6 x^2 + a_7 xy \\ a_3 + a_4 x + a_5 y + a_6 xy + a_7 y^2 \end{pmatrix}$$

This model is linear in the parameters (the a_i), which is convenient; it provides a reasonable encoding of flows resulting from 3D motions of a 2D rectangle (see figure 1.9). One may also learn linearized flow models from example data [418].

1.3.5 Flow models from kinematic models

An alternative method to build such templates is to work in 3D, and exploit the chain rule, as in the work of Bregler and Malik [51, 52]. We start with a segment in 3D, which is in some configuration and

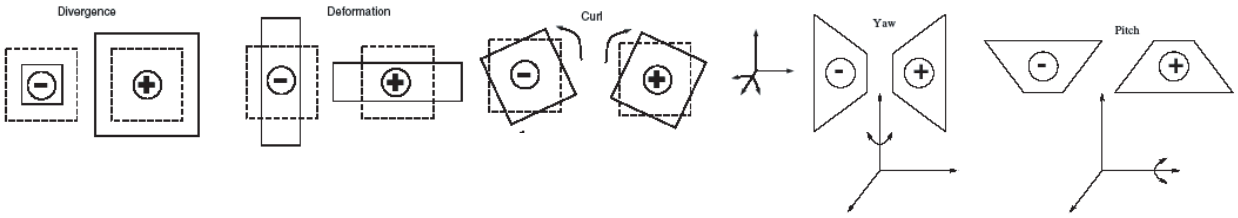


Figure 1.9: **Figure 2 from Ju, Black and Yacoob, "Cardboard People", no permission yet** *Typical flows generated by the model* $(u(\mathbf{x}), v(\mathbf{x})^T = (a_0 + a_1 x + a_2 y + a_6 x^2 + a_7 xy, a_3 + a_4 x + a_5 y + a_6 xy + a_7 y^2))$. *Different values of the* a_i *give different flows, and the model can generate flows typical of a 2D figure moving in 3D. We write* $\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$. **Divergence** *occurs when the image is scaled; for example,* $\mathbf{a} = (0, 1, 0, 0, 0, 1, 0, 0)$. **Deformation** *occurs when one direction shrinks and another grows (for example, rotation about an axis parallel to the view plane in an orthographic camera); for example,* $\mathbf{a} = (0, 1, 0, 0, 0, -1, 0, 0)$. **Curl** *can result from in plane rotation; for example,* $\mathbf{a} = (0, 0, -1, 0, 1, 0, 0, 0)$. **Yaw** *models rotation about a vertical axis in a perspective camera; for example* $\mathbf{a} = (0, 0, 0, 0, 0, 1, 0)$. *Finally,* **pitch** *models rotation about a horizontal axis in a perspective camera; for example* $\mathbf{a} = (0, 0, 0, 0, 0, 0, 1)$.

viewed with some camera. Each point on the segment produces some image value. We could think of the image values as a function — the **appearance map** — defined on the segment. This allows us to see viewing the segment as building a mapping from the points on the segment to the image domain. The image values are obtained by taking each point in the image, finding the corresponding point (if any) on the segment, and then evaluating the appearance map at this point.

All this leads to an important formal model, again under the assumption that motions in 3D do not affect the appearance map in any significant way. We have a parametrized family of maps from points on the body to the image. A flow field in the image is a vector field induced by a change in the choice of parameters (caused by either a change in joint configuration or a camera movement). We will always assume that the change in parameters from frame to frame is small. At this point, we must introduce some notation. Write the map that takes points on the segment to points in the n'th image as $\mathcal{T}_{s\to I}(\cdot;\theta_n)$, where θ_n are parameters representing camera configuration, intrinsics, etc. The point $\mathbf{p}$ on the segment appears in image n at $\mathbf{x}_n = \mathcal{T}_{s\to I}(\mathbf{p};\theta_n)$ and in image $n+1$ at $\mathbf{x}_{n+1} = \mathcal{T}_{s\to I}(\mathbf{p};\theta_{n+1})$. The tail of the flow arrow is at $\mathbf{x}_n$ and the head is at $\mathbf{x}_{n+1}$. The change in parameters, $\Delta\theta = \theta_{n+1} - \theta_n$ is small. Then the flow is

$$\mathbf{x}_{n+1} - \mathbf{x}_n = \mathcal{T}_{s\to I}(\mathbf{p};\theta_{n+1}) - \mathcal{T}_{s\to I}(\mathbf{p};\theta_n) \approx \nabla_\theta \mathcal{T}_{s\to I} \cdot \Delta\theta$$

where the gradient, $\nabla_\theta \mathcal{T}_{s\to I}$, is evaluated at $(\mathbf{p}, \theta_n)$.

1.3.5.1 Tracking a Derivative Flow Model

The main point here is that the flow at $\mathbf{x}_n$ can be obtained by fixing the relevant point $\mathbf{p}$ on the object, then considering the map taking the *parameters* to the image plane — the derivative of $\mathcal{T}_{s\to I}(\mathbf{p};\cdot)$. This is important, because the flow $\nabla_\theta \mathcal{T}_{s\to I} \cdot \Delta\theta$ is a *linear* function of $\Delta\theta$. We now have the outline of a tracking algorithm:

- Start at frame $n = 0$ and some known configuration $\theta_0 = \hat{\theta}$.
- **Fit:** Fit the best value of $\Delta\theta$ to the flow between the frame n and frame $n+1$ using the flow model given by the derivative evaluated at θ_n.

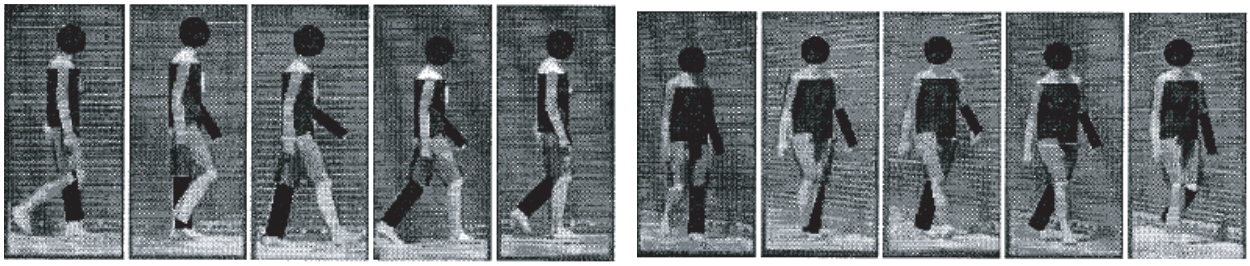


Figure 1.10: **Figure 7 from Bregler and Malik CVPR paper "Tracking people with twists and exponential maps" no permission yet** *Bregler and Malik formulate parametric flow models by modelling a person as a kinematic chain and then differentiating the maps from segment to image. They then track by searching for the parameter update that best aligns the current image pixels with those of the previous frame under this flow model. There is no dynamical model. This means that complex legacy footage, like these frames from the photographs of Eduard Muybridge [262, 261], can be tracked. Muybridge's frames are difficult to track because the frame-frame timing is not exact, and the figures can move in quite complex ways (see figure* **??***).*

- **Update:** Update the parameters by $\theta_{n+1} = \theta_n + \Delta\theta$ and set n to $n + 1$.

This should be seen as a primitive integrator, using Euler's method and inheriting all the problems that come with it. This view confirms the reasonable suspicion that fast movements are unlikely to be tracked well unless that sampling rate is high.

1.3.5.2 The Flow Model from the Chain Rule

In the special case of segments lying on a **kinematic tree** — a series of links attached by joints of known parametric form, where there are no loops — the chain rule means that the derivative takes a special form. Each segment in a kinematic tree has its own coordinate system, and the joint is represented by a map from a link's world coordinate system to that of its parent. The parent of segment k is segment $k - 1$. They are connected by a joint whose parameters at frame n are $\theta_{k,n}$. In general, in a kinematic tree, points on segments are affected by parameters at joints above them in the tree. Furthermore, we can obtain a transformation to the image by recursively concatenating transformations. Write the camera as $\mathcal{T}_{w\to i}$. Then the transformation taking a point of link k in frame n to the image can be written as

$$\mathcal{T}_{k\to i} = \mathcal{T}_{w\to i} \circ \mathcal{T}_{k-1\to w} \circ \mathcal{T}_{k\to k-1}$$

Notice that the only transformation that depends on $\theta_{k,n}$ here is $\mathcal{T}_{k\to k-1}$.

There is an advantage to changing notation at this point. Write $\mathcal{T}_{k\to k-1}$ as $\mathcal{T}_k$. The root of the tree is at segment one, and we can write $\mathcal{T}_{1\to w}$ as $\mathcal{T}_1$ and $\mathcal{T}_{w\to i}$ as $\mathcal{T}_0$. We continue to divide up the parameters θ into components, $\theta_{k,n}$ being the components associated with segment k in the n'th frame ($\theta_{0,n}$ are viewing parameters in frame n). We can now see the map from point $\mathbf{p}$ on segment k to the image as

$$\mathcal{T}_{k\to i}(\mathbf{p};\theta) = \mathcal{T}_0(\mathcal{T}_1(\mathcal{T}_2(\ldots;\theta_2);\theta_1);\theta_0)$$

This is somewhat inconvenient to write out, and it is helpful to keep track of intermediate values. Introduce the notation

$$\mathbf{p}_l = \mathcal{T}_{k\to l}(\mathbf{p};\theta)$$

for the point $\mathbf{p}$ in the coordinate system of the l'th link.

Our transformations have two types of argument: the points in space, and the camera parameters. It is useful to distinguish between two types of derivative. Write the partial derivative of a transformation $\mathcal{T}$ *with respect to its spatial arguments* as $D\mathcal{T}$. In coordinates, $\mathcal{T}$ would take the form $(f_1(x_1, x_2, x_3, \theta), f_2(x_1, x_2, x_3, \theta), f_3(x_1, x_2, x_3, \theta))$, and this derivative would be the matrix whose i, j'th element is $\partial f_i/\partial x_j$. Similarly, write the partial derivative of a transformation $\mathcal{T}$ *with respect to parameters* θ as D_θ. If we regard θ as a vector of parameters whose j'th entry is θ_j, then in coordinates this derivative would be the matrix whose i, j'th element is $\partial f_i/\partial \theta_j$.

This orgy of notation leads to a simple form for the flow. Write the flow at point $\mathbf{x}$ — which is the image of point $\mathbf{p}$ on segment k — in frame n as $\mathbf{v}(\mathbf{x}, \theta_n)$. Then

$$\mathbf{v}(\mathbf{x}, \theta_n) = D_\theta \mathcal{T}_0(\mathbf{p}_0; \theta_0) \cdot \Delta\theta_0 + D_x \mathcal{T}_0 \circ D_\theta \mathcal{T}_1(\mathbf{p}_1; \theta_1)\Delta\theta_1 \ldots + D_x \mathcal{T}_0 \circ D_x \mathcal{T}_1 \circ \ldots D_x \mathcal{T}_{k-1} \circ D_\theta \mathcal{T}_k(\mathbf{p}; \theta_k)\Delta\theta_k$$

Our indexing scheme hasn't taken into account the fact that we're dealing with a tree, but this doesn't matter; we need care only about links on the path from the relevant segment to the root. Furthermore, there is a relatively efficient algorithm for computing this derivative. We pass from the leaves to the root computing intermediate configurations $\mathbf{p}_l$ for each point $\mathbf{p}$ and the relevant parameter derivatives. We then pass from the root to the leaves concatenating spatial derivatives and summing.

1.3.5.3 Rigid-body Transformations

All the above takes a convenient and simple form for rigid-body transformations (which are likely to be the main interest in human tracking). We use homogeneous coordinates to represent points in 3D, and so a rigid body transformation takes the form

$$\mathcal{T}(\mathbf{p}, \theta) = \begin{bmatrix} \mathcal{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p}$$

where $\mathcal{R}$ is an orthonormal matrix with determinant one (a rotation matrix). The parameters are the parameters of the rotation matrix and the coefficients of the vector $\mathbf{t}$. This means the spatial derivative is the same as the transformation, which is convenient.

The derivatives with respect to the parameters are also relatively easily dealt with. Recall the definition of the **matrix exponential** as an infinite sum,

$$\exp(\mathcal{M}) = \mathcal{I} + \mathcal{M} + \frac{1}{2}\mathcal{M}^2 + \mathcal{M}^3 + \ldots + \frac{1}{n}\mathcal{M}^n \ldots$$

where the sum exists. Now it is straightforward to demonstrate that if

$$\mathcal{M} = \begin{bmatrix} \mathcal{A} & \mathbf{t} \\ \mathbf{0} & 0 \end{bmatrix}$$

and if $\mathcal{A}$ is antisymmetric, then $\exp(\mathcal{M})$ is a rigid-body transformation. The elements of the antisymmetric matrix parametrize the rotation, and the rightmost column is the translation. This is useful, because

$$\frac{\partial \left(\exp \mathcal{M}(\theta)\right)}{\partial \theta} = \left(\frac{\partial \mathcal{M}(\theta)}{\partial \theta}\right) \exp \mathcal{M}(\theta)$$

which gives straightforward forms for the parameter derivatives.

1.4 Tracking with Probability

It is convenient to see tracking as a probabilistic inference problem. In particular, we have a sequence of states $X_0, X_1, ..., X_N$ produced by some dynamical process. These states are unknown — they are sometimes called **hidden states** for this reason — but there are measurements $Y_0, Y_1, ..., Y_N$. Two problems follow naturally:

- **Tracking**, where we wish to determine some representation of $P(X_k|Y_0, ..., Y_k)$;
- **Filtering**, where we wish to determine some representation of $P(X_k|Y_0, ..., Y_N)$ (i.e. we get to use "future" measurements to infer the state).

These problems are massively simplified by two important assumptions.

- We assume measurements depend only the hidden state, that is, that $P(Y_k|X_0, ..., X_N, Y_0, ..., Y_N) = P(Y_k|X_k)$.
- We assume that the probability density for a new state is a function only of the previous state; that is, $P(X_k|X_0, ..., X_{k-1}) = P(X_k|X_{k-1})$, or, equivalently, that X_i form a **Markov chain**.

Now tracking involves three steps:

Prediction: where we construct some prediction of the future state given past measurements, or equivalently, construct a representation of $P(X_k|Y_0, ..., Y_{k-1})$. Straightforward manipulation of probability combined with the assumptions above yields that

$$P(X_k|Y_0, ..., Y_{k-1}) = \int P(X_k|X_{k-1})P(X_{k-1}|Y_0, ..., Y_{k-1})dX_{k-1}$$

Data association: where we use the predictive density — which is sometimes called the **prior** — and anything else likely to be helpful, to determine which of a pool of measurements contribute to the value of Y_k.

Correction: where we incorporate the new measurement into what is known, or, equivalently, construct a representation of $P(X_k|Y_0, ..., Y_k)$. Straightforward manipulation of probability combined with the assumptions above yields that

$$P(X_k|Y_0, ..., Y_k) = \frac{P(Y_k|X_k)P(X_k|Y_0, ..., Y_{k-1})}{\int P(Y_k|X_k)P(X_k|Y_0, ..., Y_{k-1})dX_k}$$

1.5 Linear Dynamics and the Kalman Filter

We write $X \sim N(\mu; \Sigma)$ to mean that X is a normal random variable with mean μ and covariance Σ. If the dynamics of the process we are observing are given by applying a linear map, then adding gaussian noise *and* our observations are obtained as a linear map of the state with gaussian noise added, we can write

$$X_k \sim N(\mathcal{A}_k X_{k-1}; \Sigma_k^{(d)})$$

and

$$Y_k \sim N(\mathcal{B}_k X_k; \Sigma_k^{(m)})$$

In this case, it is straightforward to demonstrate that all densities encountered are normal (or gaussians). This means that they can be represented by mean and covariance alone. In particular, both tracking and filtering boil down to producing update rules for estimates of mean and covariance of the hidden states.

Notation: We will represent the mean of $P(X_i|y_0, \ldots, y_{i-1})$ as $\overline{X}_i^-$ and the mean of $P(X_i|y_0, \ldots, y_i)$ as $\overline{X}_i^+$ — the superscripts suggest that they represent our belief about X_i immediately before and immediately after the i'th measurement arrives. Similarly, we will represent the standard deviation of $P(X_i|y_0, \ldots, y_{i-1})$ as Σ_i^- and of $P(X_i|y_0, \ldots, y_i)$ as Σ_i^+. In each case, we will assume that we know $P(X_{i-1}|y_0, \ldots, y_{i-1})$, meaning that we know $\overline{X}_{i-1}^+$ and Σ_{i-1}^+.

Prediction: Now assume we possess $\overline{X}_i^+$ and Σ_i^+. We wish to generate $\overline{X}_{i+1}^-$ and Σ_{i+1}^-. These are easy to get, because X_{i+1} is obtained by applying a linear map ($\mathcal{A}_{i+1}$) to X_i and adding zero mean noise. Recall that if U, V are random variables with mean $\overline{U}, \overline{V}$ and covariances Σ_U, ΣV respectively, and c_1, c_2 are constants, then $c_1U + c_2V$ is a random variable with mean $c_1\overline{U} + c_2\overline{V}$ and covariances $c_1^2\Sigma_U + c_2^2\Sigma_V$. This means that $\overline{X}_{i+1}^- = \mathcal{A}_{i+1}\overline{X}_i^+$ and $\Sigma_{i+1}^- = \mathcal{A}_{i+1}^T\Sigma_i^+\mathcal{A}_{i+1} + \Sigma_{i+1}^{(d)}$.

Correction: The next step is to incorporate the measurement to generate $\overline{X}_{i+1}^+$ and Σ_{i+1}^+. While the derivation of this step is relatively straightforward, it does require some notation, and we skip it for reasons of space; instead, we simply state the result in algorithm **??**.

Dynamic Model:

$$\begin{aligned} \mathbf{x}_i &\sim N(\mathcal{D}_i\mathbf{x}_{i-1}, \Sigma_{d_i}) \\ \mathbf{y}_i &\sim N(\mathcal{M}_i\mathbf{x}_i, \Sigma_{m_i}) \end{aligned}$$

Start Assumptions: $\overline{\mathbf{x}}_0^-$ and Σ_0^- are known
Update Equations: Prediction

$$\begin{aligned} \overline{\mathbf{x}}_i^- &= \mathcal{D}_i\overline{\mathbf{x}}_{i-1}^+ \\ \Sigma_i^- &= \Sigma_{d_i} + \mathcal{D}_i\sigma_{i-1}^+\mathcal{D}_i \end{aligned}$$

Update Equations: Correction

$$\begin{aligned} \mathcal{K}_i &= \Sigma_i^-\mathcal{M}_i^T\left[\mathcal{M}_i\Sigma_i^-\mathcal{M}_i^T + \Sigma_{m_i}\right]^{-1} \\ \overline{\mathbf{x}}_i^+ &= \overline{\mathbf{x}}_i^- + \mathcal{K}_i\left[\mathbf{y}_i - \mathcal{M}_i\overline{\mathbf{x}}_i^-\right] \\ \Sigma_i^+ &= \left[Id - \mathcal{K}_i\mathcal{M}_i\right]\Sigma_i^- \end{aligned}$$

Algorithm 2: *The Kalman filter updates estimates of the mean and covariance of the various distributions encountered while tracking a state variable of some fixed dimension using the given dynamic model.*

Filtering: In some applications, we must report a revised estimate of state as soon as possible after a measurement arrives, but in others one can delay reporting this estimate. If we can, we can improve the estimate by working with "future" measurements. Future measurements might affect the current estimate of state, for example, when they are in agreement with a slightly different estimate of the position of a point. Ideally, we would use all measurements and represent $P(\mathbf{X}_i|\mathbf{y}_0, \ldots, \mathbf{y}_N)$, but sometimes it is sufficient to use a small number of future measurements and work with $P(\mathbf{X}_i|\mathbf{y}_0, \ldots, \mathbf{y}_{i+k})$. This pro-

cess is known as **filtering**. There is a straightforward trick for using future measurements in a Kalman filter. First, notice that there is no reason to care about the case where $\mathcal{A}_k$ does not have full rank. This means we can run time backward, with a dynamical model

$$X_{k-1} \sim N(\mathcal{A}_k^{-1} X_k; \Sigma_{k-1}^{(d)})$$

and without changing the measurement model. Write the mean and variance of $P(\mathbf{X}_i|\mathbf{y}_{i+1}, \ldots, \mathbf{y}_N)$ as $\overline{\mathbf{X}}_i^{-b}$ and Σ_i^{-b} (the "b" for backwards) and the mean and variance of $P(\mathbf{X}_i|\mathbf{y}_i, \ldots, \mathbf{y}_N)$ as $\overline{\mathbf{X}}_i^{+b}$ and Σ_i^{+b}. Now we wish to obtain a mean and variance for $P(\mathbf{X}_i|\mathbf{y}_0, \ldots, \mathbf{y}_N)$, which we write as $\overline{X}_i^f$ and Σ_i^f. Now notice that, from the point of view of the filter running forward in time, the state estimate produced by the filter running backward is, essentially, another measurement. The measurement matrix is now the identity. We can now use the Kalman equations to combine $\overline{\mathbf{X}}_i^+$ (the forward estimate of state, representing $P(\mathbf{X}_i|\mathbf{y}_0, \ldots, \mathbf{y}_i)$) and $\overline{\mathbf{X}}_i^{-b}$ (the backward estimate of state, representing $P(\mathbf{X}_i|\mathbf{y}_{i+1}, \ldots, \mathbf{y}_N)$). We use $\overline{\mathbf{X}}_i^{-b}$, rather than $\overline{\mathbf{X}}_i^{+b}$, to avoid using the same measurement twice.

1.5.1 People Tracking with a Kalman Filter

```
Blake Hand tracker gate; Hogg body tracker gate?; Cipolla?
Other examples?
Hogg, Rohr, O'RourkeBadler, BlakesLip trackerr
```

1.6 Data Association and Multi-modal Densities

Data association involves determining which pixels or image measurements should contribute to a track. That data association is a nuisance is a persistent theme of this work. Data association is genuinely difficult to handle satisfactorily — after all, determining which pixels contribute to which decision seems to be a core — and often very difficult — computer vision problem. The problem is usually particularly difficult when one wishes to track people, for several reasons. First, standard data association techniques aren't really all that much help, as for almost every aspect the image domain covered by a person changes shape very aggressively, and can do so very fast. Second, there seem to be a lot of background objects that look like some human body parts; for example, kinematic tracking of humans in office scenes is very often complicated by the fact that many book spines (or book shelves) can look like forearms.

In tracking by detection, almost all computation is directed at data association, which is achieved by minimizing ρ with respect to the template's parameters — the support of ρ identifies the relevant pixels. Similarly, in tracking using flow, data association is achieved by choosing the parameters of a flow model to get a good match between domains in frames n and $n+1$ — the definition of the domain cuts out the relevant pixels. When these methods run awry, it is because these data association methods have failed. Either one cannot find the template, or one cannot get good parameters for the flow model.

There are a variety of simple data association strategies which exploit the presence of probability models. In particular, we have an estimate of $P(X_n|Y_0, ..., Y_{n-1})$ and we know $P(Y_n|X_n)$. From this we can obtain an estimate of $P(Y_n|Y_0, ...Y_{n-1})$, which gives us hints as to where the measurement might be.

One can use a **gate** — we look only at measurements that lie in a domain where $P(Y_n|Y_0, ..., Y_{n-1})$ is big enough. This is a method with roots in radar tracking of missiles and aeroplanes, where one must

deal with only a small number (compared with the number of pixels in an image!) of returns, but the idea has been useful in visual tracking applications.

One can use **nearest neighbours**. In the classical version, we have a small set of possible measurements, and we choose the measurement with the largest value of $P(Y_n|Y_0, ..., Y_{n-1})$. This has all the dangers of wishful thinking — we are deciding that a measurement is valid because it is consistent with our track — but is often useful in practice (see chapter *** of [120] for an example of how badly the method can behave). This strategy doesn't apply to most cases of tracking people in video because the search to find the maximising Y_n — which would likely be an image region — could be too difficult (but see section 3). However, it could be applied when one is tracking markers attached to the body — in this case, we need to know which marker is which, and this information could be obtained by allocating a measurement to the marker whose predicted position is closest.

One can use **probabilistic data association**, where we use a weighted combination of measurements within a gate, weighted using (a) the predicted measurement and (b) the probability a measurement has dropped out. Again, this method has the dangers of wishful thinking, and again does not apply to most cases of tracking people; however, it could again be applied when one is tracking markers attached to the body.

1.6.1 Multiple Modes

The Kalman filter is the workhorse of estimation, and can give useful results under many conditions. One doesn't need a guarantee of linearity to use a Kalman filter — if the logic of the application indicates that a linear model is reasonable, there is a good chance a Kalman filter will work. However, difficulties arise if relevant densities have multiple modes, because the representation adopted by a Kalman filter (the mean and covariance, sufficient statistics for a Gaussian distribution) tends to represent multimodal distributions poorly. There are several reasons one might encounter multiple modes.

First, nonlinear dynamics — or nonlinear measurement processes, or both — can create serious problems. The basic difficulty is that even quite innocuous looking setups can produce densities that are not normal, and are very difficult to represent and model. For example, let us look at only the hidden state. Assume that this is one dimensional. Now assume that state updates are *deterministic*, with $X_{i+1} = X_i + \epsilon \sin(X_i)$. If ϵ is sufficiently small, we have that for $0 < X_i < \pi$, $X_i < X_{i+1} < \pi$; for $-\pi < X_i < 0$, $-\pi < X_{i+1} < X_i$; and so on. Now assume that $P(X_0)$ is normal. For sufficiently large k, $P(X_k)$ will not be; there will be "clumps" of probability centered around the points $(2j + 1)\pi$ for j an integer. These clumps will be very difficult to represent, particularly if $P(X_0)$ has very large variance so that many clumps are important. Notice that what is creating a problem here is that quite small non-linearities in dynamics can cause probability to be concentrated in ways that are very difficult to represent. In particular, nonlinear dynamics are likely to produce densities with complicated sufficient statistics. There are cases where nonlinear dynamics does lead to densities that can be guaranteed to have finite-dimensional sufficient statistics (see [**?**, **?**, **?**]); to our knowledge, these have not been applied to human tracking.

Second, there are practical phenomena in human tracking that tend to suggest that non-normal distributions are a significant part of the problem (none of the examples we know is compelling proof that a multi-modal density occurs). Assume we wish to track a 3D model of an arm in a single image. The elbow is bent; as it straightens, it will eventually run into an **end-stop** — the forearm can't rotate further without damage. At the end-stop, the posterior on state can't be a normal distribution, because a normal distribution would have some support on the wrong side of the end-stop, and this has a significant effect on the shape of the posterior (see figure 2.5). Another case that is likely, but not guaranteed, to cause

trouble is a **kinematic singularity**. For example, if the elbow is bent, we will be able to observe rotation about the humerus, but current observation models will make this unobservable if the elbow is straight (because the outline of the arm will not change; no current method can use the changes in appearance of the hand that will result). The dimension of the state space has collapsed. The posterior might be a normal distribution in this reduced dimension space, but that would require explicitly representing the collapse. The alternative, a covariance matrix of reduced rank, creates unattractive problems of representation. Deutscher *et al.* produce evidence that, in both cases, posteriors are not, in fact, normal distributions, and show that an extended Kalman filter can lose track in these cases [92]. In fact, as we discuss in section **??**, evidence for the presence of multiple modes in people tracking problems is moot for 2D representations. Some of the difficulties with multiple modes come from kinematic ambiguity in a 2D-3D lifting process; others appear to come from data association problems.

1.6.2 Multiple Modes from Data Association

The richest source of multiple modes is data association problems. An easy example illustrates how nasty this problem can be. Assume we have a problem with linear dynamics and a linear measurement model. However, at each tick of the clock we receive more than one measurement, exactly one of which comes from the process being studied. We will continue to write the states as $\mathbf{X}_i$, the measurements as $\mathbf{Y}_i$; but we now have δ_i, an indicator variable that tells which measurement comes from the process (and is unknown). $P(\mathbf{X}_N|\mathbf{Y}_{1..N}, \delta_{1..N})$ is clearly Gaussian. We want $P(\mathbf{X}_N|\mathbf{Y}_{1..N}) = \sum_{histories} P(\mathbf{X}_N|\mathbf{Y}_{1..N}, \delta_{1..N})P(\delta_{1..N}|\mathbf{Y}_{1..N})$, which is clearly a mixture of Gaussians. The number of components is exponential in the number of frames — there is one component per history — meaning that $P(\mathbf{X}_N|\mathbf{Y}_{1..N})$ could have a very large number of modes.

1.6.2.1 The Kalman Filter with Noisy Measurements

For the moment, assume that the exponential number of histories presents no problem. We can apply the same reasoning we used for Kalman filtering to maintain a representation of $P(\mathbf{X}_N|\mathbf{Y}_{1..N})$. First, assume we have a representation of $P(\mathbf{X}_{N-1}|\mathbf{Y}_{1..N-1})$. Prediction presents no problem, because δ_N is not involved. We have

$$P(\mathbf{X}_N|\mathbf{Y}_{1..N-1}) = \int P(\mathbf{X}_N|\mathbf{X}_{N-1})$$

We have already shown that $P(\mathbf{X}_{N-1}|\mathbf{Y}_{1..N-1})$ must be a mixture of Gaussians, so $P(\mathbf{X}_N|\mathbf{Y}_{1..N-1})$ must be too if the dynamics are linear. Integration is linear, so the prediction step just involves updating the mean and covariance of each component using the rules above; the weights are not touched.

Correction is much more interesting. $\mathbf{Y}_N$ is a vector of measurements, some of which come from the object being tracked and some of which come from the noise process. We assume that the noise process is some known Gaussian, which is the same for each noise point — it might have very large covariance, so as to be approximately uniform over the domain. We know $P(\mathbf{Y}_N|\delta_N = k, \mathbf{X}_N)$ — the indicator variable says the k'th block of components of $\mathbf{Y}_N$ come from the object being tracked, and the others do not, so this density is Gaussian. $P(\delta_N = k|\mathbf{X}_N) = P(\delta_N = k)$ (because the noise process doesn't know anything about the state) and is uniform, because we have no prior basis to believe that the measurements have been sorted in some order. Then we have

$$P(\mathbf{X}_N|\mathbf{Y}_{1..N}) = \frac{\sum_{k \in \text{cases}} P(\mathbf{Y}_N|\delta_N = k, \mathbf{X}_N)P(\mathbf{X}_N|\mathbf{Y}_{1..N-1})}{\int \sum_{k \in \text{cases}} P(\mathbf{Y}_N|\delta_N = k, \mathbf{X}_N)P(\mathbf{X}_N|\mathbf{Y}_{1..N-1})d\mathbf{X}_N}$$

Now the numerator is a mixture of Gaussians. This is because each term in the sum is a mixture of Gaussians (which typically differ from term to term). The terms are a mixture of Gaussians, because $P(\mathbf{X}_N|\mathbf{Y}_{1..N-1})$ is a mixture of Gaussians and $P(\mathbf{Y}_N|\delta_N = k, \mathbf{X}_N)$ is Gaussian.

There are hard cases and easy cases in this example. The easy cases occur when the erroneous measurements are spaced very far away from the measurements predicted by the state model. In this case, we could use nearest neighbours to identify the right measurement, and need only try each measurement for the first frame to find a start point. Another easy case occurs when there are very few frames, and we can represent the mixture explicitly. In more difficult cases, we will be compelled to assume that most of the probability is concentrated in a tractable number of modes, however long the history. The result is a multiple hypothesis tracker — one Kalman filter keeps track of each of a fixed number mode, and we must determine methods to prune the number of modes. This method was used by Cham and Rehg [65], to track a 2D kinematic model of the body — note that this was adopted explicitly to cope with data association difficulties, and they make no argument that posteriors for 2D human tracking are intrinsically multimodal.

1.7 Tracking in the Presence of Multimodal Distributions

Assume we have case where the logic of the application suggests the number of clumps in the posterior is likely to be manageable. Although we regard the evidence that there are multiple modes in people tracking as moot (section **??**), there is little doubt that the posterior in this case has relatively few clumps, and that they are manageable. Algorithms for approximate inference in such cases build and manage models of the posterior. This presents some practical difficulties; for example, if we model the posterior at frame n as a mixture of Gaussians, we face the prospect that any, or each, of these Gaussians might be split by the dynamics and lead to multiple lumps in the posterior at frame $n + 1$.

Generally, to maintain a representation of the posterior, we are going to need to propagate the representation through the dynamics (prediction, above, which involves computing an integral), and then adjust the representation to take account of our measurements. Furthermore, if our representation involves a mixture, we may need to prune it to control complexity (because "lumps" of probability may split, as in the example above). When we prune, we must choose the remaining mixture elements and their weight so as to obtain a "good" approximation to the true posterior.

1.7.1 Kalman Filters – MM and IMM

There is no prospect in practice of filtering using a complete representation of the posterior, because the number of terms grows so fast. Instead, we must manage the complexity of the mixture. A natural strategy is to use a fixed number of terms in the mixture. We take the prior and compute the weights for all terms in the posterior, then prune terms with small weights and reweight the remaining terms. This case is not usually handled in tracking textbooks, which tend to concentrate on unknown dynamics (which leads to similar issues — our history is now the dynamical model applied at each past step, rather than the choice of measurement). There are two standard strategies, IMM and `what?`
[].

`Rehg used IMM? No, MHT`

1.8 Notes

The Hausdorff distance was extensively studied in the 90's [168, 327, 165, 166], and appears to have fallen out of favour since then, for reasons we can only guess at. A likely source of difficulty (which applies to any edge-based matching technique) is that scoring edge points against a template tends to perform poorly in the presence of strong natural texture, whatever the score (e.g. the discussion of verification in chapter *** of [120]). Interest in edges and edge-based methods is reviving, and we might see Hausdorff distance papers again. There are relatively few trackers, but the method was widely used for matching [59], logo recognition [67, **?**] and for face recognition [**?**, 363, 378].

Flow-based tracking maintains an appearance model implicitly — one models the appearance of a segment in a frame as the appearance most like that of the domain in the previous frame, accepting the constraints of the flow model and some noise. The difficulty is that this can drift catastrophically. For example, let a limb segment become occluded; then the appearance trackers we have described will find some image patch most like the segment, given flow and structural constraints — but this could be quite different from what the segment actually looks like. When the segment reappears, we may be unable to find it, because our segment model is now quite different from the appearance of the actual segment. Two natural strategies apply. First, we might wish to use a fixed model of segment appearance, to avoid this drift effect. Second, we might hope that better formulated dynamical constraints could help.

There is a very large range of applicable templates (for example, one might use the machinery of [113, **?**]) each of which has a somewhat different detailed form of ρ (for example, we might interpret ρ as a metric [] or as a likelihood []). Typically, we detect by identifying values$\hat{\theta}$ such that $\rho(T(\mathbf{x}|\hat{\theta}), I_n)$ (a) is a local minimum and (b) is less than some threshold. This is sufficient to build a tracker, if we ignore dynamical constraints. If the template is very reliable, it may be sufficient to detect instances and link them over time (this can work quite well for face tracking in simple circumstances; see, for example, [**?**]).

Chapter 2

Tracking: Relations between 3D and 2D

Many applications require a representation of the body in three dimensions. Such a track could come from tracking with a 3D representation — perhaps a set of body segments in 3D, modelled as surfaces, triangle meshes or sample points — or by building a kinematic track in two dimensions, then "lifting" it to produce a 3D track. If there is only one camera, relations between the 2D figure and the 3D track are complicated, and may be ambiguous. Ambiguities appear to be less significant in the case where there are multiple cameras; we review this case only briefly (section 2.1).

The heart of the question is the number of possible 3D configurations that could explain a single image. There is no doubt that there are many *if* there is no motion information and *if* only geometric correspondence information is used. In other cases, whether there is any ambiguity is uncertain, and appears to depend quite precisely on the circumstances of measurement. When reconstruction is ambiguous, one expects to encounter multimodal distributions in a tracking problem built around 3D representations, because several distinct inferred 3D configurations could have the same likelihood.

We discuss methods for reconstructing body configuration in 3D from a single view (and perhaps a dynamical history) in considerable detail in section 2.2. This leads us to the (surprisingly complex and unresolved) question of the extent and nature of ambiguities in 3D reconstruction (section 6.0.1.1). Furthermore, the lifting procedure adopted may have significant consequences for how we choose to represent body configuration (section 6.0.1.2). All this provides background information to understand tracking methods that work on 3D body models (section 3).

2.1 Kinematic Inference with Multiple Views

If one has multiple views of the body, the problem of reconstruction is considerably simplified. Ideally, the cameras are calibrated, in which case the main difficulty is localizing body parts. At least conceptually, one could lift from one frame using some method chosen from section 2.2, then search all ambiguities, evaluating by backprojecting into the other view. It is more sensible to search configurations with a cost function incorporating all views; this requires the cameras be calibrated. There are generally two questions: the score used to evaluate a particular reconstruction, and how to search for the best reconstruction.

Scores can be computed by explicitly reconstructing a three-dimensional structure from the views, then comparing the body representation to that structure. Cheung *et al*.use a volumetric reconstruction of the person — a quantized approximatation to the visual hull — obtained using five views, and then encode kinematic configuration by fitting a set of ellipsoids to the 3D reconstruction with EM [69];

the process is realtime. Kehl *et al*.use an approximate visual hull, estimated by intersecting cones over foreground regions from between 4 and 8 calibrated cameras [194] (figure 2.1). The reconstruction is produced assuming a simple background, so that the cones can be obtained. The body model is a textured 3D mesh, controlled by a skeleton (section **??**); texture maps are obtained from a modelling view. Tracking is by minimizing distance between the volumetric reconstruction and sample points on the mesh (which are a function of the skeleton's kinematic parameters). The minimization procedure itself is a sophisticated variant of stochastic gradient descent.

```
   Luck, Small
Black
```

It is not necessary to construct the visual hull explicitly. There are numerous methods that use the visual hull implicitly, by comparing the reconstructed 3D model with the silhouette in each view. Carranza *et al*.use an implicit representation, comparing the silhouette of the 3D reconstruction with silhouettes in each view using graphics hardware [63]. This yields a cost function that can be evaluated very fast, allowing real-time tracking.

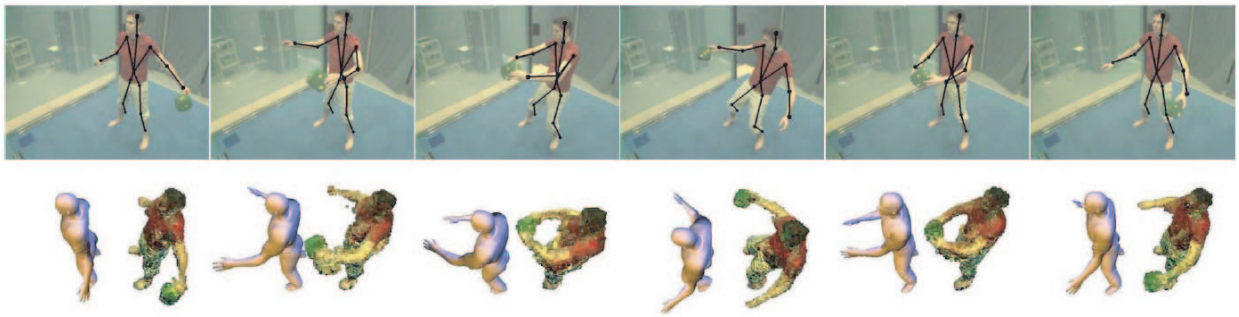


Figure 2.1: **figure 12 of kehl ea, full body tracking using multiple views, multipleview/01467432** *Kehl* et al.*represent the body as a textured 3D mesh, controlled by a skeleton with a texture map obtained from a modelling view. They obtain a volumetric reconstruction from a set of calibrated cameras, then track the body by minimizing distance between sampling points on the mesh and the volumetric reconstruction. The* **top row** *shows frames from one camera with reprojected skeleton superimposed; the* **bottom row** *shows the surface reconstruction at the* **left** *of each frame and the original volumetric reconstruction at the* **right**. *The reconstruction is accurate, despite some difficulties in the volumetric measurement.*

Stereo matches can give greater depth precision than the visual hull can provide. Plänkers and Fua estimate parameters for a model of the body consisting of a skeleton, metaball muscle model, and skin using stereo and, optionally, silhouette information [292]; the method appears to work with a complex background. Delamarre and Faugeras use a form of iterated closest point matching to produce forces that drive a 3D segment model into correspondence with the silhouette in three calibrated views [88, 89]. Drummond and Cipolla model the body with quadric segments, and track by applying a linearized flow model (as per section 1.3.5; [51, 52]) to a search for edge points close to projected sample points on the model [98] (see also [97] for more information on the formalism, and [96, 99] for information about tracking changes in camera parameters). Shahrokni *et al*.use a similar general approach, but employ a novel texture segmentation model to find silhouette points [339]. They search along a scan line near and approximately normal to the predicted silhouette to find points where there is a high posterior of a texture edge (see also an alternative method for finding texture silhouettes using a classifier in [340];

and using an entropy measure in [338]).

