Motion Analysis by Synthesis: Automatically Annotating Activities in Video

Deva Ramanan, *Member, IEEE,* D.A. Forsyth, *Member, IEEE*

D. Ramanan is with the Toyota Technological Institute, Chicago, IL 60637. D.A. Forsyth is with the Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL 61801.

E-mail:ramanan@tti-c.org,daf@cs.uiuc.edu

Abstract

An important vision problem is to automatically describe what people are doing a sequence of video. This problem is difficult for many reasons. First, tracking the articulations of people is difficult because arms and legs are small and can move quite fast. Secondly, one must sew the estimated poses from each frame together into a coherent motion path; this is difficult because people can move in unpredictable ways. Finally, one must describe the motion with some activity annotation. This appears difficult for everyday motion since there may not be a canonical vocabulary of motions.

We describe a fully automatic system that labels the activities of multiple people in a video sequence. The system decouples the choice of annotation vocabulary from the analysis procedure, allowing for easy revision of the vocabulary. The system first uses a pictorial structure model to independently detect 2D poses in each frame. The system then synthesizes 3D motion clips that looks like the 2D motions by matching poses to a stored library of motion capture data. The motion capture data is labeled off-line with a class structure that allows for multiple annotations to be composed; one may `walk`, while `wave`ing, for example.

The lack of a canonical vocabulary also makes it difficult to evaluate experimental results. We introduce a mutual information criterion that allows one to evaluate different annotation systems given labeled test footage. We demonstrate and evaluate our system on real sequences of multiple people interacting and commercial shots from a feature-length film.

Index Terms

people tracking, motion capture, surveillance

I. Introduction

Automatically describing what people are doing in a video sequence is task of great practical importance. A reliable solution would open up tremendous possibilities, such as video summary/mining – one could summarize daily activities in public areas. It would also allow for pervasive Human-Computer Interaction (HCI) – gesturing interfaces could enable smart offices and smart homes. Finally, one could analyze human movement for surveillance purposes – from medical analysis of patients to automatic flagging of suspicious behavior in high security areas. Indeed, because of the many potential applications, video-based understanding of people has been a core challenge in the vision community for over 25 years.

Understanding at a coarse scale seems to fairly well understood, but understanding activities that depend on detailed information about the body is still hard. The major difficulties appear to

be that (a) good kinematic tracking is hard; (b) models typically have too many parameters to be learned directly from data; and (c) for much everyday behavior, there isn't a natural taxonomy.

Many approaches to fine-scale activity recognition assume that a kinematic track is given, typically by manual labeling or initialization [1, 2, 3, 4]. We argue that a practical recognition system must deal with realistic errors in kinematic estimates. We demonstrate all results with an existing tracker [5, 46] that is automatic, albeit imperfect.

We focus specifically on everyday behavior. In this case, a fixed vocabulary either doesn't exist, or isn't appropriate. For example, one does not know words for behaviors that appear familiar. One way to deal with this is to work with a notation (for example, Laban notation); but such notations typically work in terms that are difficult to map to visual observables (for example, the weight of a motion). The alternatives are either to develop a canonical vocabulary or to develop an analysis procedure that is somehow vocabulary-independent. We take the third approach in this work.

Evaluating models for everyday behaviors is hard, because it is difficult to obtain a large collection of marked up video (among other things, there isn't a vocabulary in which to mark it up).We also introduce a quantitative measure based on mutual information that is *independent* of vocabulary. We use it to evaluate a large collection of annotated sequences of multiple people interacting as well as commercial footage.

A. Previous Work

There has been a substantial amount of work on recognizing activities from video (for review papers, see [6, 7, 8, 9, 10]).

There seems to be a general consensus that different taxonomies are needed for interpreting events at different time scales. Bobick [8] refers to an atomic event as a *movement*, a sequence of movements as an *activity*, and finally terms large-scale events as *actions*.

Methods for recognizing small-scale movements typically match some video signature to a **template**. If the background is stationary or uncluttered, than one can match low-level spatio-temporal descriptors without explicitly tracking arms and legs [11, 12, 13, 14, 15]. Alternatively, one could explicitly extract body pose, and directly match based on pose estimates [1, 16, 17]. These approaches are more similar to our work.

Recognizing large scale movements probably requires more sophisticated techniques; one approach is that of **temporal logics**. Here actions and events are represented with a set of logic primitives; valid temporal relations are represented by interval bounds (to pick up a cup we must first grab it) [18, 19, 20]. Given a video with low-level event detections, one obtains high level descriptions by forms of constraint propagation [21, 22]. Logic formalisms can be difficult to both author and perform inference with because they require clean input; it is not clear if they will work given missing or noisy video summaries.

By far the most popular approach is to use a HMM, where the hidden state is an activity to be inferred, and observations are image measurements. Models used have tended to be small (for example, one sees three to eight state models in [23, 24, 25]). Yamato *et al.* describe recognizing tennis strokes with HMM's [26]. Wilson and Bobick describe the use of HMM's for recognizing gestures such as pushes [27]. Yang *et al* use HMM's to recognize handwriting gestures [28].

