and we can apply the recursive definition of the cost-to-go function to get

$$\underset{X_1,\ldots,X_n}{\text{argmax}}\ f_{\text{chain}}(X_1,\ldots,X_n) = \underset{X_1}{\text{argmax}}\ \left(f_{\text{chain}}(X_1) + f^1_{\text{cost-to-go}}(X_1)\right),$$

which yields an extremely powerful maximization strategy. We start at X_n, and construct $f^{(n-1)}_{\text{cost-to-go}}(X_{n-1})$. We can represent this function as a table, giving the value of the cost-to-go function for each possible value of X_{n-1}. We build a second table giving the optimum X_n for each possible value of X_{n-1}. From this, we can build $f^{(n-2)}_{\text{cost-to-go}}(X_{n-2})$, again as a table, and also the best X_{n-1} as a function of X_{n-2}, again as a table, and so on. Now we arrive at X_1. We obtain the solution for X_1 by choosing the X_1 that yields the best value of $\left(f_{\text{chain}}(X_1) + f^2_{\text{cost-to-go}}(X_2)\right)$. But from this solution, we can obtain the solution for X_2 by looking in the table that gives the best X_2 as a function of X_1; and so on. It should be clear that this process yields a solution in polynomial time; in the exercises, you will show that, if each X_i can take one of k values, then the time is $O(nK^2)$.

This strategy will work for a model with the structure of a forest. The proof is an easy induction. If the forest has no edges (i.e., consists entirely of nodes), then it is obvious that a simple strategy applies (choose the best value for each X_i independently). This is clearly polynomial. Now assume that the algorithm yields a result in polynomial time for a forest with e edges, and show that it works for a forest with $e+1$ edges. There are two cases. The new edge could link two existing trees, in which case we could re-order the trees so the nodes that are linked are roots, construct a cost-to-go function for each root, and then choose the best pair of states for these roots from the cost-to-go functions. Otherwise, one tree had a new edge added, joining the tree to an isolated node. In this case, we reorder the tree so that this new node is the root and build a cost-to-go function from the leaves to the root. The fact that the algorithm works is a combinatorial insight, but many kinds of model have a tree structure. Models of this form are particularly important in cases of tracking and of parsing.

20.2 PARSING PEOPLE IN IMAGES

A *human parser* must produce some report of the configuration of the body in an image window. A human parse offers cues to what the person is doing, by reporting where the arms, legs, and so on are. Applications could include building a user interface that can respond to someone's gestures or building a medical support system that can tell, by watching video, whether a physically frail person is safe at home or has sustained an injury and needs care. Tracking people is a particularly useful technology (we'll discuss its applications below), and currently the most reliable technologies for human tracking involve a combination of detection and parsing.

20.2.1 Parsing with Pictorial Structure Models

Parsing can be attacked by maximizing a tree-structured model. For example, we could discretize the set of possible segments in an image by quantizing segment orientation to a fixed set of values, and quantizing the top-left corner of the segment

to the pixel grid. We set up one variable per body segment, where the value of the variable identifies which image segment corresponds to that body segment (you can think of these variables as segment pointers). This set of variables can be scored by evaluating (a) the extent to which the body segment looks like the corresponding image and (b) the extent to which segments are consistent with each other. The set of pointers that maximizes this objective function is the parse. We now use a tree-structured model to ensure the maximization component is tractable.

A *pictorial structure model* is a tree structured model, where unary terms compare parts to image observations, and binary terms evaluate relative configuration. Such models are particularly well adapted to parsing people. Assume we know the appearance of each of a set of limb segments that model a person (Figure 20.4). This means that we can build a set of unary functions that compare the image segment that X_i points to with the corresponding model segment. Because we are maximizing, larger values mean a more compatible appearance. We also obtain a set of pairwise relations for a tree-structured subset of this model. It seems natural to use the tree indicated in Figure 20.4. These terms evaluate the relative location of the image segment endpoints, and perhaps the angles between the image segments (there are numerous useful variants, as we shall see). For example, there might be a term checking that the outer end of the thigh is close to the upper end of the shin, and that the angle between the two is acceptable. Again, larger values mean that the two image segments pointed to by the variables are compatible with the relevant labels.

Models of this form can be used to find people in images in a fairly straightforward manner, and are the core technology of parsing. Felzenszwalb and Huttenlocher (2000) assume that segments have known color patterns—typically, a mixture of skin color and blue—and then compare the actual image color with these patterns for the unary terms; the binary terms ensure endpoints are close and angles are appropriate. This leads to a fairly satisfactory matcher (Figure 20.4), with the proviso that the person's clothing should be known in advance. It should look natural to extend the appearance model to score similarities in texture as well as in color, but this has not proven successful to date, most likely because folds in clothing generate strong texture noise effects.