Texture information can be registered to the body model. Starck and Hilton obtain the best configuration of a 17 joint, meshed 3D model of the human body to fit stereo, silhouette and feature matches for each frame; texture is then reprojected onto the body (in [**?**]; see also [151, 366]). The texture is then backprojected onto the reconstruction and composited to give a single texture map. In recent work, Starck and Hilton show that correspondences between texture maps induced in separate frames yield temporal correspondences and so information on how relevant surfaces deform [364]. Models of this form allow relatively straightforward synthesis of new views [365]. These methods are oriented to performance capture, and appear to have been demonstrated for simple backgrounds only.

In principle, texture information registered to the body should yield a match score and improve matches, if the texture does not move with respect to the skeleton. We are not aware of methods that use this cue, though it may prove useful if one wants a detailed surface reconstruction of a model wearing tight garments. However, one can use a flow model to register texture from frame to frame. Yamamoto *et al.*use a linear flow model derived from the kinematic model (cf section 1.3.5) with three cameras to obtain good tracks from hand-initialized data; they use three calibrated cameras [419]. The paper describes no difficulties resulting from movement of texture with respect to the body, but we expect that this effect significantly limits the precision of available reconstructions (see also figure 1.8, and the discussion in section 1.1). Theobalt *et al.*describe improved configurations obtained from the method of Carranza *et al.*([63]) by incorporating an optic flow model to correct the estimates of configuration [383]. Subjects are not wearing very tight clothing, and there again seem to be no difficulties resulting from movement of texture with respect to the body.

Generally, search methods involve either standard optimization techniques or fairly standard variants. However, Deutscher *et al.*use a form of randomized search, described in greater detail in section 2.3.1, to align a 3D model with silhouette edges [91, 93]. Sigal *et al.*use a form of belief propagation, described in greater detail in section 2.3.1.1, to infer configuration in three or four views; the method uses detectors to guide a form of search [346]. Carranza *et al.*use a surface model, controlled by a 17 joint skeleton [63]. The search for a reconstruction at a time instant uses the reconstruction at the previous instant as a start point; however, because motion can be fast, and the sampling rate is relatively slow (15 Hz, p 571), a form of grid search at each limb separately is necessary to avoid local minima. A texture estimate is obtained by rectifying all images to the surface model, and blending.

The most comprehensive and recent discussion of 3D reconstruction issues appears in two papers. Cheung *et al.*give an extensive discussion of representations of the visual hull and methods of obtaining them; the methods they describe can incorporate temporal information, color information, stereopsis and silhouette information [70]. Cheung *et al.*then use these methods to build a body model from a series of calibration sequences, which give both surface and skeleton information [71]. This model is then tracked by minimizing the sum of two scores. The first compares the deformed body model with the silhouettes in each image at a given timestep. The second compares an object reconstruction obtained at a given timestep with the silhouettes in each modelling frame. As authors note, there are 3D situations that are either kinematically ambiguous or at least very difficult for a tracking algorithm of this form. The first occurs when body parts are close together (for example, an arm pressed against the torso) and may lead to a self-intersecting reconstruction. This difficulty appears to be intrinsic to the use of silhouette features. The second occurs when the arm is straight, making rotation about the axis of the humerus ambiguous. The difficulty is that the photometric detail is too weak to force the method to the right configuration of the hand. Curiously, although Mori and Malik have shown that one can obtain landmark positions automatically [256], there appears to be no multiple view reconstruction work that identifies landmarks in several views (with, for example, the method of Mori and Malik, section 2.2.1)

and builds a geometric reconstruction this way.

2.2 Lifting to 3D

There are a variety of methods for lifting a 2D representation of the body to 3D. Different methods draw from different bodies of technique (kinematics, statistics, computational geometry, optimization, etc.), but the geometry of lifting gives clear bounds to what ambiguities may appear (subsection 2.2.1). The extent of ambiguity appears to depend on whether the ambiguous reconstructions violate kinematic constraints, and whether a dynamical history is available. The remarkable fact is that reconstruction ambiguity seems to be either quite easily evaded or not to manifest itself at all. Thus, while many papers advocate methods to manage ambiguity, almost any method appears to work — one doesn't see many records of systems failing due to ambiguity. This may be because experiments are poorly conducted; but it is more likely that the implicit folk mythology — that ambiguous reconstructions are quite easily avoided — is true. We discuss this point in section 6.0.1.1.

2.2.1 Geometric Ambiguity and Lifting by Kinematic Inference

The way that people are imaged means that there are very few cases where a scaled orthographic camera model is not appropriate. One such case to keep in mind is a person pointing towards the camera; if the hand is quite close, compared with the length of the arm, one may see distinct perspective effects over the hand and arm and in extreme cases the hand can occlude much of the body.

Regard each body segment as a cylinder, for the moment of known length. If we know the camera scale, and can mark each end of the body segment — we might do this by hand, as Taylor [379, 380] does and Barrón and Kakadiaris [23, 24] do, or by a strategy of matching image patches to marked up images as Mori and Malik do [256, 257] — then we know the cosine of the angle between the image plane and the axis of the segment, which means we have the segment in 3D up to a twofold ambiguity and translation in depth (figure 2.2 gives examples). We can reconstruct each separate segment and obtain an ambiguity of translation in depth (which is important and often forgotten) and a two-fold ambiguity at each segment.

For the moment, assume we know all segment lengths and the camera scale. We can now reconstruct the body by obtaining a reconstruction for each segment, and joining them up. Each segment has a single missing degree of freedom (depth), but the segments must join up, meaning that we have a discrete set of ambiguities. Depending on circumstances, one might work with from seven to nine body segments (the head is often omitted; the torso can reasonably be modelled with several segments), yielding from 128 to 512 possible reconstructions. These ambiguities persist for perspective images; examples appear in figure 2.4.

Barrón and Kakadiaris show that anthropometric parameters can be estimated as well [23, 24]. They do this by constructing a multivariate Gaussian prior on segment lengths, which do not vary much in size (a factor of 1.5 covers the range of human heights from four foot six to six foot nine, which deals with the vast majority of adults). Ratios of body segment lengths vary even less (e.g. see [273, 23, 24]). Barrón and Kakadiaris assume that, in any view, two segments are close to parallel to the image plane, meaning that the ratio of their image lengths is very close to the actual length ratio. They construct a discrete set of possible bodies, and use image length ratios to index into this set to obtain a start point for an optimization procedure that obtains the actual anthropometric parameters by choosing the set that agrees with the image, meets joint limit constraints, and has highest prior probability (this could be seen as an MAP estimate).

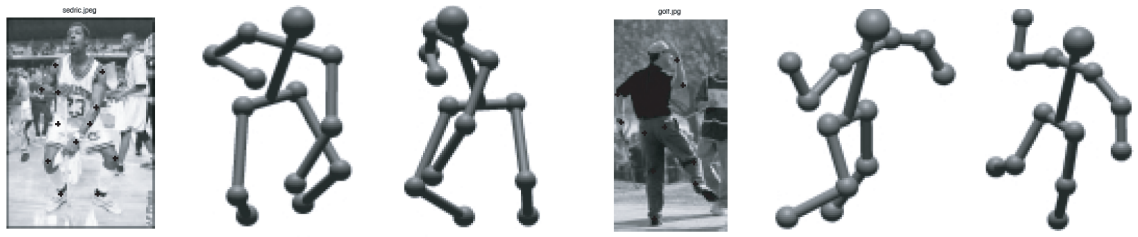


Figure 2.2: **2d3Dlift/cvpr00.pdf, figure 5, p 682, CJpaper** *Two 3D reconstructions obtained by Taylor [379], for single orthographic views of human figures. The image appears* **left***, with joint vertices on the body identified by hand (the user also identifies which vertex on each segment is closer to the camera).* **Center** *shows a rendered reconstruction in the viewing camera, and* **right** *shows a rendering from a different view direction.*

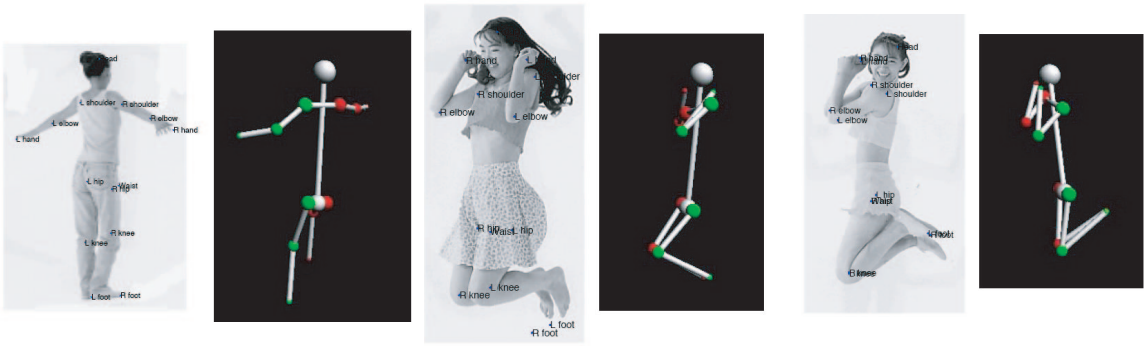


Figure 2.3: **Mori+Malik, 2d3dlift/morimecv01.pdf, figures 6 and 7** *Mori and Malik deal with discrete ambiguities by matching test image outlines to examplars, which have keypoints marked [256, 257]. The keypoint markup includes which end of the segment is closer to the view. The images on the* **left** *show example test images, with keypoints established by the matching strategy superimposed. The resulting reconstruction appears on the* **right***.*

The discrete ambiguities can be dealt with in a number of ways. One could ask the user to identify the closer endpoint of each segment (Taylor [379], p. 681). One could simply choose, as Barrón and Kakadiaris appear to do. In detail, their method uses each kinematically acceptable 3D reconstruction as a start point for the minimization procedure described above, and chooses the reconstruction with best value of the objective function. Since this procedure enforces kinematic constraints but does not apply distinct weights to distinct kinematic reconstructions, the unconstrained objective function must have a symmetry corresponding to the reconstruction ambiguity, and so the choice depends largely on random factors within the optimization procedure. It is important to notice that this doesn't seem to cause any problems, which suggests that substantial kinematic ambiguities might be rather rare. We will pick up on this point in section **??**.

Mori and Malik deal with discrete ambiguities by matching [256, 257]. They have a set of example images with joint positions marked. The outline of the body in each example is sampled, and each sample point is encoded with a **shape context** (an encoding that represents local image structure at high resolution and longer scale image structure at a lower resolution). Keypoints are marked in the examples by hand, and this marking includes a representation of which end of the body segment is closer to the

camera. The outline of the body is identified in a test image (Mori and Malik use an edge detector; a cluttered background might present issues here), and sample points on the outline are matched to sample points in examples. A global matching procedure then identifies appropriate examplars for each body segment and an appropriate 2D configuration. The body is represented as a set of segments, allowing (a) kinematic deformations in 2D and (b) different body segments in the test image to be matched to segments in different training images. The best matching example keypoint can be extracted from the matching procedure, and an estimate of the position of that keypoint in the test image is obtained from a least-squares fit transformation which aligns a number of sample points around that keypoint. The result is a markup of the test image with labelled joint positions and with which end of the segment is closest to the camera. A 3D reconstruction follows, as above (figure 2.3 gives some examples).

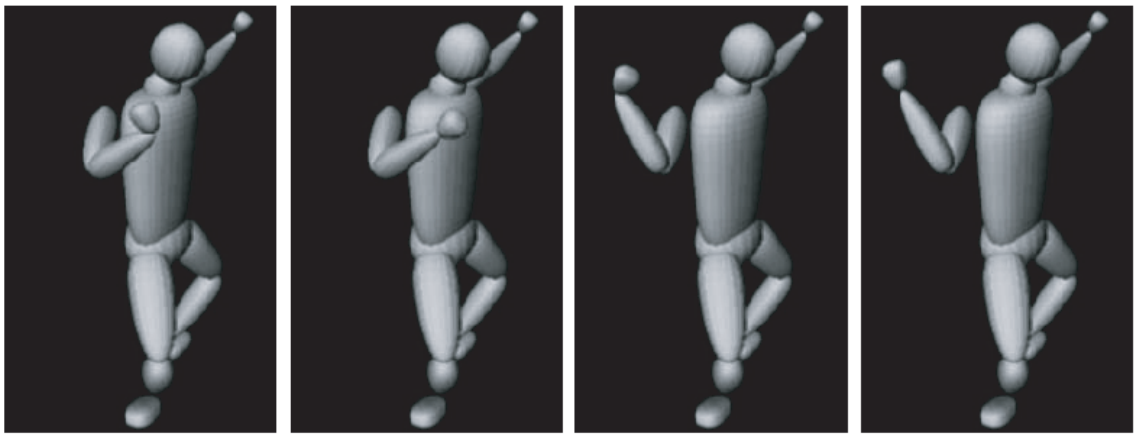


Figure 2.4: **figure 2 of sminchisescu+triggs, kinematic jump processes, kinematicambiguity/01211339** *Ambiguous reconstructions of a 3D figure, all consistent with a single view, from Sminchisescu and Triggs [357]. The ambiguities are most easily visualized by an argument about scaled orthographic cameras, given in the text, but persist for perspective views as these authors show. Note that the cocked wrist in the leftmost figure violates kinematic constraints — no person with an undamaged wrist can take this configuration.*

`fix this par`

Camera scale can be calibrated, and anthropometric parameters estimated, but the remaining discrete set of ambiguities appears to have substantial practical consequences. Furthermore, they appear to apply to whatever particular scheme of reconstruction is adopted, unless the measurement process is more informative than locating segment endpoints. Schemes that use the silhouette (section 2.6) and image matches (section **??** and section 2.2.3) appear to be subject to this set of ambiguities (section **??** discusses further observations that might help).

Current likelihood models compare some set of predicted with observed image features (typically, silhouette edges), and so must have multiple peaks corresponding to the ambiguities described. These peaks appear in the posterior (figure 2.5). While this makes the multiple peaks predictable, they are still a major nuisance. Typically, at each peak in the likelihood there are some directions where the value of the likelihood varies slowly (small eigenvalues in the Hessian). This is because localization of either landmarks or silhouette points is difficult, and large changes in the estimate of depth to a joint or of a limb angle can result in small changes to image positions. The problem directions tend to move a joint in depth (figure 2.4).

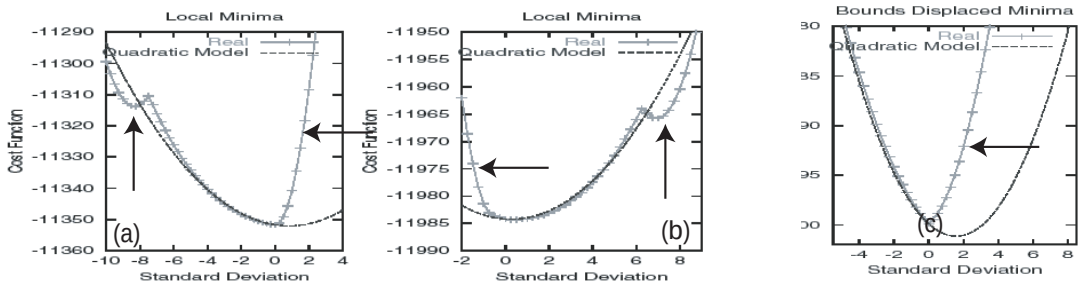


Figure 2.5: **Figure 4, 2a of Sminchisescu and Triggs, Covariance scaled..., particlefilter/009090509** *Several nasty phenomena result from kinematic ambiguities and from kinematic limits, as Sminchisescu and Triggs [353, 356]. Ambiguities in reconstruction — which are caused because body segments can be oriented in 3D either toward or away from the camera, as described in the text — result in multiple modes in the posterior. The two graphs on the* **left** *(a and b) show the fitting cost (which can be thought of as a log-likelihood) as a function of the value of two state variables (scaled by their standard deviation). The state variables refer to the kinematic configuration of the 3D model. Note the significant "bumps" from the second mode (the* **vertical arrows***). For reference, there is a quadratic approximation shown as well. Note also the the significant deformations of modes resulting from a kinematic limit (the* **horizontal arrows***). This is caused by the fact that no probability can lie on the other side of the limit, so the mode must be "squashed".*

2.2.2 Lifting by Minimization

As we have seen (section 2.1), if one has multiple views, the body configuration can be reconstructed by minimizing an error between the image and projected configuration in each view. A wide variety of view errors are available, though most involve a comparison between inferred outline points and an image silhouette. Sminchicescu and Telea show that this approach can produce a reconstruction from a single view [352] (see also [351]). Their error function includes a term to force the projected body to cover as much silhouette as possible and a term to force the projected body inside the silhouette. It is important to smooth the silhouette (from background subtraction), because noise components on the silhouette boundary can produce a difficult optimization problem. The silhouette is skeletonized and the skeleton is then pruned and "inflated" using a form of distance transform. The method produces good reconstructions, but must experience at least reconstruction ambiguities similar to those experienced by kinematic inference.

Randomized search is a reasonable strategy for attacking the minimization. Sminchisescu and Triggs describe various methods to bias the likelihood function searched by a sampler so that the state will move freely between local minima [355, 354, 359]. Sminchisescu and Triggs exploit an explicit representation of kinematic ambiguities to help this search, by making proposals for large changes of state that have a strong likelihood of being good [358]. Lee and Cohen use a markov chain Monte Carlo method to search the likelihood, using both a set of image detectors and a model of kinematic ambiguities to propose moves; this gives a set of possible reconstructions for the upper body [212] and the whole body [213].

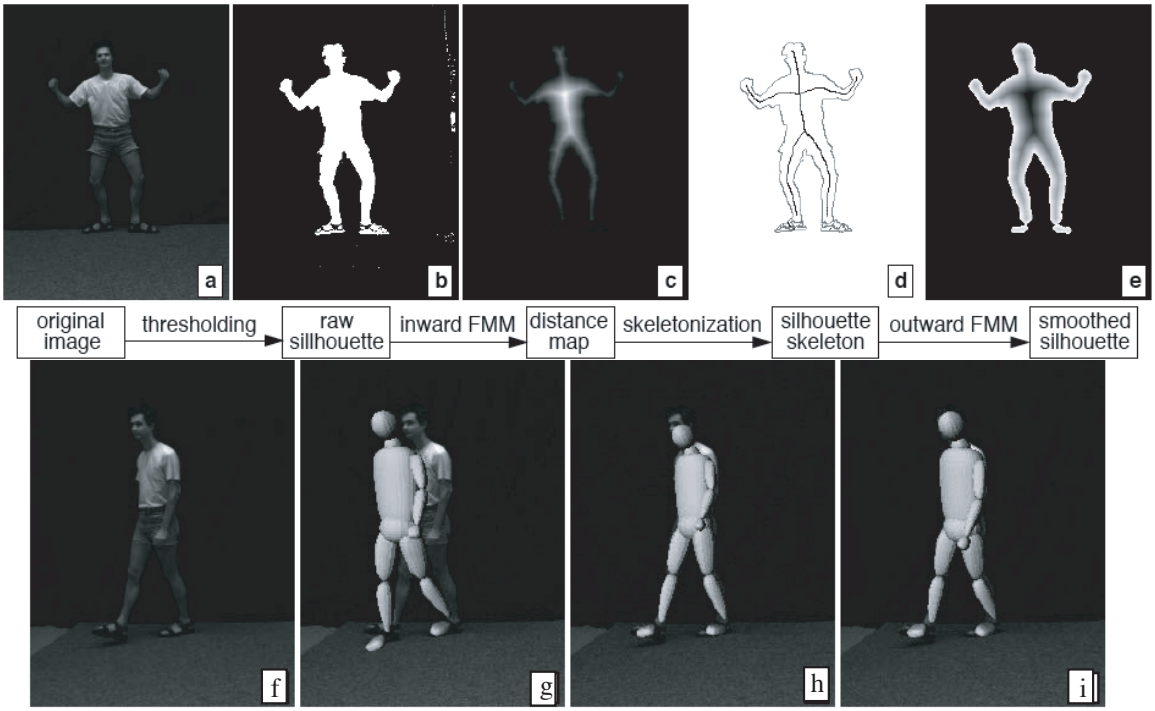


Figure 2.6: **Figure 4, figure 6 from Sminchisescu+Telea, Liftto3D/paper.pdf** *Sminchisescu and Telea lift to 3D by matching to a silhouette using a cost function that causes (a) the projected model to lie on the silhouette and (b) the silhouette to have few points unexplained by the model [352]. To do this effectively, they must first clean up the silhouette.* **Top:** *they take the raw silhouette obtained by thresholding (b), compute a distance transform (c) and skeleton (d), then prune small components of the skeleton and expand (e).* **Bottom:** *the raw image (f); the initial reconstruction, projected onto the raw image (g); the result of silhouette fitting without charging for pixels that are not explained (h); the result of silhouette fitting while charging for unexplained silhouette pixels.*

2.2.3 Lifting by Regression

Assume we are given a set of example pairs $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i$ is a vector of measurements of image properties and $\mathbf{y}_i$ is the known 3D configuration of the body for that measurement vector.

We can regard lifting as a **regression problem** — predict $\mathbf{y}$ for a new set of image measurements $\mathbf{x}$, using the training data. This regression problem has some nasty properties.

- **Dimension:** We expect $\mathbf{x}$ to be drawn from a high-dimensional space. Worse, we expect that the possible $\mathbf{x}$ that we can observe lie on a relatively low-dimensional subspace of the original space. For example, we expect to see arms and legs in a limited range of configurations; we expect to see people with arms of similar appearance; we expect to see people with legs of similar appearance; and so oon.

- **Metric distortion:** We do not expect that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ necessarily reflects the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. For example, two quite distinct body configurations could have very similar images (as a result of geometric ambiguities section 2.2.1).

- **Multiple values:** Worse, we could have two distinct values of $\mathbf{y}$ that are correctly associated with

a single value of $\mathbf{x}$, (as a result of the discrete ambiguity of section 2.2.1).

Notation: To avoid dealing with isolated constants, we will assume that one component of $\mathbf{x}$ always has the value 1.

2.2.3.1 Lifting using the Nearest Neighbour

The simplest regression method is to use the value associated with the nearest neighbour. Athitsos and Sclaroff determine 20 kinematic configuration parameters from an image of a hand by matching the image to a set of examples [14, 15]. Examples cover a wide range of viewing conditions, and the cost of obtaining the best match (in a total of 107,328 images) limits the number of distinct hand configurations to 26.

One can incorporate dynamical information into the distance cost matching entire 3D motion paths to 2D image tracks. Howe computes a match cost frame by frame, by comparing rendered motion capture data from the CMU Motion Capture collection (`http:whereisit`) with image silhouettes [162]. Views are again assumed lateral and orthographic, and are sampled every 10^o around the body. Translation and scale could be handled either by sampling, or by obtaining estimates from a bounding box. The comparison is scored with a chamfer distance. Write

$$H(S_1, S_2) = \sum_{p \in S_1} \min_{q \in S_2} d(p, q)$$

(noting a similarity with the Hausdorff distance, section **??**), θ^l for the 3D configuration of the l'th frame of motion capture data with respect to the camera (meaning that rotation, translation and scale are encoded here), P_{θ^l} for the set of pixels covered by a rendering of θ^l, and P_{S_j} for the pixels lying in the j'th silhouette. The comparison between θ^l and S_j is now scored as

$$M(\theta^l, S_j) = H(P_{\theta^l}, P_{S_j}) + H(P_{S_j}, P_{\theta^l})$$

Now write the (unknown) value of θ at time i as Θ_i — this value could be any one of the available θ^l. Howe then constructs a cost linking frames of motion capture $\Delta(\Theta_i, \Theta_{i-1})$; this cost could include a charge for extreme camera motions, though the paper does not explicitly describe this (the cost used charges for large changes in body configuration). The motion is lifted by applying dynamic programming to

$$C(\Theta_1, ..., \Theta_N) = \sum_{i=2}^{N} \Delta(\Theta_i, \Theta_{i-1}) + \sum_{i=1}^{N} M(\Theta_i, S_i)$$

There are too many frames of motion capture to implement an exact dynamic programming solution, and we allow only values θ^l of Θ_i such that $M(\theta^l, S_i)$ is less than some threshold. The method appears to produce solutions that are unambiguous, which is consistent with the view that 3D reconstruction ambiguities are probably a phenomenon of short, rather than long, time-scales. There is also some useful evidence that reconstruction errors or uncertainties do not propagate over long time-scales (figure 2.7). However, there is no attempt to use either N-best dynamic programming or beam search to identify 3D reconstructions that have cost comparable to the best cost, but are significantly different.

2.2.3.2 Snippets and Cameras

This work suggests that, while a single frame reconstruction might be ambiguous, a match from a short 2D track to a short 3D track might not be (in section **??**, we lay out evidence it is not). Howe *et*

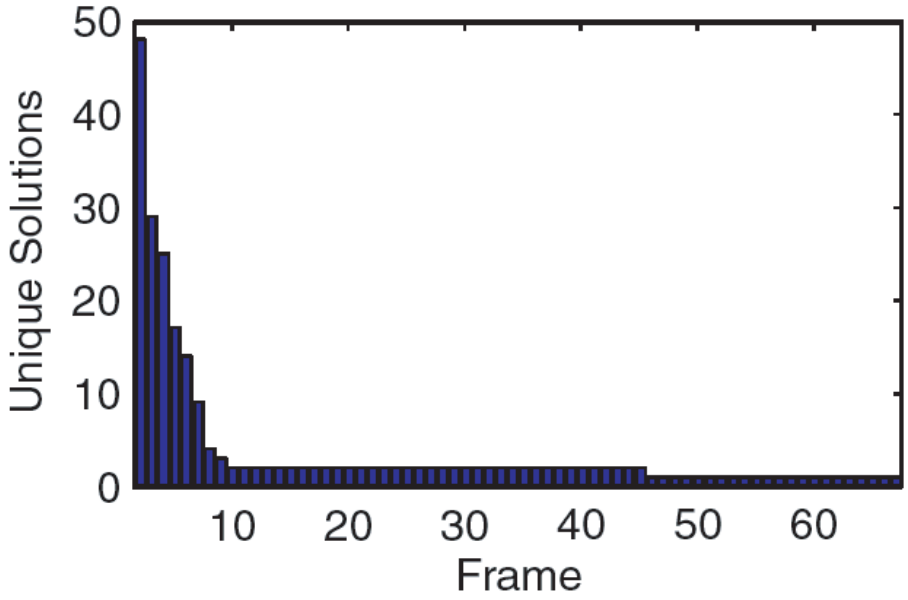


Figure 2.7: **01384804, figure 7 (8 is incomprehensible, and so omitted)** *Howe's formulation lifts to 3D by comparing projected motion capture data with image silhouettes [162]. There is a frame-frame cost for the reconstruction, and the final 3D lift is obtained by dynamic programming. In a formalism like this, one could reasonably fear that a mistaken reconstruction in one frame might result in an entirely wrong path. In practice, this does not occur. The graph is obtained by constraining the first lifted frame of a sequence to each of a 1000 different (incorrect) states; the plot shows the number of distinct states found in the succeeding frames for each path, as a function of frame. The local image evidence quickly overwhelms the effect of history; by the 10'th frame, there are only two distinct states.*

*al.*compare projected motion capture data with image tracks, but now use posterior inference to estimate dynamic parameters [163]. These parameters are an encoding of "snippets" — 11 frames of motion capture data — which are clustered using a mixture of Gaussians. Each 11 frame section of the track produces a snippet with maximal posterior, and the snippets are blended into one another to give a 3D reconstruction. While authors acknowledge the tracker loses track after a while, the lifting procedure appears to be robust and effective.

Ramanan and Forsyth use a similar approach, but apply constraints to camera dynamics, too [313]. They assume that views are lateral, estimate scale and translation from the image, and sample the remaining camera parameter (rotation about the vertical axis). They constrain the camera speed, and charge for large motions in three dimensions. The best matching sequence can then be obtained by dynamic programming. The method cannot recover the motion in depth of the root, but successfully recovers the configuration of the body with respect to the root and all root parameters but depth.

The discrete ambiguity in configuration is handled by incorporating information about surrounding frames into the match cost. In particular, the cost of matching a given image frame with a given motion capture frame is averaged over a window of image (resp. motion capture) frames centered around the frame under consideration. This means that the match uses an implicit (in the collection of motion capture) dynamical model to resolve these discrete ambiguities, at the cost of not being able to lift

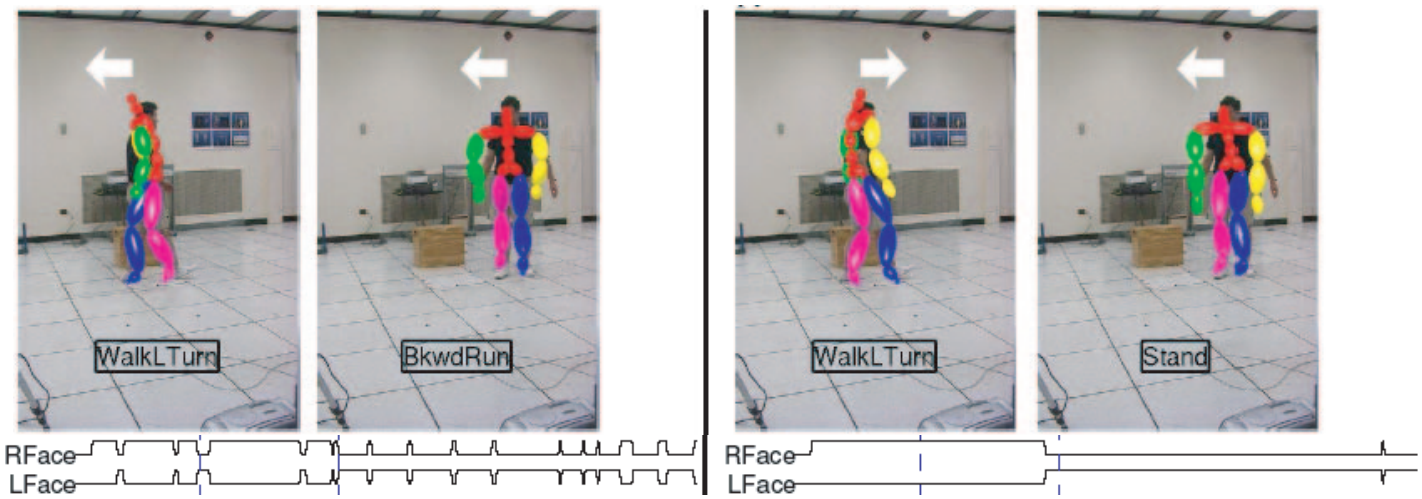


Figure 2.8: **Figure 5.4 from Ramanan's thesis Left frames** *are taken from a walking sequence, matched to motion capture data using the method of Ramanan and Forsyth [313]. Matches are independent from frame to frame. Note that the lateral view of the body (***far left***) is ambiguous, and can be reconstructed inaccurately. This ambiguity does not persist, because the camera cannot move freely from frame to frame.* **Right frames** *show reconstructions obtained using dynamic programming to enforce a model of camera cost. The correct reconstruction is usually available, because the person does not stay in an ambiguous configuration. The frames are taken from a time sequence, and the graphs* **below** *show an automatically computed annotation sequence — facing left vs. facing right — as a function of time. Note that the case on the* **left** *shows an essentially random choice of direction when the ambiguity is present (the person appears to flip from facing left to facing right regularly). This is because the free rotation of the camera means the ambiguity appears on a per-frame basis. For the case on the* **right***, the smoothing created by charging for fast camera rotations means that the labels change seldom (and are, in fact, correct).*

configurations that are not in the motion capture data.

The charge for camera rotation is reasonable, because cameras do not usually swing around the body by very large amounts, but it is also important, because Ramanan and Forsyth's model does not match heads and so has difficulty telling which way the body is facing for lateral views, particularly when the limbs are in line with the body (figure 2.8). This results in a lateral view of a standing person can be interpreted as facing either right or left; the camera rotation charge means that, if the person walks off — and so reveals the direction in which they are facing — this information can be propagated.

2.2.3.3 Regressing Pose against the Image

Shakhnarovich *et al*.train with a data set of 3D configurations and rendered frames, obtained using POSER (a program that renders human figures, from Creative Labs). They show error rates on held out data for a variety of regression methods applied to the pool of neighbours obtained using parameter sensitive hashing. Generally, performance improves with more neighbours, with using a linear (rather than constant) locally weighted regression, and if the method is robust. The best is a robust linear locally weighted regression. Their method produces estimates of joint angles with RMS errors of approximately 20^o for a 13 degree of freedom upper body model [341]; a version of this approach can produce full 3D

shape estimates [142]. Liu *et al.*demonstrate a full body reconstruction from silhouettes in five views using a similar regression model; the reconstruction is not evaluated directly, but is used to control motion synthesis [315].

2.2.3.4 Disambiguation with the Immediate Past

A major difficulty with this procedure is the possibility that a single set of image features may predict multiple poses. This could be a result of weaknesses in image features — for example, it is hard to tell which way the actor is facing in a lateral view of a standing person with current image features — but is more likely the consequence of the kinematic ambiguities described above. Reconstructions performed in the past could disambiguate the current reconstruction. Brand links images with motion capture by fitting HMM's to both motion capture data and image data; these HMM's share a dynamical model [47]. The HMM's are fitted with a variant fitting algorithm which tends to obtain models with relatively low entropy (there is some discussion in [47]; more in [46, 48]). Reconstruction in 3D is obtained by inferring a state sequence from image data, then choosing a sequence of emitted states from the motion capture model, using a smoothed approximation rather than the Viterbi sequence.

We could think of pose as lying on a set of distinct "sheets", each of which is a single valued function of image features, and then build distinct models for each sheet. This leads to tricky problems in identifying the sheets, however. Agarwal and Triggs observe that the pose in the previous frames, if correctly computed, should give a good guide to the current pose — one is unlikely to jump from sheet to sheet in a single frame [2]. This observation implies that, while $\mathbf{y}_t(\mathbf{x}_t)$ might be a multiple valued function, $\mathbf{y}_t(\mathbf{x}_t, \mathbf{y}_{t-1}, \mathbf{y}_{t-2})$ is not. At reasonable sampling rates, the pose in the last two frames should give a fair estimate of the pose in the current frame. Agarwal and Triggs first construct a regressed estimate of the pose in frame t, $\hat{\mathbf{y}}_t$ from $\mathbf{y}_{t-1}$ and $\mathbf{y}_{t-2}$ using a linear regression. They then compute a regression estimate of $\mathbf{y}_t$ from $\mathbf{x}_t$ and $\hat{\mathbf{y}}_t$, using a regression vector machine trained with a variant algorithm. The method produces estimates of joint angles with RMS errors of 4^o for 55 degrees of freedom (3 angles per joint for an 18 joint skeleton, and 1 orientation DOF with respect to the camera).

2.3 Multiple Modes, Randomized Search and Human Tracking

We have clear evidence that tracking a 3D representation of the body can result in multiple modes in the posterior and that these modes do not look Gaussian locally (figure **??**; but see section **??**). The need to manage these modes has spawned a number of methods, all of which are forms of randomized search.

Particle filters should be seen as a form of randomized search. One starts a set of points that tend to be concentrated around large values of the posterior. These are pushed through the dynamical model, to predict possible configurations in the data. The result is a sampled representation of the prior. The predictions are compared to the data, and those that compare well are given higher weights, yielding a sampled representation of the posterior. This simple view provides some insight into why particle filters in their most basic form are not particularly well adapted to kinematic tracking.

There is a problem with dimension. The state vector for most kinematic tracking problems must be high dimensional. One expects to encounter at least 20 degrees of freedom (one at each knee, two at each hip, three at each shoulder, one at each elbow and six for the root) and quite possibly many more. This means that mismatches between the prior and the likelihood can generate serious problems. Such mismatches are likely for three reasons.

First, the body can move quickly and unexpectedly, meaning that probability must be quite widely spread in the prior to account for large accelerations. It is hard to be clear on how much uncertainty

there is in the state of the body at some time given the past, and there are fair arguments either way (section **??**). However, fast movements do occur, and current methods are forced to have fairly diffuse dynamical models to cope with them.

Second, the likelihood has multiple peaks, which can be very narrow. Narrow peaks occur because some body segments — forearms are a particularly nasty example — have relatively small cross-section in the image, and so only a small range of body states will place these segments in about the right image configuration. Multiple peaks occur because there tend to be numerous objects that look somewhat like body segments (long, narrow, parallel sides, constant colour). We are now using the predictions of the prior to find the largest narrow peak in a high-dimensional likelihood — for this to have any hope of success, the predictions need to be good or to occur in very large numbers. But we know the predictions will be poor, because we know people can generate fast, unexpected movements.

Third, detectors used to produce a likelihood model may be inaccurate. This can result in small errors in inferred state, which in turn produce potentially large changes in state from frame to frame. As Sminchicescu and Triggs point out ([356], p. 372), this suggests using a relatively diffuse dynamical model as an insurance policy.

The key idea in particle filters is the randomized search. One might abandon, or at least de-emphasize, probabilistic semantics, and focus on building an effective search of the likelihood. The key difficulties are that the peaks in the likelihood are narrow (and so easy to miss) and that the configuration space is high-dimensional (so that useful search probes may be difficult to find). The narrow peaks in the likelihood could be dealt with by annealing, and good search probes may be found by considering the ambiguity of 3D reconstructions. We review these approaches in section 2.3.1.5.

2.3.1 Randomized Search with Particle Filters

There are a series of approaches to deal with problems created by the dimension of the state space. First, we could refine the search using importance sampling methods. Second, we could use sequential inference methods to obtain more efficient samples of the prior. Third, we could build lower-dimensional dynamical models. Finally, we could build more complex searches of the likelihood.

2.3.1.1 Importance Sampling

Importance sampling is a method to concentrate samples in places that seem likely to be useful. Assume we have a distribution $g(\mathbf{X}_t)$ from which we can draw samples, and which is a better guide to the likelihood than the prior $P(\mathbf{X}_t|\mathbf{Y}_0, \ldots, \mathbf{Y}_{t-1})$ is. We can then draw samples $\mathbf{X}_{ti}$ from $g(\mathbf{X}_t)$. Then the set of samples

$$(\mathbf{X}_{ti}, \frac{P(\mathbf{X}_t = \mathbf{X}_{ti}|\mathbf{Y}_0, \ldots, \mathbf{Y}_{t-1})P(\mathbf{Y}_t|\mathbf{X}_t = \mathbf{X}_{ti})}{g(\mathbf{X}_{ti})})$$

is a representation of the posterior, as can be established by pattern matching to the expressions above. Given several plausible importance functions, one could use a mixture of these functions and the prior as an importance function. Drawing samples from this mixture is straightforward; one draws a sample according to the mixing weights, and uses this to choose a sampling strategy. Image observations are a natural source of importance functions. Isard and Blake use this approach to track hands and forearms [176], using a skin detector to build an importance function. Rittscher and Blake use importance sampling methods to track contours of motions drawn from two classes (pure jump and half star jump); the tracker maintains a representation of posterior on the motion class, which can be used to distinguish

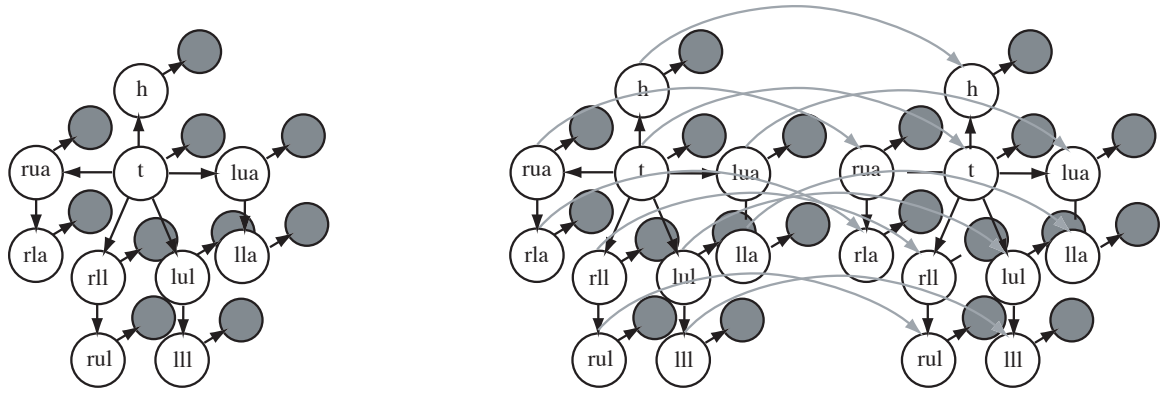


Figure 2.9: *If one models a person with a kinematic chain, then determining where a given person appears in a static image involves inference on a tree structured graphical model. On the* **left**, *a graphical model illustrating this point. In the usual language of graphical models, open nodes represent unknowns, arrows represent dependencies, and shaded nodes represent measurements. Each open node encodes the state (for example, image position; image position and orientation; 3D position, orientation and scale; and so on) of the body segment implied by the label (t: torso; lul: left upper leg; and so on). The arrow represents a model of $P(\text{variable at head}|\text{variable at tail})$. The filled nodes represent various detector responses. Notice that each open node has at most one parent, so the open nodes form a tree, so that inference is a matter of dynamic programming (or, equivalently, message passing; section 3.4 or a text such as [120, 234]). On the* **right**, *we show what happens when one has temporal dependencies. We show only two frames (there's enough clutter in the drawing), and the gray arcs are temporal links. The graphical model becomes much more complex. Most open nodes now have two parents, a spatial parent and a temporal parent, and this means that exact inference is impractical.*

between motion classes successfully [316]. Forsyth uses edge detector responses as a source of proposal mechanisms to find simple boundaries [115], and Zhu *et al.* — who call the approach data driven MCMC — use image observations to propose segmentations [392, 391, 430]. We are not aware of the method being used for kinematic tracking; however, it is a way to unify the more successful kinematic tracking methods of section **??** with particle filter based inference.

