Tracking People by Learning Their Appearance

Deva Ramanan, D.A. Forsyth, Andrew Zisserman

Abstract

An open vision problem is to automatically track the articulations of people from a video sequence. This problem is difficult because one needs to determine both the number of people in each frame and estimate their configurations. But finding people and localizing their limbs is hard because people can move fast and unpredictably, can appear in a variety of poses and clothes, and are often surrounded by limb-like clutter.

We develop a completely automatic system that works in two stages; it first builds a model of appearance of each person in a video, and then, it tracks by detecting those models in each frame ("tracking by model-building and detection"). We develop two algorithms that build models; one bottom-up approach groups together candidate body parts found throughout a sequence. We also describe a top-down approach that automatically builds people-models by detecting convenient key poses within a sequence. We finally show that building a discriminative model of appearance is quite helpful since it exploits structure in a background (without background-subtraction).

We demonstrate the resulting tracker on hundreds of thousands of frames of unscripted indoor and outdoor activity, a feature-length film ('Run Lola Run'), and legacy sports footage (from the 2002 World Series and 1998 Winter Olympics). Experiments suggest that our system can (a) count distinct individuals; (b) identify and track them; (c) recover when it loses track, for example, if individuals are occluded or briefly leave the view; (d) identify body configuration accurately; and (e) is not dependent on particular models of human motion.

Index Terms

people tracking, motion capture, surveillance

I. Introduction

One of the great open challenges in computer vision is to build a system that can kinematically track people. A reliable solution opens up tremendous possibilities, from human computer interfaces to video data mining to automated surveillance. This task is difficult because people can move fast and unpredictably, can appear in a variety of poses and clothes, and are often surrounded by clutter. Because of the technical challenge and the attractive rewards, there exists a rich body of relevant literature. However, in practice, the resulting systems typically require some limiting assumption such as multiple cameras, manual initialization, or controlled/simplistic backgrounds [2].

Much previous work has focused on high-level reasoning (such as mechanisms of inference) but in our experience the “devil is in the details” – low-level image features play a crucial role. Background subtraction can be a powerful low-level cue, but does not always apply. We want to track people who happen to stand still in front of moving backgrounds. Instead, we describe an approach that learns good image features for a particular video sequence. For example, tracking a person wearing a red shirt is much easier with a person-model that looks for red pixels. In broader terms, we argue that *model-based tracking is easier with a better model*. Thus the process of tracking becomes intertwined with the process of building a good model for the object being tracked (i.e., learning the appearance of a person’s shirt).

We describe two approaches that learn appearance; one bottom-up algorithm groups together candidate body parts found throughout a sequence, while another top-down approach builds appearance models from convenient poses (Section IV and V) . Our system then tracks by detecting the learned models in each frame. We demonstrate the final tracker on hundreds of thousands of frames of commercial and unscripted video. We find that we can accurately track people from a single view with automatic initialization in front of complex backgrounds.

II. Background

A full review of the tracking literature is beyond the scope of this paper (some are provided in [1, 2]). By far the common approach is to use a Hidden Markov Model (HMM), where the hidden states are the poses (X_t) to be estimated, and the observations are images (I_t) of a video sequence. In this work, we will regard pose X_t as a vector of 2D joint positions (methods for extending 2D pose estimates to 3D are described in [3]).

Standard first-order Markov assumptions allow us to decompose the joint probability into

$$\mathrm{P}(X_{1:T}, I_{1:T}) = \prod_t \mathrm{P}(X_t|X_{t-1})\mathrm{P}(I_t|X_t),$$

where we use the shorthand $X_{1:T} = \{X_1, \ldots, X_T\}$. Tracking corresponds to inference on this probabilitistic model; typically one searches for the maximum a posteriori (MAP) sequence of poses given an image sequence.

Inference: Much of the literature has focused on inference, which are typically variants of dynamic programming [4–6], kalman filtering [7–9] or particle filtering [10–15]. One uses the pose in the current frame and a dynamic model to predict the next pose; these predictions are then refined using image data. Particle filtering is attractive because *multiple* predictions — obtained by running samples of the prior through a model of the dynamics — are reweighted by comparing them with the local image data (the likelihood).

Dynamic model: The dynamic model $\mathrm{P}(X_t|X_{t-1})$ can be learned from motion capture data [10, 16] but this often requires choosing the specific motion (walking, running, etc…) *a priori*. Alternatively, one can build a dynamic model that selects a motion online from a set of models [17, 18]. It is difficult to apply such techniques to say, track Michelle Kwan (Figure 22), because there does not exist much figure-skating motion capture data. As such, we restrict ourselves to generic dynamic models such as constant velocity.

Likelihood model: One needs a person model to compute an image likelihood $\mathrm{P}(I_t|X_t)$. Typically one uses a template encoding appearance [5, 19–21], local motion [22] or both [23]). The template can be updated on-line or learned off-line; we will discuss these approaches in detail. In general, constructing a good likelihood model seems to be hard because it must be generic enough to represent bodies in various poses and clothes but specific enough not to be confused by background clutter.

Data association: A pre-process (such as background subtraction) oftentimes identifies the image regions of interest, and those are the observations fed into the likelihood model. In the radar tracking literature, this pre-process is known as data-association [24]. We argue that in practice, the difficulty in making any video tracking algorithm succeed lies in data-association – identifying those image regions that consist of the object to be tracked. Particle filters implicitly use the dynamic model $\mathrm{P}(X_t|X_{t-1})$ to perform data association; multiple motion predictions tell us where to look in an image. However, such an approach can drift given a weak dynamic model and a cluttered background. This is because one may need an exorbitant number of particles to accurately represent the posterior. One solution is to intelligently resample; strategies include annealing [12] and covariance-scaled sampling [14]. The alternative is to explicitly search the image at each frame for regions of interest. Indirect methods of doing this include background subtraction and skin detection. We view these approaches as too restrictive – we want to track clothed people in front of backgrounds that happen to move. A final alternative would be to use

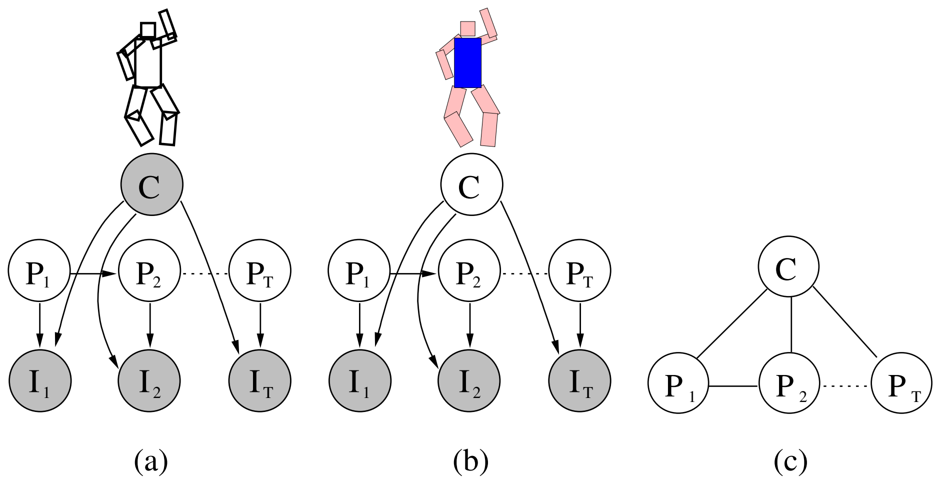


Fig. 1. In (**a**),we display the graphical model corresponding to model-based tracking. Object position P_t follows a Markovian model, while image likelihoods are defined by a model template C. Since C is given *a priori*, the template must be invariant to clothing; a common approach is to use an edge template. In this work we treat C as a random variable (**b**), and so we *build* a template specific to the particular person in a video as we track his/her position P_t. We show an undirected model in (**c**) that is equivalent to (**b**).

a template that directly identifies what images regions are people.

Templates: Most template-based trackers use an edge template, using such metrics as chamfer distance or correlation to evaluate the likelihood for a given pose. Let us assume the object state $X_t = \{P_t, C\}$ where P_t is the pose, the template is represented by an image patch C. For simplicity, assume the likelihood is measured by SSD (though we will look at alternate image descriptors):

$$\mathrm{P}(I_t|P_t, C) \propto \exp^{-||I_t(P_t)-C)||^2} \tag{1}$$

where we have ignored constants for simplicity. If we condition on C (assume the appearance template is given), our model reduces to a standard HMM (see Figure 1). Algorithms that track people by template matching follow this approach [5, 13, 19–21, 25]. Since these templates are built *a priori*, they are detuned because they must generalize across all possible people wearing all possible clothes. Such templates must necessarily be based on generic features (such as edges) and so are easily confused by background clutter. This makes them poor at data association (see Figure 15).

One could update the template on-line by including a temporally varying C_t in the object state X_t [7, 10, 26]. In practice, these markovian appearance models are hard to initialize (because we

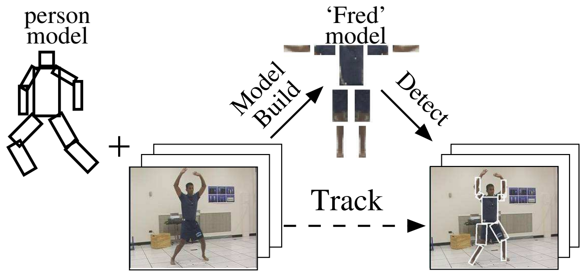


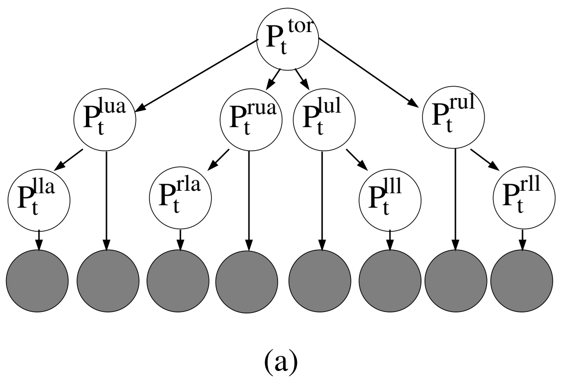
Fig. 2. We use a model-based people tracker. Initially, we use a detuned edge-template as a generic person model. From the video data, we **build** an instance-specific model capturing a person's appearance. We then track by **detecting** that model in each frame.

do not know appearance *a priori*) and can also drift. These shortcomings seem related, since one method of creating a robust tracker is to continually re-initialize it. An attractive strategy is to build a good template off-line rather than on-line. Methods that estimate human pose given a single image follow this route; a model is learned from training data, and is later used to estimate pose in a novel image. To handle the large search space of articulated poses, sampling-based algorithms [15, 27, 28] and bottom-up detection [29–32] are common. Since these models must generalize to all people, they must still be invariant to clothing and so typically require some other restriction like visible skin or stationary/uncluttered backgrounds.

III. Our approach

We treat C from Fig.1 as an unknown that is inferred automatically. We *build* a template tuned to each specific person in a video. A template that encodes the red color of a person's shirt *can* perform data association since it can quickly ignore those pixels which are not red. Under our model, the focus on tracking becomes not so much identifying where an object is, but learning what it looks like. A vital issue is the representation of the learned appearance – it should be flexible enough to handle changes in pose and illumination. Since these invariances are also required for object detection, we use a representation from that literature.

We model the human body as a puppet of rectangles, as shown in Figure 2. In the object detection community, this is often called a pictorial structure [33, 34] or body plan [32]. We describe it further in Section III-A. If a person changes clothes or undergoes extreme illumination changes, a single puppet model may not suffice. However, we posit (and verify by extensive experiments) that a single model works for many realistic situations; a red shirt tends to stay



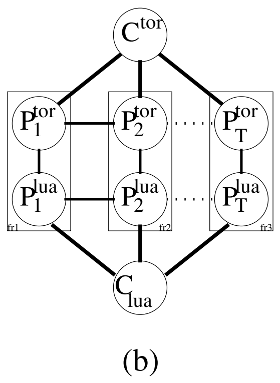


Fig. 3. Our person model (**a**) is a tree pictorial structure. This model is parameterized by probability distributions capturing geometric arrangement of parts P$(P_t^i|P_t^j)$ and local part appearances P$(I_t|P_t^i, C^i)$ (the vertical arrows into the shaded nodes). Our full temporal model replicates (a) for each frame t, adds in a part motion model P$(P_{t+1}^i|P_t^i)$, and explicitly models part appearance C^i. We show the full model for a torso-lua assembly in **(b)** (marginalizing out the image observations for convenience).

red.