A persistent nuisance with tree-structured models as we have described them is that the best parse typically will place the left leg (resp. arm) on top of the right leg (resp. arm). This is because configuration cues are usually not strong enough to force the legs (resp. arms) apart in the image, and one of the two image legs (resp. arms) will look more like the model than the other does. It is difficult to change the model to avoid this problem; inserting a term that forces the arms apart will create inference difficulties. Instead, there is a simple trick that helps. We regard the energy as the log of a probability distribution, and draw a large pool of samples from this probability distribution. Each sample is a parse, and samples with high energy will appear more commonly. We then search this pool of samples for a parse where the legs and arms do not overlap, a relatively easy test.

The process of drawing a sample is straightforward. Our tree-structured model yields a probability model $P(X_1, \ldots, X_n)$. We can use the reasoning of Section 20.1.3 to compute marginals (look at the α and β terms). Now compute the marginal $P(X_1)$, and draw a sample from that distribution to get, say, $X_1 = r$.

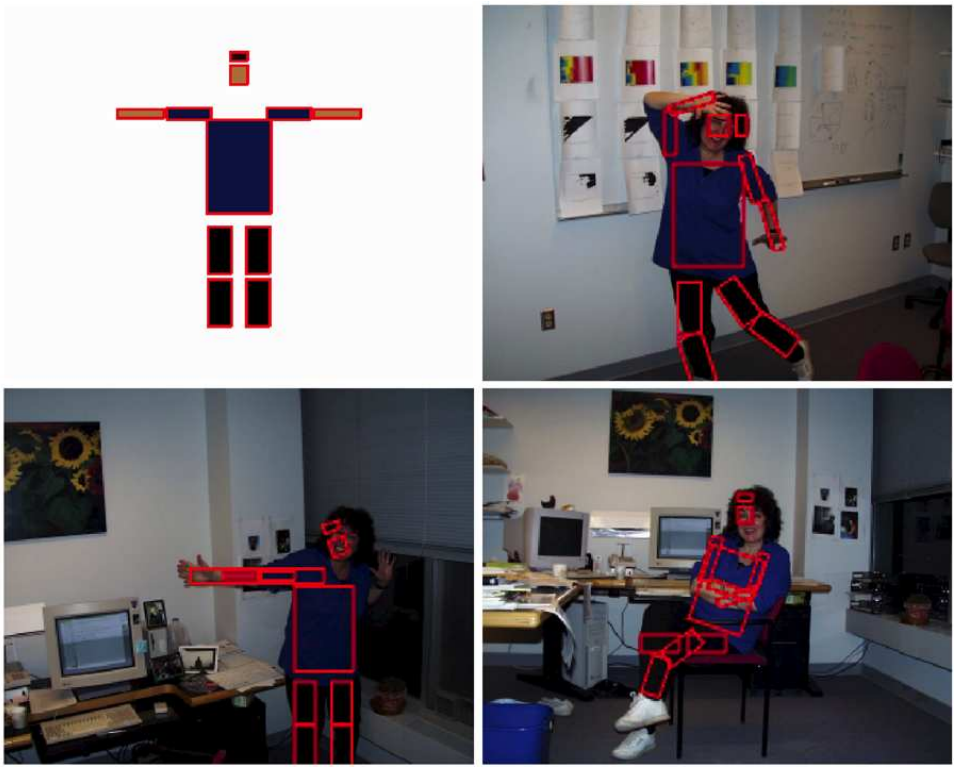


FIGURE 20.4: On the **left**, a tree-structured model of a person. Each segment is colored with the image color expected within this segment. The model attempts to find a configuration of these 11 body segments (nine limb segments, face, and hair) that (a) matches these colors and (b) is configured like a person. This can be done with dynamic programming, as described in the text. The other three frames show matches obtained using the method. *This figure was originally published as Figure 4 of "Efficient Matching of Pictorial Structures," by P. Felzenszwalb and D.P. Huttenlocher, Proc. IEEE CVPR 2000, © 2000, IEEE.*

We then draw a sample from $P(X_2|X_1 = r) = P(X_1 = r, X_2)/\sum_{X_2} P(X_1 = r, X_2)$, and so on.