If one models a person with a tree-structured kinematic model, then identifying each body segment in the image is a matter of dynamic programming (we discuss this issue in greater detail in section 3.4). However, adding temporal dependencies produces a structure that does not allow for simple exact inference, because the state of a limb in frame t has two parents: the state in time $t-1$, and the state of its parent in frame t (figure 2.9). Loopy propagation is a method for approximate inference on graphical models which are not trees. One constructs a spanning tree, passes messages with the usual algorithm on that spanning tree, and then repeats for other choices of spanning tree. This is an approximation, because some probabilities are overestimated as a result of cycles in the graph; experiment shows that, under many circumstances, the approximation gives usable and helpful results. Useful accounts of this method appear in [425, 260, 409].

Sigal *et al.* use loopy propagation, representing messages passed between nodes using a set of particles [346]. Their template is a 3D model of a person with links both in time and in space learned

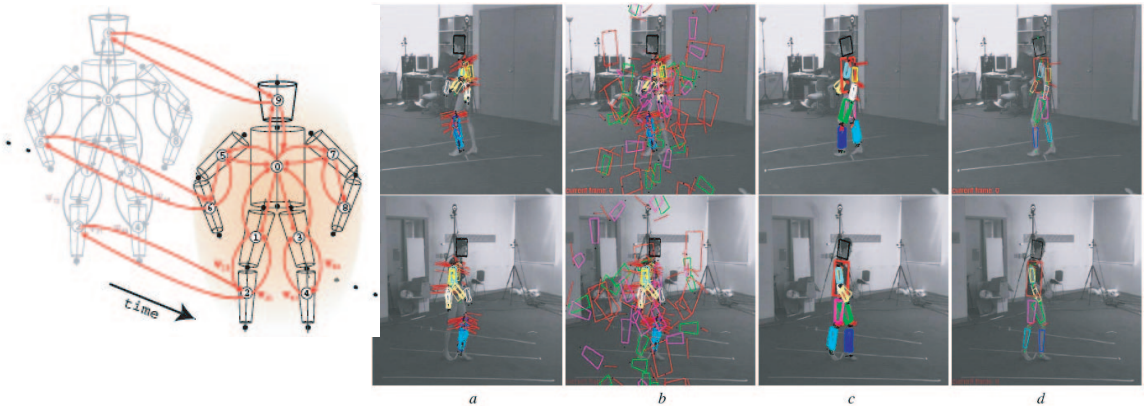


Figure 2.10: **Figures 1 and 7 from Particlefilters/013150163** *Sigal* et al. *build a 3D model of a person as a set of segments [346]. Again, the state of each segment but the root has two parents — the corresponding segment in the previous frame, and that segment's parent in the model (***left***). This yields an inference problem that is too difficult in general to do exactly. Sigal* et al. *track in multiple views using a form of particle filter adapted for loopy belief propagation. The image likelihood is a conditional exponential model. Authors use a combination of segment detectors and uniformly distributed samples to propose likely configurations of limbs in the image; these are incorporated in the inference procedure as importance functions. The figure on the* **right** *shows camera outputs with superimposed information for two of four views (***rows***); column* **(a)** *shows limb segments proposed by the detector;* **(b)** *shows proposals from a uniform distribution;* **(c)** *shows samples from the belief distribution after 30 frames of belief propagation; and* **(d)** *shows the state with the highest belief.*

from data. The likelihood is modelled with a conditional exponential model, where

$$P(\mathbf{Y}|\mathbf{X}) \propto \exp\left(-\sum_i \lambda_i g_i(\mathbf{X}, \mathbf{Y})\right)$$

with parameters λ_i learned from data. Such models, often called **maximum entropy models** and quite popular in the language modelling community, are commonly fitted by maximizing likelihood (which requires computing the partition function), using an algorithm known as **iterative scaling** (see [], and variants in []; more conventional optimization methods are now becoming popular []). Sigal *et al.* use a series of detectors which are tuned to body parts (but not, in the nature of such detectors, particularly reliable; otherwise there'd be nothing to do) to produce an importance function. Some percentage of messages passed to limb nodes are drawn from this importance function, giving strong suggestions about the configuration in 3D of a particular body segment. They demonstrate tracks of people in 3D from three views. Unusually, there is a strong evaluation component, which we describe in section **??**.

2.3.1.2 Partitioned Sampling

Partitioned sampling is a variant of importance sampling that uses a sequence of samples within each time slice. Assume that the state vector $\mathbf{X}$ has several components; notation etc. is much simpler if we assume only two, and the more general case follows, so we shall work with two and write $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$. We will also drop the subscript for time to simplify notation. Now assume that we have an importance function $I(\mathbf{X})$ that is a good guide to the likelihood (what this means will become apparent), and can

be factored as $I(\mathbf{X}) = I_1(\mathbf{x}_1)I_2(\mathbf{x}_1, \mathbf{x}_2)$ Now if $\mathbf{u}_i$ is a set of IID samples of $P(\mathbf{x}_1)$, then

$$(\mathbf{u}_i, I_1(\mathbf{u}_i))$$

represents a probability distribution proportional to $P(\mathbf{x}_1)I_1(\mathbf{x}_1)$. Take this representation and resample with replacement according to the weights, to obtain $(\mathbf{u}_j, 1)$ which must also be a representation of that distribution. Now obtain $\mathbf{v}_{kj}$, which are IID samples of $P(\mathbf{x}_2|\mathbf{x}_1 = \mathbf{u}_j)$. Then

$$((\mathbf{v}_{kj}, \mathbf{u}_j), I_2(\mathbf{u}_j, \mathbf{v}_{kj}))$$

represents a probability distribution proportional to $P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_1)I_1(\mathbf{x}_1)I_2(\mathbf{x}_1, \mathbf{x}_2)$. Take this representation and in turn resample with replacement according to the weights, to obtain $(\mathbf{v}_j, 1)$, which is also a representation of that distribution. Finally,

$$((\mathbf{v}_{lj}, \mathbf{u}_j), \frac{P(Y|\mathbf{X} = (\mathbf{v}_{lj}, \mathbf{u}_j))}{I_1(\mathbf{u}_j)I_2(\mathbf{u}_j, \mathbf{v}_{lj})})$$

represents the posterior. Notice that we have omitted various if's, and's and but's to do with the support of the importance function and so on, to get to this point. The advantage of this strategy is that we have guided the search of the likelihood using our importance function; in particular, the first resampling step discards particles that lie in spots where there is evidence — supplied by the importance function — that the marginal of the posterior will be small. Throwing these particles away allows means that, when we elaborate the particles to represent the whole state, the resulting particles should tend to lie in places where the likelihood is large. Of course, all this depends on the quality of our likelihood functions. MacCormick and Isard track hands using partitioned sampling [232]. MacCormick and Blake use this method to track multiple objects [231, 230], where one needs a method to avoid both tracks lying on the same object. The importance functions are obtained by considering each object separately, and the likelihood function is a mixture of three cases: no objects in the tracker gate, one object in the tracker gate, and two objects in the tracker gate. Again, we are aware of no kinematic trackers of humans that use this method, but see it as a way to unify the more successful kinematic tracking methods of section **??** with particle filter based inference.

2.3.1.3 Lower Dimensional State Models

Sidenbladh *et al.* build a 3D model of a human as a kinematic chain, with state encoded as the configuration and velocity of each element of this chain with respect to its parent, and the root with respect to the camera. Each segment of the model has an attached encoding of appearance, and the likelihood is computed by comparing a rendering of the state with the image, using the appearance encoding. There is a separate constant likelihood term for self-occluded segments, and a discount term for foreshortened segments, because foreshortening of a segment causes texture foreshortening. The tracker is initialized by hand. Tracks are obtained using a straightforward particle filter, using a random walk dynamical model and also using a dynamical model specialized to walking. This walking model is obtained by principal components analysis on motion captured walk data. The appearance model appears to have dynamics to account for changes in illumination; the authors point out that this advantage over a fixed appearance template comes at the cost of potentially increased tracker drift. The random walk model is shown to track a two segment arm with reasonable success, but authors indicate that more complex kinematic models are difficult to track this way. The advantage of a low dimensional model of walking dynamics is that the effective dimension of the state space at the $k + 1$'th frame is relatively small, and

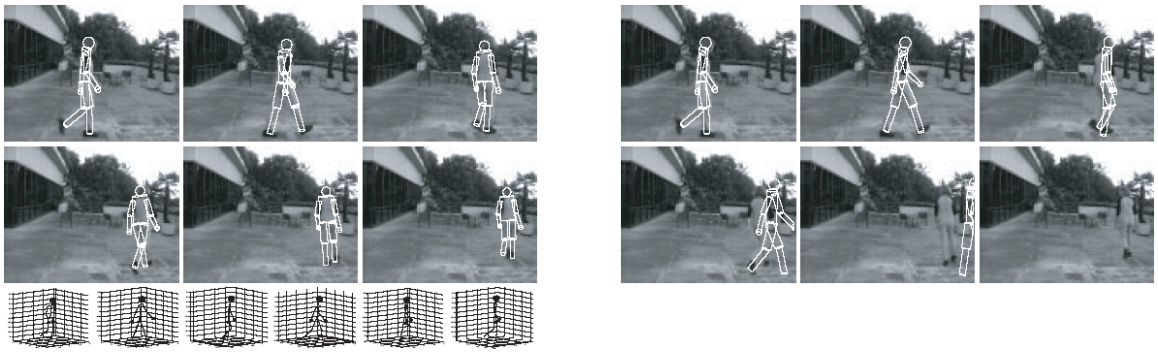


Figure 2.11: **Figure 6 and 7 from Particlefilters/eccv00.pdf** *Sidenbladh* et al. *use particle filters to track a 3D model of a walking person, using a reduced dimensional dynamical model fitted to motion capture data of walking people. This means that the dynamics are more predictable, and so the search of the likelihood is more effective; the difficulty is that one must know the activity before being able to track. On the* **left***, a track of a walking person who turns during the walk. The 3D reconstruction of this track is shown* **below left***. On the* **right***, a "track" of a walking person, initialized as on the left, but now ignoring image data; this illustrates the strength of the prior. In particular, the "track" continues to walk, but does not turn when the subject turns.*

this relatively tight motion prior allows quite good tracking of a walking figure (figure **??**). The difficulty with this approach is that one might need to choose which activity is occuring to be able to track it, and that seems difficult to do.

2.3.1.4 Probabilistic Searches of the Posterior

Choo and Fleet implement a more extensive search of the posterior using a Markov chain Monte Carlo (MCMC) method [73]. They interpret the particles at a particular step as a set of initial states for an MCMC sampler; this sampler then runs independently on each state. Any such sampler will eventually produce a fair sample of the posterior. It is reasonable to expect that running an MCMC sampler on a set of particles will produce IID samples of the posterior. These can, in turn, be passed through the particle filter and refined again. Choo and Fleet use Duane *et al.*'s hybrid Monte Carlo method to obtain samples (see [101, 264]; there is a brief account in [119]), but other methods might be used. The method is used to compute 3D configurations from images of markers. It has not been shown to cope with the dramatic problems with local maxima one associates with texture and clutter, and it seems unlikely that it can. The difficulty here is that it may take very many steps of the MCMC method to produce samples that have "forgotten" their start point. In practice, it is extremely difficult for such a sampler to pass from one local maximum of the posterior to another; this means that such a sampler is unlikely to overcome the problems created by a posterior with many narrow peaks (see [132]; in some applications, for example where there is a symmetry in the posterior, this may not be a nuisance [119], but one cannot rely on MCMC methods to discover all peaks in a posterior without quite strong proofs of good mixing behaviour).

2.3.1.5 Annealing

```
Check this; Neal paper, etc; If one is willing to abandon the semantics
of a sampled representation of a distribution, a variety of search strategies
are available.
```

A variety of search strategies are available. One strategy is to launch an **annealed search** of the likelihood. We do this by defining a set of intermediate weighting functions, to obtain $w_0(\mathbf{X}) = P(\mathbf{Y}|\mathbf{X})$, $w_1(\mathbf{X})$, ..., $w_M(\mathbf{X})$, where w_k is a somewhat smoother version of w_{k+1}. At any time step we have $\mathbf{u}_i$ a set of IID samples of $P(\mathbf{X})$. Instead of weighting these samples by the likelihood, we weight by w_M. We resample with replacement according to the weights and reset the weights to one, yielding $\mathbf{u}_j$. We take each sample and add noise drawn from a normal distribution with zero mean. We now weight the resulting samples using w_{M-1}. This process continues until each sample is weighted using the likelihood. Deutscher *et al* use this scheme to track a person moving using a 3D model viewed with multiple cameras [91, 93] (figure **??**). The likelihood is evaluated using both image values within and edge points near the projected outline; annealing in effect uses a smoothed version of this (very peaky) likelihood function to guide samples toward peaks in the likelihood. This method can be given exact probabilistic semantics by interpreting the annealing procedure as an importance function, an observation due to Neal [267, 266, 265]. Deutscher *et al* have shown that performance improvements are available by using partitioning methods together with an annealed particle filter (figure 2.13). All examples show isolated persons on black backgrounds; there is no evidence that the annealing is powerful enough to cope with the rich range of local likelihood peaks that can result from, say, texture or clutter.

2.3.2 Multiple Probes from Covariance Analysis

One difficulty with a sampled model of the posterior is that we don't know if there are larger values of the posterior close to each sample. We could regard each sample as a plausible start point for a search of the posterior. We are now no longer building a set of particles that explicitly represents the posterior in the sense above, but are using multiple states to represent the prospect that the posterior is multi-modal. Each state lies on a mode in the posterior, and we attempt to ensure that all modes have a state. The origins of this approach lie with Cham and Rehg [65], who use it to track a 2D kinematic model of the body.

Sminchisescu and Triggs elaborate this search by analysis of the Hessian of the log-posterior [353, 356]. They track a 3D model of a person, which has parameters giving the kinematic configuration, relative proportions of segments, and deformations of the surface skin. Sminchisescu and Triggs do not use a dynamical model. However, they do encode joint limits, and so must represent a model of $P(\mathbf{X}_k|\mathbf{Y}_k)$ (which we call the posterior in what follows; note that only the current measurement is involved). They regard the reconstruction at frame $k-1$ as an initial point for a search of the posterior at frame k. The likelihood is evaluated by comparing projected model points with image points, using values of edges and other image features. What is known about state is represented by a collection of tuples; the i'th tuple (c_i, μ_i, Σ_i), contains a weight c_i, a state value μ_i and a covariance matrix Σ_i. Each state value gives the state at a mode of the likelihood. The covariance is the Hessian of the negative log-posterior at the mode, and the weight is the value of the posterior at the mode. Weights are normalized to sum to one.

This information is propagated from the $k-1$'th frame to the k'th frame by using these tuples to launch searches of the likelihood. The search proceeds by:

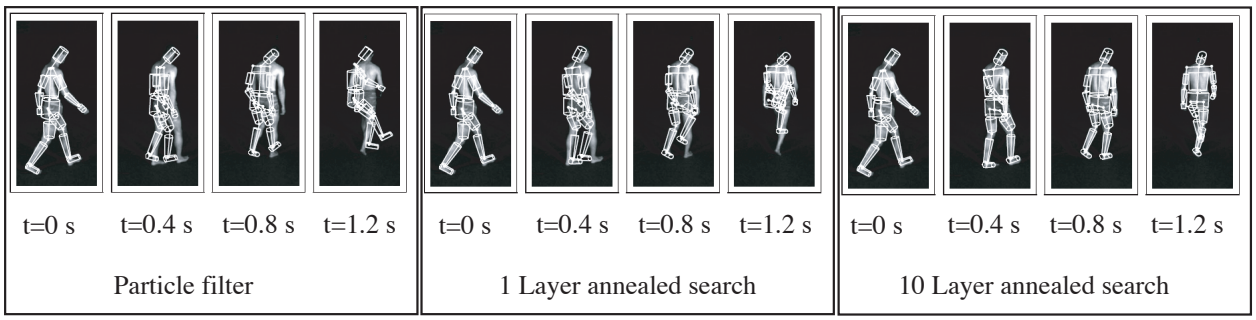


Figure 2.12: **Figure 8 of DeutscherBlakeReid** *Deutscher* et al. *[91] track a moving person in 3D using an annealed particle filter. In effect, particles are passed through the dynamic model, then weighted with a smoothed version of the likelihood. They are resampled according to weights, then perturbed randomly and weighted using a less heavily smoothed likelihood. This concentrates particles in regions where the likelihood is likely to be high. The process continues for some number of layers of annealing. The figure shows tracks for a particular set of frames using three different algorithms. On the* **left***, a straightforward particle filter, which loses track fairly quickly because searching a peaky likelihood using a smooth prior doesn't work well. In the* **center***, the results of one layer of annealing. Notice that the right leg is poorly tracked, but the track has improved. On the* **right***, the results from ten layers of annealing. Notice the much improved track. The particles no longer have any probabilistic semantics, however, and the ability of the method to deal with clutter and texture — which can hugely complicate the likelihood function — is not proven.*

- **Choosing a tuple to propagate** by drawing one of the initial tuples randomly according to the weight. Assume we have drawn tuple i.
- **Computing a local covariance scaling** by obtaining the k directions in the Σ_i where the least change in posterior is likely — these directions are those in which the state is most uncertain — by a singular value decomposition, and computing the restriction of Σ_i to this space; call the result $\Sigma_i^{'}$.
- **Generating a new tuple** by generating a sample $\mathbf{s}$ distributed as $N(\mu_i, s\Sigma_i^{'})$ for some scale parameter s (it is wise to have $s > 1$). We start an optimization procedure for the posterior at $\mathbf{s}$; this produces $\mathbf{s}^{'}$. The weight for the new tuple is the value of the posterior at this point; the mean is $\mathbf{s}^{'}$; and the covariance is the Hessian of the negative log-posterior at $\mathbf{s}^{'}$.

These steps are repeated multiple times, to produce a set of tuples representing the posterior. This set is pruned to remove tuples that represent the same mode — the states will be the same — and the result represents the new posterior. Numerous variants of this method are possible; for example, it is natural to produce a large pool of tuples, prune duplicates, and then keep only the K best. Performance comparisons between these methods appear in [356].

2.4 Notes

Reducing configuration ambiguity is one reason to use multiple cameras; another is to keep track of individuals who move out of view of a particular camera (e.g. [247, 196, 60]). Currently, this is done at a coarse scale (people are blobs).

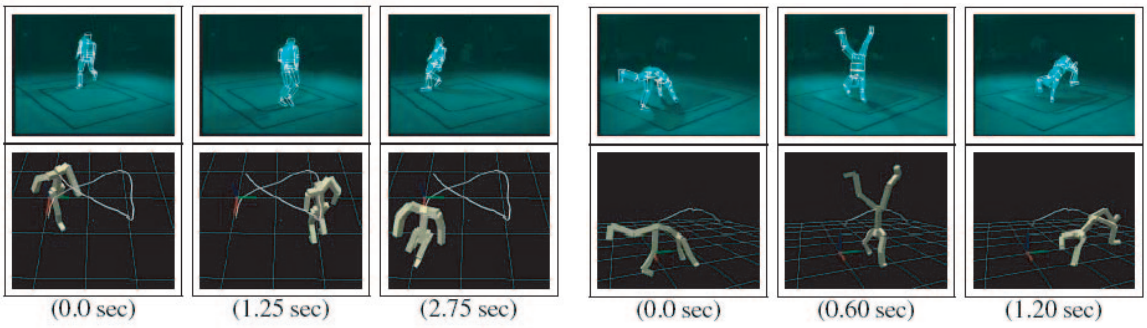


Figure 2.13: **Figure 10 and 11 of Particlefilters/deutscheretalcvpr2001** *Deutscher* et al *[93] show that one can use partitioning methods with the annealed particle filter. They track a 3D model of a person in a single view. On the* **left**, *a track and the inferred 3D configuration for a running person. On the* **right**, *a track and the inferred 3D configuration for a person doing a handstand. Again, there are no probabilistic semantics, and again the ability of the method to deal with clutter and texture is not proven.*

```
   Lots of dodgy approximate nearest neighbours papers; careful

   cite Isard Blake, SalmondGordon, Kitagawa, LiuChen, etc

   history of particle filter notes in particlefilters/iccv99-deutscher;
also Liu; also 00784636

   Obtaining dynamical models; EM; ptr to Li; North et al in particlefilters
00877523; other stuff by North and Blake

   More variant particle filters Sullivan and Rittscher particlefilters
00937536

   particlefilters/science-6 PF tracking for HCI
```

D'Souza *et al.*learn inverse kinematics for a humanoid robot with locally weighted regression [100]. Schaal *et al.*describe learning methods for a variety of robot problems, including inverse dynamics [329].

Rosales and Sclaroff use a collection of local experts ("specialized mappings") to regress hand configuration against image appearance [319].

Appendix I: Particle Filter Basics

2.4.1 The Particle Filter

Probability distributions should be thought of as devices that are used to compute expectations; this means that any procedure that can produce good estimates of expectations with respect to a distribution is a representation of that distribution. Assume we wish to represent a distribution $Q(X)$, and we can sample some other distribution $P(X)$. Take a set of points s_i which are independent identically

distributed samples of $P(\mathbf{X})$. If we attach a weight $w_i = kQ(\mathbf{s}_i)/P(\mathbf{s}_i)$ (where k is a constant, perhaps unknown) to each point, the resulting set of pairs $\{(\mathbf{s}_i, w_i)\}$ is a representation of $Q(\mathbf{X})$. This is because

$$\frac{\sum_i f(\mathbf{s}_i)w_i}{\sum_i w_i}$$

is an estimate of

$$E_Q(f) = \int f(X)Q(X)dX$$

(by the weak law of large numbers). Samples are commonly thought of as **particles**. Notice, by the way, that it is usual in the vision literature to normalize weights so that $\sum_i w_i = 1$, and in the statistics literature to divide by the sum of weights as we have done; Liu gives an argument that the latter is better practice [224].

Priors into posteriors: A useful trick allows us to obtain a representation of a posterior $P(\mathbf{X}|\mathbf{Y} = \mathbf{y})$ from a representation of a prior $P(\mathbf{X})$. Assume the prior is represented by $\{(\mathbf{s}_i, w_i)\}$. Now

$$\int f(\mathbf{X})P(\mathbf{X}|\mathbf{Y} = \mathbf{y})d\mathbf{X} = \frac{1}{K}\int f(\mathbf{X})P(\mathbf{Y} = \mathbf{y}|\mathbf{X})P(\mathbf{X})d\mathbf{X}$$

where

$$K = P(\mathbf{Y} = \mathbf{y}) = \int P(\mathbf{Y} = \mathbf{y}|\mathbf{X})P(\mathbf{X})d\mathbf{X}$$

What this means is that

$$\frac{\sum P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{s}_i)w_i}{\sum w_i}$$

is an estimate of K. In turn, this means that $(\mathbf{s}_i, w_i P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{s}_i))$ is a representation of the posterior.

Resampling: Finally, we can turn one representation of a distribution $P(\mathbf{X})$ into another representation of the same distribution by resampling the particles. The most important case is if we form a distribution π on the index from w_i, where the probability of drawing the i'th index is $w_i/\sum w_i$. We now produce a new representation $\{(\mathbf{s}_l, 1)\}$, where the l are independent identically distributed (henceforth IID) samples from π. This is a representation of $P(\mathbf{X})$. The easiest way to see this is to assume we have drawn a very large number of samples. Now if we form $\sum f(\mathbf{s}_l)$, we will see samples from the original set of particles with a frequency proportional to the original weights.

Prediction: Now assume that we have a set of points and weights $(\mathbf{s}_i, w_i)$ which represents the distribution $P(X_{k-1}|Y_0, ...Y_{k-1})$. Write $(\mathbf{q}_{j(i)})$ for a set of $N(i)$ points, indexed by j, drawn from $P(X_k|X_{k-1} = \mathbf{s}_i)$. We can represent $P(X_k|Y_0, ..., Y_{k-1})$ using the set $(\mathbf{q}_{j(i)}, w_i/N(i))$. This is because

$$\frac{\sum_i \sum_j f(\mathbf{q}_{j(i)})w_i/N(i)}{\sum_i w_i/N(i)}$$

is an estimate of

$$\int \left[\int f(X_k)P(X_k|X_{k-1})dX_k\right] P(X_{k-1}|Y_0, ..., Y_{k-1})$$

(one can see this by pattern matching to the previous paragraph). This could be impractical, because the number of samples might grow — perhaps $N(i) > 1$ for all i — but we ignore this possibility for the moment. Notice that $\mathbf{X}_t$ might live in a space of complex form — for example, multiple components of different dimension, the product of two spaces encoding very different types of state etc. — but as long

as we can obtain samples, no difficulty results (e.g. [150, 66]; various formulations and applications appear in [95, 224]).).

Correction: The result of the step above is a a set of weighted samples, $(\mathbf{q}_{j(i)}, w_i)$, which represent $P(\mathbf{X}_k|Y_0, ..., Y_{k-1})$. This is a prior. We apply the recipe above, and obtain a representation of the posterior as $(\mathbf{q}_{j(i)}, w_i P(\mathbf{Y}_k = \mathbf{y}_k|\mathbf{X}_k = \mathbf{q}_{j(i)}))$.

Initialization: Represent $P(\mathbf{X}_0)$ by a set of N samples

$$\left\{(\mathbf{s}_0^{k,-}, w_0^{k,-})\right\}$$

where

$$\mathbf{s}_0^{k,-} \sim P_s(\mathbf{S}) \text{ and } w_0^{k,-} = P(\mathbf{s}_0^{k,-})/P_s(\mathbf{S} = \mathbf{s}_0^{k,-})$$

Ideally, $P(\mathbf{X}_0)$ has a simple form and $\mathbf{s}_0^{k,-} \sim P(\mathbf{X}_0)$ and $w_0^{k,-} = 1$.
Prediction: Represent $P(\mathbf{X}_i|\mathbf{y}_0, \mathbf{y}_{i-1})$ by

$$\left\{(\mathbf{s}_i^{k,-}, w_i^{k,-})\right\}$$

where

$$\mathbf{s}_i^{k,-} = f(\mathbf{s}_{i-1}^{k,+}) + \xi_i^k \text{ and } w_i^{k,-} = w_{i-1}^{k,+} \text{ and } \xi_i^k \sim N(0, \Sigma_{d_i})$$

Correction: Represent $P(\mathbf{X}_i|\mathbf{y}_0, \mathbf{y}_i)$ by

$$\left\{(\mathbf{s}_i^{k,+}, w_i^{k,+})\right\}$$

where

$$\mathbf{s}_i^{k,+} = \mathbf{s}_i^{k,-} \text{ and } w_i^{k,+} = P(\mathbf{Y}_i = \mathbf{y}_i|\mathbf{X}_i = \mathbf{s}_i^{k,-}) w_i^{k,-}$$

Resampling: Normalise the weights so that $\sum_i w_i^{k,+} = 1$ and compute the variance of the normalised weights. If this variance exceeds some threshold, then construct a new set of samples by drawing, with replacement, N samples from the old set, using the weights as the probability that a sample will be drawn. The weight of each sample is now $1/N$.

Algorithm 3: *A practical particle filter resamples the posterior.*

2.4.2 Practicalities

The recipe above is not much use. First, we cannot manage a pool of particles that either shrinks or grows too fast. The easiest way to deal with this problem is to ensure that $N(i) = 1$, though this may not be the best. Second, as one can discover with a simple experiment, the variance of the weights tends to increase as time goes on (there is a theorem to this effect, in [224]). What this means is that one weight gets very much larger than all the others, which means that, in effect, only one sample is being used to represent the distribution. This doesn't affect the expected value of our estimates, but it does mean that their variance might be extremely high, and so they are of no practical use. This effect — known as **sample impoverishment** — can be countered by using an estimate of the variance of the weights to determine when it is occurring, and then resampling the particles. This is effective because we will tend to drop particles with low weight and keep multiple copies of particles with high weight. These

latter will spawn multiple distinct samples in the next prediction stage. As a result, we can reasonably hope that our particles tend to be concentrated near spots where the posterior is large. Algorithm 3 gives a practical particle filter; one should realize that numerous variants are possible (see [224, 95] for examples).

The particle filter can be a powerful and effective inference tool. It should be seen as a form of randomized search. One starts a set of points that tend to be concentrated around large values of the posterior. These are pushed through the dynamical model, to predict possible configurations in the data. The predictions are compared to the data, and those that compare well are given higher weights. This simple view provides some insight into the strengths of the method, and into what could go wrong.

2.5 Appendix II: Regression

2.5.1 The Nearest Neighbours

One could obtain the k-nearest neighbours, where for a new set of measurements $\mathbf{x}$, we obtain the k examples $\mathbf{x}_i$ that lie closest to $\mathbf{x}$ (using an appropriate distance function, which we discuss below), then average the $\mathbf{y}_i$ associated with the examples. There is no reason to expect good results from this method unless the examples are relatively uniformly spaced — in particular, if the k nearest neighbours that we obtain are far away from our query example, the estimate could be very poor. Worse, if $\mathbf{y}$ is a multi-valued function of $\mathbf{x}$, the method might average values associated with distinct "sheets", and produce a meaningless result.

Assume, for the moment, that $\mathbf{y}$ is not a multi-valued function of $\mathbf{x}$. We must still obtain the k nearest neighbours for a query $\mathbf{x}$. We expect to require sufficient examples that blank search through the examples is unattractive. An approximation called (r, ϵ) **nearest-neighbours** is attractive, because one can obtain a solution in time less than linear in the number of examples using an approach known as **locality sensitive hashing**. Assume a query point $\mathbf{x}$. If there is an example $\mathbf{x}_j$ such that $d(\mathbf{x}, \mathbf{x}_j) < r$, then with high probability the algorithm returns $\mathbf{x}_k$ such that $d(\mathbf{x}, \mathbf{x}_k) < (1 + \epsilon)r$, otherwise it reports no point. It does so by computing a hash key from a randomly selected subset of a **locality sensitive** family of functions; this key has the property that similar points have a high probability of colliding, and dissimilar points have a low probability of colliding. One then obtains a list of possible nearest neighbours for a query by computing a hash key for the query, obtaining all examples with which the query collides, then searching this set of examples (details appear in [133, 85, 171, 170]).

The usual difficulty in practical applications of nearest neighbours problems is that it isn't clear what distance to use. Shakhnarovich *et al.*argue that in regression problems, it is important that the distance reflect the distance between predicted parameters (rather than, say, some canonical distance between examples) [341]. While their argument applies to the general case, the application is to lifting. In this case, two quite similar image configurations $\mathbf{x}_i$ and $\mathbf{x}_j$ may predict rather different 3D configurations $\mathbf{y}_i$ and $\mathbf{y}_j$ — a query close to $\mathbf{x}_i$ should not be seen as close to $\mathbf{x}_j$, because the two predictions are different. This means that we would like the hashing functions we choose to tend to generate collisions for examples that have close values of $\mathbf{y}$.

Shakhnarovich *et al.*achieve this by noting that a hash function acts like a classifier that determines whether two points are close (when they collide) or not. By searching thresholds T and **decision stumps** — functions of the image configuration $\mathbf{x}$ with the property

$$\phi(\mathbf{x}) = \left\{ \begin{array}{cc} 1 & \text{if } \phi(\mathbf{x}) \geq T \\ -1 & \text{otherwise} \end{array} \right\}$$

they obtain a set of hash functions, on the set of training examples, tend to cause examples with similar $\mathbf{y}$ to collide and those with distinct $\mathbf{y}$ not to collide.

2.5.2 Some Regression Technique

Assume we have a set of examples likely to be relevant for a regression problem, perhaps as a result of k-nearest neighbours. How are we to obtain an estimate? in general, we must fit some approximation and obtain $\mathbf{y} = \mathbf{f}(\mathbf{x}, \beta)$, where β is a vector of parameters chosen by minimizing an error predicted and observed values. For our purposes, $\mathbf{y}$ is a high dimensional vector. This means that we will not, in general, be able to build a good model of correlations between elements of $\mathbf{y}$, and so will predict its components independent of one another.

We will concentrate on the l'th component. Our problem is now to predict a single value from some input vector. Because the components are assumed independent, we can ignore the question of which component we are dealing with, and so write $(y_i, \mathbf{x}_i)$ for the i'th example. We write

$$\mathbf{Y} = (y_1, ..., y_N)^T$$

(i.e. the values to be predicted for each example) $\mathcal{X}$ for the matrix

$$\begin{bmatrix} \mathbf{x}_1^T \\ ... \\ \mathbf{x}_N^T \end{bmatrix}$$

Recall that we assumed that one component of $\mathbf{x}$ has the value 1. We can then write the predictions produced by a vector β as $\mathcal{X}\beta$. The simplest method of obtaining a prediction is to solve for β that minimizes

$$(\mathbf{Y} - \mathcal{X}\beta)^T(\mathbf{Y} - \mathcal{X}\beta)$$

(this is **linear regression**). This is a maximum likelihood solution assuming that $p(y|\mathbf{x}, \beta, \sigma^2)$ is normal with mean zero and variance σ^2. Linear regression tends to overfit. One problem is that, if terms in $\mathbf{x}$ are strongly correlated with one another, an appropriate linear combination of these terms is noise; and multiplying this noise by a large number may lead to a good fit. This suggests solving for β that minimizes

$$(\mathbf{Y} - \mathcal{X}\beta)^T(\mathbf{Y} - \mathcal{X}\beta) + \lambda\beta^T\beta$$

(a practice known as **ridge regression**). Here λ is usually chosen by cross-validation [149]. An alternative method to control the size of the coefficients is to solve for β that minimizes

$$(\mathbf{Y} - \mathcal{X}\beta)^T(\mathbf{Y} - \mathcal{X}\beta) + \lambda \sum_i |\beta_i|$$

(a practice known as the **lasso**). The effect of using an L_1 penalty is first to make solutions non-linear in the example values, and — for sufficiently large λ — compelling some components of β to be zero.

A **relevance vector machine** (RVM) uses a Bayesian model to force weights in the regression to be small or zero. One has a model of $P(y|\mathbf{x}, \beta, \sigma^2)$ (normal, mean zero and variance σ^2, as above). We express a preference for smaller weights by writing a prior probability distribution on β, conditioned on hyperparameters α. This distribution, $p(\beta|\alpha)$, is normal with mean zero and diagonal covariance; the variance of each component is $1/\alpha$. The posterior over weights, $p(\beta|y, \mathbf{x}, \sigma^2, \alpha)$, is obtained from Bayes' rule. We then obtain $p(y|\sigma^2, \alpha)$ by marginalization, and choose a value of α that maximizes this

likelihood (say $\hat{\alpha}$). We then choose β to maximize $p(\beta|y, \mathbf{x}, \sigma^2, \hat{\alpha})$. Algorithms for each step appear in [385, 384, 108] (with a variant in [2]); an analysis in [108] shows that some components of α will tend to be infinite, implying a zero component in β.

In each of these cases, one is building a linear model that applies over the whole of the set of examples. This is unattractive for our application (for example, there is fair prospect that there is more than one $\mathbf{y}$ for a given $\mathbf{x}$, something that doesn't happen in linear models). It is natural to try to fit **local linear models**, where the model is linear, but the β varies with the input. We can do this by weighting errors with weights that depend on how close the query is to the examples.

For some input $\mathbf{x}$, we compute a diagonal weight matrix $\mathcal{W}(\mathbf{x})$. There are a variety of possibilities for the diagonal, but using

$$\exp\left[(\frac{1}{2\sigma^2})(\mathbf{x}-\mathbf{x}_j)^T(\mathbf{x}-\mathbf{x}_j)\right]$$

for the j'th entry is a sensible choice (errors close to the example have large weights, those far have small weights; σ may be chosen by cross-validation). We now use β obtained by minimizing

$$(\mathbf{Y}-\mathcal{X}\beta)^T\mathcal{W}(\mathbf{Y}-\mathcal{X}\beta)+\lambda\beta^T\beta$$

for that example. At the cost of solving a linear system for every query, we have a fit that smoothes local linear models into one another without needing to solve non-linear equations. The method has a good reputation for high-dimensional $\mathbf{x}$. Notice that the weight penalties described above can be applied to this problem as well; one computes a ridge regression, lasso, or relevance vector machine for each query as appropriate.

We are assuming that the $\mathbf{x}$ we have computed form an appropriate set of features. However, a more extensive feature representation can be obtained by comparing the query against examples using a **kernel**. Assume that we have a satisfactory **kernel function** $K(\mathbf{u}, \mathbf{v})$. A popular kernel function is

$$K(\mathbf{u}, \mathbf{v}) = \exp\left((\frac{1}{2\sigma^2})(||\mathbf{u}-\mathbf{v}||)\right)$$

(for a choice of σ usually obtained by cross-validation). We can then consider a regression of the form

$$y(\mathbf{x}) = \beta_0 + \sum_{i \in \text{examples}} \beta_i K(\mathbf{x}, \mathbf{x}_i)$$

Note that this remains linear in β, so that the methods expounded above apply. Various details and other applications of this useful trick appear in [**?**].

`Robustness; robust lwr by reweighting`

Chapter 3

Tracking: Data Association for Human Tracking

Mori thing

Agarwal and Triggs 2D tracker, 2D3Dlift/MFKNL5J etc

Tracking people is a means to an end, and trackers should be assessed in that way. Trackers should be reasonably accurate, start automatically (no practical application can use trackers that can't be started automatically), and run for long times without any particular difficulties. These are the correct criteria on which to judge trackers. In our opinion, the literature has, until quite recently, placed too much emphasis on probabilistic inference machinery, while paying insufficient attention to the (possibly dull but certainly essential) vision problems implied by data association. Furthermore, this inference machinery may, in fact, be being used to solve a non-problem (section **??**).

Early Kalman filter based human trackers (for example, Hogg's 1983 paper [], and above) could produce kinematic tracks for people moving without sudden accelerations on reasonably simple, high-contrast backgrounds if started automatically. The more recent trackers we have described use more complex inference machinery, but without any great change in competence.

read and check - is this true; look at Hogg, Chen+Lee, Rohr, Pfinder?

Improvements in competence have come with increased attention paid to tracking by detection schemes. These are well established in, say, face tracking. For example, one can build a fairly satisfactory face tracker by simply running a face detector on frames, and linking over time; smart linking schemes built around affine invariant feature patches can result in very satisfactory tracks [**?**]. Tracking by detection is now capable of building good human kinematic tracks, without relying on background subtraction.

3.1 Detecting Humans

Human detection is difficult, and important. It is difficult because people (usually!) wear clothing of widely varying appearance; because changes in body configuration can result in dramatic changes in appearance; and because different views result in dramatic changes in appearance. There are several important applications. A huge literature now deals with methods to detect pedestrians automatically,

because this is a function that autonomous or semi-autonomous motor-cars will need. There is a substantial literature on detecting and interpreting gestures for human-computer interaction purposes. There is a smaller but growing literature on using various human detection and description methods for understanding the content of various multi-media datasets. There is a small but occasionally startling literature on methods for detecting sexually explicit images. Interest in these areas is not confined to academia; in each of these areas, there are both research efforts by established companies and start-up companies appearing regularly.

No published method can find clothed people wearing unknown clothing in arbitrary configurations in complex images reliably, though, as we shall see, there is reason to believe that this situation will change. The first standard approach to this problem involves matching to one or a family of templates, which might use either spatial or temporal information (or both). We review this area in section **??**. The second standard approach is to identify parts of a person and then reasoning about an assembly of these parts to identify the person. We distinguish between two types of method, according to the type of part: First, one may use parts that are semantic in origin ("arms", "legs", "faces", and so on), and we review this approach in section **??**. Second, one may use parts that are defined by statistical criteria (for example, they form a good codebook for representing the image of the person), and we review this approach in section **??**.

For the rest of this discussion, it is helpful to recall that **face detection** — determining which parts of an image contain human faces, without reference to the individual identity of the faces — is one of the substantial successes of computer vision. Neither space nor energy allow a comprehensive review of this topic here. However, the typical approach is: One searches either rectangular or circular image windows over translation, scale and sometimes rotation; corrects illumination within these windows by methods such as histogram equalization; then presents these windows to a classifier which determines whether a face is present or not. There is then some post-processing on the classifier output to ensure that only one detect occurs at each face. This general picture appears in relatively early papers [374, 293, 375, 323, 324]. Points of variation include: the details of illumination correction; appropriate search mechanisms for rotation (cf. [326] and [333]); appropriate classifiers (cf. [375], [333], [325], [249] and [279]); building an incremental classification procedure so that many windows are rejected early and so consume little computation (see [398, 401, 188, 187] and the huge derived literature). There are a variety of strategies for detecting faces using parts, an approach that is becoming increasingly common (compare [174], [58], [214], [245, 243] and [408]; faces are becoming a common category in so-called object category recognition, see, for example, [112]).