This work describes a system that, given a video sequence of possibly multiple people, automatically tracks each person. The system first **builds** a puppet model of each person's appearance, and then tracks by **detecting** those models in each frame. Since methods for detecting pictorial structures are well understood [34], we focus primarily on algorithms for learning them (from a given video sequence).

We describe two methods of building appearance models. One is a **bottom-up** approach that looks for candidate body parts in each frame [35]. We cluster the candidates to find assemblies of parts that might be people (Section IV). Another is a **top-down** approach that looks for an entire person in a single frame [36]. We assume people tend to occupy certain key poses, and so we build models from those poses that are easy to detect (Section V). Once we have learned an appearance model by either method, we detect it in each frame to track a person (Section VI). We finally conclude with extensive results in Section VII.

A. Temporal Pictorial Structures

Our basic representation is a pictorial structure [33, 34]. We display a human body pictorial structure as a graphical model in Figure 3-(a). Following a first-order Markov model, we replicate the standard model T times, once for each frame:

$$\mathrm{P}(P_{1:T}^{1:N}, I_{1:T}|C^{1:N}) = \prod_{t}^{T}\prod_{i}^{N}\mathrm{P}(P_t^i|P_{t-1}^i)\mathrm{P}(P_t^i|P_t^{\pi(i)})\mathrm{P}(I_t|P_t^i, C^i). \tag{2}$$

As a notation convention, we use superscripts to denote body parts (i ranges over the torso plus left/right upper/lower arms/legs) and subscripts to denote frames $t \in \{1 \dots T\}$. The term $\pi(i)$ denotes the parent of part i, following the tree in Figure 3-(a). The variable P_t^i is a 3-vector capturing the position (x, y) and orientation θ of part i at time t. The first term in the right hand side (RHS) of Equation 2 is a motion model for an individual part; the last two terms are the standard geometric and local image likelihood terms in a pictorial structure [34].

For a fixed sequence of images $I_{1:T}$, we can interpret the RHS as an energy function of P_t^i and C^i. We visualize the function (for a torso-lua assembly) as an undirected graphical model in Figure 3-(b). Effectively, we want to find an arm position P_t^{lua} such that the arm lies nearby a torso $\mathrm{P}(P_t^{lua}|P_t^{tor})$, the arm lies near its position in the previous frame $\mathrm{P}(P_t^{lua}|P_{t-1}^{lua})$, and the local image patch looks like our arm model $\mathrm{P}(I_t(P_t^{lua})|P_t^{lua}, C^{lua})$.

Our image likelihood models the local image patch with a Gaussian centered at the template C^i

$$\Psi_t(P_t^i, C^i) \quad = \quad \mathrm{P}(I_t|P_t^i, C^i) \quad \propto \quad \exp^{-||I_t(P_t^i) - C^i||^2} \tag{3}$$

We ignore constants for simplicity. We discuss our image features further in Sections IV and V (we assume they are pre-scaled to have unit-variance). We model the spatial kinematics of the human body with a puppet of rectangles with freely-rotating revolute joints, using potentials of the form

$$\Psi(P_t^{lua}, P_t^{tor}) \quad = \quad \mathrm{P}(P_t^{lua}|P_t^{tor}) \quad \propto \quad \mathcal{I}(\mathcal{D}(P_t^{tor}, P_t^{lua}) < d_{max}) \tag{4}$$

where $\mathcal{I}$ is the standard identity function and $\mathcal{D}(P1, P2)$ is the distance between the hinge points for the two segments (as in [34]). We add angular bounds preventing the upper legs from point up into the torso. An alternative would be to impose a smooth penalty for mis-alignment (as in [34]); we found this did not significantly improve pose estimates, though it may be desirable for gradient-based algorithms.

Our motion model is bounded velocity

$$\Psi(P_t^i, P_{t-1}^i) \quad = \quad \mathrm{P}(P_t^i|P_{t-1}^i) \quad \propto \quad \mathcal{I}(||P_t^i - P_{t-1}^i|| < v_{max}) \tag{5}$$

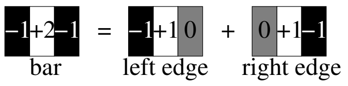


Fig. 4. One can create a rectangle detector by convolving an image with a bar template and keeping locally maximal responses. A standard bar template can be written as the summation of a left and right edge template. The resulting detector suffers from many false positives, since either a strong left or right edge will trigger a detection. A better strategy is to require both edges to be strong; such a response can be created by computing the minimum of the edge responses as opposed to the summation.

Finding an optimal track given a video sequence now corresponds to finding the maximum *a posteriori* (MAP) estimate of C_t^i and P_t^i from Figure 3-(b). Exact inference on this model is difficult because the graph contains loops and the state spaces of the variables are quite large. An attractive strategy is to ignore the loops and pass local messages [37], and to represent those messages with a set of samples [38, 39]. Since we want MAP estimates, we will pass max-product messages [40]. We present two algorithms that, although seemingly quite different, are essentially the result of different message-passing *schedules*.

IV. Building models by clustering

We present in this section an algorithm that passes messages across a set of T frames. We pass messages that infer the value of C^i, the appearance of part i; we then pass messages that infer C^j, part j's appearance, and so on (until we learn the appearance of all body parts). This interpretation is explained further in Section IV-E.

First, we procedurally describe the algorithm. An important observation is that we have some *a priori* notion of part appearance C^i as having rectangular edges. We would like to refine this model to capture the full appearance of a part. This suggests the following approach:

1) *Detect* candidate parts in each frame with an edge-based part detector.
2) *Cluster* the resulting image patches to identify body parts that look similar across time.
3) *Prune* clusters that move too fast in some frames.

A. Detecting parts with edges

We model body parts as cylinders which project to rectangles in an image. One might construct a rectangle detector using a Haar-like template of a light bar flanked by a dark background (Figure 4). To ensure a zero DC response, one would weight values in white by 2 and values in black by -1. To use the template as a detector, one convolves it with an image and defines locally maximal responses above a threshold as detections. This convolution can be performed efficiently using integral images [41]. We observe that a bar template can be decomposed into a left and right edge template $f_{bar} = f_{left} + f_{right}$. By the linearity of convolution (denoted *), we can write the response as

$$I * f_{bar} = I * f_{left} + I * f_{right}$$

In practice, using this template results in many false positives since either a single left or right edge triggers the detector. We found taking a *minimum* of a left and right edge detector resulted in response function that (when non-maximum suppressed) produced more reliable detections

$$min(I * f_{left}, I * f_{right})$$

With judicious bookkeeping, we can use the same edge templates to find dark bars on light backgrounds. We assume we know the scale of people in a given video, and so search over a single scale for each body part. We expect our local detectors to suffer from false positives and missed detections, such as those shown in Figure 5-(a). In our experiments we used a detector threshold that was manually set between 10 and 50 (assuming our edge filters are $\mathcal{L}_1$-normalized and images are scaled to 255).

B. *Clustering image patches*

Since we do not know the number of people in a video (or, for that matter, the number of segment-like things in the background), we do not know the number of clusters *a priori*. Hence, clustering segments with parametric methods like Gaussian mixture models or k-means is difficult. We opted for the mean-shift procedure [42], a non-parametric density estimation technique.

We create a feature vector for each candidate segment, consisting of a 512 dimensional RGB color histogram (8 bins for each color axis). Further cues — for example, image texture — might be added by extending the feature vector, but appear unnecessary for clustering. We scale the feature vector by an empirically-determined to yield a unit-variance model in Eq. 3.

Identifying segments with a coherent appearance across time involves finding points in this feature space that are (a) close and (b) from different frames. The mean-shift procedure is an iterative scheme in which we find the mean position of all feature points within a hypersphere

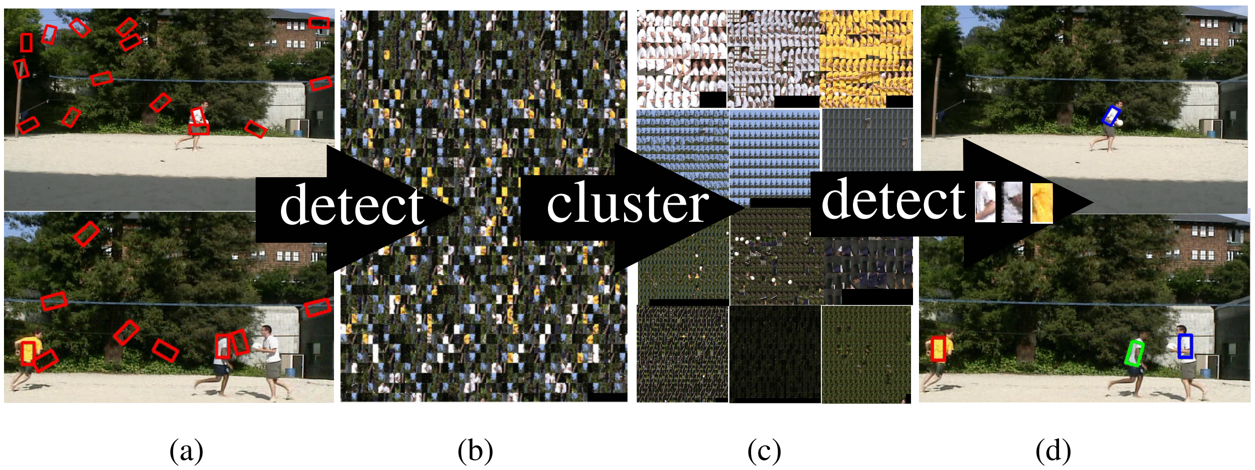


Fig. 5. Building a model of torso appearance. We detect candidate torsos using an edge template (**a**), and show the detected image patches in (**b**). We cluster the patches to enforce a constant appearance (**c**), and then prune away those clusters that do not move (all but the top 3). We now use the medoid image patch from each cluster as templates to find the (dynamically valid) torso tracks (**d**). Note that since we are using appearance and not edges to find segments, we can track against weak contrast backgrounds (the blue segment on the **lower right**).

of radius h, recenter the hypersphere around the new mean, and repeat until convergence. We initialize this procedure at each original feature point, and regard the resulting points of convergence as cluster centers. For example, for the sequence in Fig 5, starting from each original segment patch yields 12 points of convergence (denoted by the centers of the 12 clusters in (c)).

As a post-processing step we greedily merge clusters which contain members within h of each other, starting with the two closest clusters. We account for over-merging of clusters by extracting multiple valid sequences from each cluster during step (c) (for each cluster we keep extracting sequences of sufficient length until none are left; this is explained further in Section IV-C). Hence for a single arm appearance cluster, we might discover two valid tracks of a left and right arm. For our experiments, we manually varied h between .05 and .25 (assuming feature vectors are $\mathcal{L}_1$-normalized). In general, we found the clustering results to be sensitive to h.

C. *Enforcing a motion model*

For each cluster, we want to find a sequence of candidates that obeys our bounded velocity motion model defined in Equation 5. By fitting an appearance model to each cluster (typically a Gaussian, with mean at the cluster mean and standard deviation computed from the cluster), we can formulate this optimization as a straightforward dynamic programming problem. The

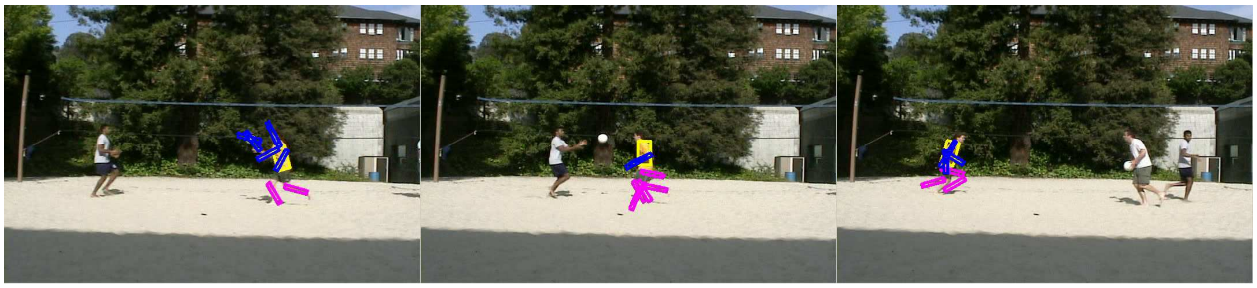


Fig. 6. Building a model of arms and legs. We search near the yellow-shirt track from Fig. 5-(d) for candidate arms (shown in **blue**) and legs (shown in **red**)). We use our kinematic model (Eq.6) to limit the search space; we only look for arms near the top of estimated torso locations. We learn arm and leg templates by clustering the candidates, as shown in Figure 7.

reward for a given candidate is its likelihood under the Gaussian appearance model, and the temporal rewards are '0' for links violating our velocity bounds and '1' otherwise. We add a dummy candidate to each frame to represent a "no match" state with a fixed charge. By applying dynamic programming, we obtain a sequence of segments, at most one per frame, where the segments are within a fixed velocity bound of one another and where *all* lie close to the cluster center in appearance.