There has been a great deal of interest in models obtained by modifying a basic activity-state HMM. The intention is to improve the expressive power of the model without complicating the processes of learning or inference. Variations include a coupled HMM (CHMM) [23, 24], a layered HMM (LHMM) [29, 30, 31], a parametric HMM (PHMM) [32], an entropic HMM (EHMM) [33], and variable length Markov models (VLMM) [34, 35]. All these HMM variations could be seen as ways of simplifying the process of *model authoring*: i.e., learning the state transition matrix. Fitting a large state space model from data is hard because one needs lots of data.

B. Our approach

We focus on recognizing small-scale activities (*movements*, in Bobick's terms). An immediate concern is what activities to recognize? For limited domains, such as tennis footage, there maybe natural categories – a forehand stroke, a backhand, etc. However, when observing everyday behavior in public areas, it is not so obvious (see Figure 1). In our opinion, in order to capture everyday movements, one must use a vocabulary that is large and expressive. To achieve this, we allow activity labels to be **composed**; one can both `walk` and `wave` simultaneously. With such a representation, we must take care to not allow invalid combinations (e.g., one cannot simultaneously `run walk`).

A large vocabulary complicates matters considerably because learning and inference are now

Fig. 1. What is the correct annotation vocabulary? Most likely, an annotation system will not contain a `'Trying to catch a frisbee between the legs'` term. This makes it difficult to construct a canonical vocabulary for everyday motion. We construct a class structure that allows for different labels to be composed; such a system might label the above motion as `crouch`ing, `reach`ing, and `kick`ing. This creates a large effective vocabulary size which makes direct analysis difficult. We build a system that decouples analysis from the choice of vocabulary by matching poses rather than annotations.

difficult. In particular, estimating the parameters of a large state-space HMM is hard because one needs an exorbitant amount of training data. We *decouple* both learning and inference from the choice of vocabulary by taking an **analysis as synthesis** approach; we recognize activities by synthesizing a motion that looks like a video. We synthesize motion by re-arranging pre-recorded clips from a motion capture database. By pre-labeling clips with activity labels, we synthesize an activity labeling "for free".

Essentially, we replace an activity state-space HMM with a pose state-space HMM. Learning a pose model is easier because poses live in a metric space; this means we can construct transition matrices by looking for poses that are nearby in this space. Such matrices are typically sparse and known as *motion graphs* in the graphics literature [36, 37, 38]. From our HMM perspective, synthesis is equivalent to inferring a sequence of hidden pose states. In practice, one performs synthesis by using graph search algorithms on an underlying motion graph.

Figure 2 shows an overview of our approach to activity recognition. We use 3 core components: annotation, detection, and motion synthesis. Initially, a user labels a collection of 3D motion capture frames with annotations; this can be done with minimal supervision (Section II). Given a new video sequence to annotate, we use a pictorial structure to detect 2D poses of each figure in a sequence [39]. We then synthesize 3D motion sequences which look like the detections by matching them to our annotated motion capture library (Section IV). We finally accept the annotations associated with the synthesized 3D motion sequence.

Our approach of linking video tracks with a motion capture database dates back to at least the