3.1.1 Finding People by Matching Static Templates

Approximately half-a-million pedestrians are killed by cars each year (1997 figures, in [128]). Car manufacturers and governments have an interest in ensuring that cars are less dangerous, and there is a considerable body of research on automated pedestrian detection. Gavrila gives an overview of the subject in [128], which covers cues such as radar, infrared, and so on, which have practical importance but are of no interest to us. For our purposes, this is an example of person detection that may be simpler than the general problem, and is certainly important.

At relatively low resolution, pedestrians tend to have a characteristic appearance. Generally, one must cope with lateral or frontal views of a walk. In these cases, one will see either a "lollipop" shape — the torso is wider than the legs, which are together in the stance phase of the walk — or a "scissor" shape — where the legs are swinging in the walk. This encourages the use of template matching.

Papageorgiou and Poggio represent 128x64 image windows with a modified wavelet expansion, and

present the expansion to a **support vector machine** (SVM), which determines whether a pedestrian is present [287]. SVM's are classifiers, trained with positive and negative examples. For a brief informative discussion of SVM's see [396] or [75]. More extensive information appears in [395, 342, 334], and discussion in the context of a variety of other classifiers is in [149]. The training data consists of windows with and without people in them; each positive example is scaled such that the person spans approximately 80 pixels from shoulder to foot. A variety of image representations are tested, with the modified wavelet expansion applied to colour images performing significantly better than wavelet expansions applied to grey-level images, low resolution pixel values for grey-level images, principal components analysis representations of grey-level images, and the like. The strength of these wavelet features appears to be that they emphasize points that are, rather roughly, outline points. This yields a method for exploiting the restricted range of contours without explicitly encoding contour templates. The wavelet expansion can be reduced in dimension to obtain a faster, though somewhat less accurate, matcher. There are several variants of this approach in the literature [283, 277, 286, 284, 285, 276].

Zhao and Thorpe use stereopsis to segment the image into blocks, then present each block to a neural network [428]. The stereo cue acts as a variant of background subtraction, because there are typically substantial discontinuities in depth between pedestrian and background. A comparison of this system with that of Papageorgiou *et al*.(the version in [284]) suggests it is more accurate, possibly because the stereo segmentation reduces the number of windows that must be searched.

There are a variety of systems that use edge templates explicitly. Gavrila describes an approach that matches image contours against a hierarchy of contour templates using a chamfer distance [127]. The method is oriented to real-time detection. The image is passed through an edge detector, and then passed through a smoothed distance transform (see [25]); a template is evaluated by computing the sum of distance transform values at template feature points, so that a small value results in a match. One needs numerous templates for such a method to be successful (distinct views; distinct phases in the walk), and Gavrila organizes the set of templates into a hierachy using an agglomerative clustering method rather like k-means. Each node of the hierarchy contains a summary template (summaries at nodes deeper in the hierarchy encode more spatial detail), and a representation of the distance of the examples from that summary. Matching proceeds by computing a cost to the representative node at the current level, and testing this against a threshold to determine whether to expand that node or not. A verification step uses radial basis functions to classify those image windows that appear to match edge templates. Gavrila *et al*.describe an improved version of this method, using stereo cues and temporal integration [129]. Broggi *et al*.describe a method that uses vertical edges, the characteristic appearance of the head and shoulders, and background subtraction to identify pedestrians [53].

Wu *et al*.build random field models of image windows with and without a pedestrian, and then detect using a likelihood ratio [417]. Shape is encoded with a random field, and measurements are assumed to be conditionally independent given the shape and some deformation parameters. There is a search over scale, translation and orientation. The considerable technical difficulties involved in evaluating the likelihood are dealt with using a variational approximation. One would expect a performance penalty for using a generative formalism in what is, in essence, a discriminative problem (does this window contain a pedestrian or not?), but ROC curves suggest the method is comparable with strong recent discriminative methods in performance.

Dalal and Triggs give a comprehensive study of features and their effects on performance for the pedestrian detection problem [84]. The method that performs best involves a histogram of oriented gradient responses (a **HOG** descriptor). This is a variant of Lowe's SIFT feature [228]. Each window is decomposed into blocks (large spatial domains) and cells (smaller spatial domains). A histogram of gradient directions (or edge orientations) is computed for each cell. In each block, a measure of

histogram "energy" is computed, and used to normalize the histogram for each cell in the block. This supplies a modicum of illumination invariance. The detection window is tiled with an overlapping grid, within each cell of which HOG descriptors are computed, and the resulting feature vector is presented to an SVM. Dalal and Triggs show this method produces no errors on the 709 image MIT dataset of [287]; they describe an expanded dataset of 1805 images. The paper compares HOG descriptors with the original method of Papageorgiou and Poggio [287]; with an extended version of the Haar wavelets of Mohan *et al*. [250]; with the PCA-Sift of Ke and Sukthankar ([193]; see also [244]); and with the shape contexts of Belongie *et al*. [28]. There is considerable detailed information on tuning of features.

3.1.2 Templates that include Motion

- spatial, spatiotemporal features and insert into classifier - structure: what features, stereo, how many templates, what classifier - also, niyogi adelson motion feature - issues: view and deformation create complexity, often quite successful for pedestrians

Static templates most likely work because the outlines of pedestrians tend to be of limited complexity. While it would be nice to have a formal notion of what this meant, the appropriate comparison is with arbitrary views of people in arbitrary configurations (say, the figure skater of figure 3.12). Pedestrians also tend to move in quite restricted ways — they are typically either standing or walking. Niyogi and Adelson point out that, if one forms an **XYT image** — a stack of frames, registered as to camera motion, originally due to Baker [19] — these motions produce quite distinctive structures (figure 3.1), which can be used to identify motions [272] or recover some gait parameters [271]. Polana and Nelson consider spatial patterns of motion energy, which also have a characteristic structure [303].

This characteristic structure can be used to detect pedestrians in a variety of ways. Papageorgiou and Poggio compute spatial wavelet features for the frame of interest and the four previous frames, stack these into a feature vector, and present this feature vector to an SVM, as above [286]. The result is a fairly significant improvement in detection rate for given false positive rate. The performance improvements that Dalal and Triggs obtain by careful feature engineering (as above) are probably available here, too. The features encode motion implicitly (by presenting the frames in sequence), but not explicitly.

Viola *et al*.use explicit motion features — obtained by computing spatial averages of differences between a frame and a previous frame, possibly shifted spatially — and obtain dramatic improvements in detection rates over static features ([402, 403]; see also the explicit use of spatial features in [77, 280, 281], which prunes detect hypotheses by looking for walking cues). This work uses a **cascade architecture**, where detection is by a sequence of classifiers, each of which operates only on windows accepted by the previous classifier. The classifiers are engineered so that they each have a low false negative rate, so that classifiers early in the cascade reject many windows, and so that the overall cascade is accurate. Features are sums of spatial averages over box-shaped windows in space and time, and so can be evaluated in large numbers extremely quickly; the techniques of classifier and features are due to Viola and Jones [401, 400, 399].

Dimitrijevic *et al*.build a spatio-temporal template as a list of spatial templates in time-order [94]. The spatial templates are edge templates giving the silhouette of the figure, and are matched with a chamfer distance, as above. The spatial templates and the spatio-temporal templates (which are acceptable sequences of spatial templates) are obtained by rendering skinned motion capture data against a blue background from a wide variety of views. The match is scored by computing the time average of chamfer distances. The detector is trained to detect the portion of the walk cycle where both feet are on the ground (other frames could be handled by various forms of temporal interpolation or tracking; see also section 3.5). The paper describes a variety of optimizations helpful to obtain a reasonable speed.

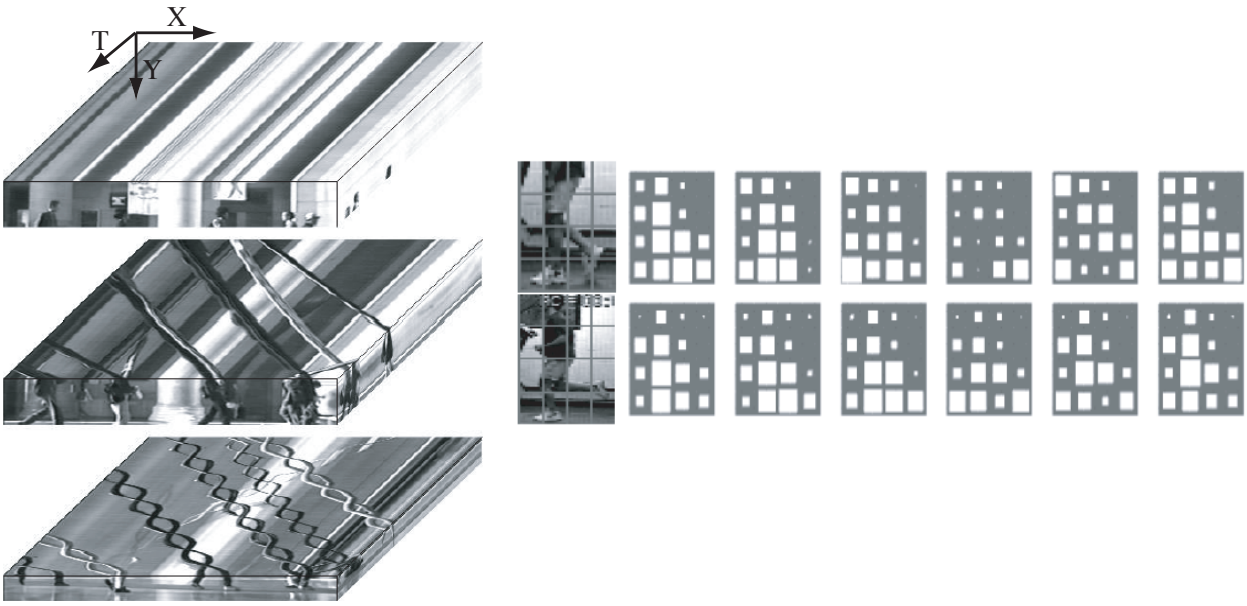


Figure 3.1: **figure 2 of polana nelson recognizing activities , figure 2 of Niyogi Adelson recognizing gait** *On the* **left**, *an XYT image of a human walker. The axes are as shown; the stack has been sliced at values of Y, to show the pattern that appears in the cross section. Notice that, at the torso there is a straight line (whose slope gives an estimate of velocity) and at the lower legs there is a characteristic "braid" pattern, first pointed out by Niyogi and Adelson [272]. On the* **right**, *a series of estimates of the spatial distribution of motion energy (larger white blocks are more energy) for different frames of a walk (***top***) and a run (***bottom***); the frame is rectified to the human figure by translation, and one image frame from each sequence is shown. Notice that, as Polana and Nelson point out, this spatial distribution is quite characteristic [303].*

3.2 Parts and Relations

Detecting pedestrians with templates most likely works because pedestrians appear in a relatively limited range of configurations and views. It appears certain that using the architecture of constructing features for whole image windows and then throwing the result into a classifiers could be used to build a person-finder for arbitrary configurations and arbitrary views only with a major engineering effort. The set of examples required would be spectacularly large, for example. This is unattractive, because this set of examples implicitly encodes a set of facts that are relatively easy to make explicit. In particular, people are made of body segments which individually have a quite simple structure, and these segments are connected into a kinematic structure which is quite well understood.

3.2.1 Traditional Parts

All this suggests finding people by finding the parts and then reasoning about their layout — essentially, building templates with complex internal kinematics. The core idea is very old (for example, one might consult [154, 278, 235, 32, 5, 4]), but the details are hard to get right and important novel formulations are a regular feature of the current research literature. It is currently usual to approach this question in terms of 2D representations, which represent a view of a person as a set of body segments — which

could be represented as image rectangles — linked by rotary (and perhaps translational) joints.

The advantage of these 2D kinematic templates is that they are relatively easy to learn. Learning 2D kinematic templates requires the relative scale of body segments, link probabilities, and an appearance encoding for each body segment. It is relatively straightforward to obtain scale information from static images. Link probabilities can be modelled in a variety of ways. It is usually better to represent translation as well as rotation of a link with respect to another; if we now use a distribution that is flat, or near to, within a useful range, we are preferring no legal kinematic configuration over any other. This isn't in accord with reality — most of the time in most footage, people are walking — but is convenient because it doesn't lock us into any particular activity. In this form, link probabilities can be modelled using either static images or anthropometric information.

3.2.1.1 Discriminative Approaches

The first difficulty is that simply identifying the body parts can be hard. This is simplified if people are not wearing clothing, because skin has a quite distinctive appearance in images. Forsyth *et al*.then search for naked people by finding extended skin regions, and testing them to tell whether they are consistent with body kinematics [118, 117]. The method is effective on their dataset (and can be extended to find horses [116]), but is not competitive with more recent methods for finding "adult" images (which typically use whole-image features [42, 11, 184, 423]). Ioffe and Forsyth formalize this process of testing, and apply it to relatively simple images of clothed people [172, 175]. Their procedure builds a classifier that accepts or rejects whole assemblies of body components; this is then projected onto factors to obtain derived classifiers that can reject partial assemblies that could never result in acceptable complete assemblies. Sprague and Luo use this approach to find clothed people in more complex images, by reasoning about image segments [362].

Figure 3.2: **from Mohan et al, figures 5 and 10** *Mohan* et al.*use SVM's to find major body parts (left arm, right arm, head/shoulders and legs) as in the training examples shown on the* **left**. *They then use these SVM's to search frames for components; the response of all part SVM's in each window is pooled and then presented to an SVM which identifies whole figures. On the* **right**, *examples showing good detects; the whole body window is outlined with lines, and the part windows with dashed lines.*

Mohan *et al*.use a discriminative approach not only to identify good assemblies of parts (as above), but also to find body parts [250]. SVM's are trained to detect the whole left arm, the whole right arm, the legs and the head/shoulders (see figure 3.2); because these body components are relatively large, and because the work focuses on pedestrians, it is possible to search for them in an image centered frame — one can inspect vertical boxes of the right size and aspect ratio to tell whether an arm is present. The SVM part detectors produce a score (distance to the separating hyperplane). For each 128x64 window, the top score for each type of part is placed in a slot in a vector, which is presented to a further SVM. Geometric consistency is enforced by finding the top score for each type of part over a subset of the window to be classified. The approach is applied to pedestrian images, and outperforms the methods of [276, **?**].

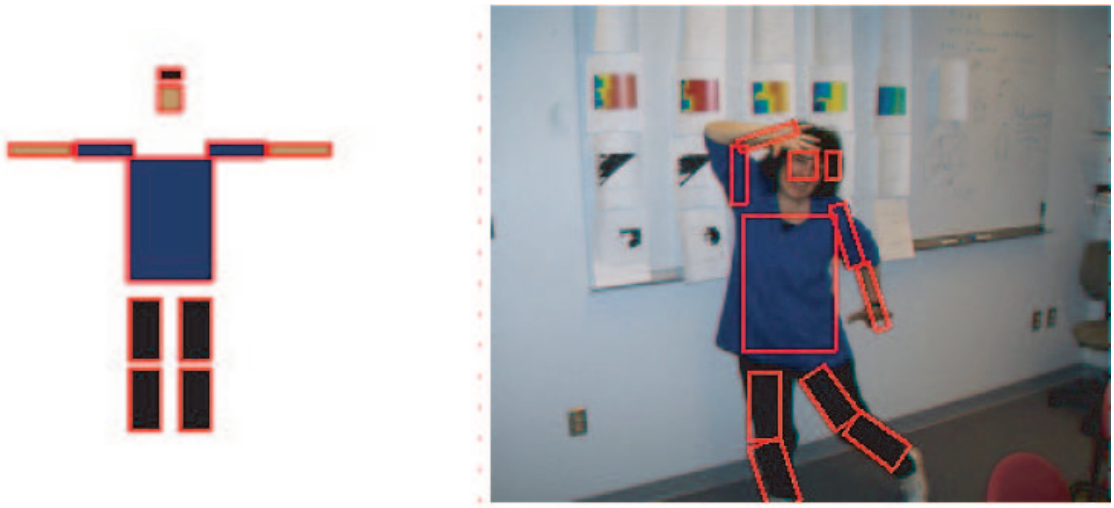


Figure 3.3: **DT's, F+H CVPR 00, fig 4 part** *A* pictorial structure *is a 2D model of appearance as a kinematic tree of segments. Each segment has configuration variables which encode the spatial support of the segment — for example, position and orientation — a local appearance model — for example, the color of a segment — and there is a cost associated with each edge in the tree — for example, the cost of finding a lower leg far from an upper leg. One can find the best instance of such a structure by discretizing the configuration variables for each segment, then using dynamic programming. Felzenswalb and Huttenlocher show that, for properly defined segment-segment costs, the cost-to-go function in the dynamic programming can be evaluated more cheaply than one would expect, meaning that localization can be fast [109, 110].*

3.2.1.2 Generative Approaches

Naked people are easier, because identifying body parts is easier. If we had an encoding of the appearance of the individual parts, this would simplify finding people, because identifying an instance involves dynamic programming; but, done in a straightforward fashion, this is slow because the likelihood evaluation is slow. Felzenswalb and Huttenlocher show how one may use **distance transforms** to speed this process up substantially [109, 110]. In particular, assume that the model is built out of a set of components, the i'th of which has some **configuration** $\mathbf{l}$. We assume that the components are linked in a tree of n nodes. Then to find the best instance, we can discretize the configurations — assume that we use m sample points — and do dynamic programming. However, this will cost $O(nm^2)$, which is unattractive because m is likely to be quite big, particularly if the configurations are high-dimensional. Felzenswalb and Huttenlocher show that, as long as the link cost has a particular form, the cost-to-go functions encountered in the dynamic programming problem are, in fact, generalized distance transforms, and so can be computed in $O(m)$ time (so that the whole thing costs $O(nm)$, which is a useful improvement). The paper demonstrates these models being used in two contexts: finding people and finding cars. People are modelled with rectangles of fixed size and known color (appearance is modelled with image color) and can be localized quite effectively (figure 3.3). Kumar *et al.*extend this model to incorporate boundaries into the likelihood and use loopy belief propagation to apply it to arbitrary graphs (rather than trees); the method is applied to pictures of cows and horses [208].

Song *et al.* use a variant of tree-structured models to identify human motion. They identify local image flows at interest points in an image, using the Lucas-Tomasi-Kanade procedure for identifying

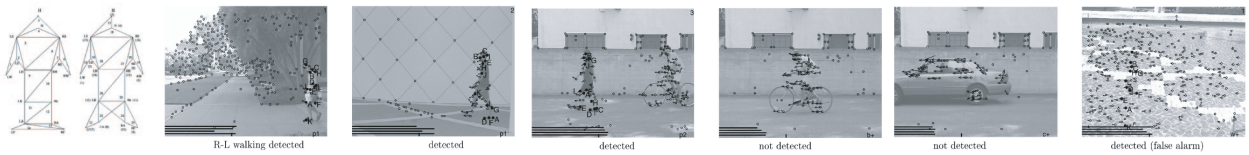


Figure 3.4: **devatracker/01206511, figs 1 and 8** *On the* **left**, *two triangulated graph models of the human figure. Each node represents the state of some interest point on the body; because the graph has a triangulated form and simple cliques, the junction tree is easy to obtain and inference is relatively straightforward (one could use dynamic programming on the junction tree). Song* et al. *use this representation to detect people engaged in known activities, using learned models to infer the form of the distributions represented by the edges of the graph [360, 361]. They detect flow at interest points in the image, then use these models to identify the maximum likelihood labelling of the image interest points in terms of the body interest points; detection is by threshold on the likelihood. On the* **right**, *some detection examples. Note the method is generally successful.*

and tracking localizable points [360, 361]. For a fixed view of a fixed activity, flows at various interest points on the body are strongly related, and discriminative. They build a triangulated graph, whose nodes represent the state of each interest point on the body and whose edges represent the existence of a probabilistic relation between the nodes. Because this graph is triangulated, the junction tree is straightforward to find and inference is relatively simple (see, for example, [183]). They then detect human motion by identifying the best correspondence between image flow features and graph nodes and testing against a threshold. One requires multiple models for multiple activities, though how many models might be needed to cover a wide range of activities and aspects is a difficult question. The method is effective at identifying human motion; note that frames are explicitly not linked over time, something that doesn't seem to cause any real difficulties for the method, which should be seen as an early track-by-detection method.

The advantage of a tree-structured kinematic model, that one can use dynamic programming for detection, extends to a mixture of such trees. However, adding temporal dependencies produces a structure that does not allow for simple exact inference, because the state of a limb in frame t has two parents: the state in time $t-1$, and the state of its parent in frame t. Ioffe and Forsyth attack this problem with a form of coordinate ascent on $P(\mathbf{X}_0, \ldots, \mathbf{X}_k | \mathbf{Y}_0, \ldots, \mathbf{Y}_k)$ [173]. They use a mixture of trees as a template. Spatial links are learned from static images and temporal links simply apply a velocity bound. The posterior is maximised by an iterative procedure, which interleaves two steps maximising over space in a particular frame while fixing all others, and maximising over time for a particular limb segment, while fixing all other segments. Each step uses dynamic programming. Segments are assumed to be white, or close; the model doesn't encode the head position, which occasionally leads to arms and legs getting confused. As figure 3.5 indicates, fair tracks are possible without a dynamical model. One should see the work of Sigal *et al.*(section 2.3.1.1; figure 2.10) as involving a similar, but more sophisticated, inference procedure.

3.2.1.3 Mixed Approaches

Ronfard *et al.*use a discriminative model to identify body parts, and then a form of generative model to construct and evaluate assemblies [318]. Their approach searches for parts that are on a finer scale than those of Mohan *et al.*(upper arms vs. arms), and these can't be found by looking for boxes of a

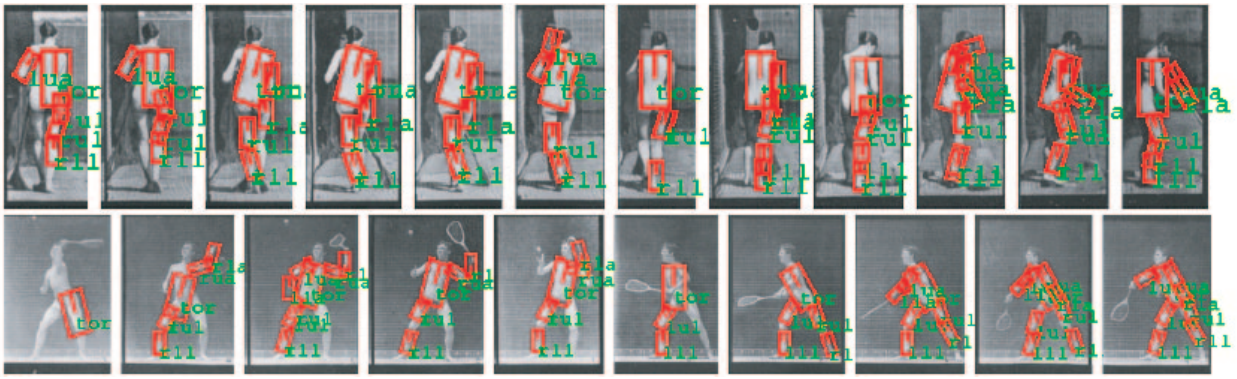


Figure 3.5: **MOT, part of figure 3** *Ioffe and Forsyth build a 2D model of a person as a set of segments, modelled using a mixture of trees to capture aspect phenomena [173]. In an image sequence, each segment except the root has two parents — the corresponding segment in the previous frame, and that segment's parent in the model. The appearance model of each individual segment is crude — segments are light bars of fixed scale. Authors find the best sequence of models by interleaving optimization over time with optimization over space; the result is a fair track, despite significant changes in aspect.*

fixed size, orientation and aspect ratio. This makes it a good idea to search for body parts over scales and orientations — in effect, a search in a part-centered coordinate system. They compare an SVM part detector and an RVM (section 2.5.2) part detector, both applied to features that consist of filtered image grey levels within the window; authors suggest that more sophisticated features, for example those of Dalal and Triggs (section 3.1.1), might give improvements. Each of the detectors produces a detection score. People are modelled as a 2D kinematic chain of parts, with link scores depending on a weighted sum of position, angle and detector scores. The chain is detected with dynamic programming, but the savings obtained by Felzenswalb and Huttenlocher (section 3.2.1.2) do not appear to be available. The weights used in the sum are obtained by a novel application of SVM's. The authors collect a large number of positive and negative examples of links, use a linear SVM with link terms as features to classify them, then use the weights produced by that linear SVM as weights in the link cost. Detection performance is strong; however, there are no standard datasets for evaluating detection of people in arbitrary configurations so comparisons are difficult.

Mikolajczyk *et al*.use discriminative part detectors, applied to orientation images and built using methods similar to those of Viola and Jones (see section 3.1.2), to identify faces, head-and-shoulders, and legs [246]. Non-maximum suppression isolates detected parts. Once a part is found, it predicts possible locations for other parts, which are used to drive a search. Finally, the assemblies that are found are presented to a likelihood ratio classifier. Micilotta *et al*.use discriminative detect hands, face and legs; a randomized search through assemblies is used to identify one with a high likelihood, which is tested against a threshold [242]. Similarly, Roberts *et al*.use a randomized search to assemble parts; parts are scored with a generative model, which is used to obtain a proposal distribution for joints [317].

3.2.2 Parts as CodeBooks

3.3 Tracking by Matching Revisited

The difficulty with background subtraction is that the background may vary in appearance in very complex ways (the camera may move; objects may wave and flutter; illumination can change; etc.) and at some point it is more effective to code the variations in the foreground than those in the background. One of the better known methods for doing so comes from an attempt to solve the likelihood problem. Toyama and Blake encode image likelihoods using a mixture built out of templates, which they call **exemplars**. Assume we have a single template — which could be a curve, or an edge map, or some such. These templates may be subject to the action of some (perhaps local) group, for example translations, rotations, scale or deformations. We model the likelihood of an image patch given a template and its deformation with an exponential distribution on distance between the image patch and the deformed template. The normalizing constant is estimated with Laplace's method. By doing this, we duck the likelihood problem. But there is a more important advantage; we can have multiple templates, which encode the important possible appearances of the foreground object. State is now (a) the template and (b) the deformation parameters, and the likelihood can be evaluated conditioned on state as above.

`must have an earlier discussion of the likelihood problem`

We can think of this method as a collection of template matchers linked over time with a dynamical model. The templates, and the dynamical model, are learned from training sequences. Because we are modelling the foreground, the training sequences can be chosen so that their background is simple, so that responses from (say) edge, curve, and the like detectors all originate on the moving person. Choosing templates now becomes a matter of clustering. Once templates have been chosen, a dynamical model is estimated by counting; authors do not discuss this point, but it seems likely that some form of smoothing would be useful, because if one has many templates and relatively short training sequences, observing that one template never follows another does not establish the probability of the event is zero. Smoothing techniques for problems of this form are a popular tool in the statistical natural language community, and several appear in Manning and Schutze's book [**?**].

What makes the resulting method attractive is that it relies on *foreground enhancement* — the template groups together image components that, taken together, imply a person is present. The main difficulty with the method is that many templates may be needed to cover all views of a moving person. Furthermore, inferring state may be quite difficult. Authors use a particle filter; but if one views a particle filter as a type of randomized search started using dynamics, as above, then it is clear that this search will be more difficult as the movement is less predictable and as the number of templates increases. Part of the difficulty is that the likelihood may change quite sharply with relatively small changes in transformation parameters.

Spatial templates can be used to identify key points on the body. Sullivan and Carlsson encode a motion sequence (of a tennis player) using a small set of templates, chosen to represent many frames well [373]. These templates are then marked up with key points on the body, and matched to frames using a score of edge distance that yields pointwise correspondence; they show that a rough face and torso track, obtained using a particle filter, improves the correspondence. The key points are transferred to the markup, and the correspondence between edge points is used to deform the matched template to line up with the image; this deformation carries the keypoints along (figure **??**). Finally, the configuration of the keypoints is significantly improved using a particle filter for backward smoothing. Loy *et al.* show that such transferred keypoints can be used to produce a three dimensional reconstruction of the

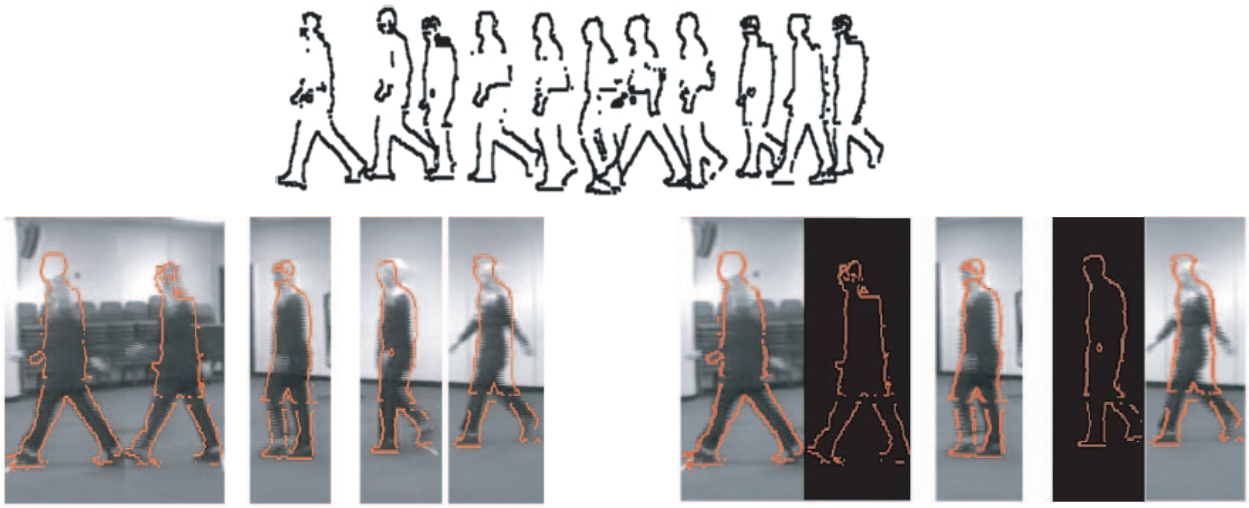


Figure 3.6: **Figure 3, 6 and 7 from Toyama-Blake IJCV paper** *Toyama and Blake [390, 389] track a 2D model of a person by learning a set of templates — which they call exemplars — from other sequences of moving people. The image consists of a deformed template and noise, and state is given by which template is rendered, and the deformation through which the template is rendered. The likelihood is obtained from a comparison between the template and the image. Tracking uses a particle filter. On the* **top**, *a typical set of templates, consisting of edge points (one may also use curves, region textures, and so on). On the* **lower left**, *a track displayed by rendering the template deformation pair with the largest posterior. On the* **lower right**, *a track of the same sequence obtained with some frames blank; notice that the dynamical model fills in reasonable templates, suggesting that such a tracker could be robust to brief occlusions.*

configuration of the body [229] (figure **??**).

3.4 Templates allowing Easy Inference

Lee and Nevatia `Devatrackers/leemotion05`

3.5 Models of Appearance

This leaves us with building a model of appearance. We must choose an encoding of appearance, and determine what appearance each segment has. The trackers we have described up to this point train models of appearance using one or another form of training data; but one could try to build these models on the sequence being tracked. The advantage of doing so is that these appearance models can be specialized to the individual being tracked — rather than attempt to encode human appearance generally, which appears to be difficult. This is the only place where, for example, we can clearly tell what color clothing is being worn by the subject.

Ramanan and Forsyth encode appearance using color — the texture changes produced by shading on folds in clothing make texture descriptors unhelpful — and determine appearance for each segment by clustering. Their algorithm assumes known scale and known link probabilities. Since individuals don't

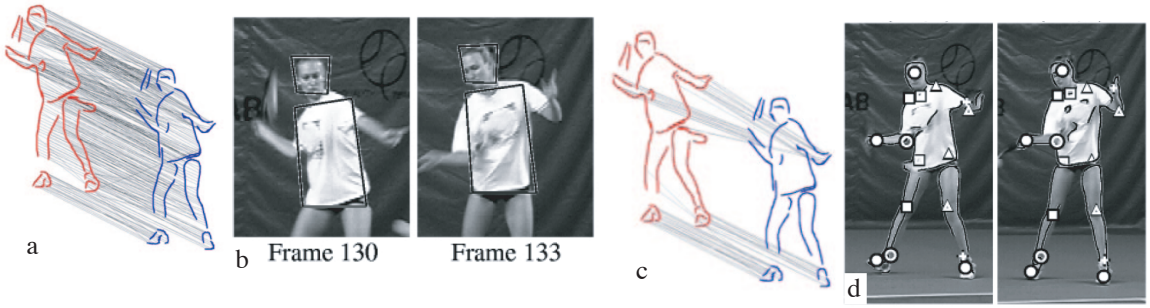


Figure 3.7: **Figure 3, part of 7, 8 part of 11 from Sullivan and Carlsson Devatracker/23500629.pdf** *Sullivan and Carlsson encode a motion sequence (of a tennis player) using a small set of templates, chosen to represent many frames well [373]. Matching uses a topological criterion that can generate large numbers of matches* **(a)***; but by obtaining a rough face and torso track* **(b)***, they can localize matches* **(c)***. This makes it possible to transfer key body points marked on the templates by hand to the frames, and by deforming the templates to obtain good estimates of the image location of these key points* **(d)***. Finally, the configuration of the keypoints is significantly improved using a particle filter for backward smoothing.*

change clothing in track sequences, one can expect that body segments look the same over the sequence, and so there should be many instances of the true segments in a long sequence. Furthermore, the correct segments lie in distinctive configurations with respect to one another in each frame, if detected. This constraint is more easily exploited by looking for torso segments first, because they're larger and tend to move more slowly. Ramanan and Forsyth use a filter tuned to parallel edges separated by a particular image distance to identify candidate torso segments; they then cluster these and prune clusters that are stationary. They look for arm and leg segments near each instance of a candidate torso segment, and if enough are found, declare that the candidate represents a true torso in appearance. Now the appearance of each arm and leg segment can be determined by finding segments near the torso that lie in the correct configuration and have coherent appearance (this is simplified by the useful observation that left and right arms and left and right legs typically look the same). Tracking now becomes a straightforward matter of detecting instances of each model in each frame, and linking those that meet a velocity constraint.

This displays some advantages of a tracking by detection framework, and the difficulties that result from relying on a dynamical model. First, recovery from occlusion, people leaving frame or dropped frames is straightforward; because we know what each individual looks like, we can detect the individual when they reappear and link the tracks (this point is widely acknowledged; see, for example, [85, 259]). Second, track errors don't propagate; when a segment is misidentified in a frame, this doesn't fatally contaminate the appearance model. Difficulties occur if different individuals look the same (although one may be able to deal with this by instancing) or if we fail to build a model.

Ramanan *et al.* demonstrate an alternative method of building a model. Assume that people occasionally adopt a pose that is (a) highly stylized (and therefore easy to detect) and (b) displays arms and legs clearly (so that appearance is easy to read off). Then, if we reliably detect at least one instance of this pose without false positives, we can read off an appearance model from the detection. Furthermore, we can make this appearance model discriminative, because we have a set of pixels that clearly do lie on the segments, and others that clearly do not. It is an empirical property that people do seem to adopt such poses, even in sequences of quite complex motions. They are relatively straightforward to detect

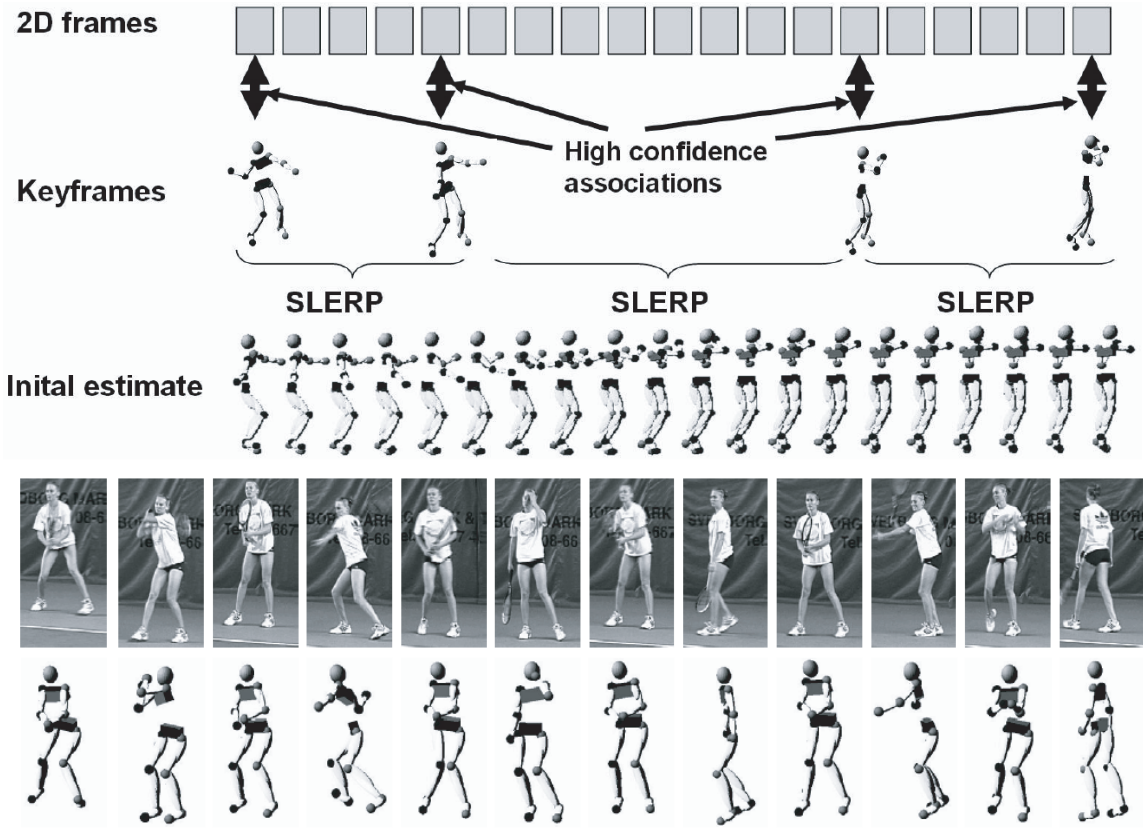


Figure 3.8: **Fig 8, Part of fig 11 from Loy, Eriksson, Sullivan, Carlsson,Devatracker/loy_ eccv04.pdf** *Loy* et al. *show that key body points, localized with a strategy similar to that of Sullivan and Carlsson, can be lifted to 3D [229]. Each 2D template has an associated, hand-built, 3D configuration. This configuration is attached to each template matched in the sequence. Intermediate 3D reconstructions are then obtained with an interpolation technique, constrained to keep limb lengths constant (***top***). On the* **bottom***, reconstructions in 3D obtained from a tennis sequence.*

by matching an edge template using a pictorial structure model. Notice that we are helped by the detection regime here — we don't need to detect every instance, just enough to build an appearance model, but we don't want false positives. Ramanan *et al*. use logistic regression to build discriminative models for each limb segment, then a pictorial structure model to detect. Again, tracking is a simple matter of detecting instances of the model and linking those that meet a velocity constraint. These discriminative models significantly reduce the difficulty of searching for an instance of a person, because much of the image is discarded by the models. In particular, the models can emphasize aspects of appearance that distinguish a particular individual from that individual's background. In his thesis, Ramanan shows that a discriminative model of appearance results in significantly better tracking behaviour (figure 3.13).

```
   Howe stuff
Shaknarovich?  sp stuff
```

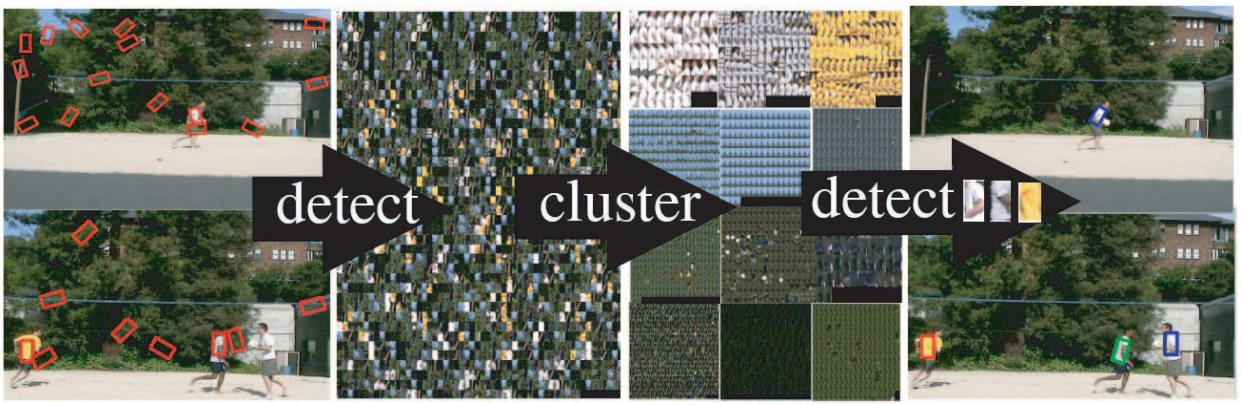


Figure 3.9: **deva's thesis** *Ramanan and Forsyth build an appearance model automatically using the sequence being tracked. They exploit the fact that they know people are to be tracked, and at what scale — meaning that the size of the segments and their layout is known. They identify putative segments using a filter tuned to parallel sets of edges, searching for torsos first. The appearance — in this case, color — within detected segments is then clustered; large clusters with instances in many frames, some of which move, are kept. Once torso appearance has been obtained, a similar procedure, now focussing on segments near the torso, yields upper and lower arm segments.*

3.6 Evaluation

`talk about most recent Black work here?`

There is no current consensus on how to evaluate a tracker. In our opinion, it is insufficient to simply apply it to several video sequences and show some resulting frames (a practice fairly widespread until recently). Counting the number of frames until the tracker fails is unhelpful: First, the tracker may not fail. Second, the causes of failure are more interesting than the implicit estimate of their frequency, which may be poor. Third, this sort of test should be conducted on a very large scale to be informative, and that is seldom practical. Trackers are — or should be — a means to a larger end, and evaluation should most likely focus on this point. In this respect, trackers are probably like edge-detectors, in that detailed evaluation is both very difficult and not wholly relevant. What matters is whether one can use the resulting representation for other purposes without too much incovenience.