We finally prune the sequences that are too small or that *never move*. This means we cannot track a person who is still for the entire duration of sequence. However, if they move at some point, the body part clusters will contain movement and will not be discarded. This means that, unlike background-subtraction techniques, we can track someone while standing still (so long as they move at some other part of the sequence).

D. Learning multiple appearance models

We use the learned appearance to build better segment detectors; e.g., we now know that the torso is a yellow rectangle, rather than just two parallel edges. We search for *new* candidates using the medoid image patch of the valid clusters from Figure 5-(c) as a template. We link up those candidates that obey our velocity constraints into the final torso track in Figure 5-(d). Since we found 3 valid torso clusters, we have 3 final torso tracks. For each track, we search near the estimated torsos $\hat{P}^{tor}$ for candidate lower-arms and lower-legs. As shown in Figure 6, we restrict the search to locations with non-zero probability under our kinematic model

$$\mathrm{P}(P^{lla}|\hat{P}^{tor}) = \sum_{P^{lua}} \mathrm{P}(P^{lla}|P^{lua})\mathrm{P}(P^{lua}|\hat{P}^{tor}). \tag{6}$$

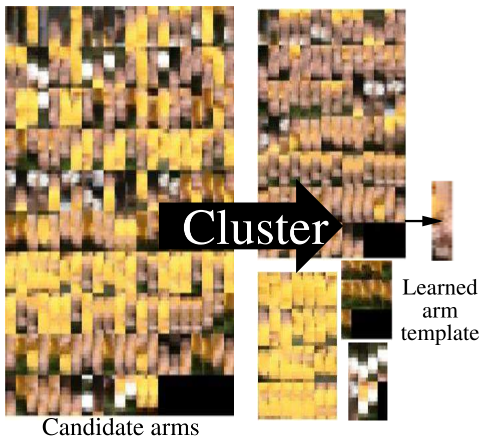


Fig. 7. Clustering the candidate arms from Figure 6 (we show the patches on the **left**). The learned template is the medoid patch of the largest cluster. We similarly cluster the candidate legs to learn a leg template. Repeating this procedure for the other two torso clusters, we learn appearance templates for all three people in this sequence (Figure 8).

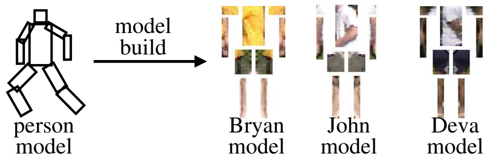


Fig. 8. Using the segment detection and clustering procedures described in Figures 5 and 6 on the shown sequence, we have automatically bootstrapped a generic person detector into a 'John', 'Bryan', and 'Deva' detector.

We cluster the candidates (as in Section IV-B) to learn lower arm and leg templates for that specific torso track. We similarly learn upper-limb templates (given $\hat{P}^{tor}$ and $\hat{P}^{lla}$ estimates) by clustering together candidate upper limbs. We repeat the procedure for each of the 3 torso tracks.

Effectively, we have taken our initial person detector, which consisted of a deformable template of parallel edges, and built from the the video a collection of *person-specific* detectors (the 'John', 'Bryan', and 'Deva' detectors in Figure 8). In Section VI, we track by independently detecting people in each frame. For long sequences, we build appearance models by clustering only an initial subset of the frames. We found that using 50-100 frames suffices, though we explore this point further in Section VII-A.

E. Approximate Inference

We now cast our algorithm in light of our temporal pictorial structure from Section III-A. We visualize our message-passing schedule with a set of trees in Figure 9. We show in [1] that the mean shift clustering algorithm finds modes in the posterior of C^i, as computed by tree (a).

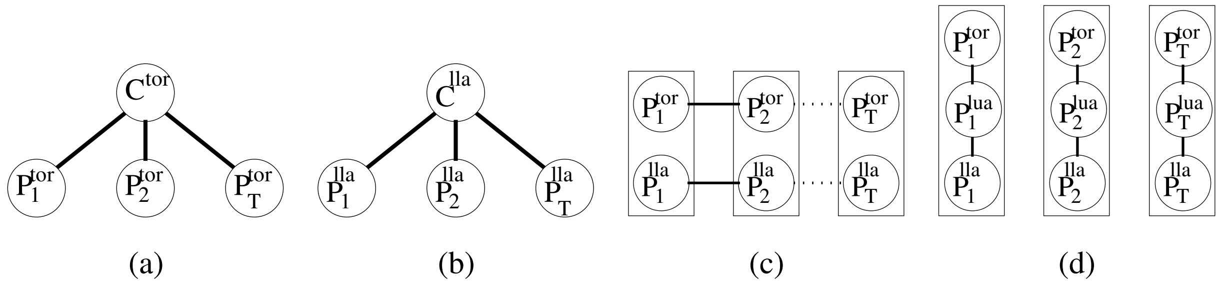


Fig. 9. A set of trees for loopy inference on our temporal pictorial structure. Trees (a) and (b) learn and apply torso and lower-arm appearance templates. Tree (c) enforces our motion model. Tree (d) restricts lower-arm and torso patches to those which obey our kinematic constraints.

We interpret each torso cluster as a *unique* person, instantiating one temporal pictorial structure for each cluster. For each instantiation, we use the mode estimate (the medoid of the cluster) to construct a new template $\hat{C}^{tor}$. We use this template to perform inference on the remaining trees from Figure 9. We perform inference on tree (c) to find a sequence of torso detections that all look like the template and that move smoothly. We then infer on (d) to find a set of lower-arm segments near the found torsos. We then infer on (b) to learn an lower-arm appearance $\hat{C}^{lla}$. We then infer on the arm chain in (c) to obtain a lower-arm track. We now use the estimated torsos and lower-arms to find candidate upper arms and learn their appearance. We repeat for the legs, learning a single appearance for left/right limbs. We learn appearance in an order which reflects the quality of our segment detectors. Our torso detector performs the best, followed by the lower limb detectors (since they are more likely to lie away from the body and have a cleaner edge profile).

Our iterative approach is similar to the partitioned sampling scheme of [43], which localizes the torso first, and then finds the remaining limbs. We suffer from the same drawbacks; namely, if the torso localization (and estimated appearance) is poor, the resulting appearance and localization estimates for the limbs will suffer. One remedy might be to continually pass messages in Figure 9 in a loopy fashion (e.g., re-estimate the torso appearance given the arm appearance).

V. Building models with stylized detectors

The approach of clustering part detectors works well when parts are reliable detected. However, building a reliable part detector is hard; a well-known difficulty of **bottom-up** approaches. An alternative strategy is to look for an entire person in single frame. This is difficult because

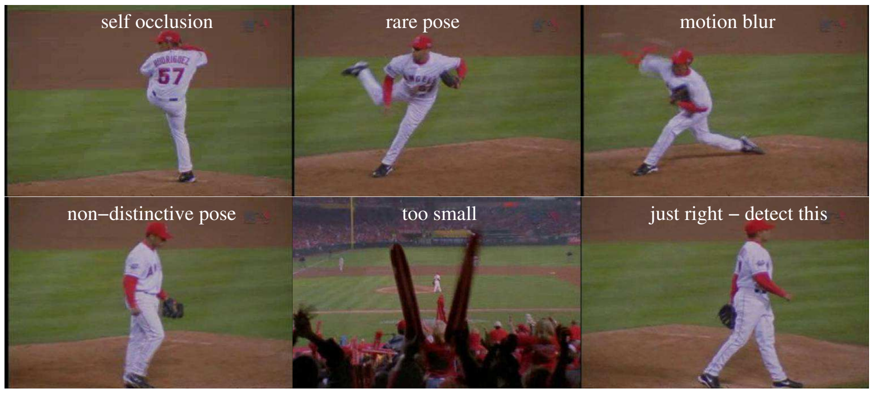


Fig. 10. What poses are easy to detect and build appearance from? We cannot learn appearance when body parts are occluded. People are hard to detect when they occupy an awkward pose, suffer from motion blur, look like the background, or are too small. As such, we build a stylized detector for large people in lateral walking poses (**bottom right**).

people are hard to detect due to variability in shape, pose, and clothing; a well-known difficulty of **top-down** approaches.

We could detect people by restricting our temporal pictorial structure from Equation 2 to a single time slice. Such a pictorial-structure detector can cope with pose variation. But since we do not know part appearances C^i *a priori*, we must use generic edge templates as part models. Such a detector will be confused by background clutter (see Figure 15).

However, our detector is not trying to "detect" a person, but rather build a model of appearance. This is an important distinction because typically one wants detectors with high precision and recall performance. In our case, we want a person detector with rather unique properties: (a) it must accurately localize limbs (since we will use the estimated limbs to build appearance models) and (b) it should have high precision (we want most detections to be of people). Given both, we can tolerate a low recall rate since we can use the learned appearance models to find the figure in those frames where the detector failed [36].

We build a person detector that only detects people in typical poses. Even though the detector will not fire on atypical poses, we can use the appearance learned from the standard poses to track in those atypical frames. This notion of **opportunistic detection** states that we can choose those poses we want to detect. This way we concentrate our efforts on easy poses rather than expending considerable effort on difficult ones. Convenient poses are ones that are (a) easy to detect and (b) easy to learn appearance from (see Figure 10). For example, consider a person walking in a lateral direction; their legs form a distinctive scissor pattern that one tends not to

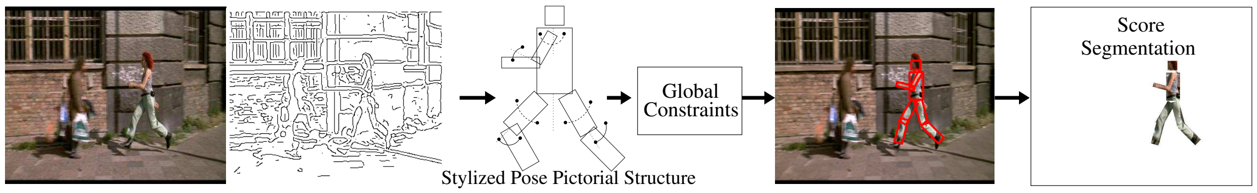


Fig. 11. Our lateral-walking pose finder. Given an edge image on the **left**, we search for a tree pictorial structure [34] using rectangle chamfer template costs to construct limb likelihoods. We restrict limbs to be positioned and oriented within bounded intervals consistent with walking left. We set these bounds (designated by the arcs overlaid on the model) by hand. We also search a mirror-flipped version of the image to find people walking right. To enforce global constraints (left and right legs should look similar), we sample from the pictorial structure posterior (using the efficient method of [34]), and re-compute a global score for the sampled configurations. The best configuration is shown on the **right**. In general, this procedure also finds walking poses in textured backgrounds; to prune away such false detections, we re-evaluate the score by computing the goodness of a segmentation into person/non-person pixels. We do this by building an appearance model for each limb (as in Figure 12) and then use the model to classify pixels from this image. We define the final cost of a walking-pose detection to be the number of mis-classified pixels.

find in backgrounds. The same pose is also fairly easy to learn appearance from since there is little self-occlusion; both the legs and arms are swinging away from the body. Following our observations, we build a single-frame people detector that finds people in the mid-stance of a lateral-walk.

Our system initially detects a lateral-walking pose with a stylized detector (Section V-A). That detection segments an image into person/background pixels. This allows us to build a **discriminative** appearance model (Section V-B) – we learn the features that discriminate the figure from its background (and assume those features will also discriminate the figure in other frames).

A. Detecting lateral walking poses

An overview of our approach to people detection is found in Figure 11. We will use a sequence from the film "Run Lola Run" as our running example (pun intended). We construct a (stylized) person detector by restricting our full temporal model from Figure 3-(b) to a single frame. We write a single-frame pictorial structure model as:

$$\mathrm{P}(P^{1:N}, I | C^{1:N}) = \prod_{i}^{N} \mathrm{P}(P^i | P^{\pi(i)}) \mathrm{P}(I | P^i, C^i). \tag{7}$$

We use an 8-part model, searching for only one arm since we assume the other arm will be occluded in our lateral walking pose. We modify our geometric and image likelihood terms (originally defined in Section III-A) to look for stylized poses.