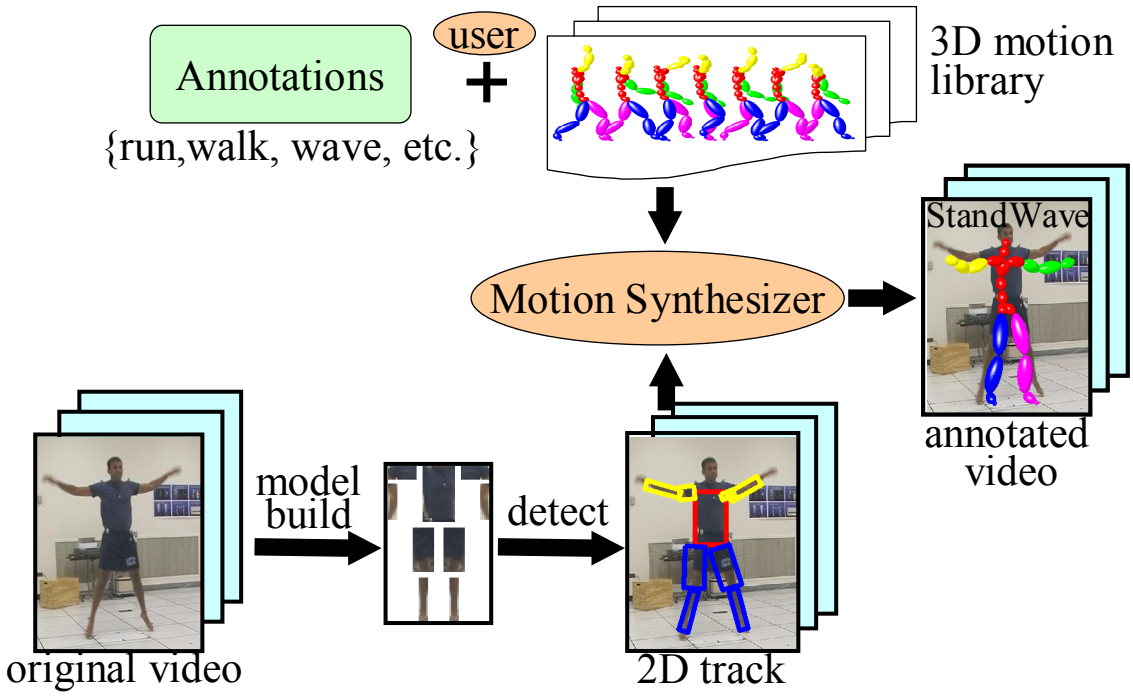

Fig. 2. Our annotation system consists of 3 main components; annotation, detection, and motion synthesis. A **user** initially labels a collection of 3D motion capture frames with annotations. Given a new video sequence to annotate, we track by detecting a pictorial structure model in each frame. We then **synthesize** 3D motion sequences which look like the 2D tracks by lifting tracks to 3D and matching them to our annotated motion capture library. We accept the annotations associated with the synthesized 3D motion sequence as annotations for the underlying video sequence.

work of Sidenbladh *et al* [40]. Similar approaches of synthesizing motions given constraints from video are taken in [38, 41]. Those works use a motion graph to enforce a human smoothness prior; we use a motion graph as a mechanism for explicit *analysis* – obtaining an activity description [42].

II. Obtaining Annotated Data

We have annotated a body of motion data using the system described in [43]. We repeat the process here for convenience.

There is no reason to believe that a canonical annotation vocabulary is available for everyday motion, meaning that the system of annotation should be flexible. In practice, this means that it should be relatively simple for a user to revise the annotations attached to the data set. Annotations should allow for composition as one can `wave` while `walk`ing, for example. We

achieve this by representing each separate term in the vocabulary as a bit in a bit string. Our annotation system attaches a bit string to each frame of motion. Each bit in the string represents annotation with a particular element of the vocabulary, meaning that elements of the vocabulary can be composed arbitrarily.

That said, we still need an initial vocabulary that can be composed/revised as needed. We use a set that is convenient for **synthesizing** motions from a given database; this suggests the vocabulary will be tied to that specific database. We use 7 minutes of football motions, collected for a commercial video game. An independent user decided a set of 13 labels would be useful for synthesizing motions from this collection; `run`, `walk`, `wave`, `jump`, `turn left`, `turn right`, `catch`, `reach`, `carry`, `backwards`, `crouch`, `stand`, and `pick up`. These labels are allowed to occur in any combination: `turn left` while `walk`ing, or `catch` while `jump`ing and `run`ning. This produces an enormous vocabulary of $2^{13} = 8192$ different annotations. Learning a HMM with such a large state space may be difficult; we avoid such difficulties by dealing with poses rather than labels. In practice, many label combinations will never be used; we can't conceive of a motion that should be annotated with both `stand` and `run`. Examining our database, we see that only 46 combinations are ever observed. We must take care to ensure that the final analysis/synthesis algorithm respects these valid combinations (see Section IV-C).

Actual annotation is simplified by using an online learning approach where a user bootstraps a classifier. Initially, a user hand-labels a random subset of frames from the database. One SVM classifier is learned for each element of our vocabulary *independently*; this means one learns a `run/not-run` classifier, a `stand/not-stand` classifier, etc. The classifiers use Gaussian kernels, and use the joint positions for one second of motion centered at the frame being classified as a feature vector. Since the motion is sampled in time, each joint has a discrete 3D trajectory in space for the second of motion centered at the frame. We used a public domain SVM library (`libsvm [44]`). The out of margin cost for the SVM is kept high to force a good fit.

After the initial training, we use the classifiers to label the remaining frames in the database. The user is incrementally presented with newly classified frames, and makes corrections as necessary. The user verified data is then added to the SVM training set and the classifier's decision boundary is re-optimized. It is our experience that after annotating 3-4 example variants of an annotation, the user rarely needs to correct the auto-annotation results. It is remarkable that

such a procedure tends to respect constraints implicit in the user-labellings; the classifiers tend not to label a new motion as both a `run` and a `stand`. This means it is possible to annotate large databases of motions quite rapidly.

III. 2D Tracking by Detection

We represent a human body as a puppet of rectangles of fixed dimension (we find people at different scales by searching an image pyramid). This parts-based model is often called a pictorial structure [39, 45]. Pictorial structures represent the body with a geometric constraints (the left leg connects to the torso) and a local appearance template for each part (the left leg is blue).