A fair proxy for this criterion is to regard the tracker as a detector, and test its accuracy at detection and localization. In particular, if one has a pool of frames each containing a known number of instances of a person, one can (a) compare the correct count with the tracker's count and (b) check that the inferred figure is in the right place. The first test can be conducted on a large scale without making unreasonable demands on human attention, but the second test is difficult to do on a large scale. Ramanan and Forsyth use these criteria; their criterion for whether a particular body segment is in the right place is to check the predicted segment intersects the image segment (which is a generous test) [314, 346].

Sigal *et al* construct a 3D reconstruction, and so can report the distance in millimetres between the true and expected positions (predicted from the posterior) of markers [346]. Figure 3.14 shows results from these evaluations.

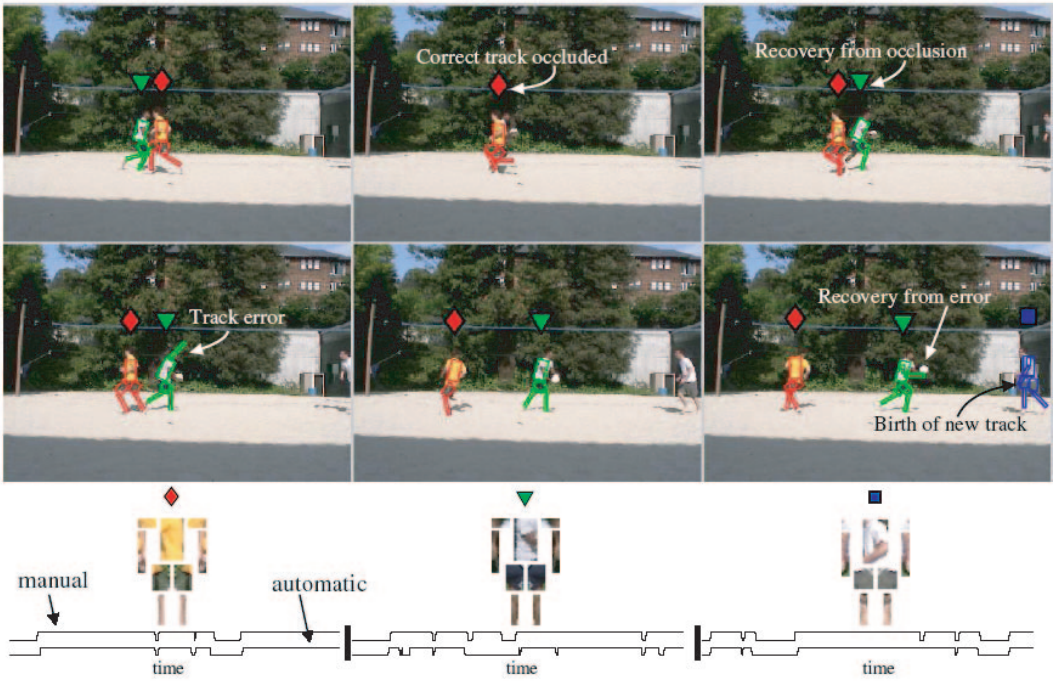


Figure 3.10: **Fig 7, CVPR 03, RamananForsyth** *Ramanan and Forsyth build an appearance model for segments in a 2D model of a person automatically, using methods described in the text and in figure* **??** *[314]. They then track by detecting instances of this appearance model in frames and linking instances across time. The advantages of this tracking by detection strategy are that one can identify particular individuals, recover from occlusions, from errors in the track and from individuals leaving the frame. The* **top** *shows frames from a tracked sequence; on the* **bottom***, appearance models for each of the three individuals identified by their appearance modelling strategy.*

3.7 Notes

There are numerous papers treating characteristic appearance of motion fields. In addition to those discussed, one might consult [301, 294, 295, 301, 298, 302, 304, 217, 44, 219, 218]. Particular efforts have been directed to detecting periodic motion; one might consult [223, 222, 335, 336, 68, 215, 381, 78, 79, 80, 81, 141, 140].

```
   other template based trackers?
Important open problems
Multiple people
Hands
Fast Gesture
Is there much in the 3D 2D issue
Tracking Applications - video puppetry
```

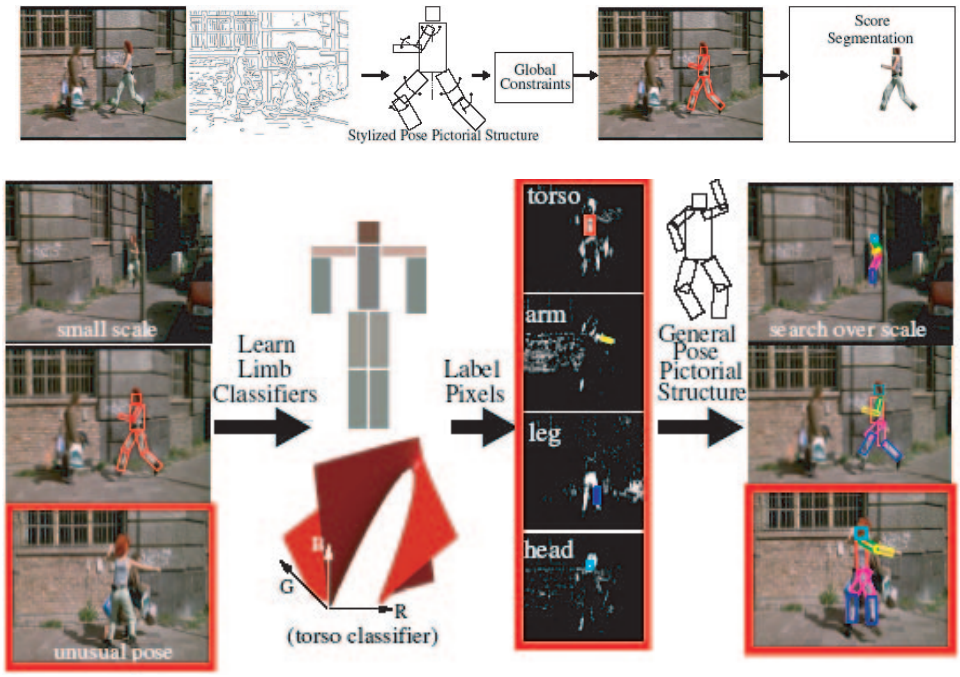


Figure 3.11: **Deva's thesis, also CVPR05** *Ramanan* et al. *demonstrate that one can build appearance models by looking for human configurations that show all limbs and are easily detected. It turns out that, even in quite short sequences of people engaging in quite extreme behaviour, one can find lateral walking views.* **Top:** *These views can be detected by using a pictorial structure model on an edge-based representation, using quite low entropy links to impose the requirement that one has a lateral view of walking. This detector is tuned to produce no false positives — false negatives are quite acceptable, as long as one instance is found.* **Bottom:** *Once an instance has been found, we have the basis of a discriminative appearance model, because we know what each limb segment looks like and we have a lot of pixels that do not lie on a limb segment. Ramanan* et al. *build a discriminative appearance model for each body segment using logistic regression, then apply a pictorial structure model to the* output *of this process — so that a good segment match contains many pixels where* $P(segment|pixel\ values)$ *are high. The resulting tracker is illustrated in figure* **??**.

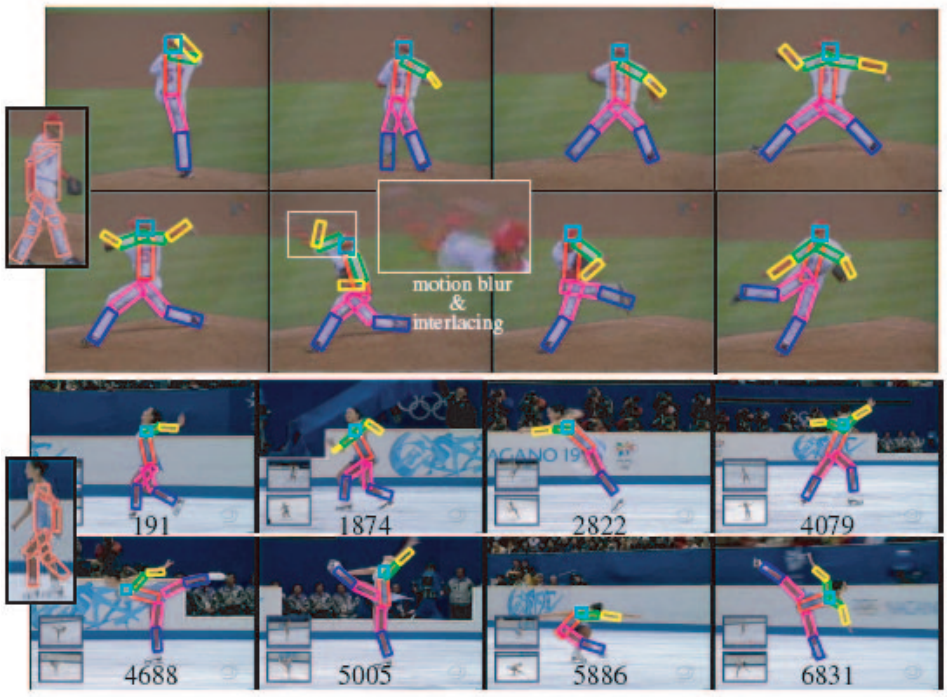
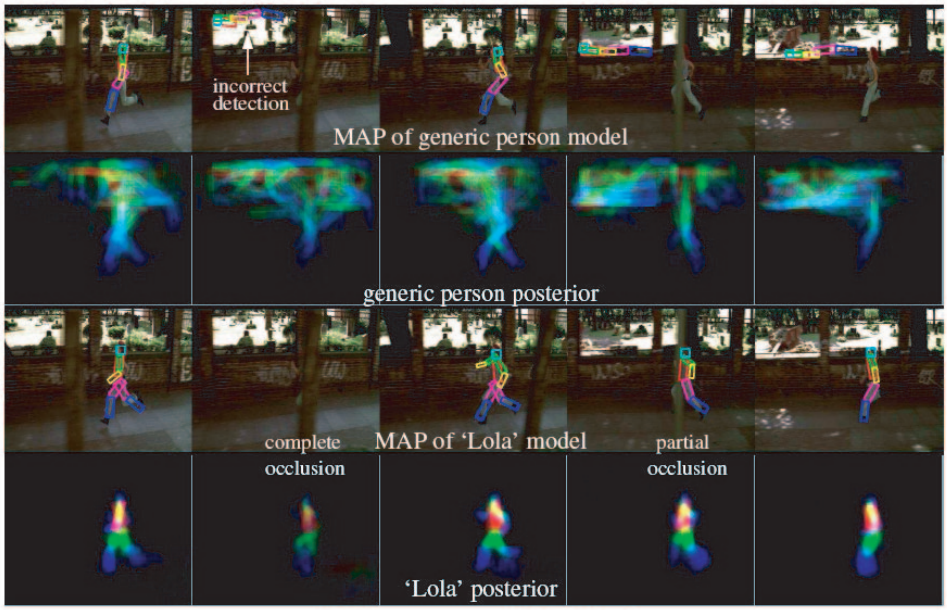


Figure 3.12: **Deva's thesis** *Frames from sequences tracked with the methods of Ramanan* et al.*, where a discriminative appearance model is built using a specialized detector (figure), and then detected in each frame using a pictorial structures model. The figure shows commercial sports footage with fast and extreme motions. On the* **top***, results from a 300 frame sequence of a baseball pitch from the 2002 World Series. On the* **bottom***, results from the complete medal-winning performance of Michelle Kwan from the 1998 Winter Olympics. We label frame numbers from the 7600-frame sequence. For each sequence, the system first runs a walking pose finder on each frame, and uses the single frame with the best score (shown in the* **left insets***) to train the discriminative appearance models. In the baseball sequence, the system is able to track through frames with excessive motion blur and interlacing effects (the* **center inset***). In the skating sequqnce, the system is able to track through extreme poses for thousands of frames. The process is fully automatic.*



% of frames correctly localized

Model	Torso	Arm	Leg
Generic	31.4	13.0	22.2
'Lola'	98.1	94.3	100

Figure 3.13: **Deva's thesis** *Ramanan shows that tracking people is easier with an instance-specific model as opposed to a generic model []. The* **top two rows** *show detections of a pictorial structure where parts are modeled with edge templates. The figure shows both the MAP pose — as boxes — and a visualization of the entire posterior obtained by overlaying translucent, lightly colored samples (so major peaks in the posterior give strong coloring). Note that the generic edge model is confused by the texture in the background, as evident by the bumpy posterior map. The* **bottom two rows** *show results using a model specialized to the subject of the sequence, using methods described above (part appearances are learned from a stylized detection). This model does a much better job of data association; it eliminates most of the background pixels. The table quantifies this phenomenon by recording the percentage of frames where limbs are accurately localized — clearly the specialized model does a much better job.*

Sequence	Torso		Arms		Legs	
	D	FA	D	FA	D	FA
Jumping Jacks	94.6	0.00	87.4	0.56	91.8	0.19
Walking	99.5	0.41	84.3	0.41	68.2	1.01
Street Pass	100	0.00	66.7	0.93	42.4	3.02
Weave Run	92.9	0.00	23.3	2.89	63.0	1.92

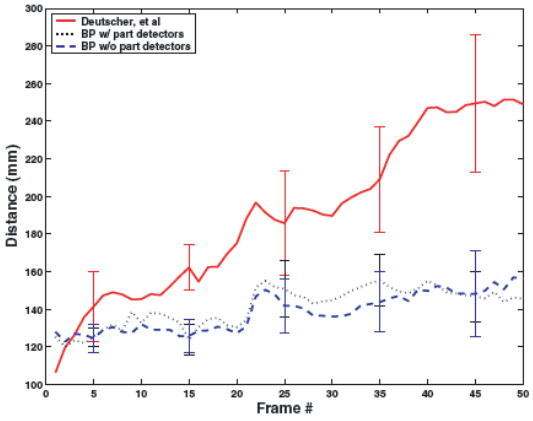


Figure 3.14: **Part of Figure 8 from devatrackers/0135063 and Table 1 of devatrackers/cvprfinal** *On the* **left**, *reports of the percentage of limb segments in the track that overlay the actual limb segments (D) and that are false alarms (FA) for a series of tracks using the methods of Ramanan and Forsyth, reported in []. On the* **right**, *distance between points on reconstructed 3D models obtained using the methods of Sigal* et al. *and tracked motion capture markers supplying ground truth; there are two baselines, the method of Deutscher* et al., *which fairly quickly loses track, and belief propagation without part detectors, which is surprisingly good.*

Chapter 4

Motion Synthesis

There are a variety of reasons to synthesize convincing looking human motion. Game platforms are now very powerful and players demand games with very rich, complex environments, which might include large numbers of non-player characters (NPC's — which are controlled by the game engine) engaged in a variety of activities. These figures need to move purposefully, react convincingly to impacts, and be able to change their activities on demand. Ideally, the motions are clean and look human; players can control characters smoothly; and there are no jumps or jerks resulting from sudden, unanticipated demands — which might originate either with a player or with game AI. Typically, this industry is willing to sacrifice a degree of quality if it can produce a very large volume of motions, and do so quickly. The film industry has traditionally been less interested in computational motion synthesis, largely because human animators — or, for that matter, actors — are still the best way to get high quality motion.

Another, perhaps less frivolous in purpose, is the simulation industry. Commodity graphics hardware has advanced to the point where many of the "immersive virtual reality" simulation and training applications which were proposed during the 1990's are now actually becoming quite practical. Many of these applications require environments that must be populated with humans. Currently, most such applications make do with minimally realistic human figures, but as recent computer games have demonstrated it is now possible to render humans with very realistic static appearance. Variations in rendering style alter a viewer's perception of motions [157, 156]. As the characters' appearance improves so too does viewer expectations concerning the characters' motion. More realistic characters with a more interesting range of behaviors present substantial challenges.

Situation simulations used for training milliary, rescue, and other hazardous-duty personnel are currently predominantly populated by unrealistic human characters. While these characters suffice for some aspects of training, they still place strong limitations of the simulation's potential effectiveness: a fire-rescue worker's response to a mannequin with the word "victim" is fundamentally different to the response that would be elicited by a character that behaves and appears like a frightened 10 year old child. Similar, but more gruesome, arguments can be advanced concerning the need for realistic humans in combat simulations [434, 435]. In either case, the desired goal is that the user become immersed in the simulation to the point where they behave as if the situation were real, and we believe that realistic simulated humans are required for this to happen.

The computer vision community needs models of human motion to recognize people and reason about what they are doing. In what follows, we argue that such models are best encoded by motion synthesis algorithms. Reasoning about what people are doing is a core problem, the solution of which would enable many significant applications. For example, current "smart" motorcars can keep and

change lanes autonomously and can determine if there's a moving obstacle in front of them, but cannot determine whether the obstacle is (say) a child (in which case it must be avoided at all costs) or a large dog (which may be hit to avoid creating a more dangerous situation). Furthermore, they cannot identify pedestrians walking along the pavement and determine whether any appears to be planning a dangerous step into the road. In another application, one seeks to understand human behaviour in public spaces. Currently, the architecture of public places is based around a set of design "rules of thumb", which are used to produce designs that people will enjoy and use; this is not always successful (e.g. [131, 422]). A better understanding of how people behave could produce enhanced architectural design rules (see, for example, [422]). As another application, it is currently possible to analyse surveillance video automatically, but only for such locations as airports and car parks. This is because in these places, one can typically determine whether activity represents a problem without knowing much about the activity — a person on a runway is a problem, pretty much whatever they're doing. However, more subtle activities such as analysing bulk video evidence to identify who passed what to whom, when, is completely out of our reach at present. Part of the problem is that it remains very difficult simply to tell whether a motion was human, and what the actor was doing.

4.1 Fundamental Notions

4.1.1 Motion Capture

Motion capture refers to special arrangements made to measure the configuration of a human body with (relatively) non-invasive processes. Early systems involved instrumented exoskeletons (e.g. []; the method is now usually seen as too invasive to be useful except in special cases) or magnetic transducers in a calibrated magnetic field (e.g. []; the method is now usually seen as unreliable in large spaces). More recent systems involve optical markers. One can use either **passive markers** (for example, make people wear tight-fitting black clothing with small white spots on them) or **active markers** (for example, flashing infrared lights attached to the body). A collection of cameras views some open space within which people wearing markers move around. The 3D configuration of the markers is reconstructed for each individual; this is then cleaned up (to remove bad matches, etc.; see below) and mapped to an appropriate skeleton.

Engineering issues: Motion capture is a complex and sophisticated technology; typical modern motion capture setups require a substantial quantity of skilled input to produce data. There are five main sources of difficulty. First, people can **move fast**, meaning that cameras must have high frame rate — 120 Hz is now common — with attendant difficulties of getting pixels out of the camera fast enough. It is now typical to use cameras that produce only reports of marker position, rather than full frames of video. To capture many fast movements, high acceleration movements or multiple people, one needs a large measurement volume (otherwise one might miss something). Furthermore, it is desirable that each marker is seen by at least two cameras, something that is difficult to achieve when there are many people because the bodyparts occlude one another. Some very fast motions can be captured only with specialized stroboscopic equipment [382]. All this leads to the second problem, that one needs **many cameras** (which is expensive, and creates calibration and algorithmic issues). Third, one requires **high resolution** in the final data, which compels the use of high resolution cameras, because the cameras can all be quite far away from the marker they view. Fourth, **marker correspondence** is a persistent nuisance. There are two issues: one must link markers across cameras to obtain a 3D reconstruction, and a correspondence error might result in a missing marker and so poor data in a particular frame; and one must link markers across time, where a correspondence error might result in a jerk in the reported

motion. Fifth, **markers slip** on the body, so that reports of marker configuration may not be accurate reports of body configuration.

Cleanup and Skeletonization: Typical workflow involves capturing 3D point positions for markers, discounting or possibly correcting any errors in correspondence by hand, then using software to link markers across time. There are usually errors, which are again discounted or corrected by hand. Motions are almost always captured to animate particular, known models. This means that one must map the representation of motion from the 3D position of markers to the configuration space of the model, which is typicaly abstracted as a **skeleton** — a kinematic tree of joints of known properties, typically all revolute, and modelled as points separated by segments of fixed, known lengths, that *approximates* the kinematics of the human body. The anatomy of the major joints of the body is extremely complex, and accurate physical modelling of a body joint may require many revolute and prismatic joints with many small segments linking them (the shoulder is a particularly nasty example [103, 393], but, for example, the drawings in [**?**] emphasize the complex kinematics of human joints). This complexity is unmanageable for most purposes, and so one must choose a much lower dimensional approximation. Different approximations have different properties — the details are a matter of folklore — and one chooses based on the needs of the application and the number of degrees of freedom of the skeleton. Skeletonization is not innocent, and it is usual to use artists to clean up skeletonized data, essentially by adjusting it until it looks good. The pernicious practice of discarding point data once it has been skeletonized is widespread, and it remains the case that data represented using one skeleton cannot necessarily be transferred to a different skeleton reliably. **Reviews** of available techniques in motion capture appear in, for example [41, 137, 241, 248, 347, 227].

Configuration Representations: For the moment, fix a skeleton. While this isn't usually an exact representation of the body's kinematics, we will assume that giving the configuration of this skeleton gives the configuration of the body. The configuration of the skeleton can be specified either in terms of its **joint angles**, or in terms of the position in 3D of the segment endpoints (**joint positions**). Not every set of points in 3D is a legal set of segment endpoints (the segments are of fixed lengths), so sets of points that are a legal set of segment endpoints must meet some **skeletal constraints**. The set of all legal configurations of the body is termed the **configuration space**; the joint angles are an explicit parametrization of this space, and sets of points in 3D taken with constraints can be seen as an implicit representation.

Skinning: In animation applications, one wants the motion capture data to drive some rendered figure — when the actor moves an arm, the virtual character should do the same. The virtual character is represented as a pool of textured polygons, and one must determine how the vertices of these polygons change when the arm is lifted. The process of building a mapping from configuration — always represented as joint angles for this purpose — to polygon vertices is referred to as **skinning**. Skinning methods typically determine an appropriate configuration for the skin for each of a set of example poses, then interpolate [251]. One represents configuration as joint angles for skinning purposes because using joint positions is unwieldy (one would have to manage the constraints; we are not aware of any advantage to be obtained by doing so).

4.1.2 Controlling Synthesized Motion

One must specify what is desired to a motion synthesis algorithm. While synthesis algorithms tend to vary quite widely, there are not many options for constraints. **Geometric constraints** may constrain: the position or position and orientation of the root; the position or position and orientation of one or more body segments; or, in extreme cases, the exact configuration of the body (in which case the **frame**

constraint can be thought of as a **keyframe**). Geometric constraints may take various forms involving either equalities or inequalities. For example, one may constrain a point to lie on a plane, a line, or a point (which are all equality constraints), or to lie within a region (an inequality constraint).

Depending on algorithmic details, constraints may be either exact or represented as a penalty function. Constraints may be either **summary constraints**, applying to the position and orientation of a summary of configuration such as the overall center of gravity or the root, or **detailed constraints**, applying to individual body segments or particular points on the body. One can apply either **instantaneous constraints**, which constrain at a particular time, or **path constraints**, which constrain to a path over a period of time. It is common, but not universal, to assume that a path constraint comes implicitly with a temporal parametrization. It is usual to assume that impossible constraints are not supplied.

Such constraints can be used to sketch out the structure of a motion in greater or lesser detail, depending on what an algorithm requires. In most cases, however, they don't determine the motion. For example, in some cases quite a precise temporal parametrization of a path may not determine whether a figure must run or walk. Usually, one would like to supply relatively few constraints (authoring constraints is a nuisance), meaning that the resulting motion is usually dramatically ambiguous. There are almost always very many ways to meet instantaneous summary constraints for the start and the end of a motion (i.e. start here at this time, end there at that time). One might dawdle at the start, then sprint; walk very slowly; run, walk, then run, then dawdle, and so on.

Annotation constraints are intended to reduce this ambiguity. These constraints are demands that a motion be of a particular type, that are painted on the timeline. The interesting issue is how one encodes the type of a motion. Arikan *et al.* choose a set of 13 terms ("run", "walk", "jump", "wave", "pick up", "crouch", "stand", "turn left", "turn right", "backwards" "reach", "catch", "carry") that appear to be useful for their dataset [12]. It is desirable to respect the fact that motions can compose — for example, one can run while carrying — and they do so by allowing any combination of these terms to be an annotation. One can visualize an annotation as a bit vector, with 13 entries, one per term. This model ignores the fact that most combinations of annotations — e.g. "stand" and "run" — are meaningless; this is deliberate, because there isn't a principled way to build a space of legal annotations and dependencies between annotations may result in nasty inference problems. Arikan *et al.* then mark up a collection of motion capture data using classifiers. The features are a representation of a pool of motion frames spanning the frame to be classified. The classifiers are trained independently, one per term, by marking up some frames, fitting a classifier, and then repeatedly classifying all frames, viewing and correcting a sample of labelled motions, and fitting a new classifier. This process converges quickly, allowing a large pool of motion to be marked up relatively quickly, probably because it is easy to view a large pool of animations and correctly identify mislabelled motions. The result is a pool of frames of motion capture data, each carrying a vector of 13 bits, each of which is determined independent of the others. Interestingly, Arikan *et al.* point out that, although their model does not exclude inconsistent annotations, relatively few of the 2^13 available annotations are actually applied, and they observe no inconsistent annotations.

4.1.3 Footskate

An important practical problem is **footskate**, where the feet of a rendered motion appear to slide on the ground plane. In the vast majority of actual motions, the feet of the actor stay fixed when they are in contact with the floor (there are exceptions — skating, various sliding movements). This property is quite sensitive to measurement problems, which tend to result in reconstructions where some point quite close to, but not on, the bottom of the foot is stationary with respect to the ground. The result is

that the foot slides on the ground (and sometimes penetrates it). The effect can be both noticeable and offensive visually. Footskate can be the result of: poorly placed markers; markers slipping; errors in correspondence across space or time; reconstruction errors; or attempts to edit, clean up or modify the motion. Part of the difficulty is that the requirement that the base of the foot lie on the ground results in complex and delicate constraints on the structure of the motion signal at many joints. These constraints appear to have the property that quite small, quite local changes in the signal violate them. It is likely that these properties are shared by other kinds of contact constraint (for example, moving with a hand on the wall), but the issue has not arisen that much in practice to date.

There are methods for cleaning up footskate. Kovar *et al*. assume that constraints that identify whether heel or toe of which foot is planted in which frame (but not where it is planted) are available [204]. One can sometimes, but not always, infer such constraints from data automatically and accurately []; such methods can fail when data is noisy, or when the motion involves skidding or sliding. Kovar *et al*. then: choose positions for each planted point, determine ankle poses to meet these constraints; adjust the root position and orientation so that the legs can meet the resulting ankles; compute legs that join the root and the ankle mainly by adjusting angles, but occasionally by adjusting leg lengths slightly; and then smooth the adjustment over multiple frames. The method is effective and succesful. One could reasonably speculate that success at cleaning up footskate is a cue to footplant constraints, suggesting simultaneous estimation of a cleaned up motion and footplant constraints. As far as we are aware, there is currently no attempt in the literature to do this.

It is typical that repeated instances of a particular motion differ by some details in timing; for example, each pace of a walk is usually slightly different. **Time alignment** is therefore a useful tool. Kovar and Gleicher score frame-frame differences using ****, and then align frames with dynamic programming. This approach is usually successful for sequences where it is meaningful — it is unwise, for example, to expect much useful out of an alignment between a sequence of standing and a sequence of jumping (see section **??** for more detail). `fix discussion of kovar gleicher`

4.1.4 Inverse Kinematics and Motion Editing

Footskate cleanup is an example of a more general problem — adjust the joint angles of a motion so that it meets some constraints on joint positions. Assume we have a fixed skeleton; we now wish to clean up a motion referred to this skeleton, perhaps moving a foot position or ensuring that a contact occurs between a hand and a doorhandle. This creates a difficulty for either representation of configuration: if we work with joint angles, we must obtain joint angles such that the constraint is met; if we work with joint positions, we must obtain a set of joint positions that meet both this constraint and the skeletal constraints. We will confine our discussion to the case of joint positions, which is more important in practice.

For the moment, let us consider only a single frame of motion. Write the vector of joint angles as θ, and the joint positions as a function of joint angles as $\mathbf{x}(\theta)$. Assume that we would like to meet a set of constraints on joint positions $\mathbf{g}(\mathbf{x}) = 0$. The problem of **inverse kinematics** is to obtain a θ such that $\mathbf{g}(\mathbf{x}(\theta)) = 0$. The constraint is important in the formulation, because we hardly ever wish to specify a change in every joint position. For example, assume we wish to move the elbow of a figure so it rests on a windowsill — we would like to adjust the kinematic configuration so that the elbow lies at a point, but we don't wish to specify every joint position to achieve this. Notice there is room for some confusion here. In the robotics and theoretical kinematics literature (e.g. []), the problem is almost always discussed in terms of choosing joint angles to constrain the endpoint configuration of a

manipulator. In graphics applications, the term refers to meeting any kinematic constraint.

Under some conditions, closed form solutions are available for at least some parameters (e.g. see [201, 386, 387, **?**]). More usually, one must see this as a numerical root finding problem. The update for Newton-Raphson method involves finding a small change in configuration $\delta\theta$ such that $\mathbf{g}(\mathbf{x}(\theta_0+\delta\theta)) = 0$. We may be able to obtain $\delta\theta$ from

$$\begin{aligned} 0 &= \mathbf{g}(\mathbf{x}(\theta_0 + \delta\theta)) \\ &\approx \mathbf{g}(\mathbf{x}_0 + \mathcal{J}_{\mathbf{x},\theta}\delta\theta) \\ &\approx \mathbf{g}(\mathbf{x}_0) + \mathcal{J}_{\mathbf{g},\mathbf{x}}\mathcal{J}_{\mathbf{x},\theta}\delta\theta \end{aligned}$$

where $\mathcal{J}_{\mathbf{x},\theta}$ is the jacobian of $\mathbf{x}$ with respect to θ, etc. In the ideal case, the product of jacobians is square and of full rank, but this seldom happens. For almost every point in the configuration space, the rank of the jacobian $\mathcal{J}_{\mathbf{x},\theta}$ should be the dimension of the configuration space (if this isn't the case, then we have a redundant angle in our parametrization; we assume that this does not happen). At some points, the rank of this jacobian will go down — these are the kinematic singularities of section 1.5.1. The practical consequence of this is that some position updates may not be attainable (for example, consider the straightened elbow of section 1.5.1; the only instantaneous hand velocity attainable is perpendicular to the forearm). The rank of $\mathcal{J}_{\mathbf{g},\mathbf{x}}$ may be small. For example, if our constraint requires that a point be in a particular place, the rank will be three. This is a manifestation of **kinematic redundancy**, which is a major nuisance. A natural strategy to deal with constraint ambiguity is to obtain a least squares solution for $\delta\theta$ — but the resulting pose may not be natural (one can use other norms, see [90]). A second source of difficulties in the optimization problem are joint limits, which mean that our optimization problem is subject to some inequality constraints on the θ. The **feasible set** of solutions that meet this constraints is not necessarily convex, which can mean the general optimization problem is hard.

Kinematic redundancy is a global rather than local matter. In particular, there may be more than one θ such that $\mathbf{g}(\mathbf{x}(\theta)) = 0$. For example, assume that we wish to constrain a figure to stand with its feet on the floor in given spots, and a hand on a given spot on a wall. Typically, there is either no solution to these constraints — the wall is too far away — or many. The collection of solutions is rather rich (stand next to a wall with your hand on the wall; you can move in all sorts of ways without having your feet move or your hand leave the wall), and could be continuous or discrete.

All this creates a nasty problem. Applying inverse kinematics on a frame-by-frame basis is may produce solutions at each frame that are inconsistent (as a result of kinematic redundancy). This is complicated by the presence of multiple solutions, and the vagaries of root finding. For example, assume we want a solution where the hand is against the wall, as above. In frame n, the root finder converges to a solution where the elbow is below the shoulder; but the start point for frame $n+1$ is slightly different from that for frame n, and the root finder could find a solution where the elbow is above the shoulder. This sort of behaviour results in noticeable and annoying "pops" in the motion. The effect can be countered by adjusting multiple frames simultaneously, but this is expensive computationally; much of the recent literature is a search for efficient approximation methods.

Gleicher shows that one can usefully edit motions — typically, so that they meet constraints that are a small revision of constraints met by the original motion — by adding a displacement [136]. Gleicher minimizes a measure of the size of the displacement subject to the new constraints. There is no guarantee that the resulting motion will necessarily look human, but for small displacements it tends to; this means that the motion author can manage constraints and update process so that the resulting motion looks human. The optimization problem is nasty. Lee and Shin obtain a more manageable optimization problem by representing the motion as a hierarchical B-spline [211]. The displacement is also a hier-

archical B-spline, and they engage in a coarse-to-fine search across the hierarchy. The IK solver at the k'th frame at the n'th level now has the $k-1$'th frame at that level and all frames at the $n-1$'th level available to generate a start point and to constrain the solution. Witkin and Popović modify motions using parametric warps, so that they pass through keyframes specified by an animator [**?**]. Shin *et al.* use similar methods to touchup motion to meet physical constraints (for example, motion not in contact is ballistic and preserves angular momentum), while sacrificing physical rigor in the formulation for speed [343]; see also [377], [**?**] and section **??**. Motion editing in this way is useful, and there are several other systems; a review appears in [138].

4.1.4.1 Resolving Kinematic Ambiguities with Examples

The danger here is that one may obtain poses that do not look human. Motion editing deals with this by being interactive, so that an animator who doesn't like the results can fiddle with the constraints until something better appears (see also [289]). An alternative is to allow relatively few degrees of freedom — for example, allow the animator to adjust only one limb at a time, as Maya does — or to require similarity to some reference pose [420, 427, 376]. This isn't always practical. An alternative, as Grochow *et al.* demonstrate, is to build a probabilistic model of poses and then obtain the best pose [144].

One can do this as follows (for consistency within this review, our notation differs from that of Grochow *et al.*). Write $\mathbf{y}$ for a feature vector describing a pose $\mathbf{x}$ (the feature vector could contain such information as joint positions, velocities, accelerations, etc.). Write $\mathbf{u}$ for the (unknown) values of a low dimensional parametrization of the space of poses. Use the subscript i to identify values associated with the i'th example. Now assume we have a regression model $P(\mathbf{y}|\mathbf{u},\theta)$ for θ some parameters (which in this case choose a model and weight components with respect to one another). We could obtain an inverse kinematic solution by maximizing

$$P(\mathbf{y}(\mathbf{x}),\mathbf{u}|\theta)$$

with respect to $\mathbf{x}$, subject to some kinematic constraints $g(\mathbf{x})=0$. The regression model is built using N examples $\mathbf{y}_i$ (note we do not know $\mathbf{u}_i$ for these examples). We assume the examples are independent and identically distributed (note the independence assumption needs care with motion data; frames may be correlated over quite long timescales), and obtain $\mathbf{u}_i,\theta$ to maximise

$$P(\mathbf{u}_i,\theta|\mathbf{y}_i)$$

Grochow *et al.* use a scaled gaussian process latent variable model as a regression model, and note that some simpler models tend to overfit dramatically. The method produces very good results; authors note that a form of rough-and-ready smoothing (obtained by interpolating between parameters obtained with clean training data and training data with added noise) seems to produce useful models that allow a greater range of legal poses.

While motion editing does not offer direct insight into representing motion, the artifacts produced by this work have been useful, and it has produced several helpful insights. The first is that it is quite dangerous to require large changes in a motion signal; typically, the resulting motion path does not look human (e.g. [137]). The second is that enforcing some criteria — for example, conservation of momentum and angular momentum [343]; requiring the zero-moment point lies within the support polygon [343, 200, 86] — can improve motion editing results quite significantly. However, note that one can generate bad motions without violating any of these constraints, because motion is the result of extremely complex considerations. The third is that examples can help produce quite good results.

motionediting/ulsca2003.pdf edits speech lip signals using ICA Popovic Witkin

4.2 The Motion Graph

Motion capture data is used in very large quantities by, for example, the movie and computer game industries. For each title that will contain human motion, an appropriate script of motions is produced; typically, this involves a relatively small set of "complete" motions that can be joined up in a variety of different ways. This script is captured, and then motions are generated within the game by attaching an appropriate set of these motion building blocks together. Motions captured for a particular title are then usually discarded as re-use presents both economic and legal difficulties.

This suggests a form of directed graph structure encoding legal transitions between motions. The attraction is that if we have such a graph, then any path is a legal motion; thus, with some luck, much of the work of motion synthesis could be done in advance. Furthermore, it may be possible to issue quality guarantees for any synthesized motion if we can do so locally within the graph. This hope has not yet materialized, but remains an attraction of the representation.

There are several ways to implement this graph structure, but the important matter here is a representation of legal motion transitions. The simplest, which we favour as a conceptual (but not necessarily computational) device is to regard every frame of motion as a node and insert a directed edge from a frame to any frame that could succeed it. We will call this object a **motion graph**, and always have this representation in mind when we use the term. An alternative representation is to build a set of unique clips (runs of frames where there is no choice of successor — one could build these by clumping together nodes in the previous representation that have only one successor), use the unique clips as edges and make **choice points** into nodes. In this representation, one thinks of running one clip which ends in a node where we can choose which clip to run next. Finally, we could make each clip be a node, and then insert edges between nodes that allow a cut. Here we must be careful with the semantics, because there could be more than one edge from node to node — it may be possible to cut from clip A to clip B in different ways — and our edges need to carry information about where they leave the source clip and where they arrive at in the target clip. There is no difference of substance between the representations; we favour the first, as we find it easier to think about.

4.2.1 Building a Motion Graph

A set of observed motion sequences is a motion graph (there is a pool of frames, and a set of **observed edges**). This graph can be made significantly more useful by adding directed edges — which we call **computed edges** — from each frame to any frame that could succeed it in some sequence. Typically, we do so by identifying places where we can build a **transition** — a sequence of frames that starts at one frame in the graph (say, frame A_i in sequence A), ends at another (B_j in sequence B), and joins the frames preceding the start to those succeeding finish in a natural motion — and blending as in section **??** to build these transitions. This involves adding frames of interpolated motion.

Links by transitions: Kovar *et al.* build links by testing pairs of frames A_i and B_j to tell whether a transition of fixed length is possible between them, then building that transition [203]. They compare a window of fixed length into the future of frame A with a window of the same length into the past of frame B. Each window is represented as a set of points in 3D, and there are implicit correspondences. The distance is then the minimum sum of weighted squared distances between corresponding points

available by choice of rigid-body transformation applied to one sequence. The weights are necessary because errors in some joint positions appear to be more noticeable than errors in other positions. This distance is computed for every pair of frames for which it exists (the future or the past might be too short). They build transitions between pairs of frames where the distance is a local minimum (the topology being supplied by the order of frames in the original sequences) and is lower than a threshold. The transition is built by aligning the windows with a rigid body transformation, then blending them. Footskate is avoided by identifying frames with footplant constraints, and blending in such a way as to preserve these constraints. There is no time or space deformation (c.f. section **??**). If the motion graph is to be used in game applications, there is real value in allowing a designer to interact with this process, as Gleicher *et al.*show [139]. In this work, the designer can choose among possible "to" frames for a given "from" frame, and can disallow (resp. allow) transitions suggested (resp. discouraged) by the criterion above.

Links by similarity: Lee *et al.* test for a possible link from A_i to B_j by testing a distance between A_i and B_{j-1}, the logic being that if these two are sufficiently similar, then their futures could be interchanged [210]. Notice that this suggests that if A_i can be linked with B_j, then we should be able to link from B_j to A_{i+1}. The distance is obtained as a weighted sum of differences in joint angles, summed with differences in velocities at various points across the body (the choice of weights is important; see below). Two frames can then be linked if the distance is sufficiently small, the velocity term ensuring that the temporal ordering of motion is respected. Links between frames with dissimilar contact states, or that are not local maxima (again, the topology is given by the order of frames in the original sequences), are pruned. Once it is known that a pair of frames can be linked, the future of the from frame is blended with the past of the two frame to build a transition, again inserting new frames.