20.2.2 Estimating the Appearance of Clothing

One crucial difficulty with the pictorial structure model, as we have described it, is that we need to know the appearance of the body segments. We could avoid this difficulty by changing the segment appearance models. Body segments are extended, and we expect some contrast at either side, so the segment appearance model could just require that there be strong edges on either side of the segment. It turns out that this model works poorly, because there tend to be numerous such segments in the image.

However, as Ramanan (2006) points out, this model can be used to start a

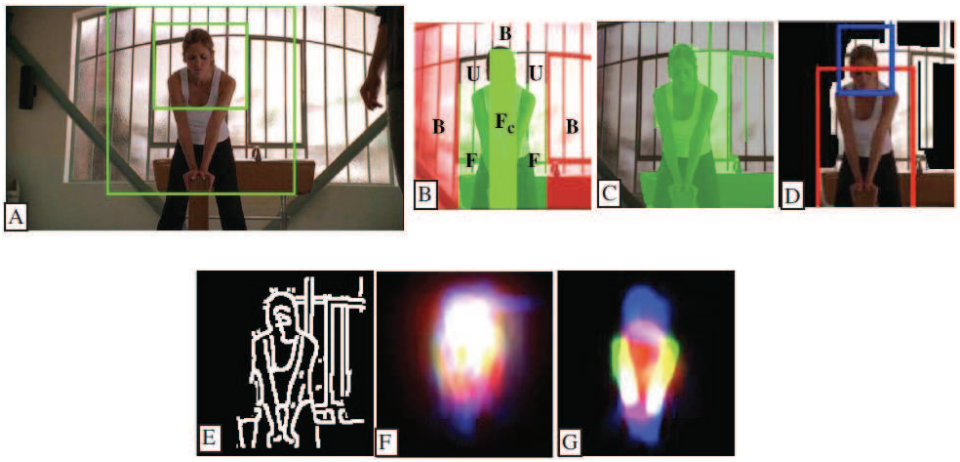


FIGURE 20.5: The human parser of Ramanan (2006) is a search of all spatial layouts in the image to find one that is consistent with the constraints we know on appearance. Ferrari *et al.* (2008) show that reducing the search space improves the results. First, one finds upper bodies, and builds a box around those detections using constraints on the body size (**A**). Outside this box is background, and some pixels inside this box are, too. In **B**, body constraints mean that pixels labeled F_c and F are very likely foreground, U are unknown, and B are very likely background. One then builds color models for foreground and background using this information, then uses an interactive segmenter to segment, requiring that F_c pixels be foreground, to get **C**. The result is a much reduced search domain for the human parser, which starts using an edge map **D**, to get an initial parse **E**, and, after iterating, produces **F**. *This figure was originally published as Figure 2 of "Progressive search space reduction for human pose estimation," by V. Ferrari, M. Marín-Jiménez, and A. Zisserman, Proc. IEEE CVPR 2008,* © *IEEE 2003.*

process that first estimates appearance, then parses, then re-estimates appearance, and so on. We start by assuming that segments have edges on their boundaries. We use this model to generate multiple estimates of configuration, using the procedure for sampling in Section 20.2.1. In turn, we can use these estimates to build a map of the posterior probability a pixel is, for example, a head pixel, by rendering the head segment for each of the sampled estimates of configuration and then summing the images. In turn, this means we have a set of weighted head/non-head pixels, which can be used to build a discriminative appearance model for the head. From this and other such discriminative appearance models, we can re-estimate the configuration (and then re-estimate appearance, and so on). The technical details are beyond the scope of this chapter, but the procedure can produce simultaneous estimates of parses and appearance models for complex images.

If the person covers a relatively small percentage of the image pixels, then this strategy will work poorly because there is a strong chance the initial estimate of configuration might be completely wrong, and then re-estimation is unlikely to help. Ferrari *et al.* (2008) show improved parses obtained by pruning the search domain using appearance information. They first detect the figure's upper body, and then use that information to derive a set of bounds. Everything outside a large

box computed from the torso cannot be on the body (because the arms have fixed length, and so on). Similarly, a smaller box can be guaranteed to line on the body, because we have found the upper body. We can now use an interactive segmentation method (Section 20.2.1) to segment an estimate of the person from the background. The background color model can be estimated from pixels outside the box, and some inside the box; the foreground color model can be estimated from some of the pixels inside the box; and we can constrain some pixels to be foreground in the final segmentation. Because the segmentation might not be precise, we can dilate it (Algorithm 16.3) to get a somewhat larger domain. We now have a relatively small search domain and a very rough initial estimate of configuration to start the iterative re-estimation process. Further constraints are available if we are working with a motion sequence; these are explored in Section 20.3.