Geometric Model: Because we wish to track people performing a variety of activities, we employ weak geometric constraints (parts must be connected and respect reasonable joint angle limits). This is in contrast to many other approaches that use strong priors (typically for walking). In our opinion, tracking should be performed with weak priors. Even though we might expect people to behave predictably most of the time, we especially want to understand *unexpected* poses and movement (e.g., to evaluate suspicious behavior in an airport).

Part Template: We have described methods of learning part models *automatically* from a video in previous work [5, 46]. In [46], we describe a bottom-up system that initially looks for candidate body parts in each frame (using a detuned edge-based part template). The system builds tuned part templates by grouping together candidate detections that look similar and that are arranged in valid geometric layouts. In [5], we describe a system that initially uses a edge-based pictorial structure tuned to find people in stylized poses. From those frames where it fires, the system automatically builds appearance templates for each part. After the appearance templates are learned, people are re-detected in previous and successive frames in unrestricted poses. We assume that a person is symmetric in appearance and so learn a single part appearance template for left/right limbs. For a detailed description, we refer the readers to those works. However, we note that in many situations the part model is given (for say, sports footage where team uniforms are known *a priori*).

Detection: One practical difficulty of tracking people is dealing with self-occlusion; poses where we see both arms and legs are quite rare. We match a *single arm, single leg* pictorial structure to a given frame from a video. This chain-model can be efficiently matched to an

image in a few seconds using the method of Felzenschwalb and Huttenlocher [39]. Fixing the torso at the estimated image position, we search for an additional arm/leg that does not overlap the original limb estimate (only keeping those matches above a threshold). Such a procedure implies that the detected 2D poses will suffer from left/right ambiguities and missed detections of joints.

For simplicity, we assume that each actor in a video looks different. This means we can track a particular person simply by detecting their appearance model at each frame. The resulting 2D track looks jittery because the pose is independently estimated in each frame. As in [5], we low-pass filter the pose estimates (by reporting the average pose in a 3-frame window). This local smoothing also allows us to track through single-frame occlusions. If a video contains multiple people that look similar, detecting them is still straightforward; we instance a single pictorial structure multiple times in each frame. However, tracking multiple similar-looking people probably requires more sophisticated data association algorithms (such as gating [47]).

IV. 3D Motion Synthesis

We can think of the pictorial-structure matching as generating a sequence of 2D stick figures. Intuitively, it might seem that identifying the rough 3D pose is possible by looking at a single 2D skeleton; we define a matching criteria that does this in Section IV-A. However, a 2D stick figure is still ambiguous in many ways; it is difficult to recover the orientation of the torso with respect to the camera, and it is difficult to label the left/right limbs. We resolve these ambiguities in Section IV-B by propagating information throughout a video.

A. Matching by minimizing reprojection error

In this section, we describe an approach for matching 3D poses to 2D poses, assuming a scaled orthographic camera model. We represent each 2D pose as a point cloud of joint positions. We define a joint position at the upper & lower torso, shoulders, elbows, wrists, knees, and ankles (for a total of 12 points). We can similarly represent each pose from our motion capture library as a cloud of 12 corresponding 3D points. Note that this is a subset of the 30 joints in our original motion capture skeleton (see Figure 2). Hence our matching process also recovers *extra* degrees of freedoms that are difficult to directly estimate from video. Computing the matching

3D pose reduces to the well-studied problem of matching a 3D point cloud to a 2D point cloud, where correspondences are known [48, 49].

To incorporate local dynamics (such as velocities and accelerations), we represent all poses by a point cloud of joint positions collected from 15 frame ($\frac{1}{2}$ second) windows. We write the 2D point cloud extracted from the t^{th} frame as m_t, a $2X(12 * 15)$ matrix. We compute m_t at every frame, and so the windows overlap. Similarly, we represent our motion capture library as a collection of 3D point clouds M_i (where i varies over the 11000 poses in our database). For convenience, we subtract out the centroids and subsample the clouds to be equal to the video frame rate.

Background stabilization: To compute meaningful local dynamics, we assume the joint positions m_t are defined with respect to a coordinate system that is stationary over a 15-frame window. This is trivially true for video recorded with a stationary camera. Interestingly, one can also factor-out camera movement *if* most of the foreground is a tracked person(s). Such sequences are common in sports footage and feature-film shots in which the camera pans to follow someone (we show examples from the film 'Run Lola Run' in SectionV). Recall that our 2D pose estimates are obtained without any background assumption (Sec.III). This means we can use the pose estimates to *identify* the foreground/background pixels. Using only the background pixels (the pixels outside the rectangle masks), we align successive pairs of frame (by finding the translation minimizing SSD error). Such a procedure will produce gross errors in registration over a long sequence, but provides reasonable estimates over local 15-frame windows. We find this approach also works for sequences containing moving objects in the background (see Figure 8).