`No - how does this work?`

Arikan and Forsyth represent frames as a sets of points in 3D in a coordinate frame centered on the torso, and obtain a distance by summing squared differences in positions and velocities in that frame taken together with the differences in velocity and acceleration of the torso frame itself [13]. If this distance lies below a threshold for a pair of frames, we can insert Any edge where that distance lies below a threshold is inserted as a computed edge, with the direction being obtained from considerations of smoothness as below. They do not require any particular combinatorial structure in their graph, and so do not post process.

Cleanup: Some applications require a fast decision at each choice point, meaning it may be hard to look far ahead in the graph when making that decision. In these cases, it is helpful to remove nodes that lack outgoing edges and graph components that cannot be escaped (see figure 4.1). This is best achieved by computing the strongly connected components of the graph (components such that, for any pair of nodes in the component, there is a directed path between them) and keeping the largest [203, 210].

Open issues: The methods we have described have generally been successful at producing usable motion graphs. There remain a number of open issues in building a motion graph. Identifying pairs of frames that allow one to build a transition is probably the right approach, but one could quibble with current implementations. First, it remains difficult to know whether one can or can't build transitions between a pair of frames (see section **??** above). Wang and Bodenheimer demonstrate that the choice of weights in Lee's algorithm is important, and show that better weights than those used in the original paper can be learned from data [406]. One could reasonably hope for a more extensive criterion than just requiring the frames to be close and this seems like a productive area of study. Second, requiring the transitions be of fixed length creates problems can be justified on practical grounds, but seems unwise in general. In fact, there is some tension between requiring that the transition be of fixed length, and that the distance between linked frames be a local minimum. Wang and Bodenheimer show that the size of

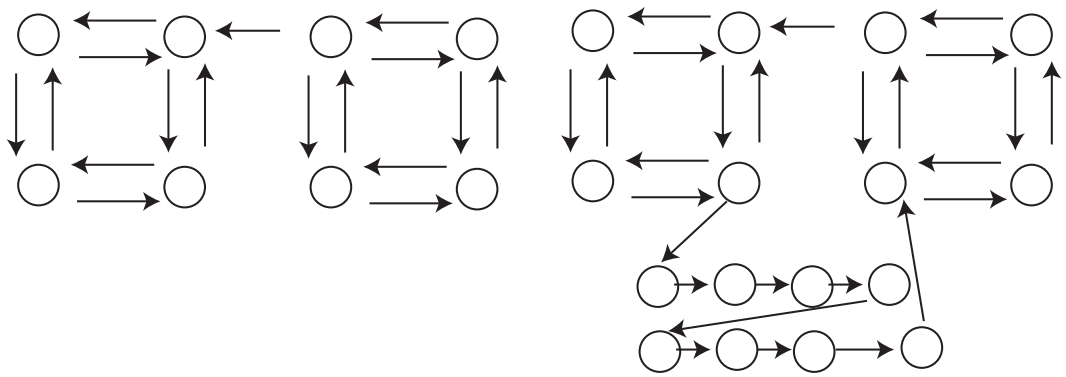


Figure 4.1: *Examples of bad motion graphs. On the* **left**, *a motion graph where it is possible to get stuck in one component. This problem can be avoided by computing strongly connected components and taking the largest, at the possible cost of excluding some frames. The graph on the* **right** *has the difficulty that it is possible to get caught in a motion where no alternatives are available for many frames. This presents a difficulty if one wishes the motion to be responsive. Typically, there is a tension between obtaining high quality motions — which tend to require relatively few edges in the graph — and responsive motions — which tend to need as many edges leaving nodes.*

the difference between frames gives some cue to the length of the transition, as does the velocity [407]. It is probably possible to take the methods described in section **??** (for enriching collections of observed motions) and, by incorporating some form of greedy transition builder, obtain very heavily enriched motion graphs.

`Where is this going? Generally, there is a complex and poorly understood tension between the complexity of a motion graph,`

4.2.2 Searching a Motion Graph

We assume that our method of constructing edges is satisfactory, which means that any path in the motion graph is a motion. We can construct paths in the motion graph using local or global properties. A **local search** involves looking ahead some fixed number of frames. This means that the motion can respond to inputs, but may mean that some constraints can't be met. A **global search** involves looking at entire paths. The resulting motion is less responsive, but more easily constrained.

Local search methods: Kovar *et al* concentrate on choosing the next frame of motion, or, equivalently, choosing one of the outgoing edges at a choice node in the motion graph [203]. Nodes without outgoing edges are a problem; they can be removed with a simple graph algorithm. The choice of edge can be made in a variety of ways: one could look at a game controller, look at the local tangent direction of the desired root path, look at an annotation constraint (figure 4.2), or use a random variable. This latter approach can generate very good background motion when used with care. The trick is not to cut between motion sequences too often (because all methods of constructing motion graphs have flaws, and a path that contains mainly computed edges in the motion graph will tend to explore those flaws, and look bad). This can be achieved, for example, by choosing observed edges with rather higher probability than computed edges.

Local searches can run into problems. The motion graph might contain some frames that can be

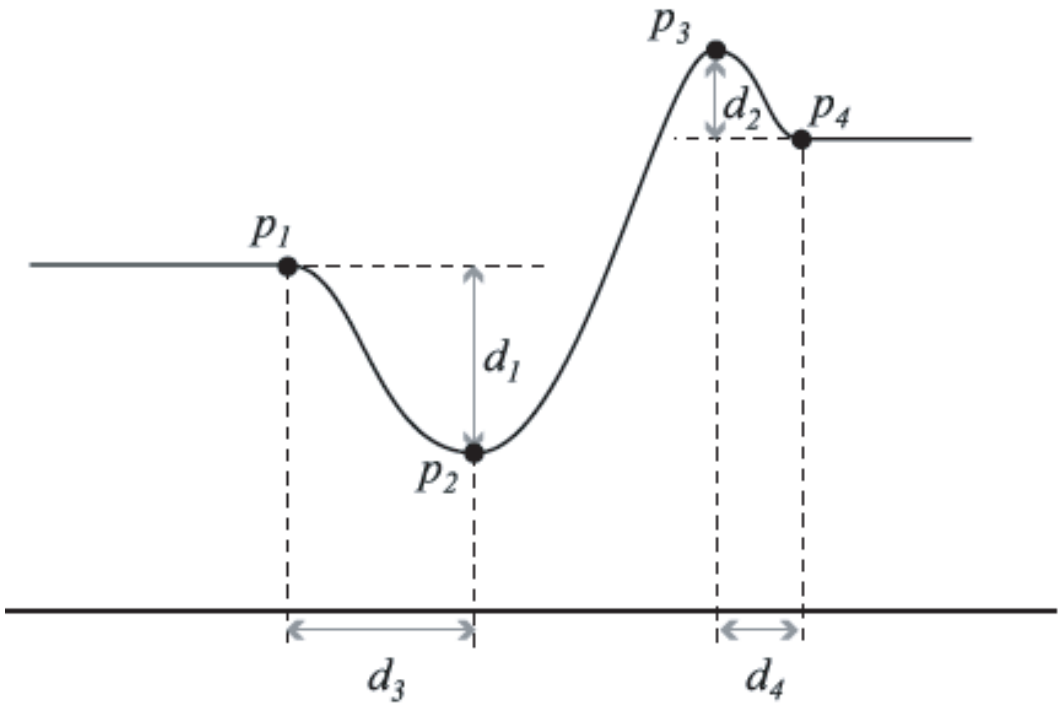


Figure 4.2: **Fig 8, Mographs/mograph.pdf** *These figures show motions synthesized using the motion graph method of Kovar* et al.*to meet path constraints and annotation constraints. The demand path is the coloured path on the ground plane; this is yellow for **"walking"**, green for **"sneaking"** and blue for **"martial arts move"**. The black path shows the projected root path, and the figures are frames sampled at even intervals to give a sense of the motion.*

reached only by making the right choice at a choice point many frames away. In this case, choosing based only on a local criterion could make it impossible to meet some constraints (this is the **horizon problem** — a choice now might lead to trouble that is invisible, because it is on the other side of the horizon separating the future cases we consider from those we don't). We can cope with the horizon problem either by using a representation of available futures when making a choice, by choosing paths using some form of global search, or by enriching the motion graph (section **??**).

Taking the future into account: The body is capable of very fast accelerations. This suggests that, in a motion graph built with enough data, there is a fairly short path from any one frame to any other. In turn, this suggests that the horizon problem wouldn't be a problem if the horizon looked forward sufficiently far in time. However, in this case the range of futures available from a particular frame must be very large. Lee *et al.* encode the future in terms of clusters of frames [210]. These clusters form a graph, where each cluster is a node and there is an edge from one node to another if there is an edge from a frame in the cluster represented by the "from" node to a frame in the cluster represented by the "to" node. A given frame in the motion graph is associated with some node in this **cluster graph**. For any node in the cluster graph, we can construct a **cluster tree** — a tree, rooted at the node under consideration, that gives the nodes in the cluster graph accessible with a fixed number of hops. We now represent the available futures at a given frame by the cluster tree associated with that frame (there is a **cluster path** from the root to each leaf — see figure 4.3). Motions are controlled using either a

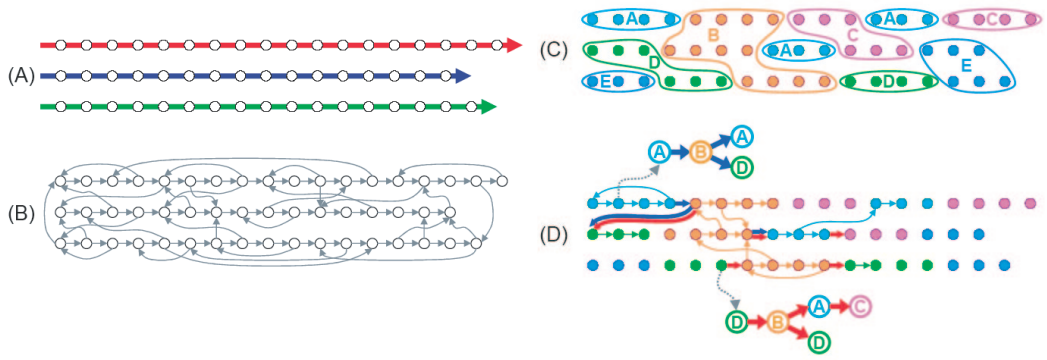


Figure 4.3: **Figure 4 from mographs/lee02interactive** *The figure shows steps in building the motion representation of Lee* et al.*[210]. (A) shows the original motion signals, with observed motion edges. (B) shows computed edges inserted using the methods described in the text; though the figure suggests that no new frames are created in building transitions, this is not necessarily the case. (C) shows a clustering of frames, by appropriate clustering criteria. The choice of clustering criterion is delicate, and relevant considerations are described in the text. (D) shows cluster trees attached to some frames; these trees indicate what types of frame are available from a fixed number of hops leaving the current frame, and give a compressed encoding of the possible futures available from the frame in consideration.*

choice based interface (where the animator chooses at each choice point), a sketch interface — where the sketch provides a demand signal — or a vision interface — where background subtracted frames from multiple viewpoints provide a demand signal. In both the sketch and the vision interface, frames are chosen by scoring the available cluster paths against the demand signal.

For this method, the choice of clustering criterion depends on the application. The alternatives are to represent the body relative to the root of the body, relative to the root of the body in the frame at the root of the cluster path, or in absolute coordinates. The first case is appropriate in uncluttered environments, where one can reasonably expect that any frame can occur at any location and in any orientation. The second can be appropriate when one needs anticipation — for example, synthesizing the run-up to a jump which must leave the ground at a point chosen during the synthesis procedure; this is a need one associates with animations in computer games that emphasize complex movements like jumps. The third case is appropriate to a cluttered environment, where a frame may be usable in only one spot in the motion domain.

Ambiguity: The family of acceptable paths through a motion graph that meet a given set of motion constraints is usually very large, a phenomenon we refer to as **motion ambiguity**. **Local motion ambiguity** arises because most motion data collections contain multiple copies of some motions — typically, walking and running — and that there is a rich collection of links between frames in these motions. As a result, there is a spectacular number of walking motion paths available. One could deal with this issue by clustering, but it isn't the major source of difficulty. The real problem is an important general peculiarity — it is very seldom possible to author constraints on a motion animation that are unambiguous, because the number of constraints required would be unnaturally large. This seems to be a result of the ways in which people find it natural to think about human motion (this issue will re-surface in our discussion of activity representations). For example, if I am instructed to go from point A to point B in some period of time, I can do so in a very large number of ways unless the constraints imply maximum

velocity at all times. One consequence of all this is that the horizon problem should not be a problem in practice because there are lots of paths that meet a set of constraints. Another is that searches for a global motion path can be complicated, because of the number of paths available.

Global Search Methods: Arikan and Forsyth search for complete motion paths that meet given constraints [13]. Such searches are intrinsically off-line so one must sacrifice the goal of interaction, but if the search is fast enough it can be used for authoring animations. Motion ambiguity means that simply applying Dijkstra's algorithm doesn't work, because the algorithm must manage too many intermediate paths. Arikan and Forsyth use a variant of the motion graph where each clip of observed motion is a node, and edges represent acceptable cuts. This means that edges need to be tagged with "from" and "to" frames within the node, and that there are typically multiple self-edges and multiple edges between any pair of nodes. They produce a sequence of compressed version of this graph by clustering edges, so that a pool of edges with similar "from" and "to" frames can be replaced by a single edge with approximating "from" and "to" frames in the more heavily clustered version. They then use a randomized search to find a pool of paths in the most heavily compressed version of the graph; these paths are either refined locally to produce paths in less heavily compressed graphs, or modified. The best resulting path is then reported. They report a trick that can be used to make synthesized motion paths look as though actors are interacting. One obtains measurements of an interaction, then uses frame constraints to construct paths into and out of the interaction.

Low entropy: Human motion appears to be quite predictable in the sense that one can predict the frame that will occur a short while in the future rather well using the current frame — we use the term **low temporal entropy** to refer to this property. This is in tension with what we have seen already (that motion constraints are ambiguous, and that it is generally fairly easy to move between any two frames in the motion graph quite quickly). The most plausible interpretation is that, while one could move between any two frames quickly, people generally don't. This entropy property allows useful approximations for search algorithms.

Annotation based synthesis: One method to control motion ambiguity is to require the synthesis process to produce motions that meet annotation constraints (described in section 4.1.2). Arikan *et al*.use demands that either require the annotation to be present, to be absent, or are "don't care" [12]. The annotations are painted on the timeline. Frames in the motion graph carry annotations, and we must produce a path that meets position and frame constraints, and carries the required annotation at the required time. For the moment, assume that the only geometric constraint is on the start point. Then building a path that meets annotation constraints is a matter of dynamic programming (there are local costs for failing to meet annotation demands, and frame-frame costs for continuity). The dynamic programming problem is too hard to solve in that form, because there are too many frames of motion. Instead, Arikan *et al*.coarsely quantize the graph into blocks of frames that form sequences and then use dynamic programming on a random subset of these blocks. There are then two search activities: refining blocks, and changing the (randomly chosen) working set of blocks. This works well, because ambiguity means that one doesn't miss much structure by random sampling and low entropy means that a quantized path represents the actual solution quite well.

General points: There are several useful conclusions that can be drawn from all this. First, motion capture data clusters well — in the sense that several of the methods we have described cluster motion capture data, and no significant problems appear to result. This is important, because it isn't obvious that useful clusters are available for such high-dimensional data. Second, there is some indirect evidence that motion graphs can be quite well connected. For example, in the method of Lee *et al*., there is an assumption that elapsed time in cluster nodes need not be represented (this means that, when we plan a motion using the cluster graph, we are assuming that when the motion enters a particular cluster

node it isn't going to be stuck there for a very long time before being able to leave the node). This assumption clearly causes no problems, probably because it is a property of reasonably sized motion datasets that a motion can move fairly freely between cluster nodes (it's hard to prove this, but we see the consequences of this property regularly). Motion ambiguity is partially a result of this phenomenon, which creates the rich variety of paths; the other part is an apparent sense on the part of motion authors that relatively few constraints should specify a motion. Finally, low temporal entropy is a phenomenon that will make a nuisance of itself later; for example, the low temporal entropy of motion is the reason that one could obtain apparently excellent annotation results by simply labelling every action in every frame of a surveillance video with ***"walking"***.

4.2.3 Enriching a Motion Collection

Motion graph methods can produce good motions when the data is there. But there will never be enough motion capture data to make motion graph methods able to cope with very general synthesis problems. The methods are attractive because it is fairly straightforward to synthesize motions given one has the data. This has spawned a search for methods that can enrich existing motion collections using what is known about human motion and (perhaps) physics.

There is not yet a comprehensive understanding of how to approach this problem. Part of the difficulty is that human motions quite clearly have some properties allowing composition over the body and over time. These properties are a formidable source of complexity of a form that will defeat naive data-driven methods — for example, to synthesize an actor walking while scratching with the left hand, do we really need to see this particular action? does this mean we need to see walking while scratching with the right hand to synthesize that, too? must we observe scratching different locations with each hand, too?

4.2.3.1 New Motions by Cut and Paste

Simple methods can produce good results for some composition across the body, but not for all cases. Ikemoto and Forsyth build new motions from old by cutting arms or upper bodies off one motion and attaching them to another [169]. Pairs of motions are selected by several different randomized proposal mechanisms, components transplanted between them, and the two results then presented to a classifier which attempts to tag sequences that do not look human. The classifier is quite reliable when presented with motions that are reasonably similar to examples, but tends to be less reliable when presented with dramatically different motions; this is a difficulty, because the whole point of understanding composition is to synthesize good motions that are dramatically different from examples.

What is important here is that the classifier is necessary; many such transplants are successful, but some apparently innocuous transplants generate motions that are extremely bad. It is difficult to be precise about the source of difficulty, but at least one kind of problem appears to result from passive reactions. For example, assume the actor punches his left arm in the air very hard; then there is typically a small transient wiggle in the right arm. If one transplants the right arm to another sequence where there is no such punch, the resulting sequence often looks very bad, with the right arm apparently the culprit. One might speculate that humans can identify movements that both don't look like as though they have been commanded by the central nervous system and can't be explained as a passive phenomenon.

4.2.3.2 Motion Fill-in by Nonparametric Regression

`Pullen+Bregler`

The idea that motion of one part of the body leaves a signature in the motion of other parts of the body is confirmed by work of Pullen and Bregler [310], who built a motion synthesis system that allows animators to sketch part of the motion of the body, and then uses a non-parametric regression method to fill in the details. Joint angle signals are segmented at local extrema. The segments are represented at multiple temporal scales. Animators can then sketch part of a motion — for example, hip and knee angles at a coarse temporal scale — and the system then obtains fragments of joint angle for the other joints and other scales. These are found by matching the fragments of sketched motion to a motion capture dataset (allowing a degree of scaling in both time and angle in the matching process). Typically, there are multiple matches for each fragment. The set of resulting fragments is searched to produce signals that tend to have as many **consecutive fragments** — fragments that succeed one another in the observed data — as possible. These signals may not be continuous (and usually are not, unless the fragments are consecutive), so discontinuous joins are smoothed using a blending technique. Multiple motions can result from this process, and it is up to the animator to choose the best.

```
Other Pullen+Bregler paper
```

The method produces rather good motions, using examples and motion demands from the same "type" of activity. Conditioning on the kind of motion appears to be important — one couldn't reasonably expect that it would be possible to synthesize good football motions from observations of dance — but it is difficult to be precise about what one is actually conditioning on. The fact the method works can be used as evidence in support of the idea that motions have some form of structure that takes in the whole of the body. It is probably unwise to use this view to argue against a compositional representation of motion, because the experiments in the paper don't establish that there is only one possible path for, say, the upper body given a particular set of lower body motions.

4.2.3.3 Enriching Motion Collections with Variational Methods

It is not always necessary to enforce that a motion be physical.

Sulejmanpašić and Popović modify existing motions to obtain revised motions that meet physical constraints using a full dynamical model [372].

4.2.4 How good is a Motion Graph?

```
Pollard and Reitsma stuff

   work through citations in optimsynth paper-1.pdf
motion editors that preserve physical properties also cited in paper-1.pdf
```

In **motion interpolation**, one attempts to produce motions that interpolate between, or extrapolate from, existing motion-capture measurements. A natural procedure is to produce a controller that can track the measurements and then, when measurements are no longer available, produce motions by controlling some body parameters. A variety of approaches that make use of physical simulation have been developed along these lines. Controllers that track motion data provide a useful mechanism for smoothing recorded errors while also adjusting for disturbances not present in the recorded motion [106, 306, 432, 433]. Other approaches make use of hand designed or optimized controllers that operate independently from recorded motion [104, 105, 145, 158, 309]. Building controllers that generate

human-like motion remains an open research problem.

```
ptr to parametric methods and primitives
```

4.2.5 The Limits of the Pure Statistical View

The methods we have described so far in this section take what can loosely be described as a statistical view of motion — in essence, we are expecting, usually implicitly, that a model that is good at representing the motions that one has seen will be good at representing the motions that one will see. This property of a model and a dataset is known as **generalization** in the machine learning community, where quite strong guarantees are available if one has an appropriately representative data set and if the model adopted meets certain criteria (e.g. see []). There is no reason to believe that these guarantees are available in the case of human motion; it appears likely that they never will be.

This is a problem that has to do with both data and models.

There is an important issue of datasets here that clouds the picture somewhat. In our opinion, it probably is the case that many significant motion distinctions are "large" — in the sense that they involve huge changes in kinematic configuration — and so quite simple clustering and dimension reduction methods can expose much structure in motion. What remains uncertain is the extent to which the vocabulary of motions that are well-behaved in this way can be used to encode what one does every day — current experimental work covers relatively small ranges of motion, because motion data is difficult to collect in large volumes. Furthermore, it isn't currently possible to collect data without being intrusive — there are no collections of motion data that can be said to represent "what people do". Finally, there is a significant difficulty with rare motions. In some applications, not encoding a motion that people do relatively seldom is entirely appropriate (for most animation applications, for example, relatively small amounts of sensibly collected motion data is quite sufficient). In other applications, one should be able to encode even very rare behaviours (think contortionist), so that they can be reported.

```
The compositional confusion

Synthesis and objects

link to kovar and gleicher align and linear smooth
link to safanova hodgins thing at SCA 05
Difficulty of the optimization problem
Introduce with IK; then Witkin Kass, various path editing, hand off to
Gleicher review

Witkin+Kass?
```

Motion graph methods produce motions with non-parametric models. This is all very well when appropriate data is available, but the methods have little to say about what to do when there isn't any relevant data. An alternative is to build what are, in essence, parametric models. One can do so by seeing human motion as a physical phenomenon. There is a linkage of body segments, with known masses, inertias, etc. which moves under some set of physical laws subject to forces that are obtained either from notions of efficiency (section **??**) or from controllers (section **??**). Of course, one must then either determine appropriate notions of efficiency or appropriate controllers to obtain motion that looks good.

It remains uncertain how helpful physical constraints are in producing good motions. Instead of emphasizing the physical aspects of the problem, one could attempt to build statistical models of motion directly (section **??**). Such models are usually built around dimension reduction techniques, which we review briefly in that section.

The general area of building parametric models of human motion is changing fast. At time of writing, the most useful results appear to be coming from attempts to enrich sets of observed motions by building deformations of one form or another that attempt to remain physically viable in a rather general way. We discuss this strategy in section **??**.

4.3 Motion from Efficiency

The motion editing methods we have seen do not require that deformed motions be physical. In fact, these methods are simplifications that originate in a body of research to generate human motion from considerations of physical constraint and energy. This work originates with Witkin and Kass, who introduced the use of **variational methods**, widely known as **spacetime constraints** [415].

We have a jointed figure, whose configuration can be represented by some set of parameters $\mathbf{q}$. These coordinates can be **reduced coordinates**, where any set of values represents a legal configuration of the figure — these could be, for example, root coordinates and joint angles. An alternative is to use **generalized coordinates** `is this right`
, where not every choice of values represents a legal configuration of the figure — these could be, for example, the pose of each separate limb segment; in this case we need constraints to ensure that the limbs don't fly apart. The configuration of this figure is subject to some constraints. For example, a figure that is sliding on the floor will be constrained to have each foot on the floor. This figure is subjected to a set of forces and torques $\mathbf{f}$. Assume the figure is moving for the time interval I. From mechanics, the motion of this figure achieves an extremal value of the time integral of the **Lagrangian** (see, for example [**?**, **?**, **?**, **?**]). We write the Lagrangian as $L(\mathbf{f}(t), \mathbf{q}(t), \lambda, t)$, where λ are the **Lagrange multipliers** (which can be interpreted as the coefficients of generalized workless constraint forces that ensure the motion meets the constraints). Some constraints are **dynamical constraints** (which refer to forces, torques, momenta and the like); we shall write this set as $\mathbf{D}_e(\mathbf{f}, \mathbf{q}, \lambda, t) = 0$ and $\mathbf{D}_i(\mathbf{f}, \mathbf{q}, \lambda, t) \leq 0$. Others are **kinematic constraints** (which constrain configuration); we shall write this set of constraints as $\mathbf{K}_e(\mathbf{q}, t) = 0$ and $\mathbf{K}_i(\mathbf{q}, t) \leq 0$.

Let us confine our attention to an interval where we know which kinematic constraints are active (i.e. which components of $\mathbf{K}_i$ are equal to 0), and write the set of active kinematic constraints including all the equality constraints as $\mathbf{P}(\mathbf{q}, t) = 0$. Write the remaining set of kinematic inequality constraints as $\mathbf{P}_i(\mathbf{q}, t) < 0$. Any physical motion extremizes the Lagrangian subject to these constraints, and, from variational calculus, we obtain the **Euler-Lagrange equations**, which are differential equations satisfied by any motion that does extremize the Lagrangian. We adopt the notation where differentiating by a vector results in a vector of derivatives with respect to each component. Write the Euler-Lagrange equations as

$$\mathbf{E}(\mathbf{f}, \mathbf{q}, \lambda, t) = 0 = \begin{pmatrix} \frac{d\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right)}{dt} - \frac{\partial L}{\partial \mathbf{q}} - \lambda^T \frac{\partial P}{\partial \mathbf{q}} - \mathbf{f} \\ \mathbf{P}(\mathbf{q}, t) \end{pmatrix}$$

Notice that we now have algebraic equations that constrain derivatives. Equations of this form are known as **differential-algebraic equations**; they have a (well-deserved) reputation for creating nasty numerical problems (a fair place to start is [152, 153]).

Now we wish to choose a motion that meets the dynamical constraints, and where some other criterion — which might measure, for example, work — is extremised. Write this criterion as

$$\int G(\mathbf{q}, \mathbf{f}, \lambda, t)dt$$

The problem becomes

$$\begin{aligned} \text{Maximize } & \int G(\mathbf{q}, \mathbf{f}, \lambda, t)dt \\ & \text{Subject to:} \\ \mathbf{E}(\mathbf{f}, \mathbf{q}, \lambda, t) &= 0 \\ \mathbf{D}_e(\mathbf{f}, \mathbf{q}, \lambda, t) &= 0 \\ \mathbf{D}_i(\mathbf{f}, \mathbf{q}, \lambda, t) &\leq 0 \\ \mathbf{P}_i(\mathbf{q}, t) &< 0 \end{aligned}$$

Witkin and Kass did not use the idea to generate human motions, but demonstrated very attractive animations of a bouncing lamp produced using this method. There are very serious practical difficulties in producing animations of human motion like this. The actual minimization process might be extremely difficult. In fact, there is no prospect of getting a useful result by simply dropping this problem into a commercial optimization package. The state space has complex geometry caused by the internal degrees of freedom, joint limits and the like. Contact and frame constraints can produce unpleasant feasible sets, and one should expect the problem not to be convex. One must encode the function $\mathbf{x}(t)$ with some finite dimensional parameter space, and the choice of encoding may create difficulties; for example, contact constraints tend to produce quite high frequency terms in the motion signal (or, equivalently but rather easier to observe, smoothing the motion signal tends to lead to footskate). There is some reason to believe that a coarse-to-fine representation is useful [226]. One may simplify optimization difficulties by choosing simplified characters (e.g. [388, 308, 307, 106]; freefall diving is a particular interest [76, 225]) or by exploiting interaction with an animator (e.g. [74]). Ngo and Marks produce motions for quite complex characters using spacetime optimization by building motions out of **stimulus-response** pairs — parametric packets of motion that are triggered by some parametric test ([270, 269]; see also [239] for other motions built out of packets). The precise set of packets, and the parameters of those packets, are chosen using search by a genetic algorithm (see also the work of Sims [348]). There is no claim that these motions necessarily appear human.

The choice of objective function can affect the resulting motion and is by no manner of means obvious. It is occasionally asserted that human motion should minimize some choice of mechanical energy. One should place little weight on this idea for most motions because there are too many other important considerations that shape how we move. For example, Wu and Popović need a specially crafted objective function that allows for the enormous energy expenditure required at takeoff to obtain convincing bird flights [72]. As another example, the energy saved by using a slow reaching motion might be far outweighed by that lost by getting to the target fruit too late. For that matter, even more energy could be saved by not moving at all; but at some cost.

For these reasons, spacetime optimization has not to our knowledge been used to generate complete human motions over long periods. The closest to doing so is the work of Rose *et al.*, who generate **motion transitions** — short sequences of motion that join specified frames "naturally"— using an optimization procedure that minimizes the total squared torque moving the upper body [321]. The legs are controlled kinematically, using either manual or automatically supplied constraints for footplants. Spacetime optimization has, however, been of tremendous value in deforming existing motions.

4.3.1 Simplified Characters, Modified Physics and Reduced Dimensions

Popović and Witkin use characters with simplified kinematics, and model muscle forces explicitly (the muscle is modelled as a **proportional-derivative controller** attempting to drive a degree of freedom to a setpoint) [308]. Their method produces physically plausible motions that meet constraints and are close to observations. They represent major features of motion using **handles** — vector functions of configuration, typically a map onto some lower dimensional space, the details of which vary between applications. For example, if one wished to ensure a motion preserved contact, appropriate handles might be the position of points on the figure. A spacetime optimization is used to fit the simplified model to observed motion data, resulting in handles $\mathbf{h}_s(\mathbf{q}_s)$; a second spacetime optimization produces a simplified model that meets the constraints with handles $\mathbf{h}_t(\mathbf{q}_t)$; and the handles for the observed data are $\mathbf{h}_o(\mathbf{q}_o)$. They now seek to produce a final motion $\mathbf{q}_f$ with handles $\mathbf{h}_f(\mathbf{q}_f) = \mathbf{h}_o(\mathbf{q}_o) + (\mathbf{h}_t(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s))$ (that is, displace the handles of the original motion with a displacement computed from the simplified figure).

They do this by optimizing an objective function that penalizes **mass displacement**, which is computed as a sum of squared magnitudes of differences in positions between corresponding sample points on the final motion and the observed motion, weighted by the mass at that sample point. As a result, degrees of freedom in the final animation that are not constrained by the handles are derived from the original motion. The optimization is constrained by the requirement on the handles (above) and physical constraints on the motion. The parameters are configuration and muscle demands. The spacetime method appears to benefit considerably from the relatively few degrees of freedom in the simplified character and the presence of an initial point (the observed motion).

Liu and Popović produce character animations from rough initial sketches using an optimization method by breaking the motion into phases, simplifying the physical constraints, and, where necessary, exploiting the animator's input [221] They then identify transitions — where the figure moves from one set of constraints applying to another — and require the animator to provide frames for these transitions, which tend to be a particular source of difficulty for optimization methods. They must now produce a series of motion clips to fill in between these transitions. There are two important cases: **ballistic motion**, where there is no contact — the body is in flight, as in jumping, diving, etc. — and **constrained motion**, where there is some contact. In ballistic motion, if we use reduced coordinates, then all external forces are due to gravity (so the acceleration of the center of mass is $\mathbf{g}$) and angular momentum is conserved. Constrained motions are required to have a momentum curve of a particular form (figure 4.4), which is consistent with biomechanical observations.

The objective function is a sum of three terms: a measure of mass displacement ; a measure of **coordinate velocity**, which penalizes large changes in the degrees of freedom to enforce frame-frame coherence; and a measure of **static balance**, which penalizes large distances between the center of mass and the location of point constraints. The objective function and the constraints are functions of $\mathbf{q}(t)$ (and its derivatives) and the control points for the momentum curve. The method does not constrain forces or torques at joint, and they do not participate in the objective function, which means that they can be ignored (this doesn't mean the motion isn't physical; it means that we assume that the body will supply whatever internal forces or torques are required to follow the motion path). Abe *et al.*drop the mass displacement and coordinate velocity terms in favour of a similarity term, and use a variety of different momentum profiles to produce further variations on motion capture data [1].

There is a real advantage to not constraining forces and torques and not allowing them to participate in the objective function: one does not need to compute them. This means that computing various Jacobians that arise in the optimization procedure can be made linear (rather than quadratic) in the

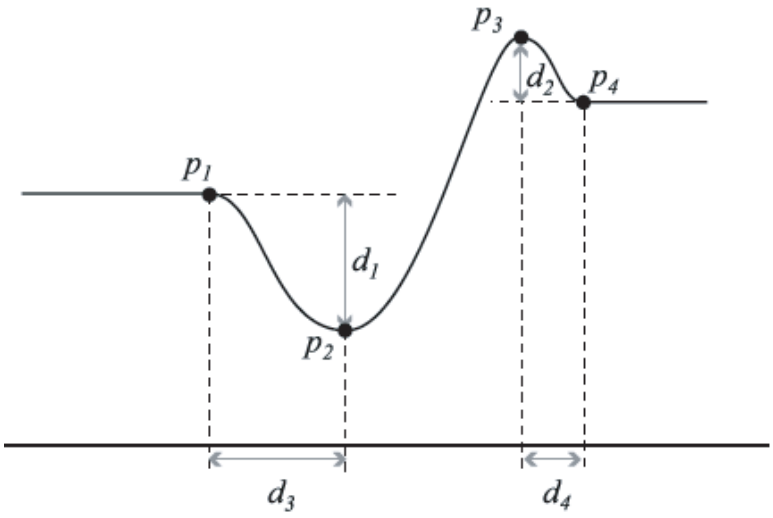


Figure 4.4: **Physicalsmoothing/paper1.pdf, figure 6** *The angular momentum curve for a whole motion for the method of Liu and Popović [221], showing total angular momentum as a function of time. The motion before p_1 and after p_4 is ballistic, so the total angular momentum is a constant. The form of the momentum curve is taken from biomechanical models [282, 199]. The form is imposed by smoothly interpolating p_1, p_2, p_3 and p_4, requiring that $p_2 < p_1$, $d_2 < d_1$ and $(p_2 - p_4)(p_3 - p_4) < 0$. Figure 4.5 shows a motion obtained using this method.*

number of degrees of freedom, as Fang and Pollard show [107].

4.3.2 Start Points

4.3.3 How to Build Objective Functions

Liu *et al*.show a method to obtain simulation parameters from examples [220]. They build a full physical simulation of a jointed body with 18 nodes, 29 joint DOF's and 6 root DOF's, with passive effects at each joint

4.4 Controllers

4.4.1 Motion Blending

The space of human motions: For the moment, let us adopt some encoding of the state of the body (the details don't matter for this discussion, but we'd expect to see the configuration of the root, the configuration of the body relative to the root, velocities and most likely accelerations in this encoding). Because segment lengths don't vary, because velocities are limited and because there are torque limits, not every point in this state space represents a legal motion. It is useful to think of the legal motions as forming a "sheet" in this space. We make no claim on the topology of this object, not even that it is a manifold. We can think of motions as functions from time to this space. These functions must meet some obvious constraints — for example, velocities computed as time derivatives of kinematic configuration need to be the same as corresponding velocities recorded in the state vector. We expect

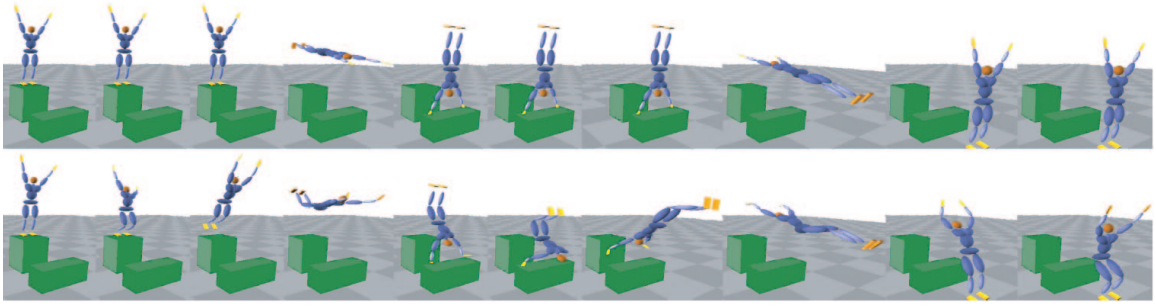


Figure 4.5: **Physicalsmoothing/paper1.pdf, figure 6 Top:** *a motion demand supplied by an animator and* **bottom** *a motion synthesized using the procedure of Liu and Popovč [221]. The motion is obtained by (a) inferring constraints from the demand; (b) extracting transitions from an animator; and then (c) computing a set of clips that meet these transitions and the inferred constraints, have angular momentum curves of the form of figure 4.4 and extremize an objective function that penalizes mass-displacement, coordinate velocities, and out-of-balance configurations.*

other local constraints, too, resulting from torque limits and the like. There should be some form of structure at long time scales — we know, for example, that it is possible to walk backwards for long distances, but that it is very seldom done — but not much is known about these constraints (we discuss the matter in section **??**). We can represent the space of human motions by all acceptable functions from time to our space. It is difficult to make this model more precise, but in the form described it is already useful.

Assume we have two legal states $\mathbf{x}_1$ and $\mathbf{x}_2$ that are close. For many such pairs (though not for all — see figure **??**) we can expect that states that lie on the line segment joining them are also legal. Another way to put this point is that, if the two states are sufficiently close, then the vector $\mathbf{x}_2 - \mathbf{x}_1$ should lie on the tangent space at $\mathbf{x}_1$. Now assume we have two observed motions $\mathbf{f}_1(t)$ and $\mathbf{f}_2(t)$, which run for similar time periods and are sufficiently similar to one another (it is not known how to be precise what this means). We expect — and can observe in data — that repeated versions of the same movement have slightly different temporal parametrizations. For example, each step of a walk can take a slightly different span of time. This means that we will need to massage the temporal parametrization. Assume we can place states in correspondence by a small — again, it isn't currently possible to be precise — change in temporal parametrization $\tau(t)$, so that $\mathbf{f}_1(t)$ is close to $\mathbf{f}_2(\tau(t))$. Under these circumstances, we can expect that $\mathbf{f}_1(t) - \mathbf{f}_2(\tau(t))$ lies close to the tangent space to the space of human motions (it would be on the tangent if we defined what small meant, divided by size, and took an appropriate limit — but it isn't currently easy to be formal about this).

Using tangents to the space of motions: The attraction of finding tangents — or near tangents — to the space of motions is that adding a small amount of a tangent to a motion will most likely result in a good motion. Bruderlin and Williams describe several constructions that identify plausible tangents [56]. They operate in joint angle space, meaning that issues with the root, described below, are hidden but not non-existent. The first potential tangent they identify is $\mathbf{f}_1(t) - \mathbf{f}_1(\tau(t))$ — equivalently, they observe that small temporal scaling of a good motion usually results in another good motion, with perhaps a difference in style. A second plausible tangent involves adding a small offset to joint angles. This is difficult to represent usefully in our notation because we haven't been precise about what's in $\mathbf{x}$; however, the observation that adding joint offsets can result in good motions is confirmed in, for example, [169]. Again, it simply isn't known what offsets will and won't work; Bruderlin and Williams

rely on an animator to look at the motion. A third plausible tangent involves linear filtering of joint angles. Again, some filters work and some don't, but an animator can look at the motion. Finally, they identify the tangent $\mathbf{f}_1 - \mathbf{f}_2(\tau(t))$. This tangent allows **blending** — for at least some values of λ, we expect that

$$\mathbf{f}_1 + \lambda \left(\mathbf{f}_1 - \mathbf{f}_2(\tau(t))\right)$$

is a good motion. In fact, the blending weights can change with time without creating a problem; this may offer some more insight into the structure of the tangent space.

Kovar and Gleicher align multiple motions using monotonically increasing functions of time, obtained using dynamic programming [202]. The motions are then blended linearly. However, doing all this requires careful handling of the root. If our representation contains the root, then we will be able to blend very few motions because even motions that are similar may occur in different places, which is clearly a waste of data. However, we cannot simply strip every frame of root information, because the root path is often quite strongly correlated with the body pose. For example, people use different gaits for fast and slow translational movements. As another example, an actor trying to move quickly along a root path with a sharp kink in it typically makes a form of braking and pivoting step.

The solution seems to be to (a) ignore root information for the whole sequence (rather than per frame) and (b) allow small deformations of the root paths so they line up with one another. One could think of this as identifying $\mathbf{f}_1(t)$, $\mathbf{f}_2(t)$, $\tau(t)$ and a rigid body transformation $\mathcal{T}(t)$ such that $\mathbf{f}_1(t) - \mathcal{T}\mathbf{f}_2(\tau(t))$ is small. Kovar and Gleicher do this by first identifying $\tau(t)$ by minimizing a Euclidean invariant distance function on the frame representation using dynamic programming, and then obtaining $\mathcal{T}(t)$ by transforming frames to align.