P$(\boldsymbol{P^i}|\boldsymbol{P^{\pi(i)}})$: We manually set our kinematic shape potentials to be uniform within a bounded range consistent with walking laterally (Figure 11). For example, we force θ for our upper legs to be between 45 and 15 degrees with respect to the torso axis. We do not allow them to be 0 degrees because we want to detect people in a distinctive scissor-leg pattern. Learning these potentials automatically from data is interesting future work.

P$(\boldsymbol{I}|\boldsymbol{P^i}, \boldsymbol{C^i})$: We evaluate the local image likelihood with a chamfer template edge mask [44]. Our part template C^i must be invariant to clothing variations and so we use a rectangular edge template (Figure 11). Given an image, the chamfer cost of an edge template is the average distance between each edge in the template and the closest edge in the image. We compute this efficiently by convolving the distance-transformed edge image with the edge template. To exploit edge orientation cues, we quantize edge pixels into one of 12 orientations, and compute the chamfer cost separately for each orientation (and add the costs together). To capture the deformations from Figure 11, we convolve using rotated versions of our templates.

Since we use our lateral-walking detector in a high-precision/low-recall regime, we need to look only at those configurations where all the limbs have high likelihoods. Before evaluating the kinematic potentials, we perform non-maximum suppression on the chamfer likelihood response functions (and only keep candidate limbs above a likelihood threshold). We also throw away arm candidates that are vertical or horizontal (since there tends to be many vertical and horizontal rectangles in images of man-made structures). This is again justified for high-precision/low-recall detection; even though the arm of a person may in fact be horizontal or vertical, we *choose* not to learn their appearance in this pose, since we would encounter many false positive detections (and build incorrect appearance models).

Global constraints: We found it useful to enforce global constraints in our person model. For example, left and right legs tend to be similar in appearance [31]. Also, our kinematic leg potentials still allow for overlap if the left leg happens to be translated over onto the right leg. These dependencies cannot be captured by a tree pictorial structure. Instead of finding the MAP estimate of Eq.7, we generate samples from the posterior (using the efficient method of [34]), and re-score the samples under the global constraints. We generate 2000 samples per image, and

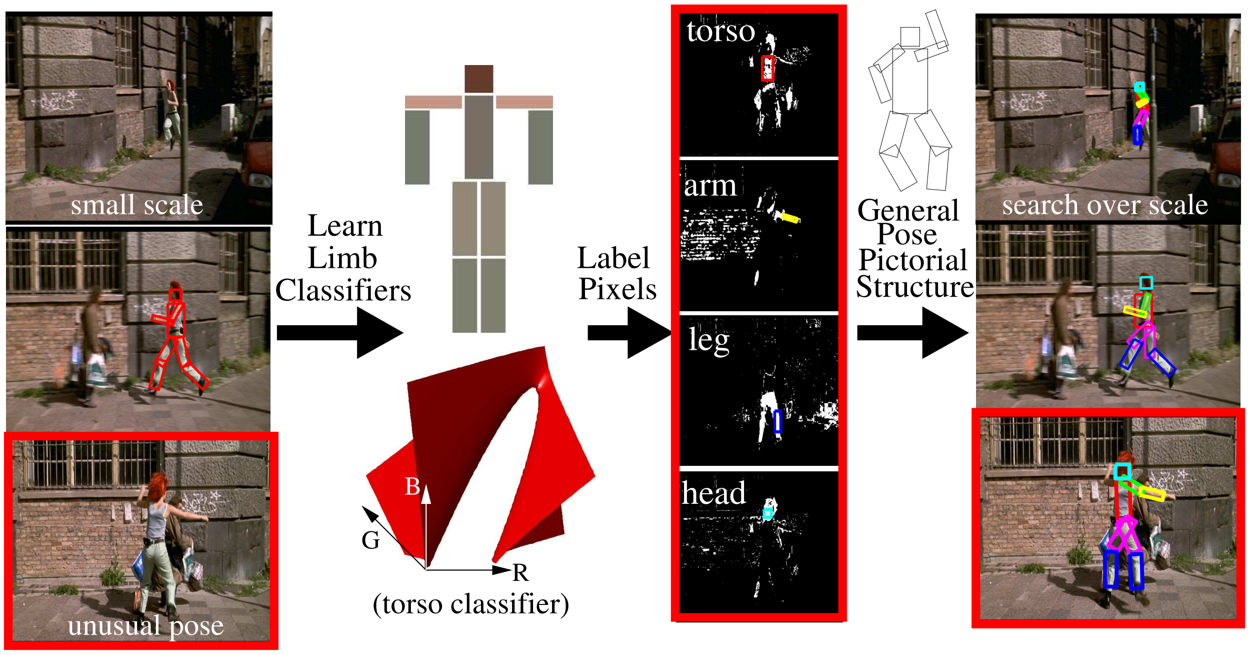


Fig. 12. An overview of our approach; given a video sequence, we run a single-scale walking pose detector on each frame. Our detector fails on the small scale figure and the on a-typical pose, but correctly detects the walking pose (**left**). Given the estimated limb positions from that detection, we learn a quadratic logistic regression classifier for each limb in RGB space, using the masked limb pixels as positives and all non-person pixels as negatives. In the **middle left**, we show the learned decision boundary for the torso and crudely visualize the remaining limb classifiers with a Gaussian fit to the positive pixels. Note that the visual models appear to be poor; many models look like the background because some of the limb pixels happen to be in shadow. The classifiers are successful precisely because they learn to ignore these pixels (since they do not help discriminate between positive and negative examples). We then run the classifiers on *all* frames from a sequence to obtain limb masks on the **middle right** (we show pixels from the third frame classified as torso, lower arm, lower leg, and head). We then search these masks for candidate limbs arranged in a pictorial structure [34], searching over general pose deformations at multiple scales. This yields the recovered configurations on the **right**. We show additional frames in Figure 23

define the initial cost of each sample to be its negative-log probability. To find configurations where the left and right legs look similar, we add the disparity in leg appearance (as measured by the $\mathcal{L}_2$ distance between color histograms) to the cost of each sample. To force left and right legs to be far apart, we discard samples where leg endpoints are within a distance d of each other, where d is the width of the torso. We finally keep the sample with the lowest cost.

Segmentation score: Given an image with a laterally walking person, the procedure above tends to correctly localize the limbs of the figure. But it does not perform well as a people detector; it fires happily on textured regions. We add a region-based cue to the detection score. We can interpret the recovered figure as a proposed segmentation of the image (into person/non-

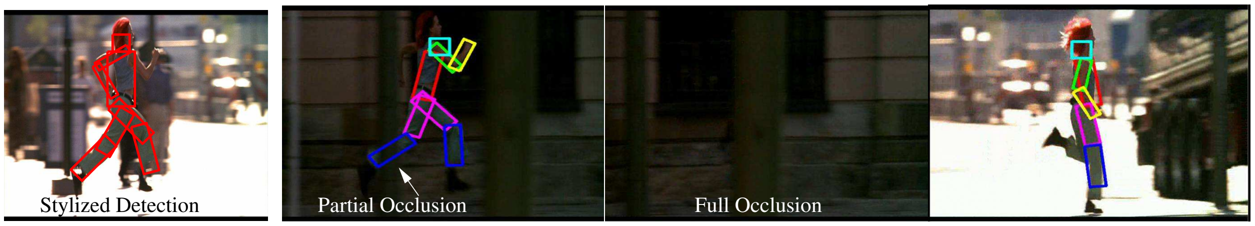


Fig. 13. We show tracking results for a sequence with large changes in illumination. On the **left**, we show the frame on which our walking pose detector fired. Our system automatically learns discriminative limb appearance models from that *single* frame, and uses those models to track the figure when the background changes (**right**). This suggests that our logistic regression appearance model is quite generalizable. Note that since our system tracks by detection, it can track through partial and full occlusion.

person pixels), and directly evaluate the segmentation [31] as the final detection cost. Rather than use a standard segmentation measure, we adopt a simpler approach.

We build classifiers (in RGB space) for each limb, as described in Section V-B. For each limb classifier, we create a test pool of limb pixels (from inside the corresponding limb mask) and background pixels (from identically-sized rectangles flanking both sides of the true limb). We then classify the all test pixels, and define the cost of the segmentation to be the total number of misclassified pixels. Note that this strategy would not work if we used classifiers with high Vapnik-Chervonenkis (VC) dimension (a nearest neighbor classifier always returns 0 errors when training and testing on the same data [45]). Restricting ourselves to a near-linear classifier (such as quadratic logistic regression) seems to address this issue. We threshold this final segmentation score to obtain good stylized-pose detections.

B. *Discriminative appearance models*

Since our person detector localizes a complete person in a single frame, we know both the person pixels *and* the non-person pixels. This suggests we can build a discriminative model of appearance. We assume each limb is (more or less) constant colored, and train a quadratic logistic regression classifier. One could also use more expressive classifiers (such as SVMs) with complex decision boundaries, but we did not find this necessary. We use all pixels inside the estimated limb rectangle as positives, and use all non-person pixels (not inside any limb mask) as negatives. Our appearance model for each limb is a quadratic surface that splits RGB space into limb/non-limb pixels (Figure 12). Recall our set of limbs are the head, torso, upper/lower arm, and left/right upper/lower leg. We fit one model for the upper leg using examples from both left and right limbs (and similarly for the lower leg).

Our classifiers learn some illumination invariance (since illumination tends to be poor feature with which to classify the training examples; see Figure 12). This suggests our models naturally cope with illumination changes (Figure 13). Alternatively, one could build an explicit temporal illumination variable as in [46].

VI. Tracking by model detection

Given either model-building method (from Section IV or V), we now have a representation of the appearance of each part C^i. Since people tend to be symmetric in appearance, we maintain a single appearance for left and right limbs. The representation may be *generative* (a template patch) or *discriminative* (a classifier). To score a template, we evaluate candidate patches under the (RGB-histogram) gaussian model fit to each limb cluster from Section IV-B. For speed up, we only evaluate patches with a good SSD score (which can be computed efficiently by a convolution with the template patch). We evaluate the (log) likelihood of a classifier by summing up the number of misclassified pixels in a local image region (by convolving the limb masks in Fig. 12 with the rectangle filters of Fig 4). We weight the ($\mathcal{L}_1$-normalized) filter response by an empirically-determined value of $1/3$ for our stylized-pose experiments.

Since our learned part models C^i provide a good description of a person's appearance, we can localize a person quite well just by looking at a single frame. As such, we use a single frame pictorial structure (using the known part appearances) to detect each person in each frame.

Multiple Scales: To detect pictorial structures at multiple scales, our system searches over an image pyramid. It selects the largest scale at which a person was detected. This means that although both our model-building algorithms operate at a single scale, our tracker can still track through scale changes. For computational speed-up, we avoid this step unless we know a sequence contains scale changes. We find that searching over scale does not degrade performance due to the quality of our limb models (see Figure 23).

Occlusion: One practical difficulty of tracking people is dealing with self-occlusion; poses where we see both arms and legs are quite rare. Rather than use a formal occlusion model, we found the following procedure to work well. We draw 1000 samples from the posterior of P$(P^{1:N}|I, C^{1:N})$ for a *single arm, single leg* pictorial structure model. We use the efficient

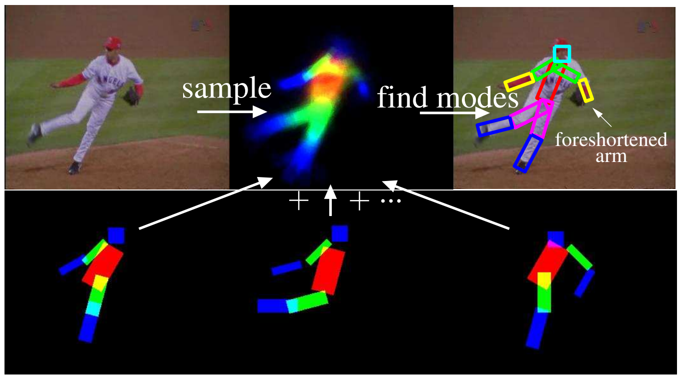


Fig. 14. We detect people by sampling from a one-leg, one-arm pictorial structure P$(P^{1:N}|C^{1:N}, I)$. For the image on the **top left**, we can visualize the posterior (**top center**) by superimposing the samples (**bottom**). We find individual legs and arms by finding modes in the posterior; in images where only one arm is visible, we will find only one arm mode. Our mode-finding algorithm smooths the posterior. When we represent the samples in an appropriate pose space, smoothing allowing us to recover a foreshortened arm even when all sampled arms are too long (**top right**).

method of Felzenszwalb and Huttenlocher [34]. The samples tend to lie on actual arms and legs because of the quality of our limbs masks. We can visualize this posterior by rendering the samples on top of one another (Figure 14). Interestingly, we see two distinct legs in this map; they correspond to different *modes* in the posterior. The *uncertainty* in the matches captures the presence of two arms and two legs. We can explicitly find the modes with the mean shift algorithm [42]. Recall this algorithm performs gradient ascent from an initial starting point. We represent each sample pose as a vector of 2D limb endpoints. We start from the sample with the highest posterior value (we typically need a few mean shift iterations to reach a mode). We remove those samples whose legs and arms overlap, and repeat to find a second mode (only keeping the new mode if its above some threshold). This procedure works well when both limbs are well separated, but suffers from missed detections when limbs partially overlap. In practice, one can often recover the missing limb by temporal smoothing and reasoning about aspect [3]; if only one leg is found, there is a good chance that the other is partially occluded by it.