Alignment: An important issue when comparing point clouds is that of alignment [48, 49]. We find the transformation that best aligns the 3D pose M_i with the 2D pose m_t, in terms of minimizing the reprojection error. We search over scaled-Euclidean transformations; a rotation, translation, and an isotropic scale. We explicitly search over rotations, rendering M_i from different orthographic cameras. We assume that the camera is at 0° elevation, and explicitly search over 20 azimuth (ϕ) values. Since the camera is fixed over a sequence, we can interpret ϕ as the orientation of the root body (i.e. the center torso) of M_i with respect to the camera. Given a pose M_i and orthographic camera matrix R_ϕ, we compute a 2D rendered point cloud $R_\phi M_i$. Aligning two 2D point clouds is straightforward [48]; we translate and scale $R_\phi M_i$ so that its centroid and variance match that of m_t.

Reprojection error: We write the squared reprojection error after alignment as $||R_\phi M_i - m_t||^2$. We cannot directly compute this error because of left/right ambiguities. The pictorial structure model cannot distinguish between left and right arms because they look similar. This means that correspondence is not fully known. Interestingly, the alignment procedure described thus far still applies; the centroid and variance of a point cloud do not change when points are permuted. We compute a final reprojection error by finding the left/right assignment that minimizes the reprojection error:

$$||R_\phi M_i - m_t||^2_{lr} = \sum_{j=1}^{15} \min_k ||R_\phi M_i(j) - m_t(j)F_k||^2, \quad (1)$$

where we write $M_i(j)$ for the joint positions from the j^{th} frame in the 15-frame window. We write F_k for a $12X12$ binary matrix that permutes the left/right leg/arm joints of $m_t(j)$; hence $k \in \{1, 2, 3, 4\}$.

Missed detections: In most frames, our pictorial structure will not recover all the limbs. We would like to omit any missing joint positions from the 2D and 3D clouds when performing alignment and computing the reprojection error. However, missing limbs complicate alignment; if we detect only one leg in a frame of m_t, we do not know whether to align it to a left or right reference leg in M_i (if we detect two legs, correspondence doesn't matter as explained above). We explicitly search over the 4 left/right labellings of the center frame for alignment. To avoid searching over labellings for the remaining frames in our window, we only use unambiguous points for computing alignment (points from the center frame, points on the torso, and points from limbs for those frames when both limbs are detected). After alignment, we use all detected joint positions to compute the reprojection error, as in Equation 1.

Alternatives: One could build an aspect model exploiting missed detections. If the left side of a person is not detected, this suggests something about his/her orientation with respect to the camera. We can formalize this notion by computing the visibility of joint positions of the rendered 3D point clouds $R_\phi M_i$. When matching to a specific m_t with a given set of detected (i.e., visible) limbs, we add a penalty for mis-matched visibility flags (because they suggest a false positive or missed limb detection). In our experience, this aspect model did not significantly improve results.

Rather than computing strict $\mathcal{L}_2$ reprojection error, one might want to compute a weighted error where joint positions from the center frame are weighted more heavily. All our steps (our

alignment procedure and reprojection error computation) still follow; in our experience this did not significantly change the final results.

B. Synthesis by Dynamic Programming

Given the matching procedure from Section IV-A, one might find the pose M_i that best matches m_t for each frame independently. For some poses, such as lateral views of the stance phase of a walk, it is hard to estimate both the torso orientation and left/right assignment from the 2D stick figure m_t. If someone supplied the correct camera azimuth (or equivalently the torso orientation) and left/right assignment, the recovered pose M_i would match m_t rather well. We propagate information temporally to achieve this effect; we find a sequence of poses M_i that match m_t *and* that change smoothly from frame to frame.

For each frame t, let us compute the best match M_i given a particular camera orientation $\{1, 2, \ldots, 20\}$ and given a particular left/right assignment of the center frame. This yields a pool of $20 \times 4 = 80$ 3D point clouds rotated, translated, and scaled to align with m_t. Let us write these *aligned* point clouds as $M_t(l_t)$ and their reprojection error as $\epsilon_t(l_t)$ where $l_t \in \{1, 2, \ldots, 80\}$.

We now want to find a sequence of 3D point clouds that all match the image data well and that match future and past 3D estimates well. We can efficiently compute this by dynamic programming. Let us define

$$f(l_{1:T}) = \sum_t \epsilon_t(l_t) + w||M_t(l_t) - M_{t-1}(l_{t-1})||_o^2, \tag{2}$$

where the user-defined weight w controls how closely the synthesized motion should follow the image data (the first term) versus how continuous it should be (the second term). We use $w = .1$. We compute the sequence of $\hat{l_t}$ that minimizes Equation 2 by dynamic programming. Doing so recovers a sequence of 3D clouds $M_t(\hat{l_t})$ that all match the image data, and that temporally match each other well.