20.3 TRACKING PEOPLE

Tracking people in video is an important practical problem. If we could tell how people behave inside and outside buildings, it might be possible to design more effective buildings. If we could reliably report the location of arms, legs, torso, and head in video sequences, we could build much-improved game interfaces and surveillance systems.

20.3.1 Why Human Tracking Is Hard

Any tracking system, for any target, must balance two kinds of evidence to produce tracks. The first kind is direct measurements of state. In the extreme case, if we can detect perfectly, building tracking systems isn't that demanding. The second kind is predictable dynamics, which allows a system to pool evidence over multiple frames and produce good state estimates even when measurements are poor.

Tracking people is difficult, because detecting people is difficult and because human motion can be quite unpredictable. Detection is hard because many effects cause people to look different from window to window. There is a range of body shapes and sizes. Changes in body configuration and in viewpoint can produce dramatic changes in appearance. The appearance of clothing also can vary widely. At time of writing, no published method can find clothed people wearing unknown clothing in arbitrary configurations in complex scenes reliably (but see Section 17.1.2). The main cues to help overcome these difficulties are the fairly strong constraints on the layout of the body, and the relatively restricted appearance of a range of human body parts and configurations.

Motion cues present more subtle difficulties. If the people we are observing are engaged in known activities, their motions might be quite predictable. But the body can accelerate very quickly—think of the degree of motion blur in sports videos as an example—and the body parts that can engage in the most unpredictable motions tend also to be the ones that are hardest to detect. Forearms turn out to be difficult to track (small and fast moving), hands are even harder, and we are not aware of finger trackers that behave reliably for the full range of (potentially very fast-changing) finger motions.

Even so, motion is almost certainly a useful cue for detecting people or segments. Motion also can contribute by predicting plausible locations for detections

in the next frame, through some form of filtering procedure. Although body configurations change quickly from frame to frame, appearance changes very slowly, particularly if one is careful about illumination. This is because people tend not to change clothes from frame to frame. Generally, building a good person tracker seems to involve paying close attention to image appearance and data association, rather than to dynamical models or probabilistic inference. As a result, recent methods strongly emphasize various tracking by detection ideas, and the main kinds of distinction between methods are the same as those for detection.

There is a rich range of options for representing the body when we track, and a range of levels of detail are useful. Representing a person as a single point is sometimes useful; for example, such representations are enough to tell where and when people gather in a public space, or during a fire drill. Alternatives include: representing the head and torso; representing the head, torso, and arms; representing head, torso, arms, and legs; and so on, down to the fingers. Tracking becomes increasingly difficult as the number of degrees of freedom goes up, and we are not aware of any successful attempts to track the body from torso to fingers (they are a lot smaller than torsos, which introduces other problems). Most procedures for tracking single point representations use the methods of Chapter 11 directly, typically combining background subtraction with some form of blob appearance tracker.

We focus on trackers that try to represent the body with fairly detailed kinematic models, because such trackers use procedures specialized for tracking people. The state of the body could be represented in 3D or in 2D. If there are many cameras, a 3D state representation is natural, and multicamera tracking of people against constrained backgrounds now works rather well (see the notes). The flavor of this subject is more like reconstruction than like detection or recognition, and it doesn't fit very well into general pattern of single camera tracking. In many important cases—for example, an interface to a computer game—there will be only one camera. If we require a representation of the body in three dimensions, then we could use a 3D representation of state, perhaps joint locations in 3D, or a set of body segments in 3D modeled as surfaces. Alternatively, we could track the body using a 2D state representation, and then "lift" it to produce a 3D track. Relations between the 2D figure and the 3D track are complicated and might be ambiguous. The heart of the question is the number of possible 3D configurations that could explain a single image, and this depends quite a lot on what we observe in the image.

Generally, we favor tracking using a 2D representation then lifting the track to 3D, and we will discuss only this strategy in any detail. This is mainly a question of clarity. Methods for tracking using 3D state representations must deal with data association and with lifting ambiguity simultaneously, and this leads to complexity. In contrast, tracking in 2D is a data association problem alone, and lifting the track is a problem of ambiguity alone. Another advantage to working in 2D first, then lifting, is that the lifting process can use image evidence on longer timescales without having any significant effect on the complexity of the tracking algorithm. We will return to this argument in Section 20.4.