Spatial Smoothing: The above procedure has a neat side affect; it spatially smooths the posterior function P$(P^{1:N}|I, C^{1:N})$. The amount of smoothing is proportional to the bandwidth of the mean shift procedure. We found a "smoothed mode" pose to be better than a direct MAP estimate in two regards; 1) the smoothed pose tends to be stable since nearby poses also have high posterior values and 2) the smoothed pose contains "sub-pixel" accuracy since it is a local average. Note that this is sub-pixel in the pose space. Recall our poses are now represented as a vector of 2D joint positions. All poses contain arms of equal length, but by averaging joint positions we might obtain arms of different lengths. Interestingly, this averaging often captures the foreshortening of a limb (see Figure 14). Felzenszwalb and Huttenlocher [34] explicitly search over foreshortenings when detecting their articulated models; we avoid the explicit search (making detection an order of magnitude faster) but appear to get the same qualitative results.

Temporal Smoothing: Independently estimating the pose at each frame has its obvious drawbacks; the final track will look jittery. One can produce a smooth track by feeding the pose posterior at each frame (as represented by the 1000 samples) into a formal motion model. We perform local smoothing at a given frame by adding all samples from the previous and next frame when performing our mode-finding procedure. Our final pose estimate will then be a weighted average of nearby poses from nearby frames. Note our procedure does *not* give us left/right correspondence over time. In each frame of our recovered tracks, there is one bit of ambiguity in the left/right labels of the arms and legs. One can resolve much of this ambiguity by searching over the bit when performing motion analysis [3].

Multiple People: In general, we must account for multiple people in a video. The clustering procedure from Section IV naturally handles multiple people with multiple clusters. Given a set of walking-pose detections from Section V, we need to automatically establish the number of different people that are actually present. For each detection, we learn a *generative* appearance model (by fitting a Gaussian in RGB space for each limb mask). This returns a vector of RGB values. We cluster these vectors to obtain sets of people models with similar appearance, again using the mean shift procedure. After obtaining clusters of similar looking people, we use positive and negative examples from across the cluster when training the logistic regression for each limb appearance. We then use these people models as described in the next two paragraphs.

Multiple instances: If a video has multiple people that look similar, our algorithms might only learn a single set of part models C^i (consider a video of a soccer team). In this case, when visualizing the posterior of the pictorial structure, we will see many modes corresponding to different instances. We use the same mode finding procedure to find a set of unique detections.

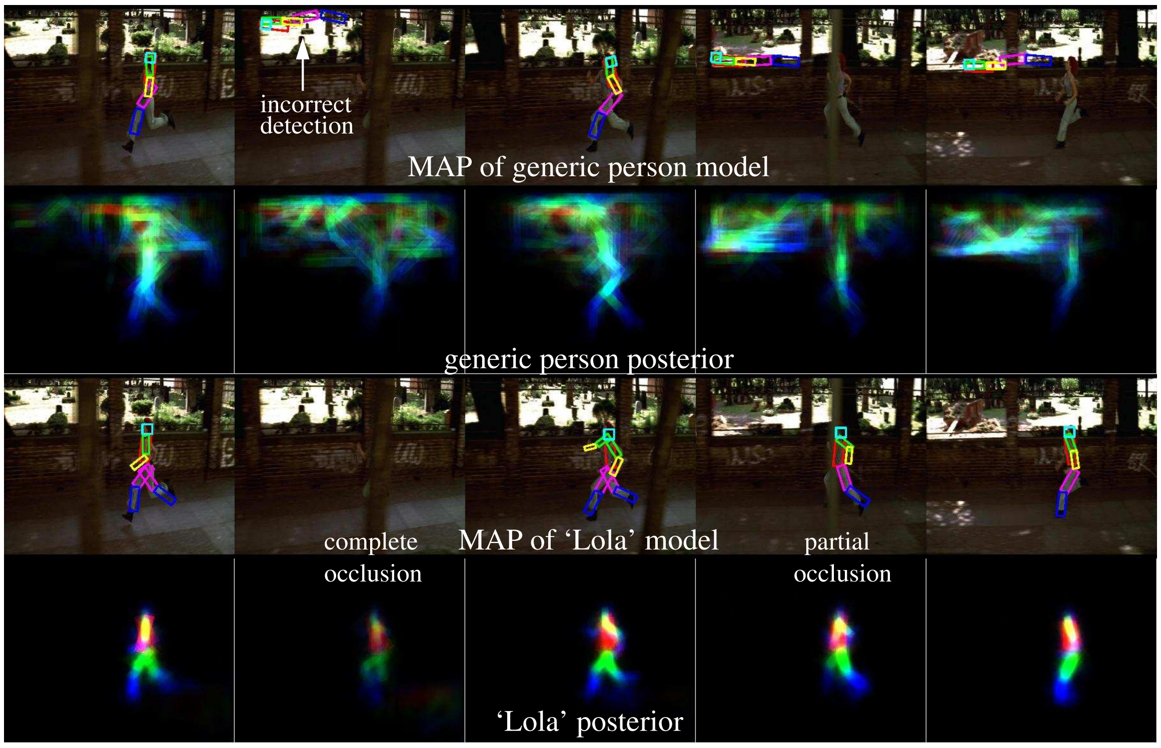
In general, we will have multiple appearance models, each possibly instanced multiple times. For each model, we independently find all instances of it in a frame. Many models will compete to explain the same or overlapping image regions. We use a simple greedy assignment; we first assign the best-scoring instance to the image pixels it covers. For all the remaining instances that do not overlap, we find the best-scoring one, assign it, and repeat. This greedy strategy works best when people are well separated in an image (Figure 24).

VII. Experimental results

Scoring: Typically, a primary criterion for evaluating a tracker is its expected time-to-failure. This pre-supposes that all trackers eventually fail — we argue this is not true. If one tracks a face by running a face detector on each frame, one can track for essentially *infinitely* long [41]. In this case, the quality of the track can measured by the quality of the face detector, in terms of detection rates. We extend this analysis to the detection of individual body parts. We define a part to be correctly localized when the majority of pixels covered by the estimated part have the correct labeling, an admittedly generous test. The accuracy required for a potential application such as activity recognition is not well understood; as such, we use the overlap criteria as a proxy. For sequences without ground truth, we manually label a random set of 100 frames. To avoid the ambiguity in scoring partial occlusions, we score only the first lower-arm and lower-leg found for a given person detection. If the torso and lower limb estimates are accurate, upper limbs are more or less constrained to be correct by the kinematic model.

Do better models help? The fundamental premise of this work is that tracking is easier with an instance-specific person model, rather than a generic one. A natural question is: how well one could do simply with a generic model? We examine this in Figure 15. We construct two pictorial structures using *identical* code except for the part appearance model P$(I|P^i, C^i)$. We use edge templates for the generic model, and we use a color classifier for the 'Lola' model (trained on a stylized detection). Looking at the posterior map, the 'Lola' model performs much better at *data association*; it readily ignores most of the background clutter. Background clutter confuses the generic model, causing spurious modes in its posterior. These modes also suggest why propagating a dynamic model is hard; it must maintain uncertainty across all of these modes. The MAP estimates of the 'Lola' model also produce better tracks.

Computation: Both our tracking systems operate in two stages. In the first stage, pictorial structure models are built for (possibly multiple) people in a sequence. Both systems build models by batch-processing a collection of frames. Once the models are built, the second stage proceeds in an on-line fashion. Each model is detected as a new frame arrives (with a one-frame delay for the local smoothing described in Section VI). Both stages require 7-10 seconds of



% of frames correctly localized

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

Fig. 15. Tracking people is easier with an instance-specific model as opposed to a generic model. In the **top** 2 rows, we show detections of a pictorial structure where parts are modeled with edge templates. We show both the MAP pose and visualize the entire posterior using the method of Figure 14. Note that the generic edge model is confused by the texture in the background, as evident by the bumpy posterior map. In the **bottom** 2 rows, we show results using a 'Lola' model where part appearances are learned from a stylized detection (Section V). This model does a much better job of *data association*; it eliminates most of the background pixels. We can quantify this by looking at the percentage of frames where limbs are accurately localized (the **table**). We score only the first lower-arm and lower-leg found by the procedure from Section VI. We define a part to be correctly localized when the majority of pixels covered by the estimated part have the correct labeling. Note that because we track by detection, our system can recover from partial and complete occlusions.

processing per frame in matlab. We have implemented a real-time version of our stylized-pose system (with various heuristics for faster model-building and online detection).

A. Building models by clustering

We tested our clustering algorithm on four different sequences. "Jumping Jacks" and "Walk" were both taken indoors with motion-captured ground-truth, while "Street Pass" and "Weave

% of frames correctly localized (clustering)

Sequence	Torso	Arm	Leg
J. Jacks	94.6	87.4	91.8
Walk	99.5	84.3	68.2
Street Pass	91.2	57.4	38.7
Weave Run	90.3	21.2	61.8

TABLE I. Localization performance for various sequences, using the conventions of Fig. 15. Our torso rates are quite good, while rates for limbs suffer in the challenging outdoor sequences.

Run" were both taken outdoors without ground truth. We clustered an initial subset of the total frames; 75/100, 150/288, 200/380, and 150/300 frames respectively (the fractions were chosen arbitrarily). In general, we localize torsos quite well but limbs are hard, particularly in the challenging outdoor sequences (Table I).

Self-starting: None of these tracks were hand initialized. However, we do optimize thresholds for the segment detectors and the bandwidth for the mean-shift procedure within the ranges defined in Section IV. More sophisticated segment detection and clustering algorithms may eliminate the need for tweaking.

Multiple activities: In Figure 16, we see frames from the two indoor sequences; "Jumping Jacks" and "Walk." In both left rows, we show the original edge-based candidates that clustered together. When limbs are close to the body or surrounded by a weak-contrast background, few candidates are found. By building an appearance model (using the surrounding frames where candidates *are* detected), we now can track using the learned appearance. Because we track by detection without a strong motion model, we can track both activities with the same system.

Lack of background subtraction: In Figure 17, we show frames from a sequence that contains a moving background. We still learn accurate appearance models, although varying lighting conditions often result in poor matches (as the poor localization results for arms and legs in Table I). By using a metric more robust to lighting changes, the appearance models learned in the clustering and the segments found may be more accurate. Alternatively, one might add explicit illumination variables to C^i to deal with temporal changes in brightness (as in [46]).