Continuity cost: We use $||.||_o^2$ to denote the squared error of the points that overlap temporally (the first 14 frame window in M_t and the last 14 frame window in M_{t-1}). We make the error translation-invariant by subtracting out the centroid of each point cloud [48]. Note that by doing so, we are ignoring the global dynamics present in the video. For example, assume the 2D tracked figure in the video has long legs and walks 10 meters in the video clip. Now, assume

we are using a 3D skeleton with short legs. There are two motions we might want to synthesize. One, the global dynamics could match; the short skeleton also moves 10 meters, but has to run to do it. Alternatively, the local dynamics could match; the short skeleton walks, but only moves half the distance. Different applications may require different behaviors. For our task of activity recognition (discriminating `walk` versus `run`), we argue local dynamics are more important and so ignore global dynamics in the continuity cost.

Alternatives: Ideally, one would perform dynamic programming over all poses in our 3D library, but this is computationally prohibitive. An alternative is to synthesize a 3D motion by explicitly walking along a motion graph [38, 40, 41]. Such an approach guarantees that the synthesized motion will be globally continuous and human-looking. However, it can suffer from drift, in the sense that the synthesized motion can get "stuck" in a part of the graph whose poses do not follow the 2D tracks well. Essentially, enforcing global continuity might force poor matches for local dynamics. Such an approach is not appropriate for our task since local dynamics seem to define fine-scale activities.

C. Smoothing

For each frame t, the recovered point cloud $M_t(\hat{l}_t)$ spans a 15 frame window. This means, for each frame t, we have 15 possible poses, obtained from neighboring frames with overlapping windows. Since our 3D point clouds $M_t(\hat{l}_t)$ are subsampled representations of larger point clouds (constructed from skeletons with additional joints), we align the full point clouds to the images using the recovered scaled-Euclidean transformations. To compute a final 3D pose, we average the joint positions from the 15 skeleton poses that overlap frame t.

Recall that each 3D pose was labeled off-line with a binary vector of 13 flags representing an annotation. To compute an activity vector for the t^{th} frame, one might be tempted to average the binary vectors of the overlapping poses. However, this might produce a final annotation where both `run` and `stand` are 'on'. To respect the constraints implicit in the off-line labeling, we use a weighted voting scheme. Each overlapping pose votes for 1 of the 2^{13} possible annotation vectors, and we finally choose the annotation vector with the most votes. In practice, we precompute all valid annotation vectors from the database (of which there are 46), and only record votes for one of them.