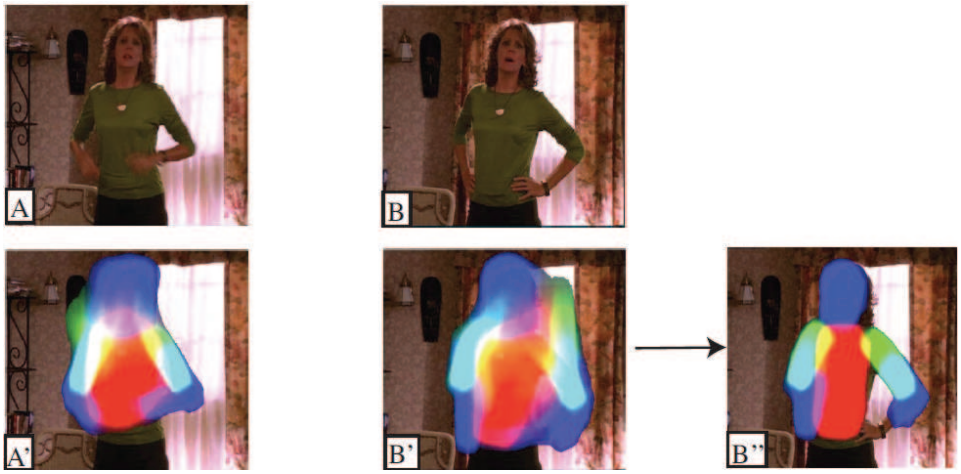


FIGURE 20.6: Human body segments do not change appearance much over time, so using multiple frames can yield a better appearance model and so a better parse. **A** shows a frame, and **A'** shows its parse, derived by the method of Ferrari *et al.* (2008), described in Section 20.2.2 and Figure 20.5. In this case, the parse has relatively low entropy, and we have a fairly accurate model of where everything is. The frame in **B** is more difficult, and a single frame method produces the parse of **B'**, which has relatively high entropy. By requiring that appearance be coherent over time, and that segments not move much from frame to frame, we can obtain the tighter parse of **B"**. *This figure was originally published as Figure 6 of "Progressive search space reduction for human pose estimation," by V. Ferrari, M. Marín-Jiménez, and A. Zisserman, Proc. IEEE CVPR 2008, © IEEE 2003.*

20.3.2 Kinematic Tracking by Appearance

In Section 20.3.2, we described methods to identify an appearance model for a person from a single image. Generally, the strategy was to find a small but plausible spatial domain in the image, then iterate configuration estimation and appearance estimation in that domain. In a motion sequence, we can build a much better appearance model by exploiting the fact that body segment appearance doesn't change over time. Furthermore, the sampling time of the video is relatively fast compared to body movement, which means we know roughly which search domain in the $n+1$th frame corresponds to which in the nth frame. This means that we can strengthen the appearance model by using multiple frames to estimate appearance. We can improve configuration estimates both by using the improved appearance model, and by exploiting the fact that segments move relatively slowly. Ferarri *et al.* show significant improvements in practice for upper body models estimated using these two constraints (Figure 20.6).

There is an alternative method to obtain an appearance model. It turns out that people adopt a lateral walking configuration rather often, meaning that if we have a long enough sequence (minutes are usually enough), we will detect this configuration somewhere. Once we have detected it, we can read off an appearance model because we know where the arms, legs, torso, and head are. The pictorial

structure model can detect lateral walking configurations without knowing the color or texture of body segments. We set up ϕ to score whether there are image edges close to the edges of the segment rectangles, and use strong angular constraints in ψ to detect only the lateral walking configuration. The resulting detector can be tuned to have a very low false positive rate, though it will then have a low detect rate, too. Now we run this lateral walking detector over every frame in the sequence. Because the detector has a low false positive rate, we know when it responds that we have found a real person; and because we have localized the torso, arms, legs, and head, we know what these segments look like.