4.4.1.1 Difficulties with Blending

Assume we have two motions both captured at the same frequency. Both contain temporally localized large accelerations (for example, they might be grabbing or hitting motions). The temporal parametrization of the motions is slightly different, meaning that the samples are aligned slightly differently in time with respect to the motions. Even at the best possible time alignment, if we blend these motions we expect to lose some of the structure at high temporal frequencies — which would be the large accelerations. The result is a motion that can be "squashy" in appearance and can lose its temporal crispness. This problem doesn't always occur, and might be manageable if one is careful (for example, it might be worth reconstructing motions using some form of interpolation, resampling at very high frequencies, then aligning the resampled motions). A version of this problem occurs for dimension reduction methods, too. `figure`

`Reparametrization, Prolonged Actions, and Tangents`

`p105-park, locomotion based on blending`

4.4.2 Motion Primitives

Our very rough model of the space of motions above doesn't really take long time structure of motions into account. Such structure is evident in how people move on a daily basis. One can walk backward for long distances, but one doesn't; one can intersperse; for that matter, some can walk on their hands,

but few do for long periods. This sort of structure needs to be thought of in terms that are probabilistic, rather than deterministic (because the semantics are that one could but one tends not to).

A natural method for building models of motion on these time scales is to identify clusters of motion of the same type and then consider the statistics of how these **motion primitives** are strung together. There are pragmatic advantages to this approach: we can avoid blending between motions that are obviously different; we can model and account for long term temporal structure in motion; and we may be able to compress our representation of motion with the right choice of primitive model. There is some neuroscientific evidence that motor activities are encoded using motor primitives of one form or another (for example, see []). In animation, the idea dates at least to the work of Rose *et al.*, who describe motion **verbs** — our primitives — and **adverbs** — parameters that can be supplied to choose a particular instance from a scattered data interpolate [320]. The verbs appear to be chosen by hand; within a particular primitive, motions are aligned (c.f. section **??**) and then a scattered data interpolate produces an instance. There is a **verb graph** which gives the combinatorial structure of how verbs can be joined up.

4.4.3 Primitives by Segmenting and Clustering

Primitives are sometimes called **movemes**. Matarić *et al*. represent motor primitives with force fields used to drive controllers for joint torque on a rigid-body model of the upper body [236, 237]. These force fields have a stationary point at a desired hand configuration; different force fields can be superposed to obtain different endpoints. The primitives appear to be chosen by hand. The motions are 3D motion captured arm movement; segment boundaries are obtained by looking for points where the sum of squares of velocity at all joints is small. Del Vecchio *et al*. define primitives by considering all possible motions generated by a parametric family of linear time-invariant systems; if a split of the parameter space results in two sets of motions that are always distinct, that split can be used to derive primitives [397]. The definition of the primitives results in a segmentation algorithm, and authors show that reaching and drawing motions can be distinguished in this framework.

`pointer to primitives/01024148 - which is recognition based on primitives`
`Perona's primitive stuff`

There is quite a lot of evidence that motions segment and cluster well — meaning that one can use various segmentation and clustering processes as intermediate steps in motion synthesis, without serious difficulties resulting. This is not something one would expect, given the dimension of most motion representations. Barbič *et al*.compare three motion segmenters, each using a purely kinematic representation of motion [21]. Their principal components analysis (PCA — see, for example, [120] for this useful standard technique) moves along a sequence of frames adding frames to the pool, computing a representation of the pool using the first k principal components, and looking for sharp increases in the residual error of this representation. Their Gaussian mixture model segmenter regards frames as IID samples from a Gaussian mixture model, then computes the mixture component from which a frame arises. Their probabilistic PCA segmenter works like the PCA segmenter, but obtains a normal probability density from the principal component analysis and then compute the Mahalanobis of new frames from the mean of this model; this segmenter appears to be the best of the three. While there is no agreed way to evaluate a motion segmentation, Barbič *et al*.report segmentations that look good. For our purposes, the most significant point here is that distinct movements tend to be dramatically distinct — one doesn't need to look at fine details of dynamics to segment such motions as ***"walk"***, ***"stand"***, ***"sit down"*** and ***"run"***.

Dimension Reduction: It is natural to expect that any primitive structure in motions could be exposed by reducing the dimension of the data. Furthermore, dimension reduction methods could yield a conveniently compressed encoding of a motion primitive. Fod *et al*. construct primitives by segmenting motions at points of low total velocity, then subjecting the segments to principal component analysis and clustering [114]. Jenkins and Mataric segment motions using kinematic considerations, then use a variant of Isomap (detailed in [181]) that incorporates temporal information by reducing distances between frames that have similar temporal neighbours to obtain an embedding for kinematic variables [182]. They cluster in the resulting space to obtain motion primitives over short temporal scales, then apply isomap again to obtain primitives on longer temporal scales; they report plausible motions.

There is other evidence that relatively few measurements can yield the kinematic configuration of the body — that is, that a low dimensional representation of configuration applies. Chai and Hodgins demonstrate a form of **video puppetry** — where an animated figure is controlled by observations of an actor — using relatively few markers; this approach most likely works because motions tend to be confined to a low dimensional subspace [**?**]. Safanova *et al*.are able to produce plausible figure animations using optimization techniques confined to a low-dimensional space (see [**?**], figure **??** and section **??**).

```
   Safanova figures 2 and 3; dimensionreduction/lowdoptim
Pollard low dimensional stuff
Perhaps Zordan's controller here
Problems with dimension reduction
```

4.4.3.1 Linking Segmentation to the Primitive Model

Segmentation and encoding should interact — we can reasonably expect a good segmentation results in good primitives, but the other way works, too; if one has a good representation of each particular primitive, that could drive segmentation. This is now a commonplace in the machine learning community. Li *et al*. segment and model motions simultaneously using a linear dynamical system model of each separate primitive and a Markov model to string the primitives together by specifying the likelihood of encountering a primitive given the previous primitive [216]. For the moment, assume the segmentation is known and we wish to identify a primitive from some set of observations that have been determined to come from that primitive. We assume that each primitive consists of a sequence of observations $\mathbf{Y}_t$, each generated by a hidden state $\mathbf{x}_t$. We would like the system to have second order dynamics so that the model takes accelerations into account; this is equivalent to assuming that x_t is a linear function of x_{t-1} and x_{t-2}. We can obtain a Markovian model by stacking two state vectors to obtain $\mathbf{X}_t = \left[\mathbf{x}_t, \mathbf{x}_{t-1}\right]^T$. The model of each primitive now takes the form

$$\begin{aligned} \mathbf{X}_t &= \mathcal{A}_t \mathbf{X}_{t-1} + \mathbf{V}_t \\ \mathbf{Y}_t &= \mathcal{B}_t \mathbf{X}_t + \mathbf{W}_t \end{aligned}$$

where $\mathbf{V}_t$ and $\mathbf{W}_t$ are normal random variables with known mean and variance. Notice that $\mathcal{A}_t$ will have the form

$$\left(\begin{array}{cc} \dashv_t \dashv_{t-1} & \mathcal{I} 0 \end{array} \right)$$

(so that one has the right behaviour from the stacked components of the state vector). You should compare this model to the HMM's used for tracking; we have the same model, but now we wish to obtain the values of $\mathcal{A}$ and $\mathcal{B}$ from observations of $\mathbf{Y}_t$, rather than estimate the states. The difficulty

here is that the model is not uniquely specified in this form. For example, assume that $\mathcal{C}_t$ is a sequence of matrices of full rank, then the state sequence $\hat{\mathbf{X}}_t = \mathcal{C}_t \mathbf{X}_t$ taken with matrices $\mathcal{A}_t \mathcal{C}_t^{-1}$ and $\mathcal{B}_t \mathcal{C}_t^{-1}$, has the same likelihood. Li *et al* deal with this by insisting that the states be the projection of the observations on to a subset of the principal components of the observations, and can then estimate $\mathcal{A}_t$ and $\mathcal{B}_t$ with maximum likelihood.

Of course, the segmentation is not known. We will estimate the segmentation and primitives together with an iterative procedure: fix the primitives, estimate the best segmentation; now re-estimate the primitives with that segmentation; etc. This mirrors EM, but one is now using the maximum likelihood segmentation conditioned on the primitive parameters as an estimate of the expected segmentation conditioned on the parameters. The segmentation can be obtained with dynamic programming (Li *et al*. assume that each primitive emits at least 60 frames, which complicates the representation only very slightly). To see that the best segmentation of some sequence of length N into M primitives of length no shorter than L is available using dynamic programming, we build a graph whose nodes consist of statements that frames i to $i+k$ of the sequence were produced by primitive j; there can be no more than $N^2 M$ such nodes. Each node is labelled with the negative log-likelihood of the relevant sequence under the relevant dynamical model. There is a directed edge from each node to any node that can succeed it, labelled with the negative log-likelihood that the one primitive follows the other under the Markov model. We now obtain the minimum value path through this (acyclic, directed) graph using dynamic programming.

The resulting model can be used generatively to produce new motions. Li *et al*. obtain their best results by specifying the body configuration at each change of primitive — so that the model interpolates between these frames. This avoids phenomena like drift (which must occur because of the random noise component) causing minor but annoying effects like the feet floating above or below the ground.

Chapter 5

Describing Activities: Fundamentals

```
ontologies and non-parametric methods
intentions
linguistic ideas
```

Understanding what people are doing is one of the great unsolved problems of computer vision. A fair solution opens tremendous application possibilities, including: improved surveillance systems; a better understanding of what people do in public; better architectural design; and better human computer interfaces. We argue that the time is right for substantial advances in the understanding of this area, and propose to use existing tools from the speech and object recognition community to achieve them.

While there has been extensive study of this topic, it still isn't terribly well understood. One can obtain statistics of some behaviours from coarse scale tracks (e.g. for car parks, see [371]; for architectural domains, see [422]). But understanding activities that depend on detailed information about the body is still hard. We contend that the major difficulties have been (a) that good kinematic tracking is hard; (b) that models typically have too many parameters to be learned directly from data; and (c) for much everyday behaviour, there isn't a clear taxonomy into which to classify observations.

There is a long tradition of research on interpreting activities in the vision community (see, for example, the extensive survey in [164]). There are three major threads. First, one can use temporal logics to represent crucial order relations between states that constrain activities. Second, one can use spatio-temporal templates to identify instances of activities. Third, one can use (typically, hidden Markov) models of dynamics.

5.1 What should an Activity Representation do?

It has been recognized for some time that there are other helpful distinctions (e.g. Bobick [37] distinguishes between movements, activity and actions, corresponding to longer timescales and increasing complexity of representation; some variants are described in two useful review papers [3, 130]). First, we distinguish between short, medium and long timescales. Second, we distinguish between motions that can be sustained (walking, running, waving) and motions that have a localizable character (catch, throw, punch, kick). Since we want our complex, composite motions to share a vocabulary of base units, we use the kinematic configuration of the body, limb velocities, and perhaps accelerations as distinctive features at **short** timescales — which might be of the order of a small number of frames. We define **acts** to be frame labels that can be decided on such very short timescale features — such labels, (for

example, walk-right-leg-stance-left-leg-swing) tend not to have directly useful semantics.

At **medium** timescales, we have **activities** — motions like walking, running, jumping, standing, waving, whose temporal extent can be short (but may be long); such motions are typically composites of multiple acts. Furthermore, activities can be sustained for long periods. We use the term **actions** for motions that have a localizable character and require medium timescales to identify. Both actions and activities may be difficult to identify with only a few frames but are relatively easy to identify from hundreds to thousands of frames. Both actions and activities allow a degree of composition — for example, one could walk and scratch at the same time.

One's interpretation of a view of moving humans is strongly affected by objects nearby. For example, a person standing in an isolated field may be behaving strangely; the same person in the same configuration next to a bus stop is waiting for a bus. We believe that the most natural level at which to start inserting considerations of context into activity recognition is that of activities — where one can pool object detector responses over a long enough sequence of frames to expect quite good behaviour — and define the next layer of the representation to be **motions in context**. Context applies to both activities and actions. These occur at medium timescales, but the nature of the motion in context is determined by both the actions in the sequence and the response of object detectors. We use the term **behaviour** to cover motions at **long** timescales — typically, behaviours such as fighting, exercising or visiting an ATM might be composed of a selection of different motions in context, linked up by activities, and organized in a variety of possible ways and meeting a variety of constraints on temporal ordering.

Our goal is a theory and mechanism for recognizing a wide range of behaviours. There are some important constraints on solutions to this problem. First, we expect that typical behaviours are a composite of many activities, and this composite is not unique — the same behaviour may be represented by multiple sequences of actions, as long as these sequences observe an internal structure. For example, one may scratch or groom at any time while visiting an ATM, but one must type a PIN before retrieving money, and insert a card before typing a PIN (notice that one can't retrieve a card before inserting it, but at some machines one might retrieve the card before typing a PIN; at others, the card is retrieved after typing the PIN and before recovering money). Each activity may itself be one of several different composites of multiple actions, in the same way. Each action might also have compositional properties — for example, one may walk with three-quarters of one's body while scratching with the fourth limb. The modelling strategy must respect both this hierarchical structure and the compositional nature of motion.

Second, we expect that there is not labelled data for each possible case; we cannot simply learn models without any human interaction. This applies to models of actions, activities and behaviours. This difficulty is created by the compositional nature of human motion; the sheer richness of available motions defeats pure data-driven strategies. An important criterion for choosing a modelling strategy is that it be easy for humans to author and to assess rich models quickly. Such models should be amenable to parameter learning from data, but it should not be necessary to see an example of every possible instance of a behaviour to build a model.

Third, we expect that the supervised data that is available may be marked up somewhat inaccurately. Typically, a behaviour will be marked up with activity names (an activity with actions, respectively), but the boundaries of the markup are unlikely to be accurate. We expect the learning algorithm to be robust to some segmentation noise.

Finally, we expect that basic activities with the model — model building, composition, and inference — be relatively straightforward. In this, we follow the experience of the statistical natural language community, that trading expressiveness in models for simplicity of authoring and inference is often

advantageous.

Constraints on Data: An important part of design here is to keep into account what kinds of data are easy to obtain and what difficult, so as to plan model authoring around what is practical. Experience suggests that it is possible to get from minutes to hours of reasonable quality motion capture data; relatively few minutes of video labelled as to actions (these labels are very difficult to produce because they require frame accuracy); minutes to hours of video labelled in reasonable detail with respect to activities and behaviours, accepting poor temporal resolution in the labels; and of the order of months of public observation video. It is relatively straightforward to look at large volumes of labelled motion capture data and correct labels, not least because one can observe many frames simultaneously (e.g. see [12]).

One important source of difficulty is that it is hard to tell which aspects of behaviour should be modelled accurately in order to perform useful tasks. Resolving this requires (a) study of ideas in sufficient generality that they transfer between tasks and (b) some example tasks. But the selection of example tasks is not innocuous. In particular, a distinctive feature of everyday activity is the number of behaviours that appear familiar, but for which the observer may not know a word or even a compact description. In contrast, in some domains (e.g. ballet [61]; gymnastics [8];tai chi [45, 50]; tennis [373]; walking [62]) there are quite specific vocabularies that refer to very precisely delineated behaviours. This is an advantage for building demonstration systems, because one can evaluate them, but may avoid the real difficulty, which is that for most activities we want to classify the activity without knowing a precise or canonical set of classes.

5.2 Methods based around temporal logics

Pinhanez and Bobick [290, 291] describe a method for detecting what we have called behaviours using a representation derived from Allen's interval algebra [9], a method for representing temporal relations between a set of intervals. One authors a description of the behaviour in terms of primitives, which are indivisible and occupy temporal intervals. The description incorporates a set of legal relations between the primitive intervals; a description is consistent if at least one set of intervals, together with an allocation of those intervals to primitives, satisfies it. One determines whether an event is past, now or future by solving a consistent labelling problem, allowing temporal propagation. There is no dynamical model — sets of intervals produced by processes with quite different dynamics could be a consistent labelling; this can be an advantage at the behaviour level, but probably is a source of difficulties at the action/activity level. Papers do not show the method applied to noisy detectors; there are results using simulated detectors on real data. Siskind [350, 349] describes methods to infer activities related to objects — such as throw, pick up, carry, and so on — from an event logic formulated around a set of physical primitives —- such as translation, support relations, contact relations, and the like — from a representation of video. A combination of spatial and temporal criteria are required to infer both relations and events, using a form of logical inference.

The methods are focussed on activity representation, and do not use real video data; there is no mechanism to account for missing or noisy interpretations of video.

5.3 Methods based on Templates

The notion that a motion produces a characteristic spatio-temporal pattern dates at least to Polana and Nelson [296, 297, 299, 300, 305]. Spatio-temporal patterns are used to recognize actions in work by

Bobick and Davis [38] and Davis and Bobick [87]. Ben-Arie et al [30, 29] recognize actions by first finding and tracking body parts using a form of template matcher and voting on lifted tracks; the tracks are lifted to 3D and a spatio-temporal representation of each body segment votes separately for an action. The action with the most votes is chosen. The method is successful, and has the advantage that it is robust to composition — if all but the left arm is walking, the action will still be recognized. However, the vocabulary consists of eight items (jump, kneel, pick, put, run, sit, stand, walk) and the vocabulary cannot be composed. An alternative is to match gestural information directly, incorporating a timewarp to improve the match. Bobick and Wilson [39, 40] use a state-based method that encodes gestures as a string of vector-quantized observation segments; this preserves order, but drops dynamical information. The advantage is relatively fast training.

5.4 Hidden Markov Models and Finite State Representations

Hidden Markov models (HMM's) pervade studies of motion, gesture and activity, and a complete review of their applications here may now be impossible. HMM's are models of sequences, and at their heart is a clock. One has a set of hidden states; at each tick of the clock, a Markov process chooses a new state, dependent on the previous state and nothing else; and an emission process produces an observation from the new state. There are clean solutions for the standard problems of learning (determining an appropriate state transition model and emission model for a given state model) and inference (determine which hidden states occurred given a set of observed states). HMM's have been used for understanding human behaviour but typically with quite small state models.

Very large state models are common in speech recognition, where HMM's have been hugely influential. We do not propose to engage in speech research, and so do not review the area here. It is purely a source of inspiration by analogy. Viewed from a great height, a typical speech system has a series of components: a language model showing how words are built up into sentences; a pronunciation dictionary, giving sequences of context independent phones that correspond to words; a context dependency model, showing how local influences produce context dependent phones (cphones hereafter) from context independent phones; an acoustic observation model showing how acoustic observations result from context dependent phones (this is an extremely compact description of a highly sophisticated area; more extensive descriptions appear in [180, 312]). The resulting object is a vast HMM — in our example, states can be thought of as being tagged with word-cphone-phone-sample — to explain each sample.

This HMM has some important, attractive features. Learning and authoring can be broken into tractable subproblems — the language model might be learned with one kind of dataset, the pronunciation dictionary with another — and as a result, we obtain an HMM on a massive scale, but with little difficulty in authoring it. While the state space is so big that dynamic programming must be sacrificed for a beam search, the state transition model is not impossible to learn, because most state transitions don't occur. Furthermore, the model is forced to share parameters in important ways — a phoneme in one word has the same model as that phoneme in a different word. The currently dominant method for authoring such models involves finite state transducers (section **??**); we propose to adopt this approach.

Finite state models have had considerable success in the speech and language community. We introduce some terminology here, from the reviews by Mohri and others [254, 253, 252]. A *finite state automaton* is a directed graph, whose nodes are known as states. There is at least one final state and one initial state; each edge is labelled with an element of an alphabet. The automaton accepts any string corresponding to a path from an initial state to a final state. In a *finite state transducer*, transitions are labelled with both an element of an input alphabet and an element of an output alphabet; any string

accepted by the transducer results in a string of output symbols, and so the transducer can be seen as representing a relation between families of strings. Transducers (representing relations between strings) can be composed, and there are efficient algorithms for computing the composition of two transducers.

In a *string-to-weight transducer*, the output alphabet consists of weights (typically, in a semiring or better; non-negative reals with addition and min is common, because it corresponds to the case of Viterbi and negative log-probabilities); there are initial and final weights. If a string-to-weight transducer accepts some string, its output for that string is defined as the minimum sum of weights over the paths accepting the string. Particularly attractive are subsequential string-to-weight transducers, where there is only one path accepting any given string. Not all transducers can be transformed to this form; there are algorithms for this process, known as *determinization* when it is possible. Furthermore, there are *minimization* algorithms, that can produce the unique (up to automorphism) smallest transducer that implements the same set of mappings as a given transducer.

Each of the components of a speech architecture (language model; a pronunciation dictionary; context dependency model; acoustic observation model) is a string-to-weight transducer. In principle, one could compose the lot to produce a single, enormous string-to-weight transducer, determinize it, minimize the result, and search that (this is equivalent to recognizing that, in the final analysis, the composition of each component produces an HMM with an enormous state space). In practice, the object involved is far too large. Instead, one uses a *beam search* to produce a reduced string-to-weight transducer (*the word lattice*) that contains a reduced pool of higher probability paths. Determinizing and minimizing this transducer is practical and useful; the result is very much faster searches.

There are two reasons that this material is of interest to us. First, the trick of reducing a speech signal to a (determinized and minimized) word lattice produces a highly compact representation of a large number of different transcriptions (each corresponding to a path through the string-to-weight transducer) that is easy to search and manage. We argue below that we can produce act, action and activity models which will allow reduction of video to an action/activity lattice with the same attractive properties. Second, a finite state automaton (whose states represent actions and activities) is a reasonable representation for a behaviour. If one determinizes and minimizes this, standard algorithms allow one to identify weights associated with instances of such a transducer in a word lattice extremely fast (we have used this idea in a vision context to produce fast searches for words in mediaeval latin handwriting [102]). This means we should be able to engage in fast searches for behaviours.

LeCun et al identify other useful building blocks associated with finite state models [209]. Their *graph transformers* take (weighted directed) graphs as inputs and produce graphs as outputs; an example of a transformer would be composition with a fixed transducer. Particularly useful is the idea of a Viterbi transformer, a process that (using our terminology) takes a string-to-weight transducer and applies a beam search to produce a reduced string-to-weight transducer which is effectively a word lattice. They demonstrate that gradient based learning can usefully be applied to architectures of such objects.

5.5 Activity Recognition Methods based around HMM's

HMM's have been very widely adopted in activity recognition, but the models used have tended to be small (for example, one sees three and five state models in [45, 50]). Yamato *et al.* describe recognizing tennis strokes with HMM's [421]. Wilson and Bobick describe the use of HMM's for recognizing gestures such as pushes [412]. Yang et al use HMM's to recognize handwriting gestures [**?**]. Feng and Perona [111] call actions "movelets", and build a vocabulary by vector quantizing a representation of image shape, as a collection of rectangle, varying over time. These codewords are then strung together

by an HMM, representing activities; there is one HMM per activity. We can then identify a new video by computing the image representation for each frame, obtaining the movelets, and choosing the particular model that generated the keyword sequence by a form of maximum likelihood. The method is not view invariant, depending on an image centered representation.

There has been a great deal of interest in models obtained by modifying the HMM structure. The intention is to improve the expressive power of the model without complicating the processes of learning or inference. Brand et al use coupled HMM's (CHMM's), which involve some number of simultaneous HMM's operating to the same clock, where the choice of a particular model's hidden state is affected by all other model's states [45, 50]. Such an object is clearly itself an HMM, but authors demonstrate a training method that reduces the number of parameters to learn by coupling but with very much enlarged state space; however, instead of estimating the parameters of that object, one projects the parameter estimates to transition parameters for each separate model. This means that one learns parameters for each separate model that tend to couple the two models. They show these models can distinguish between a set of T'ai Chi moves.

Oliver et al [275, 274] represent behaviours using layered hidden Markov models (LHMM's). These models involve a bank of HMM's at the lowest level, each generating some portion of the observation. The observations at higher levels are the maximum likelihood hidden state sequences for the lower levels. One then obtains for each HMM the maximum likelihood hidden state sequence. At the next level, the observations are these states, and this continues recursively. The resulting object is an HMM, but of complex structure; the LHMM form offers authoring advantages. This representation outperforms a straightforward HMM in recognizing such activities as phone conversation from both vision and acoustic data.Similarly, Mori et al build a hierarchical representation out of HMM's to recognize everyday gesture [258].

Wilson and Bobick [413] use a form of HMM where an unknown, global parameter applies to all emission models (which they call a parametric hidden Markov model or PHMM) to model gestures with a parametric form (such as might accompany "it was *this* big"). Data is from stereo or a Polhemus. There are recognition results for classes of gesture such as pointing. Kettnaker and Brand [195](also, Brand and Kettnaker, [49]) fit an HMM while penalizing model entropy; this tends to reduce the number of non-zero parameters, so that one can fit models with quite large state spaces satisfactorily (such models are sometimes known as Entropic HMM's or EHMM's). Galata *et al*. use variable length Markov models (VLMM's: a model that generates a state stochastically based on a variable but bounded length history) to encode behaviour and obtain a reduction in perplexity by doing so [123, 124].

Building variant HMM's is a way to simplify learning the state transition process from data (if the state space is large, the number of parameters is a problem). But there is an alternative — one could author the state transition process in such a way that it has relatively few free parameters, despite a very large state space, and then learn those parameters.

Finite state methods have been used directly. Hongeng *et al*. demonstrate recognition of multiperson activities from video of people at coarse scales (few kinematic details are available); activities include conversing and blocking [160]. Zhao and Nevatia use a finite-state model of walking, running and standing, built from motion capture [429]. Hong *et al*. use finite state machines to model gesture [159]. We are not aware of material that attempts to build large hierarchical finite state machines, patterned after speech recognition programs, and using opportunistic learning, as we propose to do.

5.6 Notes

Wiley and Hahn describe methods to interpolate between motions [411]. Guo and Roberge

5.6.0.2 Gesture

5.6.0.3 Event Detection

5.6.0.4 Sign Language Recognition

The best-known system for sign matching is due to Starner and Pentland [367, 368]. Features are image moments of the hand region; signers either wear coloured gloves, or hands are identified using a skin filter. A Hidden Markov Model (HMM) is used to model individual signs; signs are strung together with a rigid language model (`pronoun verbnoun adjective pronoun`). Authors report a recognition rate of 90% with a vocabulary of 40 signs. Grobel and Assan recognize isolated signs under similar conditions for a 262-word vocabulary using HMM's [197]. This work was extended to recognize continuous German sign language with a vocabulary of 97 signs by Bauer and Hienz [26]. Vogler and Metaxas have built a system that uses estimates of arm position, recovered either from a physical sensor mounted on the body or from a system of three cameras that measures arm position fairly accurately [**?**, **?**, **?**]. For a vocabulary of 53 words, and an independent word language model, they report a word recognition accuracy of the order of 90%. A more recent system attempted to recognize phonemes with HMM's; Vogler and Metaxas were able to recognize signs from a 22 word vocabulary with similar recognition rates for phoneme and word models (without handshapes in [404], with handshapes in [405]).

Kadous transduced isolated Australian sign language signs with a powerglove, reporting a recognition rate of 80% using decision trees [263]. Matsuo *et al* transduced Japanese sign language with stereo cameras, using decision tree methods to recognize a vocabulary of 38 signs [238]. Kim *et al.* transduce Korean sign language using datagloves, reporting 94% accuracy in recognition for 131 Korean signs [198]. Al-Jarrah and Halawani report high recognition accuracy for 30 Arabic manual alphabet signs recognized from monocular views of a signer using a fuzzy inference system [6]. Gao *et al.* describe recognizing isolated signs drawn from a vocabulary of 5177 using datagloves and an HMM model [126, 410]. Their system is not speaker-independent: they describe relatively high accuracy for the original signer, and a significant reduction in performance for other signers. Similarly, Zieren and Kraiss report high, but not speaker independent, accuracy for monocular recognition of German sign language drawn from a vocabulary of 152 signs [431]. Akyol and Canzler describe an information terminal which can recognize 16 signs with a high, user-independent, recognition rate; their system uses HMM's to infer signs from monocular views of users wearing coloured gloves [10]. Bowden *et al.* use independent component analysis to obtain state estimates from a set of discriminative visual features; each sign is encoded as a Markov chain, learned from a single example [43]. They report high accuracy recognition from a lexicon of 49 signs using a very small training set.

Chapter 6

Discussion

```
standard datasets for human detection in arbitrary configurations
datasets in general
```

6.0.1 Which is better, 2D or 3D?

One could represent the body in 2D or in 3D. But which representation is better? There is very little compelling evidence either way. For many applications, it is important to have some form of representation of the body in 3D. For example, this might be used to match to labelled data (section **??** and []); to produce motion capture data (**video motion capture**; section **??** and []); or to control animated figures (**video puppetry**; section **??** and []). As we have seen, it is not particularly difficult to lift from 2D to 3D, so this is not a compelling point.

It may be the case that inference is much more difficult for 3D models than for 2D models. This could occur if the relationship between 2D and 3D was highly ambiguous — many distinct 3D poses are implied by a particular 2D observation. In this case, one could expect a single probability modes in the 2D posterior to result in multiple separated modes in the 3D posterior. There is some evidence that this occurs, but that it is a local phenomenon.

There is no evidence that any application demands tracking using a 3D representation, as opposed to (for example) tracking with a 2D representation and then lifting to 3D. In some cases, it may be sufficient to produce a 2D representation — for example, if one wishes to composite a figure away from or into a scene — but there is little evidence that 2D representations are easier to produce or unequivocally better. This is probably best regarded as a matter of taste.

The evidence here is moot. Deutscher suggests posterior isn't Gaussian, and this should be accepted - kinematic singularities are probably a minor problem, but endstops are important.

Sminchisescu and Triggs have some stuff on observability.

There is a question of 3D vs 2D representation here.

6.0.1.1 Is 3D Configuration Ambiguous?

```
Why not use the discrete ambiguity?
Why not make improved measurements?
Are they all there?
Do dynamics make much difference?
```

```
Mori + Malik
What about image dependencies created by the camera?
```

Localization difficulties interact with these am

Localizing segment endpoints is difficult, and one expects some inaccuracy.

This interacts with the depth ambiguity to produce significantly larger

The depth ambiguity interacts with difficulties in localizing segment endpoints to produce significantly larger uncertainty in some directions in joint angle space; typically, these are directions that tend to move a limb in depth (figure 2.4). This means that the posterior at a particular frame can have quite large covariance in some directions. But it doesn't mean the posterior conditioned on many measurements necessarily has large covariance — a sufficient past history combined with some dynamical limits may mean that, *over a longer timescale than a single frame*, the reconstruction is close to unique.

The situation is complicated by the discrete ambiguities, which appear to guarantee the existence of multiple modes in the posterior under some circumstances. In practice, these multiple modes occur (see figure 2.5). However, this ambiguity does not appear to be persistent over long time scales. Formally, $P(\mathbf{X}_i|\mathbf{Y}_1, ..., \mathbf{Y}_i)$ may be multimodal, but $P(\mathbf{X}_i|\mathbf{Y}_1, ..., \mathbf{Y}_{i+n})$ tends not to be for sufficiently large n. This point is remarked on by Sminchisescu and Triggs ([356], p 372, "In practice, choosing the wrong minimum rapidly leads to mistracking..."), and is important for the methods described below. The modes in the multimodal posterior may have quite large covariance in some directions (because of the depth ambiguity), meaning that poorly placed samples — or failing to search at a sample — may miss the few important modes for the next frame. It appears that a fixed number of modes (2^k, where k is the number of body segments) is probably sufficient, though we're not aware of anyone benefiting from this observation directly.

```
short vs long histories; why;
```

6.0.1.2 What Representation of 3D Configuration should be Adopted?

Predicting components independent of one another has implications for the encoding of body kinematics. For example, this suggests adopting a representation based on joint angles rather than on positions of key points on the body (because some pairs of these points are a fixed distance apart). Joint angles might themselves be quite strongly correlated with one another. The vast majority of the human activity that one observes is walking, where there is a characteristic oscillatory motion of both upper and lower body, 180^o out of phase with one another. To address this question sensibly, one needs a model of the type of error that must be avoided. If the intention is to perform kinematic reconstructions for configurations whose frequencies are well represented by typical motion capture or video data (in outdoor data, lots of walking, some running, and other activities very infrequent), then the correlations between joint angles typical of walking are very important. One should choose a representation of configuration where components are independent. This would most likely look like a single variable giving the phase of the walk, some variables encoding speed, frequency and the like, and then offsets of arm and leg from typical configurations for that phase. Principal components analysis might also yield an appropriate set of variables, although the estimate could be poor because of the dimension.

Alternatively, one might wish to obtain reconstructions for activities whose frequencies are not well represented by typical motion capture or video data — sporting activities and the like. In this case, joint angles could be weakly correlated or independent. There are some small reasons to think that they are not, however. For example, in some throwing or striking motions, there is a clear proximal-distal

sequence by which the joints are activated, leading to a whip-like motion (for some cases, see [7, 16, 311]; the effect does not occur in all sports [121]; it can be used in animation [31]). This means that near-straight elbow implies a particular shoulder position, so that joint angles will be correlated. The topic has not been sufficiently addressed in regression studies of kinematic configuration.

6.0.2 What is the status of dynamical models?

`probably are good, but hard to build right`

6.0.3 What Motion is Human?

Some of the practices we have described — cut and paste, for example — may disrupt some of the detailed structure of the motion signal to which people are often extremely sensitive. Several researchers have used light-dot displays, also referred to as biological motion stimuli, to study perception of human movements [122]. The light-dot displays show only dots or patches of light that move with the main joints of walking figures, but even these minimal cues have been shown to be sufficient for viewers to make detailed assessments of the nature of both the motion and the underlying figure[185]. Work by Cutting and Kozlowski showed that viewers easily recognized friends by their walking gaits on light-dot displays[82]. They also reported that the gender of unfamiliar walkers was readily identifiable, even after the number of lights had been reduced to just two located on the ankles[205]. In a published note, they later explained that the two light-dot decisions were probably attributable to stride length[206]. Continuing this work, Barclay, Cutting, and Kozlowski showed that gender recognition based on walking gait required between 1.6 and 2.7 seconds of display, or about two step cycles[22, 83].

Not much is known about what inclines people toward or away from the judgement that a motion is "good" or "natural". It is known that the choice of rendering has an effect, with more naturalistic renderings making people more inclined to reject motions [157, 156].

There have been several attempts to build methods to score the "goodness" of a motion. Ikemoto and Forsyth use a large pool of labelled frames and a support vector machine to build a classifier that takes a set of frames, centered on the current frame, and labels the current frame with "human" or "not human" []. The classifier has a respectable total error rate (approx ****) but, as one would expect, is not particularly reliable when applied to motions that are very different from the training set. **** *et al.*

6.0.4 Notes

The use of inverse kinematics in animation dates to at least the work of Girard and Maciejewski [135]; see also [134] and [146]. Methods for handling singularities are discussed in [233]. A good summary of early work in animation is [17]. Tolani *et al.* contains a considerable body of helpful background and review material [387].

Zhao and Badler approach inverse kinematics as a nonlinear programming problem — using our notation, find $\arg\min \mid \mathbf{g}(\mathbf{x}(\theta)) \mid$ subject to joint constraints, etc. — and use a variant of a standard optimization method; it is not possible to guarantee a global minimum (neither the objective function nor the constraints are convex) [426]. Incompatible constraints can be handled by a scheme allocating different priorities to constraints [18].

Shin *et al.* obtain a real-time solver for a puppetry application by linking a fast frame-by-frame solver using a mixed analytical-numerical strategy with a Kalman filter smoother [344].

Joints other than the shoulder have been studied in some detail [255, 288].

```
   choosing timescales for transitions; Mographs/sca2004wb.pdf
mglight snap together, particularly how the graph should be built
```

6.0.5 Notes

Non-parametric synthesis methods have been successful in several areas. The idea appears to originate with Shannon [], who used these methods to synthesize text. These methods have revolutionized texture synthesis; there is an enormous literature, starting with Efros and Leung []. There are now a range of very elaborate methods, including methods that can transfer texture "styles" (brushstrokes and the like). The analogy with texture synthesis seems to have motivated non-parametric motion synthesis, and one could reasonably believe that there is more to be found in the analogy. A good set of places to start looking is [].

6.0.6 Notes

Mass displacement comes from **Physicalsmoothing/p11-popovic**

Index

Bibliography

[1] Y. Abe, C. K. Liu, and Z. Popović. Momentum-based parameterization of dynamic character motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 173–182, New York, NY, USA, 2004. ACM Press.

[2] A. Agarwal and B. Triggs. Learning to track 3d human motion from silhouettes. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 2, New York, NY, USA, 2004. ACM Press.

[3] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding: CVIU*, 73(3):428–440, 1999.

[4] G. Agin and T. Binford. Computer description of curved objects. In *IJCAI73*, pages 629–640, 1973.

[5] G. Agin and T. Binford. Computer description of curved objects. *TC*, 25(4):439–449, April 1976.

[6] O. Al-Jarrah and A. Halawani. Recognition of gestures in arabic sign language using neuro-fuzzy systems. *AI*, 133(1-2):117–138, December 2001.

[7] R. M. Alexander. Optimum timing of muscle activation for simple models of throwing. *J. Theor. Biol.*, 150:349–372, 1991.

[8] G. M. Alexei A. Efros, Alexander C. Berg and J. Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision (ICCV'03)*, 2003.

[9] J. F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984.

[10] A. S. annd Canzler U. An information terminal using vision based sign language recognition. In *ITEA Workshop on Virtual Home Environments, VHE Middleware Consortium*, pages 61–68, 2002.

[11] W. Arentz and B. Olstad. Classifying offensive sites based on image content. *CVIU*, 94(1-3):295–310, April 2004.

[12] O. Arikan, D. Forsyth, and J. O'Brien. Motion synthesis from annotations. *SIGGRAPH*, 2003.

[13] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490. ACM Press, 2002.

[14] V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. In *AFGR02*, pages 40–45, 2002.

[15] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR03*, pages II: 432–439, 2003.

[16] A. Atwater. Biomechanics of overarm throwing movements and of throwing injuries. *Exerc. Sport. Sci. Rev.*, 7:43–85, 1979.

[17] N. I. Badler, B. A. Barsky, and D. Zeltzer, editors. *Making them move: mechanics, control, and animation of articulated figures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[18] P. Baerlocher and R. Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6):402–417, 2004.

[19] H. Baker. Building surfaces of evolution: The weaving wall. *IJCV*, 3(1. May 1989):51–72, May 1989.

[20] Y. Bar-Shalom and X.-R. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[21] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 185–194, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[22] C. D. Barclay, J. E. Cutting, and L. T. Kozlowski. Temporal and spatial factors in gait perception that influence gender recognition. *Perception & Psychophysics*, 23(2):145–152, 1978.

[23] C. Barron and I. Kakadiaris. Estimating anthropometry and pose from a single image. In *CVPR00*, pages I: 669–676, 2000.

[24] C. Barron and I. Kakadiaris. Estimating anthropometry and pose from a single uncalibrated image. *CVIU*, 81(3):269–284, March 2001.

[25] H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI77*, pages 659–663, 1977.

[26] B. Bauer and H. Hienz. Relevant features for video-based continuous sign language recognition. In *IEEE Workshop on Automatic Face and Gesture Recognition*, pages 440–445, 2000.

[27] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *ECCV (1)*, pages 299–308, 1994.

[28] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, April 2002.

[29] J. Ben-Arie, P. Pandit, and S. Rajaram. View-based human activity recognition by indexing and sequencing. In *CVPR01*, pages II:78–83, 2001.

[30] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram. Human activity recognition using multidimensional indexing. *PAMI*, 24(8):1091–1104, August 2002.

[31] D. Bhat and J. Kearney. On animating whip-type motions. *The Journal of Visualization and Computer Animation*, 5:229–249, 1996.

[32] T. Binford. Inferring surfaces from images. *AI*, 17(1-3):205–244, August 1981.

[33] M. Black, Y. Yacoob, A. Jepson, and D. Fleet. Learning parameterized models of image motion. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 561–567, 1997.

[34] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[35] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[36] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[37] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. *Royal*, B-352:1257–1265, 1997.

[38] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *PAMI*, 23(3):257–267, March 2001.

[39] A. Bobick and A. Wilson. A state based technique for the summarization and recognition of gesture. In *Int. Conf. on Computer Vision*, pages 382–388, 1995.

[40] A. Bobick and A. Wilson. A state based approach to the representation and recognition of gesture. *IEEE T. Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, December 1997.

[41] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97. Proceedings of the Eurographics Workshop*, 1997.

[42] A. Bosson, G. Cawley, Y. Chan, and R. Harvey. Non-retrieval: Blocking pornographic images. In *CIVR02*, pages 50–59, 2002.

[43] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference on Computer Vision*, pages Vol I: 390–401, 2004.

[44] J. Boyd and J. Little. Phase in model-free perception of gait. In *HUMO00*, pages 3–10, 2000.

[45] M. Brand. Coupled hidden markov models for complex action recognition. Media lab vision and modelling tr-407, MIT, 1997.

[46] M. Brand. An entropic estimator for structure discovery. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 723–729, Cambridge, MA, USA, 1999. MIT Press.