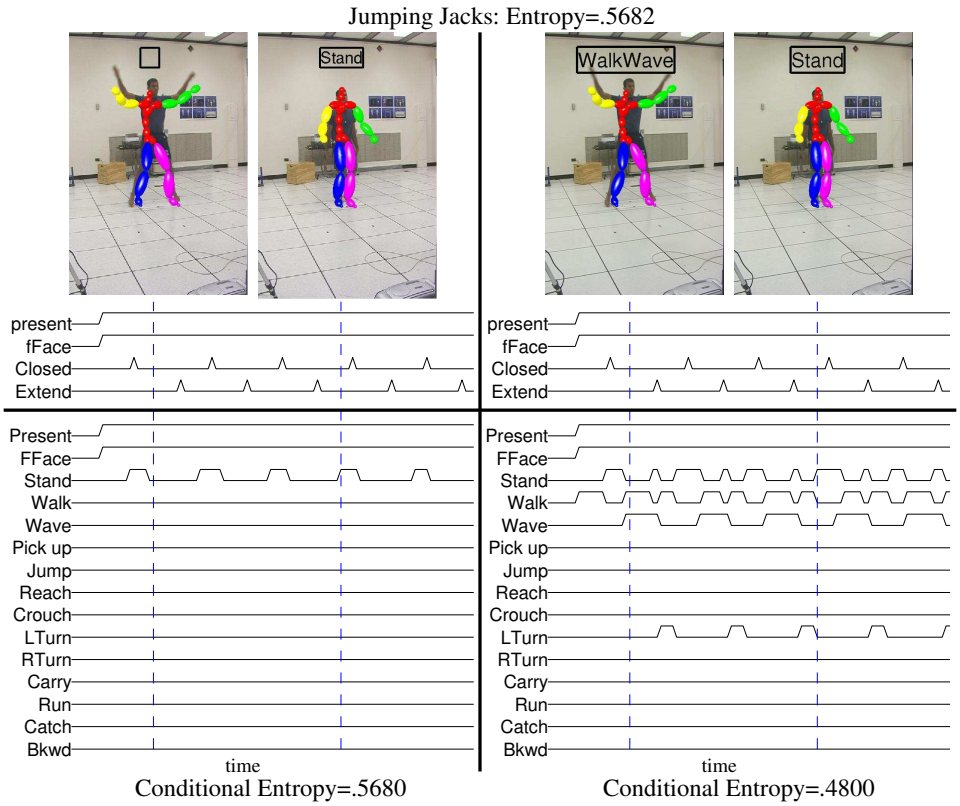

Fig. 3. Unfamiliar configurations can either be annotated with 'null' or with the closest match. We show annotation results for a sequence of jumping jacks (sometimes known as star jumps) from two such annotation systems. In the top row, we show the same two frames run through each system. The recovered 3D pose from Section IV-C has been reprojected back to the image. In the bottom, we show signals representing annotation bits over time. The manual annotator records whether or not the figure is `present`, `front faceing`, in a `closed` stance, and/or in an `extended` stance. The automatic annotation consists of a total of 16 bits; `present`, `front faceing`, plus the 13 bits from the annotation vocabulary of Section II. In first dotted line, corresponding to the image above it, the manual annotator asserts the figure is `present`, frontally `faceing`, and about to reach the `extended` stance. The automatic annotator asserts the figure is `present`, frontally `faceing`, and is **not** `standing`, **not** `jumping`, etc. The annotations for both systems are reasonable *given there are no corresponding categories available* (this is like describing a movement that is totally unfamiliar). On the **left**, we freely allow 'null' annotations (where no annotation bit is set). On the **right**, we discourage 'null' annotations as described in Section V. Configurations near the `extend` stance are now labeled as `walkwave`, a reasonable approximation. We quantitatively show this approach results in better annotations, by computing the reduction in entropy of the user signal given the automatic reports (as described in Section V.).

V. Experimental Results

We use a motion database of 118 motions of football players. Each frame was annotated using the procedure and vocabulary of Section II by a user who had not seen the videos to be annotated. We tested our system on six sequences; indoor sequences of a person walking