Multiple people, recovery from occlusion and error: In Figure 18, we show frames from the "Weave Run" sequence, in which three figures are running in a weave fashion. In the top row, we see two tracks crossing. When the two figures lie on top of each other, we correctly

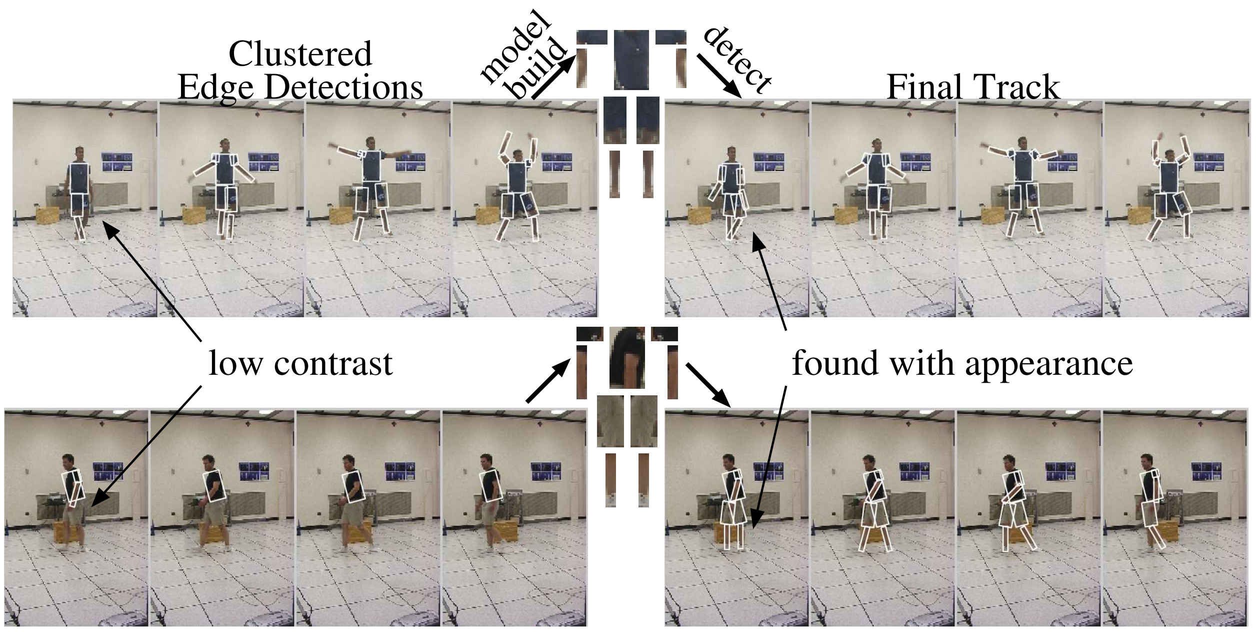


Fig. 16. Self-starting tracker on “Jumping Jacks” (**top**) and “Walk” (**bottom**) sequences. In both **left** rows, we show the original edge-based candidates that clustered together. When limbs are close to the body or surrounded by a weak-contrast background, few candidates are found. By building an appearance model (using the surrounding frames where candidates *are* detected), we now can track using the learned appearance (**right**).

disambiguate who is in front, and furthermore, recover the interrupted track of the occluded figure. In the bottom row, a track finds a false arm in the background but later recovers. We also see a new track being born, increasing the count of tracked people to three.

Number of frames to cluster: For long sequences, we only cluster the first K frames. A natural question is: how does K affect the final performance? We evaluate localization performance versus K in Figure 19 for the “Walk” sequence. We also show results for models learned with part detectors augmented with a skin classifier. Both detectors in Figure 19 happen to perform well for small K since our subject is initially against a uncluttered background. As we increase K and our subject walks into the room, our edge detectors pick up extraneous background candidates which cluster into poor appearance models. However, both perform well as K is increased sufficiently. This result suggests that high-level clustering can compensate for poor low-level detection, given we cluster over enough frames. Also note that performance does not change if we evaluate the detectors on the initial set of K training frames or the entire sequence - this suggests our learned appearance models generalize well.

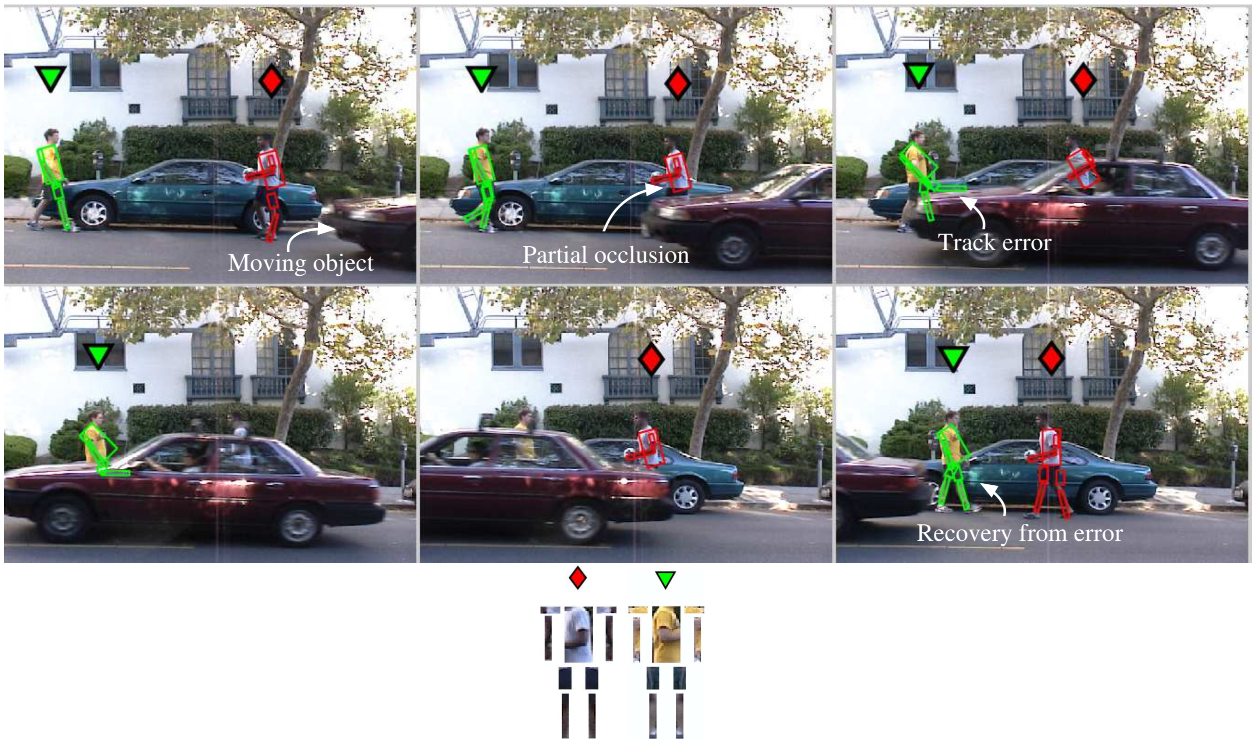


Fig. 17. Self-starting tracker on “Street Pass” sequence containing multiple moving objects. The learned appearance templates are shown **below** select frames from the track. We denote individual tracks by a token displayed above the figure. The tracker successfully learns the correct number of appearance models, and does not mistake the moving car for a new person. We are also able to recover from partial and complete occlusion, as well as from errors in configuration (which drifting appearance models would typically fail on).

B. Building models with a stylized detector

Note that our stylized pose system is straightforward to parallelize because it operates on individual frames. As such, we could implement it on our cluster, allowing us to test our system on hundreds of thousands of frames. Our dataset includes the feature length film “Run Lola Run”, an hour of footage of a local park, and long sequences of legacy sports footage, including Michelle Kwan’s 1998 Olympic performance.

Our automatic tracker consists of two stages. The system first runs a stylized pose detector in each frame of a sequence to find select frames from which to build discriminative appearance models (Section V). It then tracks by detecting the learned appearance models in all frames (Section VI). We evaluate each component separately.

Automatic Initialization: We stress that all these tracks were obtained completely automatically with no manual intervention; the same program with identical parameters was used in

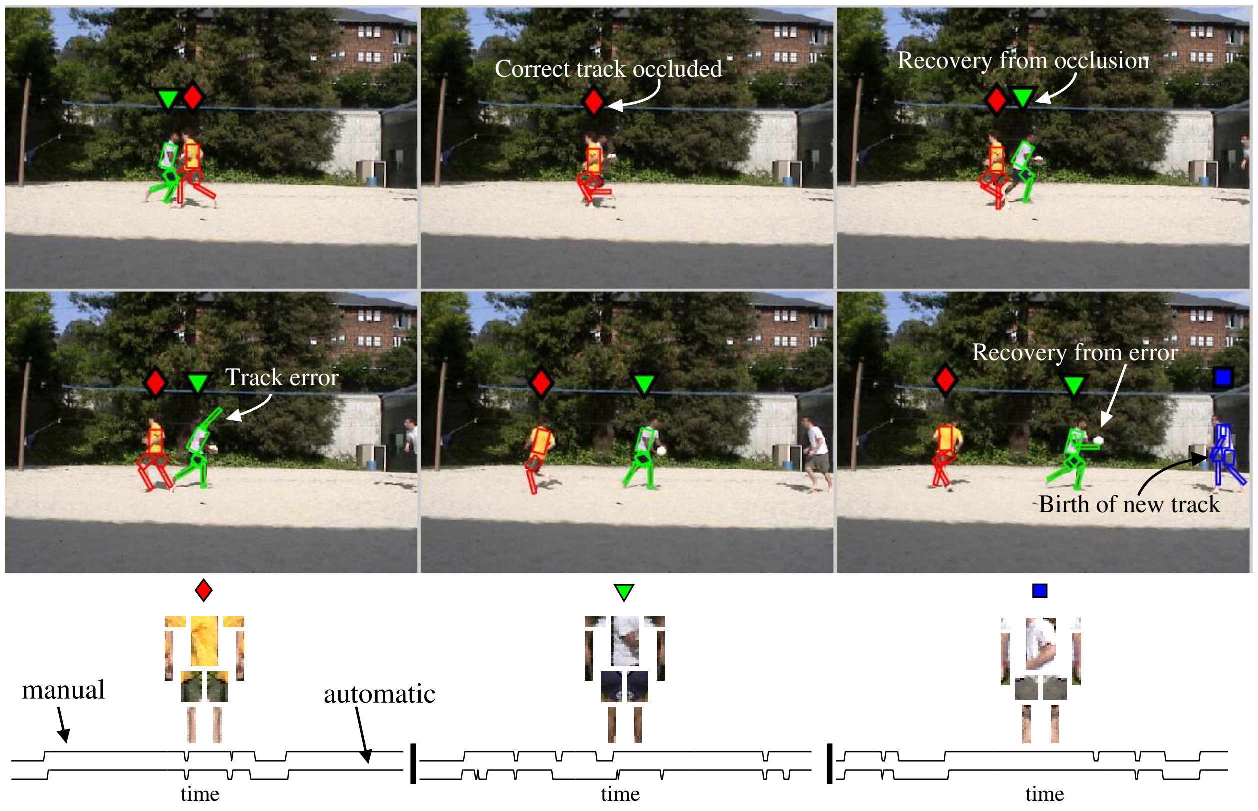


Fig. 18. Self-starting tracker on “Weave Run”. We show a subset of frames illustrating one figure passing another **above** the set of learned appearance templates. The correct figure is occluded and the correct track is recovered once it reappears. An earlier incorrect arm estimate is also fixed (this would prove difficult assuming a drifting appearance model). In the final frame, a new track is born, increasing the count of found people to three. **Below** each figure we show detection signals, as a function of time, specifying when it is in view. The manual and automatic signal extracted from the tracker agree – the tracker tends to detect figures when they are in view and misses them when they are not.

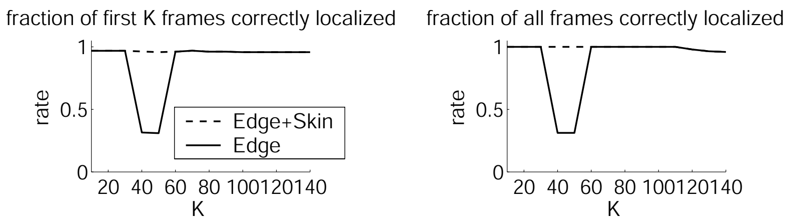


Fig. 19. How do our generic part detectors affect performance? We plot performance for edge versus edge+skin detectors for the 288-frame “Walk” sequence. We vary the size of the initial K frames used to cluster and learn an appearance model. Early in the sequence (for small K), the person is fortuitously standing against an uncluttered background, and both detectors learn a good model. As K increases, background clutter leads our edge detectors to construct poor appearance models. For large K, clustering yields working models irrespective of the detectors. On the **left**, we evaluate the learned detectors on the initial K frames. On the **right**, we evaluate the same detectors on the entire sequence. In both cases, performance is nearly identical; appearance models learned from an initial K frames seem to generalize well to whole sequence.