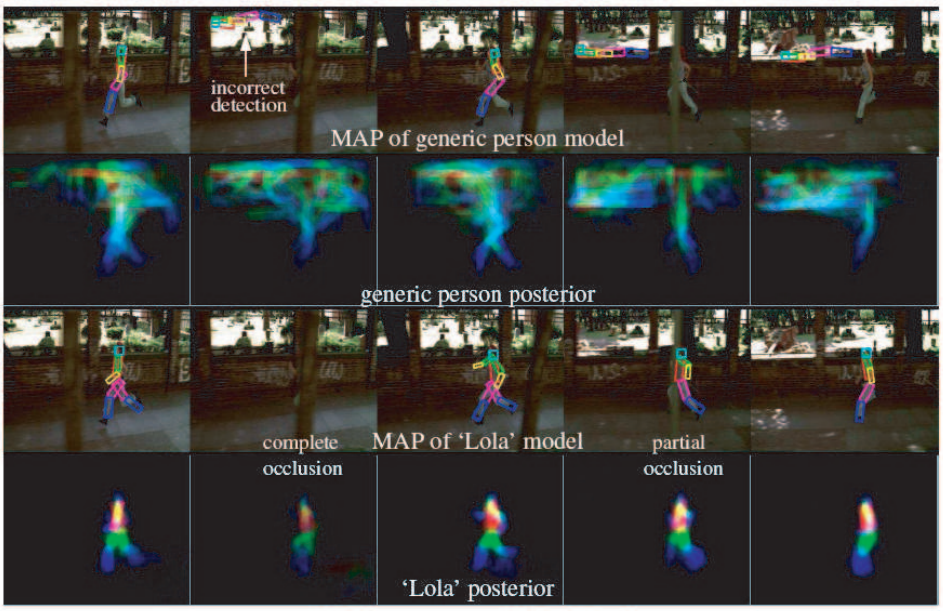
We can now build a discriminative appearance model for arms, legs, etc., and use this in a new pictorial structure model to detect each instance of the person. We take example pixels from each detected segment and from its background, and use, say, logistic regression to build a classifier that gives a one at segment pixels and a zero otherwise. Applying these to the images yields a set of segment maps, and the ϕ for each segment scores how many ones appear inside the image rectangle on the relevant segment map. We can now pass over the video again, using a pictorial structure with weak constraints to detect instances of this person.

20.3.3 Kinematic Human Tracking Using Templates

Some human motions—walking, jumping, dancing—are highly repetitive, and the relatively free structure of a fully deformable model is not necessary to track them. If we are confident that we will be dealing with such motions, then we could benefit by using more restrictive models of spatial layout. For example, if we are tracking only walking people in lateral views, then there are relatively few configurations that we will see, and so our estimate of layout should be better. There is another advantage to doing this: we can identify body configurations that are wholly out of line with what we expect, and report unusual behavior.

Toyama and Blake (2002) encode image likelihoods using a mixture built out of templates, which they call *exemplars* (see also Toyama and Blake (2001)). 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 (one could regard this as a simplified maximum entropy model; we are not aware of successful attempts to add complexity at this point). The normalizing constant is estimated with Laplace's method. Multiple templates can be used to 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.

We can think of this method as a collection of template matchers linked over time with a dynamical model (Figure 20.8). The templates, and the dynamical model, are learned from training sequences. Because we are modeling the foreground, the training sequences can be chosen so that their background is simple, so that responses from (say) edge, curve, and related detectors all originate on the moving person. Choosing templates now becomes a matter of clustering. Once



% of frames correctly localized

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

FIGURE 20.7: Ramanan (2005) 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. *Figure reprinted from D. Ramanan's UC Berkeley PhD thesis, "Tracking People and Recognizing their Activities," 2005, © 2005 D. Ramanan.*

templates have been chosen, a dynamical model is estimated by counting.

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 might be needed to cover all views of a moving person. Furthermore, inferring state might be quite difficult.

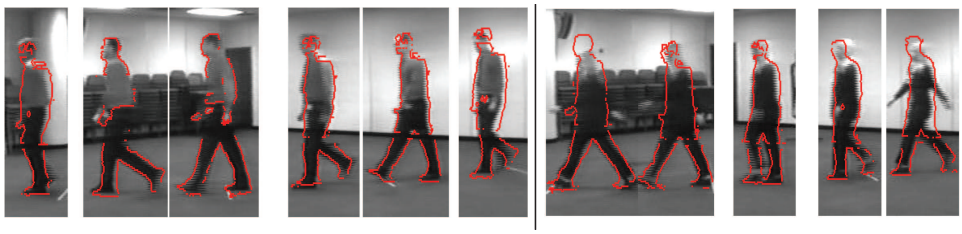


FIGURE 20.8: Toyama and Blake (2001) show that human motion can be tracked by matching templates then linking the templates over time. The templates encode possible body configuration, and are allowed to deform to account for camera variations. This representation has the advantage that a template can pool otherwise possibly unreliable edge evidence; tracking uses a particle filter (Section 11.5). The figure shows frames from two motion sequences, with the best matching template superimposed (ignore the horizontal line structure in the frames; this is just interlacing effects in the video). On the **left** of the bar, frames from a test sequence showing a person who also appears in the training sequences (i.e., it's the same actor, but not the same frames). Templates generalize across individuals well; the **right** shows frames from a test sequence featuring an actor who does not appear in the training sequences. *This figure was originally published as Figures 1 and 4 of "Probabilistic Tracking in a Metric Space," by K. Toyama and A. Blake, Proc. IEEE ICCV 2001,* © *IEEE, 2001.*