[47] M. Brand. Shadow puppetry. In *ICCV99*, pages 1237–1244, 1999.

[48] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Comput.*, 11(5):1155–1182, 1999.

[49] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE T. Pattern Analysis and Machine Intelligence*, 22(8):844–851, August 2000.

[50] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[51] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR98*, pages 8–15, 1998.

[52] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *IJCV*, 56(3):179–194, February 2004.

[53] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi. Shape-based pedestrian detection. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 215–220, 2000.

[54] R. Brooks. Model-based three-dimensional interpretations of two-dimensional images. *PAMI*, 5(2):140–149, March 1983.

[55] L. Brown. *A Radar History of World War II: Technical and Military Imperatives*. Institute of Physics Press, 2000.

[56] A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA, 1995. ACM Press.

[57] R. Buderi. *The Invention that Changed the World*. Touchstone Press, 1998. reprint.

[58] M. Burl, T. Leung, and P. Perona. Face localisation via shape statistics. In *Int. Workshop on Automatic Face and Gesture Recognition*, 1995.

[59] T. Caelli and S. Nagendran. Fast edge-only matching techniques for robot pattern matching. *CVGIP*, 39(2):131–143, August 1987.

[60] Q. Cai and J. K. Aggarwal. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 356, Washington, DC, USA, 1998. IEEE Computer Society.

[61] L. W. Campbell and A. F. Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, pages 624–630, 1995.

[62] S. Carlsson. Recognizing walking people. In *European Conference on Computer Vision*, pages I: 472–486, 2000.

[63] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003.

[64] A. Cavallaro and T. Ebrahimi. Video object extraction based on adaptive background and statistical change detection. In *SPIE*, pages 465–475, 4310.

[65] T. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages II: 239–245, 1999.

[66] N. Checka, K. Wilson, M. Siracusa, and T. Darrell. Multiple person and speaker activity tracking with a particle filter. In *IEEE Conf. on Acoustics, Speech, and Signal Processing*, volume 5, pages 881–4, 2004.

[67] J. Chen, M. Leung, and Y. Gao. Noisy logo recognition using line segment hausdorff distance. *PR*, 36(4):943–955, April 2003.

[68] F. Cheng, W. Christmas, and J. Kittler. Periodic human motion description for sports video databases. In *ICPR04*, pages III: 870–873, 2004.

[69] K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, volume 2, pages 714 – 720, June 2000.

[70] K.-M. G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *Int. J. Comput. Vision*, 62(3):221–247, 2005.

[71] K.-M. G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time part ii: Applications to human modeling and markerless motion tracking. *Int. J. Comput. Vision*, 63(3):225–245, 2005.

[72] J. chi Wu and Z. Popović. Realistic modeling of bird flight animations. *ACM Trans. Graph.*, 22(3):888–895, 2003.

[73] K. Choo and D. Fleet. People tracking using hybrid monte carlo filtering. In *ICCV01*, pages II: 321–328, 2001.

[74] M. F. Cohen. Interactive spacetime control for animation. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 293–302, New York, NY, USA, 1992. ACM Press.

[75] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–97, 1995.

[76] L. Crawford and S. Sastry. Biological motor control approaches for a planar diver. In *IEEE Conf. on Decision and Control*, pages 3881–3886, 1995.

[77] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, and W. von Seelen. Walking pedestrian recognition. *Intelligent Transportation Systems*, 1(3):155–163, September 2000.

[78] R. Cutler and L. Davis. View-based detection and analysis of periodic motion. In *ICPR98*, pages Vol I: 495–500, 1998.

[79] R. Cutler and L. Davis. Real-time periodic motion detection, analysis, and applications. In *CVPR99*, pages II: 326–332, 1999.

[80] R. Cutler and L. Davis. Robust periodic motion and motion symmetry detection. In *CVPR00*, pages II: 615–622, 2000.

[81] R. Cutler and L. Davis. Robust real-time periodic motion detection, analysis, and applications. *PAMI*, 22(8):781–796, August 2000.

[82] J. E. Cutting and L. T. Kozlowski. Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin of the Psychonomic Society*, 9(5):353–356, 1977.

[83] J. E. Cutting, D. R. Proffitt, and L. T. Kozlowski. A biomechanical invariant for gait perception. *Journal of Experimental Psychology: Human Perception and Performance*, 4(3):357–372, 1978.

[84] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR05*, pages I: 886–893, 2005.

[85] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *IJCV*, 37(2):175–185, June 2000.

[86] A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. In *1999 IEEE International Conference on Robotics & Automation*, pages 1044–1049, 1999.

[87] J. Davis and A. Bobick. The representation and recognition of action using temporal templates. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[88] Q. Delamarre and O. Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 716, Washington, DC, USA, 1999. IEEE Computer Society.

[89] Q. Delamarre and O. Faugeras. 3d articulated models and multiview tracking with physical forces. *Comput. Vis. Image Underst.*, 81(3):328–357, 2001.

[90] A. S. Deo and I. D. Walker. Minimum effort inverse kinematics for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 13(6), 1997.

[91] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR00*, pages II: 126–133, 2000.

[92] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *ICCV99*, pages 1144–1149, 1999.

[93] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61(2):185–205, February 2005.

[94] M. Dimitrijevic, V. Lepetit, and P. Fua. Human body pose recognition using spatio-temporal templates. In *ICCV workshop on Modeling People and Human Interaction*, 2005.

[95] A. Doucet, N. D. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[96] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. In T. P. Pridmore and D. Elliman, editors, *Proceedings of the British Machine Vision Conference 1999, BMVC 1999, Nottingham, 13-16 September 1999*, 1999.

[97] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 20–36, London, UK, 2000. Springer-Verlag.

[98] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *ICCV*, pages 315–320, 2001.

[99] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *PAMI*, 24(7):932–946, July 2002.

[100] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *Int. Conf. Intelligent Robots and Systems*, pages 298–303, 2001.

[101] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195:216–222, 1987.

[102] J. Edwards, Y. W. Teh, D. Forsyth, R. Bock, M. Maire, and G. Vesom. Making mediaeval latin manuscripts searchable using ghmm's. In *Proc. NIPS*, 2004.

[103] A. Engin and S. Tumer. Three-dimensional kinematic modelling of the human shoulder complex - part i: Physical model and determination of joint sinus cones. *ASME Journal of Biomechanical Engineering*, 111:107–112, 1989.

[104] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 251–260, Aug. 2001.

[105] P. Faloutsos, M. van de Panne, and D. Terzopoulos. The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics*, 25(6):933–953, Dec. 2001.

[106] A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics*, 22(3):417–426, July 2003.

[107] A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion. *ACM Trans. Graph.*, 22(3):417–426, 2003.

[108] A. Faul and M. Tipping. Analysis of sparse bayesian learning. In *Advances in Neural Information Processing Systems 14*, pages 383–389. MIT Press, 2002.

[109] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *CVPR00*, pages II: 66–73, 2000.

[110] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, January 2005.

[111] X. Feng and P. Perona. Human action recognition by sequence of movelet codewords. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 717–721, 2002.

[112] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR03*, pages II: 264–271, 2003.

[113] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*, May 2004.

[114] A. Fod, M. J. Matarić, and O. C. Jenkins. Automated derivation of primitives for movement classification. *Auton. Robots*, 12(1):39–54, 2002.

[115] D. Forsyth. Sampling, resampling and colour constancy. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages I: 300–305, 1999.

[116] D. Forsyth and M. Fleck. Body plans. In *CVPR97*, pages 678–683, 1997.

[117] D. Forsyth and M. Fleck. Automatic detection of human nudes. *IJCV*, 32(1):63–77, August 1999.

[118] D. Forsyth, M. Fleck, and C. Bregler. Finding naked people. In *European Conference on Computer Vision*, pages 593–602, 1996.

[119] D. Forsyth, J. Haddon, and S. Ioffe. The joy of sampling. *Int. J. Computer Vision*, 2001.

[120] D. Forsyth and J. Ponce. *Computer Vision: a modern approach*. Prentice-Hall, 2002.

[121] L. Fradet, M. Botcazou, C. Durocher, A. Cretual, F. Multon, J. Prioux, and P. Delamarche. Do handball throws always exhibit a proximal-to-distal segmental sequence? *Journal of Sports Sciences*, 22(5):439–447, 2004.

[122] J. Freyd. Dynamic mental representations. *Psychological Review*, 94(4):427–438, 1987.

[123] A. Galata, N. Johnson, and D. Hogg. Learning behaviour models of human activities. In *BMVC99*, page Modelling Human Behaviour, 1999.

[124] A. Galata, N. Johnson, and D. Hogg. Learning structured behavior models using variable length markov models. In *MPeople99*, pages xx–yy, 1999.

[125] D. Gao, J. Zhou, and L. Xin. A novel algorithm of adaptive background estimation. In *ICIP01*, pages II: 395–398, 2001.

[126] W. Gao, J. Ma, X. Chen, et al. Handtalker: a multimodal dialog system using sign language and 3d virtual human. In *Proc. Third Int. Conf. Multimodal Interface*, pages 564–571, 2000.

[127] D. Gavrila. Pedestrian detection from a moving vehicle. In *ECCV00*, pages II: 37–49, 2000.

[128] D. Gavrila. Sensor-based pedestrian protection. *Intelligent Transportation Systems*, pages 77–81, 2001.

[129] D. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: the protector system. In *Intelligent Vehicle Symposium*, pages 13–18, 2004.

[130] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding: CVIU*, 73(1):82–98, 1999.

[131] J. Gehl. *Life between buildings: Using public space*. Architektens Vorlag, 1996.

[132] W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov chain Monte Carlo in practice*. Chapman and Hall, 1996.

[133] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *25th International Conference on Very Large Data Bases (VLDB 99)*, pages 518–529, 1999.

[134] M. Girard. Interactive design of 3-d computer-animated legged animal motion. In *SI3D '86: Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 131–150, New York, NY, USA, 1987. ACM Press.

[135] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1985. ACM Press.

[136] M. Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 1997.

[137] M. Gleicher. Animation from observation: Motion capture and motion editing. *Computer Graphics*, 1999.

[138] M. Gleicher. Comparing constraint-based motion editing methods. *Graphical Models*, 2001.

[139] M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen. Snap-together motion: assembling run-time animations. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 181–188, New York, NY, USA, 2003. ACM Press.

[140] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. "dynamism of a dog on a leash" or behavior classification by eigen-decomposition of periodic motions. In *ECCV02*, page I: 461 ff., 2002.

[141] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Behavior classification by eigendecomposition of periodic motions. *PR*, 38(7):1033–1043, July 2005.

[142] K. Grauman, G. Shakhnarovich, and T. Darrell. Virtual visual hulls: Example-based 3d shape inference from silhouettes. In *SMVP04*, pages 26–37, 2004.

[143] W. Grimson, L. Lee, R. Romano, and C. Stauffer. Using adaptive tracking to classify and monitor activities in a site. In *CVPR98*, pages 22–29, 1998.

[144] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[145] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 9–20, July 1998.

[146] J. K. Hahn. Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 299–308, New York, NY, USA, 1988. ACM Press.

[147] I. Haritaoglu, D. Harwood, and L. Davis. W4s: A real-time system for detecting and tracking people in 2 1/2-d. In *European Conference on Computer Vision*, page I: 877, 1998.

[148] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. *IEEE T. Pattern Analysis and Machine Intelligence*, 22:809–830, 2000.

[149] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag, 2001.

[150] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *ICCV98*, pages 344–349, 1998.

[151] A. Hilton and J. Starck. Multiple view reconstruction of people. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004), 6-9 September 2004, Thessaloniki, Greece*, pages 357–364, 2004.

[152] A. C. Hindmarsh and L. R. Petzold. Algorithms and software for ordinary differential equations and differential-algebraic equations, part i: Euler methods and error estimation. *Comput. Phys.*, 9(1):34–41, 1995.

[153] A. C. Hindmarsh and L. R. Petzold. Algorithms and software for ordinary differential equations and differential-algebraic equations, part ii: higher-order methods and software packages. *Comput. Phys.*, 9(2):148–155, 1995.

[154] G. Hinton. Relaxation and its role in vision. Technical report, University of Edinburgh, 1978. PhD Thesis.

[155] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, editors. *Understanding Robust and Exploratory Data Analysis*. John Wiley, 1983.

[156] J. K. Hodgins, J. F. O'Brien, and J. Tumblin. Do geometric models affect judgments of human motion? In *Graphics Interface '97*, pages 17–25, May 1997.

[157] J. K. Hodgins, J. F. O'Brien, and J. Tumblin. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):307–316, Oct. 1998.

[158] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 71–78, Aug. 1995.

[159] P. Hong, M. Turk, and T. Huang. Gesture modeling and recognition using finite state machines. In *AFGR00*, pages 410–415, 2000.

[160] S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *CVIU*, 96(2):129–162, November 2004.

[161] B. Horn and B. Schunck. Determining optical flow. *AI*, 17(1-3):185–203, August 1981.

[162] N. Howe. Silhouette lookup for automatic pose tracking. In *Non-Rigid04*, page 15, 2004.

[163] N. R. Howe, M. E. Leventon, and W. T. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 820–26. MIT Press, 2000.

[164] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE transactions on systems, man, and cyberneticspart c: applications and reviews*, 34(3), 2004.

[165] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance under translation. In *CVPR92*, pages 654–656, 1992.

[166] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *PAMI*, 15(9):850–863, September 1993.

[167] D. Huttenlocher, J. Noh, and W. Rucklidge. Tracking non-rigid objects in complex scenes. In *ICCV93*, pages 93–101, 1993.

[168] D. Huttenlocher and W. Rucklidge. Multi-resolution technique for comparing images using the hausdorff distance. In *CVPR93*, pages 705–706, 1993.

[169] L. Ikemoto and D. Forsyth. Enriching a motion collection by transplanting limbs. In *Proc. Symposium on Computer Animation*, 2004.

[170] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM Press.

[171] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625, New York, NY, USA, 1997. ACM Press.

[172] S. Ioffe and D. Forsyth. Learning to find pictures of people. In *NIPS*, 1998.

[173] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. In *ICCV01*, pages I: 690–695, 2001.

[174] S. Ioffe and D. Forsyth. Mixtures of trees for object recognition. In *CVPR*, 2001.

[175] S. Ioffe and D. Forsyth. Probabilistic methods for finding people. *IJCV*, 43(1):45–68, June 2001.

[176] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *European Conference on Computer Vision*, page I: 893, 1998.

[177] Y. Ivanov, A. Bobick, and J. Liu. Fast lighting independent background subtraction. In *Vismod*, 1997.

[178] Y. Ivanov, A. Bobick, and J. Liu. Fast lighting independent background subtraction. *IJCV*, 37(2):199–207, June 2000.

[179] O. Javed and M. Shah. Tracking and object classification for automated surveillance. In *ECCV02*, page IV: 343 ff., 2002.

[180] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA, 1998.

[181] O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 56, New York, NY, USA, 2004. ACM Press.

[182] O. C. Jenkins and M. J. Mataric. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 225–232, New York, NY, USA, 2003. ACM Press.

[183] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[184] C. Jeong, J. Kim, and K. Hong. Appearance-based nude image detection. In *ICPR04*, pages IV: 467–470, 2004.

[185] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

[186] M. Jones and T. Poggio. Model-based matching of line drawings by linear combinations of prototypes. In *ICCV95*, pages 531–536, 1995.

[187] M. Jones and P. Viola. Face recognition using boosted local features. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[188] M. Jones and P. Viola. Fast multi-view face detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[189] R. V. Jones. *Most Secret War*. Wordsworth Military Library, 1998. reprint.

[190] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. Int. Conference on Face and Gesture*, pages 561–567, 1996.

[191] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *ICCV87*, pages 259–268, 1987.

[192] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, January 1988.

[193] Y. Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *CVPR04*, pages II: 506–513, 2004.

[194] R. Kehl, M. Bray, and L. V. Gool. Full body tracking from multiple views using stochastic sampling. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 129–136, Washington, DC, USA, 2005. IEEE Computer Society.

[195] V. Kettnaker and M. Brand. Minimum-entropy models of scene activity. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages I:281–286, 1999.

[196] V. Kettnaker and R. Zabih. Counting people from multiple cameras. In *ICMCS '99: Proceedings of the IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2*, page 267, Washington, DC, USA, 1999. IEEE Computer Society.

[197] K.Grobel and M.Assan. Isolated sign language recognition using hidden markov models. In *Proc. Int. Conf. System Man and Cybernetics*, pages 162–167, 1997.

[198] J. Kim, W. Jang, and Z. Bien. A dynamic gesture recognition system for the korean sign language (ksl). *Systems, Man and Cybernetics-B*, 26(2):354–359, April 1996.

[199] D. King. Generating vertical velocity and angular momentum during skating jumps. In *23rd Annual Meeting of the American Society of Biomechanics*, 1999.

[200] H. Ko and N. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Application*, 16(2):50–59, 1996.

[201] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, pages 71–81, 1982.

[202] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 214–224, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[203] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482. ACM Press, 2002.

[204] L. Kovar, J. Schreiner, and M. Gleicher. Footskate cleanup for motion capture editing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104. ACM Press, 2002.

[205] L. T. Kozlowski and J. E. Cutting. Recognizing the sex of a walker from a dynamic point-light display. *Perception & Psychophysics*, 21(6):575–580, 1977.

[206] L. T. Kozlowski and J. E. Cutting. Recognizing the gender of walkers from point-lights mounted on ankles: Some second thoughts. *Perception & Psychophysics*, 23(5):459, 1978.

[207] D. Kriegman and J. Ponce. On recognizing and positioning curved 3-d objects from image contours. *PAMI*, 12(12):1127–1137, December 1990.

[208] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proceedings of the British Machine Vision Conference*, 2004.

[209] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[210] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. *Proc. SIGGRAPH 2002*, 2002.

[211] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48. ACM Press/Addison-Wesley Publishing Co., 1999.

[212] M. Lee and I. Cohen. Human upper body pose estimation in static images. In *ECCV04*, pages Vol II: 126–138, 2004.

[213] M. Lee and I. Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *CVPR04*, pages II: 334–341, 2004.

[214] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labelled graph matching. In *Int. Conf. on Computer Vision*, 1995.

[215] B. Li and H. Holstein. Recognition of human periodic motion: A frequency domain approach. In *ICPR02*, pages I: 311–314, 2002.

[216] Y. Li, T. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472. ACM Press, 2002.

[217] J. Little and J. Boyd. Describing motion for recognition. In *SCV95*, pages 235–240, 1995.

[218] J. Little and J. Boyd. Recognizing people by their gait: The shape of motion. *Videre*, 1(2):xx–yy, 1998.

[219] J. Little and J. Boyd. Shape of motion and the perception of human gaits. In *EEMCV98*, page xx, 1998.

[220] C. K. Liu, A. Hertzmann, and Z. Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph.*, 24(3):1071–1081, 2005.

[221] C. K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 408–416, New York, NY, USA, 2002. ACM Press.

[222] F. Liu and R. Picard. Detecting and segmenting periodic motion. Media lab vision and modelling tr-400, MIT, 1996.

[223] F. Liu and R. Picard. Finding periodicity in space and time. In *ICCV98*, pages 376–383, 1998.

[224] J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer, 2001.

[225] Z. Liu and M. F. Cohen. Decomposition of linked figure motion: Diving. In *5th Eurographics Workshop on Animation and Simulation*, 1994.

[226] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA, 1994. ACM Press.

[227] M. Liverman. *The Animator's Motion Capture Guide : Organizing, Managing,Editing*. Charles River Media, 2004.

[228] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, November 2004.

[229] G. Loy, M. Eriksson, J. Sullivan, and S. Carlsson. Monocular 3d reconstruction of human motion in long action sequences. In *European Conference on Computer Vision*, pages Vol IV: 442–455, 2004.

[230] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *ICCV99*, pages 572–578, 1999.

[231] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. *IJCV*, 39(1):57–71, August 2000.

[232] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *European Conference on Computer Vision*, pages II: 3–19, 2000.

[233] A. A. Maciejewski. Motion simulation: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Comput. Graph. Appl.*, 10(3):63–71, 1990.

[234] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[235] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *RoyalP*, B-200:269–294, 1978.

[236] M. J. Matarić, V. B. Zordan, and Z. Mason. Movement control methods for complex, dynamically simulated agents: Adonis dances the macarena. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 317–324, New York, NY, USA, 1998. ACM Press.

[237] M. J. Matarić, V. B. Zordan, and M. M. Williamson. Making complex articulated agents dance. *Autonomous Agents and Multi-Agent Systems*, 2(1):23–43, 1999.

[238] H. Matsuo, S. Igi, S. Lu, Y. Nagashima, Y. Takata, and T. Teshima. The recognition algorithm with non-contact for japanese sign language using morphological analysis. In *Proc. Int. Gesture Workshop*, pages 273–284, 1997.

[239] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 29–38, New York, NY, USA, 1990. ACM Press.

[240] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *CVIU*, 80(1):42–56, October 2000.

[241] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan-Kaufmann, 1999.

[242] A. Micilotta, E. Ong, and R. Bowden. Detection and tracking of humans by probabilistic body part assembly. In *In Proc. BMVC05*, volume 1, pages 429–438, 2005.

[243] K. Mikolajczyk. Face detector. Technical report, INRIA Rhone-Alpes. Ph.D report.

[244] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE T. Pattern Analysis and Machine Intelligence*, 2004. accepted.

[245] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV04*, pages Vol I: 69–82, 2004.

[246] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV04*, pages Vol I: 69–82, 2004.

[247] A. Mittal and L. S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *Int. J. Comput. Vision*, 51(3):189–203, 2003.

[248] T. Moeslund. Summaries of 107 computer vision-based human motion capture papers. Technical Report LLA 99-01, University of Aalborg, 1999.

[249] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *ICCV95*, pages 786–793, 1995.

[250] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *PAMI*, 23(4):349–361, April 2001.

[251] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM TOG*, 22(3):562–568, 2003.

[252] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2), 1997.

[253] M. Mohri, F. C. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[254] M. Mohri and M. Riley. Weighted finite-state transducers in speech recognition (tutorial). In *Proceedings of the International Conference on Spoken Language Processing 2002 (ICSLP '02)*, 2002.

[255] G. Monheit and N. I. Badler. A kinematic model of the human spine and torso. *IEEE Comput. Graph. Appl.*, 11(2):29–38, 1991.

[256] G. Mori, , and J. Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision LNCS 2352*, volume 3, pages 666–680, 2002.

[257] G. Mori and J. Malik. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. to appear.

[258] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato. Hierarchical recognition of daily human actions based on continuous hidden markov models. In *Automatic Face and Gesture Recognition*, pages 779–784, 2004.

[259] J. Mundy and C.-F. Chang. Fusion of intensity, texture, and color in video tracking based on mutual information. In *Applied Imagery Pattern Recognition Workshop*, pages 10– 15, 2004.

[260] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[261] E. Muybridge. *Animals in Motion*. Dover, 1957.

[262] E. Muybridge. *The Human Figure in Motion*. Dover, 1989.

[263] M.W.Kadous. Machine recognition of auslan signs using powergloves: towards large lexicon integration of sign language. In *Proc. Workshop on the Integration of Gesture in Language and Speech*, pages 165–174, 1996.

[264] R. Neal. Probabilistic inference using markov chain monte carlo methods. Computer science tech report crg-tr-93-1, University of Toronto, 1993.

[265] R. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1996.

[266] R. Neal. Annealed importance sampling. Technical report, Technical Report No. 9805 (revised), Dept. of Statistics, University of Toronto, 1998.

[267] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[268] R. Nevatia. *Computer Analysis of Scenes of 3-D Curved Objects*. Book, 1976.

[269] J. T. Ngo and J. Marks. Physically realistic motion synthesis in animation. *Evol. Comput.*, 1(3):235–268, 1993.

[270] J. T. Ngo and J. Marks. Spacetime constraints revisited. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 343–350, New York, NY, USA, 1993. ACM Press.

[271] S. Niyogi and E. Adelson. Analyzing gait with spatiotemporal surfaces. In *Proc. IEEE Workshop on Nonrigid and Articulated Motion*, pages 64– 69, 1994.

[272] S. Niyogi and E. Adelson. Analyzing and recognizing walking figures in xyt. Media lab vision and modelling tr-223, MIT, 1995.

[273] S. of Anthropology Research Project, editor. *Anthropometric Source Book*. Webb Associates, 1978. NASA reference publication 1024, 3 Vols.

[274] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *CVIU*, 96(2):163–180, November 2004.

[275] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 3–8, 2002.

[276] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *CVPR97*, pages 193–199, 1997.

[277] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. A trainable system for people detection. In *DARPA97*, pages 207–214, 1997.

[278] J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *PAMI*, 2(6):522–536, November 1980.

[279] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 130–6, 1997.

[280] C. Pai, H. Tyan, Y. Liang, H. Liao, and S. Chen. Pedestrian detection and tracking at crossroads. In *ICIP03*, pages II: 101–104, 2003.

[281] C. Pai, H. Tyan, Y. Liang, H. Liao, and S. Chen. Pedestrian detection and tracking at crossroads. *PR*, 37(5):1025–1034, May 2004.

[282] M. Pandy, F. Anderson, and D. Hull. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *J. of Biomech. Eng.*, pages 450–460, 1992.

[283] C. Papageorgiou. A trainable system for object detection in images and video sequences constantine. In *Ph. D.*, 2000.

[284] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable object detection system. In *DARPA98*, pages 1019–1024, 1998.

[285] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *ICCV98*, pages 555–562, 1998.

[286] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *ICIP99*, pages IV:35–39, 1999.

[287] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 38(1):15–33, June 2000.

[288] V. Parenti-Castelli, A. Leardini, R. D. Gregorio, and J. J. O'Connor. On the modeling of passive motion of the human knee joint by means of equivalent planar and spatial parallel mechanisms. *Auton. Robots*, 16(2):219–232, 2004.

[289] C. B. Phillips, J. Zhao, and N. I. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 245–250, New York, NY, USA, 1990. ACM Press.

[290] C. Pinhanez and A. Bobick. Pnf propagation and the detection of actions described by temporal intervals. In *DARPA97*, pages 227–234, 1997.

[291] C. Pinhanez and A. Bobick. Human action detection using pnf propagation of temporal constraints. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 898–904, 1998.

[292] R. Plänkers and P. Fua. Tracking and modeling people in video sequences. *Comput. Vis. Image Underst.*, 81(3):285–302, 2001.

[293] T. Poggio and K.-K. Sung. Finding human faces with a gaussian mixture distribution-based face model. In *Asian Conf. on Computer Vision*, pages 435–440, 1995.

[294] R. Polana and R. Nelson. Detecting activities. In *DARPA93*, pages 569–574, 1993.

[295] R. Polana and R. Nelson. Detecting activities. In *CVPR93*, pages 2–7, 1993.

[296] R. Polana and R. Nelson. Detecting activities. In *DARPA93*, pages 569–574, 1993.

[297] R. Polana and R. Nelson. Detecting activities. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2–7, 1993.

[298] R. Polana and R. Nelson. Detecting activities. *JVCIR*, 5:172–180, 1994.

[299] R. Polana and R. Nelson. Detecting activities. In *ICPR94*, pages A:815–818, 1994.

[300] R. Polana and R. Nelson. Detecting activities. *JVCIR*, 5:172–180, 1994.

[301] R. Polana and R. Nelson. Low level recognition of human motion. In *Non-Rigid94*, pages XX–YY, 1994.

[302] R. Polana and R. Nelson. Recognition of nonrigid motion. In *ARPA94*, pages II:1219–1224, 1994.

[303] R. Polana and R. Nelson. Recognizing activities. In *ICPR94*, pages A:815–818, 1994.

[304] R. Polana and R. Nelson. Detection and recognition of periodic, nonrigid motion. *IJCV*, 23(3):261–282, June 1997.

[305] R. Polana and R. Nelson. Detection and recognition of periodic, nonrigid motion. *Int. J. Computer Vision*, 23(3):261–282, 1997.

[306] N. S. Pollard and F. Behmaram-Mosavat. Force-based motion editing for locomotion tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[307] N. S. Pollard and F. Behmaram-Mosavat. Force-based motion editing for locomotion tasks. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[308] Z. Popović and A. Witkin. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[309] Z. Popović and A. P. Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 11–20, Aug. 1999.

[310] K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. *SIGGRAPH 02*, 2002.

[311] C. Putnam. A segment interaction analysis of proximal-to-distal sequential segment motion patterns. *Med. Sci. Sports. Exerc.*, 23:130–144, 1991.

[312] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[313] D. Ramanan and D. Forsyth. Automatic annotation of everyday movements. In *Proc. NIPS*, 2003.

[314] D. Ramanan and D. Forsyth. Finding and tracking people from the bottom up. In *CVPR03*, pages II: 467–474, 2003.

[315] L. Ren, G. Shakhnarovich, J. K. Hodgins, H. Pfister, and P. Viola. Learning silhouette features for control of human motion. *ACM Trans. Graph.*, 24(4):1303–1331, 2005.

[316] J. Rittscher and A. Blake. Classification of human body motion. In *ICCV99*, pages 634–639, 1999.

[317] T. Roberts, S. McKenna, and I. Ricketts. Human pose estimation using learnt probabilistic region similarities and partial configurations. In *ECCV04*, pages Vol IV: 291–303, 2004.

[318] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision*, page IV: 700 ff., 2002.

[319] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3d hand pose reconstruction using specialized mappings. In *ICCV01*, pages I: 378–385, 2001.

[320] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[321] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 147–154, New York, NY, USA, 1996. ACM Press.

[322] P. Rousseeuw. *Robust Regression and Outlier Detection*. Wiley, 1987.

[323] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing 8*, pages 875–881, 1996.

[324] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 203–8, 1996.

[325] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE T. Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[326] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[327] W. Rucklidge. Efficiently locating objects using the hausdorff distance. *IJCV*, 24(3):251–270, October 1997.

[328] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.

[329] S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[330] C. Schmid and R. Mohr. Combining grey value invariants with local constraints for object recognition. In *CVPR96*, pages 872–877, 1996.

[331] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–535, May 1997.

[332] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR00*, pages I: 746–751, 2000.

[333] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR00*, pages I: 746–751, 2000.

[334] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[335] S. Seitz and C. Dyer. Affine invariant detection of periodic motion. In *CVPR94*, pages 970–975, 1994.

[336] S. Seitz and C. Dyer. View invariant analysis of cyclic motion. *IJCV*, 25(3):231–251, December 1997.

[337] A. Senior. Tracking people with probabilistic appearance models. In *PETS02*, pages 48–55, 2002.

[338] A. Shahrokni, T. Drummond, and P. Fua. Fast Texture-Based Tracking and Delineation Using Texture Entropy. In *International Conference on Computer Vision*, 2005.

[339] A. Shahrokni, T. Drummond, V. Lepetit, and P. Fua. Markov-based Silhouette Extraction for Three–Dimensional Body Tracking in Presence of Cluttered Background. In *British Machine Vision Conference*, Kingston, UK, 2004.

[340] A. Shahrokni, F. Fleuret, and P. Fua. Classifier-based Contour Tracking for Rigid and Deformable Objects. In *British Machine Vision Conference*, Oxford, UK, 2005.

[341] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV03*, pages 750–757, 2003.

[342] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[343] H. J. Shin, L. Kovar, and M. Gleicher. Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 194, Washington, DC, USA, 2003. IEEE Computer Society.

[344] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001.

[345] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *European Conference on Computer Vision*, 2000.

[346] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard. Tracking loose-limbed people. In *CVPR04*, pages I: 421–428, 2004.

[347] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and global skeleton fitting techniques for optical motion capture. In *Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, Nov. 1998. Proceedings of CAPTECH '98.

[348] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 1994. ACM Press.

[349] J. M. Siskind. Grounding language in perception. *Artificial Intelligence Review*, 8:371–391, 1995.

[350] J. M. Siskind. Reconstructing force-dynamic models from video sequences. *Artificial Intelligence*, 151:91–154, 2003.

[351] C. Sminchisescu. Consistency and coupling in human model likelihoods. In *Proceedings International Conference on Automatic Face and Gesture Recognition*, pages 22–27, 2002.

[352] C. Sminchisescu and A. Telea. Human pose estimation from silhouettes: A consistent approach using distance level sets. In *WSCG02*, page 413, 2002.

[353] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. In *CVPR01*, pages I:447–454, 2001.

[354] C. Sminchisescu and B. Triggs. Building roadmaps of local minima of visual models. In *ECCV02*, page I: 566 ff., 2002.

[355] C. Sminchisescu and B. Triggs. Hyperdynamics importance sampling. In *ECCV02*, page I: 769 ff., 2002.

[356] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6):371–391, 2003.

[357] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In *CVPR03*, pages I: 69–76, 2003.

[358] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In *CVPR03*, pages I: 69–76, 2003.

[359] C. Sminchisescu and B. Triggs. Building roadmaps of minima and transitions in visual models. *IJCV*, 61(1):81–101, January 2005.

[360] Y. Song, X. Feng, and P. Perona. Towards detection of human motion. In *CVPR00*, pages I: 810–817, 2000.

[361] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *PAMI*, 25(7):814–827, July 2003.

[362] N. Sprague and J. Luo. Clothed people detection in still images. In *ICPR02*, pages III: 585–589, 2002.

[363] S. Srisuk and W. Kurutach. New robust hausdorff distance-based face detection. In *ICIP01*, pages I: 1022–1025, 2001.

[364] J. Starck and A. Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. In *ICCV*, 2005.

[365] J. Starck and A. Hilton. Virtual view synthesis of people from multiple view video sequences. *Graphical Models*, 67(6):600–620, 2005.

[366] J. Starck, A. Hilton, and J. Illingworth. Human shape estimation in a multi-camera studio. In *BMVC*, 2001.

[367] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. Technical report, MIT, 1996.

[368] T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE T. Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[369] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages II: 246–252, 1999.

[370] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR99*, pages II: 246–252, 1999.

[371] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE T. Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[372] A. Sulejmanpašić and J. Popović. Adaptation of performed ballistic motion. *ACM Trans. Graph.*, 24(1):165–179, 2005.

[373] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *European Conference on Computer Vision*, page I: 629 ff., 2002.

[374] K.-K. Sung and T. Poggio. Example based learning for view based face detection. Ai memo 1521, MIT, 1994.

[375] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE T. Pattern Analysis and Machine Intelligence*, 20:39–51, 1998.

[376] S. Tak and H. Ko. Example guided inverse kinematics. In *International Conference on Computer Graphics and Imaging*, pages 19–23, 2000.

[377] S. Tak and H.-S. Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[378] B. Takacs. Comparing face images using the modified hausdorff distance. *PR*, 31(12):1873–1881, December 1998.

[379] C. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 677–84, 2000.

[380] C. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *CVIU*, 80(3):349–363, December 2000.

[381] A. Thangali and S. Sclaroff. Periodic motion detection and estimation via space-time sampling. In *Motion05*, pages II: 176–182, 2005.

[382] C. Theobalt, I. Albrecht, J. Haber, M. Magnor, and H.-P. Seidel. Pitching a baseball: tracking high-speed motion with multi-exposure images. *ACM Trans. Graph.*, 23(3):540–547, 2004.

[383] C. Theobalt, J. Carranza, M. A. Magnor, and H.-P. Seidel. Enhancing silhouette-based human motion capture with 3d motion fields. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 185, Washington, DC, USA, 2003. IEEE Computer Society.

[384] M. E. Tipping. The relevance vector machine. In *In Advances in Neural Information Processing Systems 12*, pages 332–388. MIT Press, 2000.

[385] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, 2001.

[386] D. Tolani and N. I. Badler. Real-time inverse kinematics of the human arm. *Presence*, 5(4):393–401, 1996.

[387] D. Tolani, A. Goswami, and N. I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models*, 62:353–388, 2000.

[388] N. Torkos and M. V. de Panne. Footprint-based quadruped motion synthesis. In *Graphics Interface 98*, pages 151–160, 1998.

[389] K. Toyama and A. Blake. Probabilistic tracking in a metric space. In *ICCV01*, pages II: 50–57, 2001.

[390] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *IJCV*, 48(1):9–19, June 2002.

[391] Z. Tu and S. Zhu. Image segmentation by data-driven markov chain monte carlo. In *ICCV01*, pages II: 131–138, 2001.

[392] Z. Tu and S. Zhu. Image segmentation by data-driven markov chain monte carlo. *PAMI*, 24(5):657–673, May 2002.

[393] S. Tumer and A. Engin. Three-dimensional kinematic modelling of the human shoulder complex - part ii: Mathematical modelling and solution via optimization. *ASME Journal of Biomechanical Engineering*, 111:113–121, 1989.

[394] S. Ullman and R. Basri. Recognition by linear combinations of models. *PAMI*, 13(10):992–1006, October 1991.

[395] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1996.

[396] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[397] D. D. Vecchio, R. Murray, and P. Perona. Decomposition of human motion into dynamics-based primitives with application to drawing tasks. *Automatica*, 39(12):2085–2098, 2003.

[398] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[399] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR01*, pages I:511–518, 2001.

[400] P. Viola and M. Jones. Robust real-time face detection. In *ICCV01*, page II: 747, 2001.

[401] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, May 2004.

[402] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV03*, pages 734–741, 2003.

[403] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2):153–161, July 2005.

[404] C. Vogler and D. Metaxas. Toward scalability in asl recognition: breaking down signs into phonemes. In *Gesture workshop 99*, 1999.

[405] C. Vogler and D. Metaxas. Handshapes and movements: Multiple-channel american sign language recognition. In *GW03*, pages 247–258, 2003.

[406] J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 232–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[407] J. Wang and B. Bodenheimer. Computing the duration of motion transitions: an empirical approach. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 335–344, New York, NY, USA, 2004. ACM Press.

[408] M. Weber, W. Einhauser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 20–7, 2000.

[409] Y. Weiss. Belief propagation and revision in networks with loops. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1997.

[410] W.Gao, J. Ma, J. Wu, and C. Wang. Sign language recognition based on hmm/ann/dp. *Int. J. Pattern Recognition and Artificial Intelligence*, 14(5):587–602, 2000.

[411] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl.*, 17(6):39–45, 1997.

[412] A. Wilson and A. Bobick. Learning visual behavior for gesture analysis. In *IEEE Symposium on Computer Vision*, pages 229–234, 1995.

[413] A. Wilson and A. Bobick. Parametric hidden markov models for gesture recognition. *IEEE T. Pattern Analysis and Machine Intelligence*, 21(9):884–900, September 1999.

[414] A. G. with Staff of the Analytical Sciences Corporation. *Applied Optimal Estimation*. MIT Press, 1974.

[415] A. Witkin and M. Kass. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM Press.

[416] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, July 1997.

[417] Y. Wu, T. Yu, and G. Hua. A statistical field model for pedestrian detection. In *CVPR05*, pages I: 1023–1030, 2005.

[418] Y. Yacoob and L. Davis. Learned models for estimation of rigid and articulated human motion from stationary or moving camera. *IJCV*, 36(1):5–30, January 2000.

[419] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki. Incremental tracking of human actions from multiple views. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 2, Washington, DC, USA, 1998. IEEE Computer Society.

[420] K. Yamane and Y. Nakamura. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):352–360, 2003.

[421] J. Yamato, J. Ohya, and K. Ishii. Recognising human action in time sequential images using hidden markov model. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[422] W. Yan and D. Forsyth. Learning the behaviour of users in a public space through video tracking. In *CVPR*, 2004. In review.

[423] J. Yang, Z. Fu, T. Tan, and W. Hu. A novel approach to detecting adult images. In *ICPR04*, pages IV: 479–482, 2004.

[424] M. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *PAMI*, 24(1):34–58, January 2002.

[425] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[426] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.*, 13(4):313–336, 1994.

[427] L. Zhao and N. Badler. Gesticulation behaviors for virtual humans. In *PG '98: Proceedings of the 6th Pacific Conference on Computer Graphics and Applications*, page 161, Washington, DC, USA, 1998. IEEE Computer Society.

[428] L. Zhao and C. Thorpe. Stereo- and neural network-based pedestrian detection. *Intelligent Transportation Systems*, 1(3):148–154, September 2000.

[429] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *PAMI*, 26(9):1208–1221, September 2004.

[430] S. Zhu, R. Zhang, and Z. Tu. Integrating bottom-up/top-down for object recognition by data driven markov chain monte carlo. In *CVPR00*, pages I: 738–745, 2000.

[431] J. Zieren and K.-F. Kraiss. Non-intrusive sign language recognition for human computer interaction. In *Proc. IFAC/IFIP/IFORS/IEA symposium on analysis, design and evaluation of human machine systems*, 2004.

[432] V. B. Zordan and J. K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation '99*, Sept. 1999.

[433] V. B. Zordan and J. K. Hodgins. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 89–96, July 2002.

[434] M. Zyda, J. Hiles, A. Mayberry, C. Wardynski, M. Capps, B. Osborn, R. Shilling, M. Robaszewski, and M. Davis. Entertainment r&d for defense. *IEEE Computer Graphics and Applications*, pages 28–36, 2003.

[435] M. Zyda, A. Mayberry, C. Wardynski, R. Shilling, and M. Davis. The moves institute's america's army operations game. In *Proceedings of the ACM SIGGRAPH 2003 Symposium on Interactive 3D Graphics*, pages 217–218, 2003.