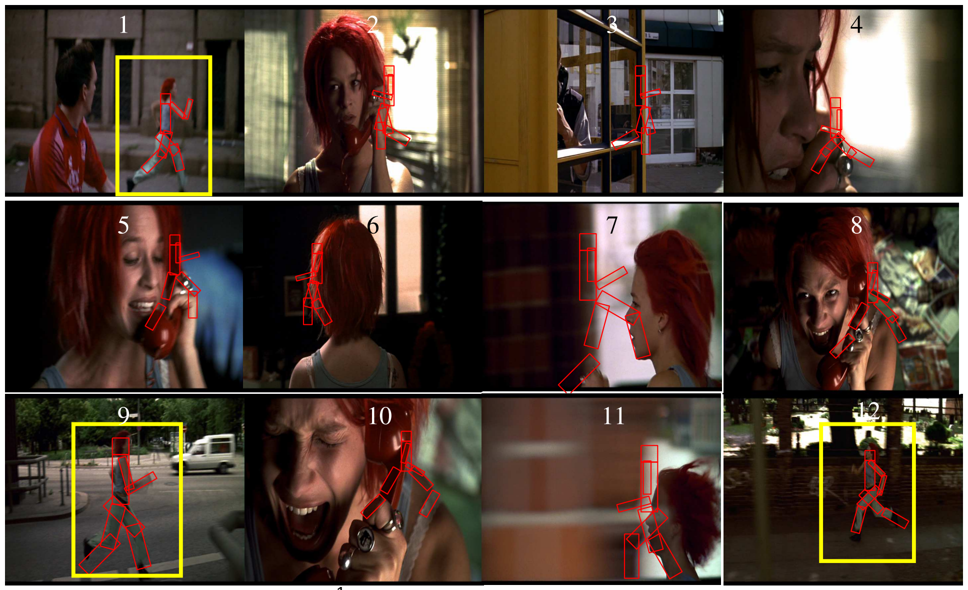
each case. We did scale the video clips so that the figure was of a size expected by the stylized detector.

1) Lateral-Walking Pose Detection: Evaluating our walking pose detector is a difficult task by itself. Labeling false positives is straightforward; if the detector finds a person in the background, that is incorrect. But labeling missed detections is difficult because our detector is not trying to detect all people; only people in certain configurations.

In the case of “Run Lola Run”, we can exploit *shots* (sequences where the camera is filming continuously) in our evaluation. We label each shot (as determined by a histogram-based shot detector) as containing a full-body figure or not. We define the score of a shot to be the best score from our walking detector on its set of frames; the implicit assumption is that if a shot contains a person, our walking pose detector will fire at some point. In Figure 20, we show precision-recall curves for the task of detecting shots with full-body figures using our walking detector. Note that we would expect to perform much better if we use a learned appearance model to track throughout a video, and not just in the shot it was found (particularly for “Run Lola Run” since Lola never changes clothes!). Even without exploiting that fact, we still do quite reasonably at high-precision/low-recall regions of the graph, and significantly better than chance.

For the park sequence, there are no natural shots, making recall measurements awkward to define. Since this video contains multiple people (many of which look similar), we cluster the appearance models to obtain a set of different-looking people models, and then use them to search the entire video. In this case, we would like to select a detector threshold for our walking detector where most of the accepted detections are correct and we still accept enough different looking models to capture most people in the video. As such, we plot precision versus number of appearance model clusters spanned by the accepted detections in Figure 21. We do quite reasonably; we can find most of the different looking people in the video while still maintaining about 50% precision. In other words, if our appearance model detection was perfect (Section VII-B.2) and we were willing to deal with 50% of the tracks being junk, we could track all the people in the video.

We look at the ability of our detector to find people performing unusual activities in Figure 22. Perhaps surprisingly, we are still able to find frames where our detector fires. This means our algorithm is capable of tracking long and challenging sports footage, where people are moving



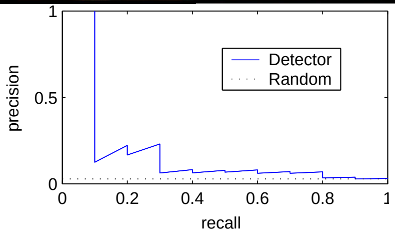
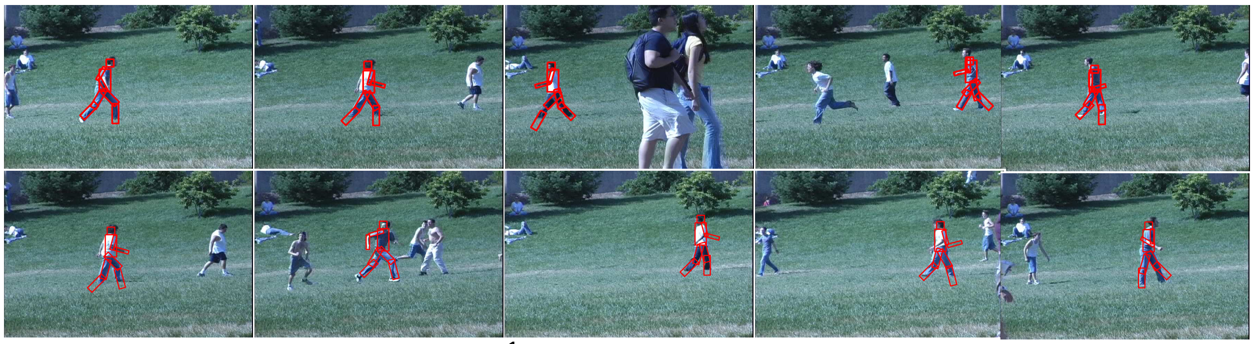


Fig. 20. Evaluating our stylized detector on “Run Lola Run”. We score each shot with the best detection score from our stylized detector. On the **top** show the top 12 shots from the first 30000 frames of the movie. On the **bottom**, we show precision-recall curves for detecting shots with a full-bodied figure. Our detector performs well at high-precision/low-recall (as designed), although in general, detecting people is hard. Many running shots of Lola show her running toward or away from the camera, for which our detector does not fire. If we use the model learned from one shot across the whole video, we would expect to do significantly better (since Lola never changes clothes!).

fast and taking on extreme poses. We show results on a baseball pitch from the 2002 World Series and Michelle Kwan’s medal winning performance from the 1998 Winter Olympics.

2) Appearance model detection: In the second phase of our algorithm, we track by detecting the learned appearance models in each frame. Following the convention from Section VII-A, we evaluate performance by considering localization results for limbs (Table II). The results for our commercial sequences are impressive considering the fast movement and the complexity of the background. Our success is due to the simple fact that people often dress to be visually distinctive. If baseball players wore green uniforms, it would be hard to spot one’s teammate on a playing field. Likewise, filmmakers often want characters to stand out so they can be seen by an audience; a wonderful example is Lola’s red hair. Once our person model learns to look for



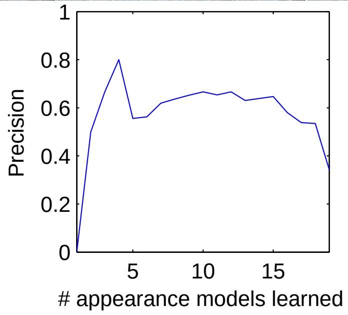


Fig. 21. Evaluating our stylized detector on unscripted outdoor footage. On the **top**, we show the top 10 detections for the first 30000 frames of the video. Even though multiple people are frequently interacting (see Figure 24), our walking pose detector tends to fire on frames where the figures are well separated, since they have a better detection score. Note that we do not use any form of background subtraction. On the **bottom**, we show precision curves. Recall is awkward to define since we are not trying to detect people in every frame; rather we want to fire at least once on different looking people in the sequence. We obtain a set of models of different looking people by clustering the correct detections of our walking detector (which we validate by hand). For a given detector threshold, we can explicitly calculate precision (how many of the reported detections are correct) and the number of different models spanned by correct detections. As we lower the threshold, we span more models.

red hair (Figure 23) or a white uniform (Figure 22), it is extremely unlikely it will loose track (since *by design* there is little white or red in the background).

In our park sequence (Figure 24), we learn multiple appearance models for multiple people. Here, we consider a localization to be correct if it fires on any person in the image; we do not look for consistency of detections from one frame to the next. Since many people in the sequence look like each other, we need additional constraints to pull out individual tracks (such as motion). Our results are also good, though not near the performance we achieve on our commercial sequences. We do quite well at detecting torsos, with about 90% accuracy, while arms are still difficult because they are small and move fast. This data is hard for many reasons; the video is washed out, there are significant shadow effects, there are many small people interacting with each other (Figure 24).

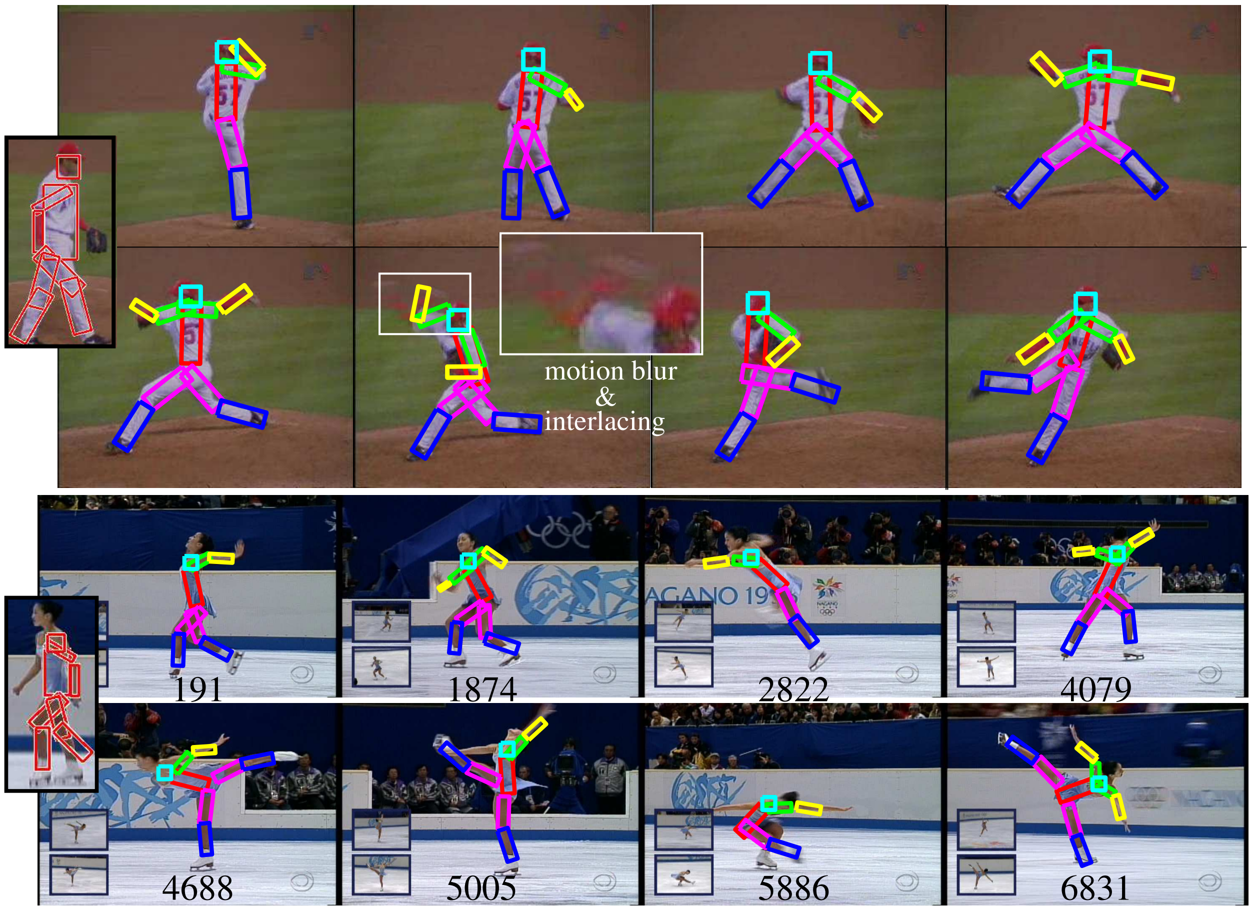


Fig. 22. Our automatic tracker on commercial sports footage with fast and extreme motions. On the **top**, we show results from a 300 frame sequence of a baseball pitch from the 2002 World Series. On the **bottom**, we show results from the *complete* medal-winning performance of Michelle Kwan from the 1998 Winter Olympics. We label frame numbers from the 7600-frame sequence. For each sequence, our 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, our system is able to track through frames with excessive motion blur and interlacing effects (the **center inset**). In the skating sequence, our system is able to track through extreme poses for thousands of frames. This means that our system can track long sequences with extreme poses, complex backgrounds, and fast movement *without manual intervention*.

VIII. Discussion

This paper presents an approach to people-tracking that bootstraps a generic person model into a instance-specific one. The basic framework is to (1) run a generic model on a video, (2) build a specific model from the detections, and then (3) use the specific model to track.