20.4 3D FROM 2D: LIFTING

Surprisingly, the 2D configuration of a person in an image allows reconstructing that person's 3D configuration, from some straightforward geometric reasoning. There are two kinds of reconstruction. An *absolute reconstruction* reconstructs the configuration of the body with respect to a global world coordinate system. A *relative reconstruction* yields the configuration of body segments with respect to some *root coordinate system*. The root coordinate system is carried with the body, with its origin typically in the torso.

Absolute reconstruction is difficult, even with motion information, because each separate frame is missing a translation in depth, and motion information is not usually enough to recover this. Absolute reconstruction with a moving camera is particularly tricky, because one would need good camera egomotion estimates to produce such a reconstruction (we are not aware of any such reconstructions in the literature at the time of writing). Relative reconstruction is enough for most purposes. For example, absolute reconstruction doesn't seem to be necessary to label activities.

Reconstructions appear to be ambiguous, but might not be. There are methods for avoiding ambiguity that exploit appearance details (Section 20.4.2). Furthermore, there may be disambiguating information in motion (Section 20.4.3).

20.4.1 Reconstruction in an Orthographic View

People in pictures typically are far from the camera compared to the range of depths they span (the body is quite flat), and so a scaled orthographic camera model is usually appropriate. One case where it fails is a person pointing toward

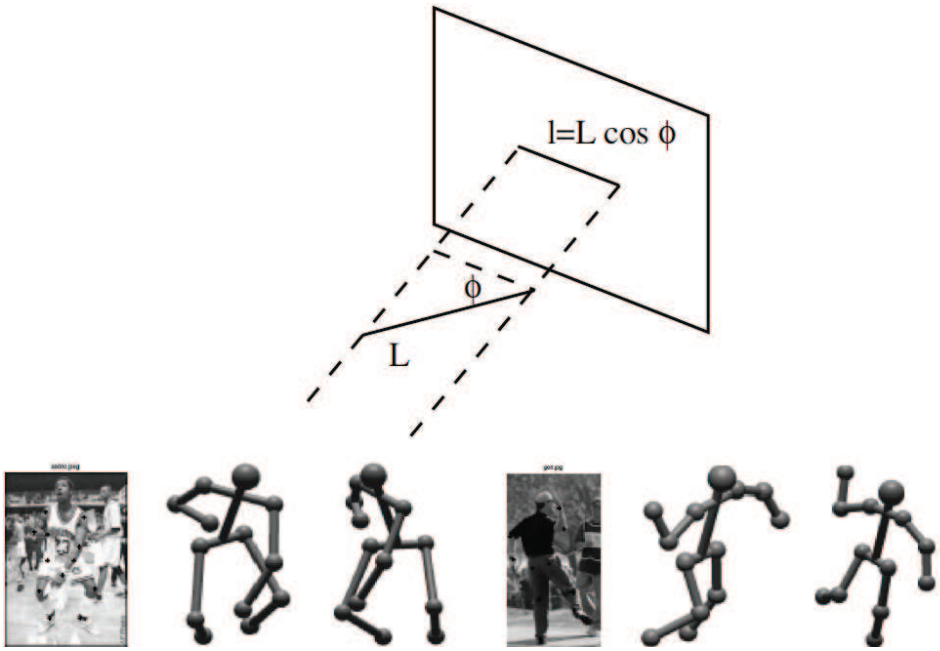


FIGURE 20.9: An orthographic view of a segment of known length L will have length $sL\cos\phi$, where ϕ is the angle of inclination of the segment to the camera and s is the camera scale linking meters to pixels (which is one in the figure above). In turn, this means that if we know the length of the body segment and can guess the camera scale, we can estimate $\cos\phi$ and so know the angle of inclination to the frame up to a twofold ambiguity. This method is effective; **below** we show two 3D reconstructions obtained by Taylor (2000), 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. *This figure was originally published as Figures 1 and 4 of "Reconstruction of articulated objects from point correspondences in a single uncalibrated image," by C.J. Taylor, Proc. IEEE CVPR, 2000* © *2000 IEEE.*

the camera; if the hand is quite close, compared with the length of the arm there may be 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 and assume we know its length. If we know the camera scale, and can mark each end of the body segment, 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 20.9 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. 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 9 to 11 body segments (the head is often omitted; the torso can reasonably be modeled with several segments), yielding from 512 to 2,048 possible reconstructions. These ambiguities persist for perspective images; examples appear in Figure 20.10.