Fundamentally, we build models by looking for *coherence* in the detections; we develop an algorithm in Section IV that operationalizes this notion by clustering candidate body parts. A

% of frames correctly localized (stylized pose)

Sequence	Torso	Arm	Leg
Baseball	98.4	93.75	95.3
Skating	99.2	77.5	97.6
Lola	95.6	89.8	92.6
Park	79.1	29.2	58.3
Walk	99.3	99.3	98.0

TABLE II. We evaluate our appearance model detection by calculating how often individual limbs are correctly localized, using the same conventions as Figure 15. For "Run Lola Run", we average performance over three shots on which the stylized detector correctly fired (a separate appearance model was learned for each shot; see Figures 13,15,23). Our torso and leg detection tends to be quite good, with arms being harder to track because they are small. The outdoor "Park" sequence proves difficult because of crowd scenes and lighting changes. We compare results with our clustering approach on the 'Walk' sequence from Table I. The stylized-pose system performs much better than the clustering algorithm because of its discriminative appearance model. The limb classifiers can exploit the structure in the background. Our performance is impressive given the difficulty and length of sequences we test on; our commercial sequences contain extremely fast movement and non-stationary complex backgrounds, and are tracked *completely automatically*.

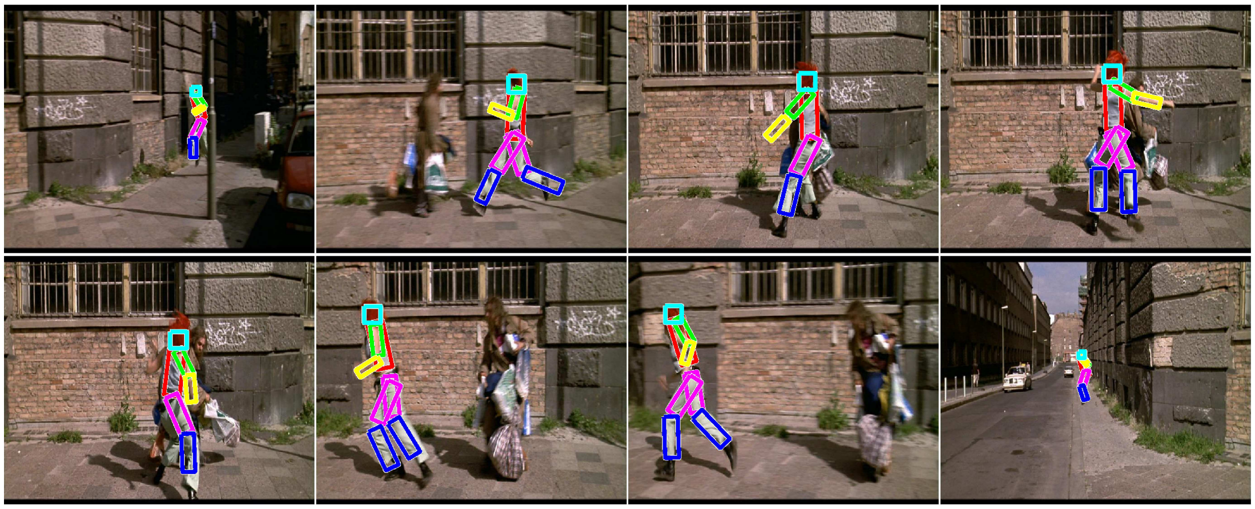


Fig. 23. A sequence from "Run Lola Run" of Lola running around corner and bumping into a character while undergoing extreme scale changes. Note that the character is wearing bulky clothing and so our person detector has no hope of finding him. Our initial walking detector is run at a single scale; once we learn an appearance model (as shown in Figure 12), we track over multiple scales by searching an image pyramid at each frame.

quite useful observation is that this initial detection can be done *opportunistically*; we describe an algorithm in Section V that looks for stylized poses that are reliably detected and easy to build appearance from. One could also look for stylized motions over short frames; such a

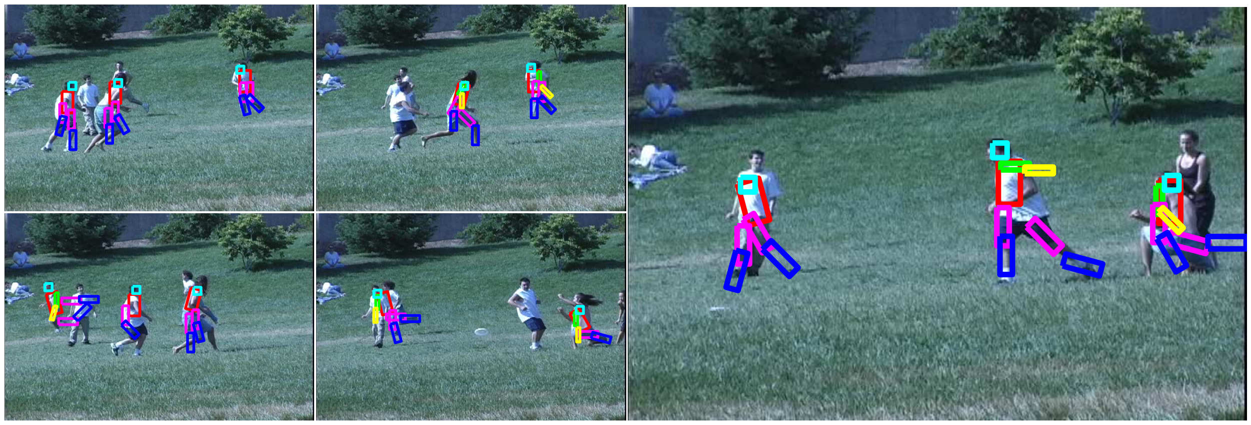


Fig. 24. Automatic tracking of a sequence from the 30000 frame park sequence. This sequence is harder than our commercial footage; we have poor color resolution, many small figures are in shadow and many are occluding each other while performing fast motions. We still obtain good detections and reasonable localization of arms and legs. Since many of the learned models look similar, we do not try to disambiguate instances from frame to frame. One might do that using motion constraints.

detector might perform better since it pools information from across frames. Another practical observation is that *discriminative* appearance models learned from a few frames can discriminate an object in other frames. Discriminative features for tracking are not new [47], but by learning them from select frames in which we trust our detections, we make them quite powerful.

Comparison of model-building algorithms: We find the two model-building algorithms complementary. The stylized-pose system appears more robust, since a single set of parameters worked for all the sequences shown. If we can observe people for a long time, or if we expect them to behave predictably, detecting stylized poses is likely the better approach. These scenarios might be realistic for public areas monitored by security cameras or athletic performances. However, if we observe a person moving for a short time in a contrived manner, clustering a pool of candidate parts may work better than a stylized detector. A simple example of this is our jumping jack sequence (Figure 16); here, the clustering algorithm easily learns the body appearance, but stylized pose algorithm fails because the figure never walks laterally. This suggests another improvement for our stylized system; increase the set of stylized poses to cover different aspects.

We use the 'Walk' sequence from Figure 16 to compare our clustering and stylized-pose systems (Table I versus Table II). Both methods succeed in building accurate models; the frames with high-contrast backgrounds provides good part detections for the clustering system, while the frequent lateral-walking poses trigger the stylized-pose system. The difference in performance is mainly due to the representation of appearance. The stylized-pose system performs much better because of its discriminative appearance model. The limb classifiers can exploit the structure in the background. This structure seems to be present even when backgrounds apparently change (as suggested by our commercial sequences).

REFERENCES

[1] D. Ramanan, "Tracking people and recognizing their activities," Ph.D. dissertation, U.C. Berkeley, 2005.

[2] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding: CVIU*, vol. 73, no. 1, pp. 82–98, 1999.

[3] D. Ramanan and D. A. Forsyth, "Automatic annotation of everyday movements," in *NIPS*, 2003.

[4] D. Hogg, "Model based vision: a program to see a walking person," *Image and Vision Computing*, vol. 1, no. 1, pp. 5–20, 1983.

[5] S. Ioffe and D. A. Forsyth, "Human tracking with mixtures of trees," in *ICCV*, 2001.

[6] J. O'Rourke and N. Badler, "Model-based image analysis of human motion using constraint propagation," *IEEE T. Pattern Analysis and Machine Intelligence*, vol. 2, pp. 522–546, 1980.

[7] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *CVPR*, 1998, pp. 8–15.

[8] D. Gavrila and L. Davis, "3d model-based tracking of humans in action: a multi-view approach," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 1996, pp. 73–80.

[9] K. Rohr, "Incremental recognition of pedestrians from image sequences," in *CVPR*, 1993, pp. 9–13.

[10] 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.

[11] A. Blake and M. Isard, "Condensation - conditional density propagation for visual tracking," *Int. J. Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[12] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2000, pp. II:126–133.

[13] K. Toyama and A. Blake, "Probabilistic tracking with exemplars in a metric space," *Int. J. Computer Vision*, vol. 48, no. 1, pp. 9–19, 2002.

[14] C. Sminchisescu and B. Triggs, "Covariance scaled sampling for monocular 3d body tracking," in *CVPR*, 2001.

[15] L. Sigal, S. Bhatia, S. Roth, M.Black, and M. Isard, "Tracking loose-limbed people," in *CVPR*, 2004.

[16] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *ECCV*, 2000.

[17] V. Pavlovic, J. Rehg, T.-J. Cham, and K. Murphy, "A dynamic bayesian network approach to figure tracking using learned dynamic models," in *Int. Conf. on Computer Vision*, 1999, pp. 94–101.

[18] A. Agarwal and B. Triggs, "Tracking articulated motion with piecewise learned dynamic models," in *ECCV*, 2004.

[19] G. Mori and J. Malik, "Estimating human body configurations using shape context matching," in *ECCV*, 2002.

[20] J. Sullivan and S. Carlsson, "Recognizing and tracking human action," in *European Conference on Computer Vision*, 2002.

[21] D. M. Gavrila, "Pedestrian detection from a moving vehicle," in *ECCV*, 2000, pp. 37–49.

[22] Y. Song, X. Feng, and P. Perona, "Towards detection of human motion," in *CVPR*, 2000, pp. 810–17.

[23] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *ICCV*, 2003.

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

[25] J. Rehg and T. Kanade, "Model-based tracking of self-occluding articulated objects," in *ICCV-95*, 1995, pp. 612–617.

[26] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," *PAMI*, vol. 25, no. 10, pp. 1296–1311, 2003.

[27] M.-H. Y. Gang Hua and Y. Wu, "Learning to estimate human pose with data driven belief propagation," in *CVPR*, 2005.

[28] M. Lee and I. Cohen, "Proposal maps driven mcmc for estimating human body pose in static images," in *CVPR*, 2004.

[29] T. Roberts, S. J. McKenna, and I. W. Ricketts, "Human pose estimation using learnt probabilistic region similarities and partial configurations," in *ECCV*, 2004.

[30] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse picture of people," in *ECCV*, 2002.

[31] G. Mori, X. Ren, A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *CVPR*, 2004.

[32] S. Ioffe and D. Forsyth, "Probabilistic methods for finding people," *Int. J. Computer Vision*, 2001.

[33] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computer*, vol. 1, no. 22, pp. 67–92, January 1973.

[34] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Int. J. Computer Vision*, vol. 61, no. 1, January 2005.

[35] D. Ramanan and D. A. Forsyth, "Finding and tracking people from the bottom up," in *CVPR*, 2003.

[36] D. Ramanan, D. Forsyth, and A. Zisserman, "Strike a pose: Tracking people by finding stylized poses," in *CVPR*, June 2005.

[37] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *International Joint Conference on Artificial Intelligence*, August 2001.

[38] M. Isard, "Pampas: Real-valued graphical models for computer vision," in *CVPR*, 2003.

[39] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky, "Nonparametric belief propagation," in *CVPR*, 2003.

[40] W. T. Freeman and Y. Weiss, "On the fixed points of the max-product algorithm," *IEEE T. Information Threory*, 2000.

[41] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[42] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE T. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[43] J. Deutscher, A. Davison, and I. Reid, "Automatic partitioning of high dimensional search spaces associated with articulated body motion capture," in *CVPR*, 2001.

[44] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *CVPR*, 2003.

[45] V. N. Vapnik, *Statistical Learning Theory*. John Wiley and Sons, 1998.

[46] M. Kumar, P. Torr, and A. Zisserman, "Learning layered pictorial structures from video," in *Indian Conference on Vision, Graphics and Image Processing*, 2004.

[47] R. Collins and Y. Liu, "On-line selection of discriminitive tracking features," in *ICCV*, 2003.