FIGURE 20.10: Ambiguous reconstructions of a 3D figure, all consistent with a single view, from Sminchisescu and Triggs (2003). 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. *This figure was originally published as Figure 2 of "Kinematic jump processes for monocular 3D human tracking," by C. Sminchisescu and W. Triggs, Proc. IEEE CVPR, 2003* © *2003 IEEE.*

In this very simple model of the body, 3D reconstruction from a single image is ambiguous. However, the model oversimplifies in some important ways, and the true extent of ambiguity in this case is quite uncertain. One important oversimplification is that we assume that all 3D configurations are available. In practice, there are many constraints on the available joint rotations (for example, your elbow will move through about 70°), so some of the ambiguous configurations might not be consistent with the kinematics of the body. Unfortunately, there is clear evidence that there are multiple kinematically acceptable reconstructions consistent with a single image (Figure 20.10). It is not known whether there are multiple acceptable reconstructions associated with most images, or with only a few images.

20.4.2 Exploiting Appearance for Unambiguous Reconstructions

Mori and Malik (2005) deal with discrete ambiguities by matching (see also Mori *et al.* (2002)). 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 back-

FIGURE 20.11: Mori *et al.* (2002) deal with discrete ambiguities by matching test image outlines to examplars, which have keypoints marked. 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**. See also Mori and Malik (2005). *This figure was originally published as Figures 6 and 7 of "Estimating Human Body Configurations using Shape Context Matching," by G. Mori and J. Malik, IEEE Workshop on Models versus Exemplars in Computer Vision 2001* © *IEEE, 2001.*

ground 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 that aligns a number of sample points around that keypoint. The result is a markup of the test image with labeled joint positions and with which end of the segment is closest to the camera. A 3D reconstruction follows, as above (Figure 20.11 gives some examples).

An alternative is to regress the joint angles against an image of the body. The simplest regression method is to match the input to its nearest neighbor in a large training set then output the value associated with that nearest neighbor. Shakhnarovich *et al.* (2003) built 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 neighbors obtained using parameter-sensitive hashing. Generally, performance improves with more neighbors, with using a linear locally weighted regression (where one builds a linear regression model out of a pool of nearest neighbors), and when the method is robust. The best is a robust, linear, locally weighted regression. Their method produces estimates of joint angles with root mean square errors of approximately 20° for a 13 degree of freedom upper body model; a version of this approach can produce full 3D shape estimates (Grauman *et al.* 2004).

FIGURE 20.12: The 3D configuration of the body can be reconstructed using a form of nonparametric regression. Shakhnarovich *et al.* (2003) match the input frame (**top row**) to a large selection of labeled frames. The nearest neighbors are shown in the **center row**; these give a fair reconstruction in most cases, but can be improved by finding multiple nearest neighbors and building a robust linear regression (**bottom row**). *This figure was originally published as Figure 5 of "Fast Pose Estimation with Parameter-Sensitive Hashing," by G. Shakhnarovich, P. Viola, and T. Darrell, Proc. CVPR 2003, 2003. © IEEE, 2003.*

20.4.3 Exploiting Motion for Unambiguous Reconstructions

In many applications there is a video sequence of a moving person. In such cases, it does not make sense to infer the 3D structure for each frame. It is a reliable rule of thumb from the animation community that most body motions are quite slow compared to reasonable video frame rates. Evidence includes, for example, the relative ease with which motion capture sequences can be compressed with minimal loss (Arikan 2006). This means that reconstructed body configurations for each frame will not be independent, so that future (or past) frames may disambiguate the current reconstruction.

Howe (2004) incorporates dynamical information into the distance cost, by matching entire 3D motion paths to 2D image tracks. For each frame of a motion sequence, we render every motion capture frame in our collection using a discretized grid containing every possible camera and every possible root coordinate system. Now we must construct a sequence of 3D motion reconstructions that (a) joins up well and (b) looks like the tracked frames. This is an optimization problem. We build a transition cost for going from each triple of (motion capture frame, camera, root coordinate system) to every other such triple. This cost should penalize excessively large body segment and camera velocities. We compute a match cost comparing the rendered frame with the tracked frame. Write F_i for the ith frame in tracked sequence, S for a reconstruction of that sequence, and (L_i, C_i, R_i) for the reconstruction frame and camera corresponding to F_i. The cost function for a