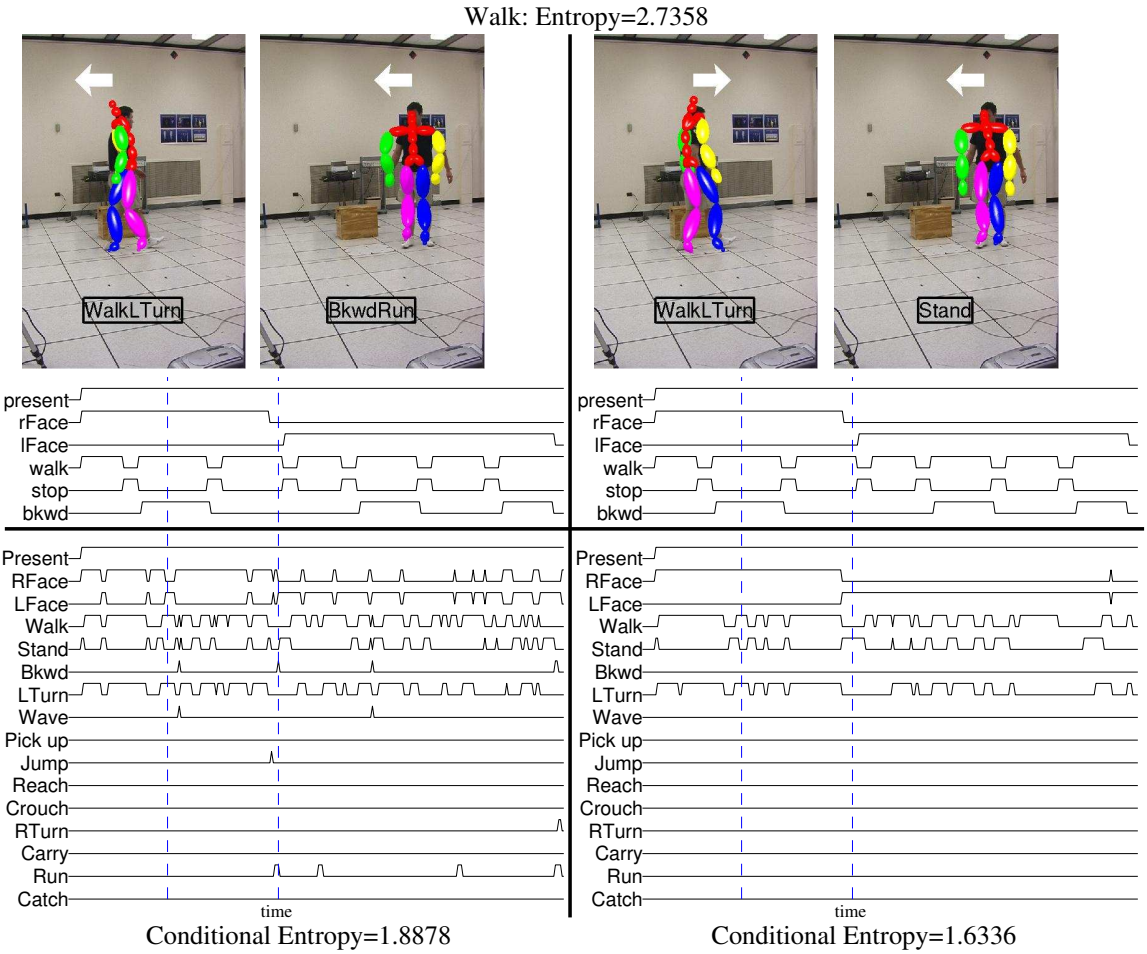

Fig. 4. We show annotation results for a walking sequence from two versions of our system using the convention of Figure 3. Null matches are allowed. On the **left**, we infer the 3D pose M_i (and associated annotation) independently for each frame. On the **right**, we synthesize a smooth set of poses (and associated annotations) by dynamic programming (Section IV-B). Each image is labeled with an arrow pointing in the direction the inferred figure is facing, not moving. By enforcing smoothness, we are able to fix spurious `run`'s and incorrect torso orientations present on the **left** (i.e., the first image frame and the automatic `left face`ing and `right face`ing annotation bits). The system on the **right** correlates better with the user annotations, as shown by a lower conditional entropy score.

and doing jumping-jacks, an outdoor sequence of multiple people passing a ball back and forth, and various shots from the feature-length film 'Run Lola Run'. For each video, our system automatically builds a pictorial structure for each figure, and then detects each model in each frame (Section III). Our system then synthesizes a 3D motion by matching annotated motion clips to the 2D detections (Section IV).

Evaluation of general-purpose activity recognition systems is difficult precisely because there

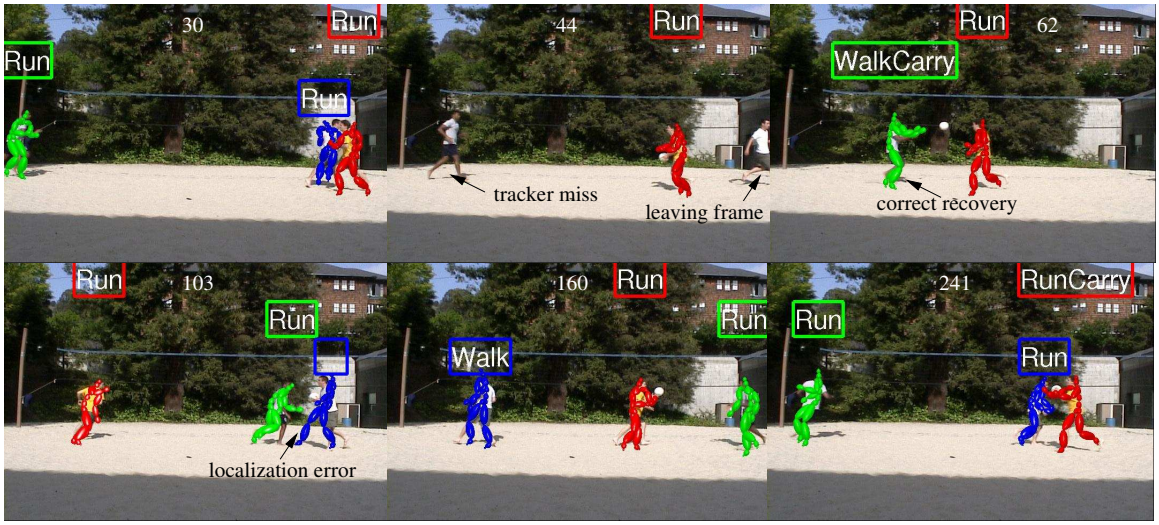

Fig. 5. Frames sampled from a 21 second sequence of three actors playing with a ball. The numbers on each frame give the order in which the frame appears in the sequence; the spacing is roughly even. The white annotations are automatically generated by our system (we manually add the black words for clarity). Overlaid on each frame is the best configuration chosen for the body of each of the three actors detected — both number and appearance are obtained automatically — using camera consistency as in Section IV-B. Individuals are associated with a color and a fixed-height annotation label to show the tracker has consistently identified them. We see two tracks interrupted, one because of a missed detection and the other because the figure leaves the view. Both tracks are recovered, but we see an incorrect pose estimation because of a missed leg detection.

is no canonical vocabulary. Given a test video, we manually annotate it with labels appropriate for that video. For example, given a video of a figure performing jumping jacks, we use the labels `closed` and `extended` to describe phases of the jump. These labels need not correspond with those used to describe a collection of football motions. This makes applying standard evaluation criteria such as ROC curves or confusion matrices awkward, since there is no clear correct detection. Furthermore, there is no meaningful standard with which to compare.

Scoring by plotting signals: Qualitatively, we lay out a visual comparison between the human and automatic annotations signals. We show an example in Figure 3, which displays annotation results for a 91-frame jumping jack sequence. The top 4 lower case annotations are hand-labeled over the entire 91 frame sequence. Generally, automatic annotation is successful: the figure is detected correctly, oriented correctly (this is recovered from the torso orientation estimates ϕ), and the description of the figure's activities is largely correct.

Scoring by conditional entropy: We quantify the degree to which the user and automatic

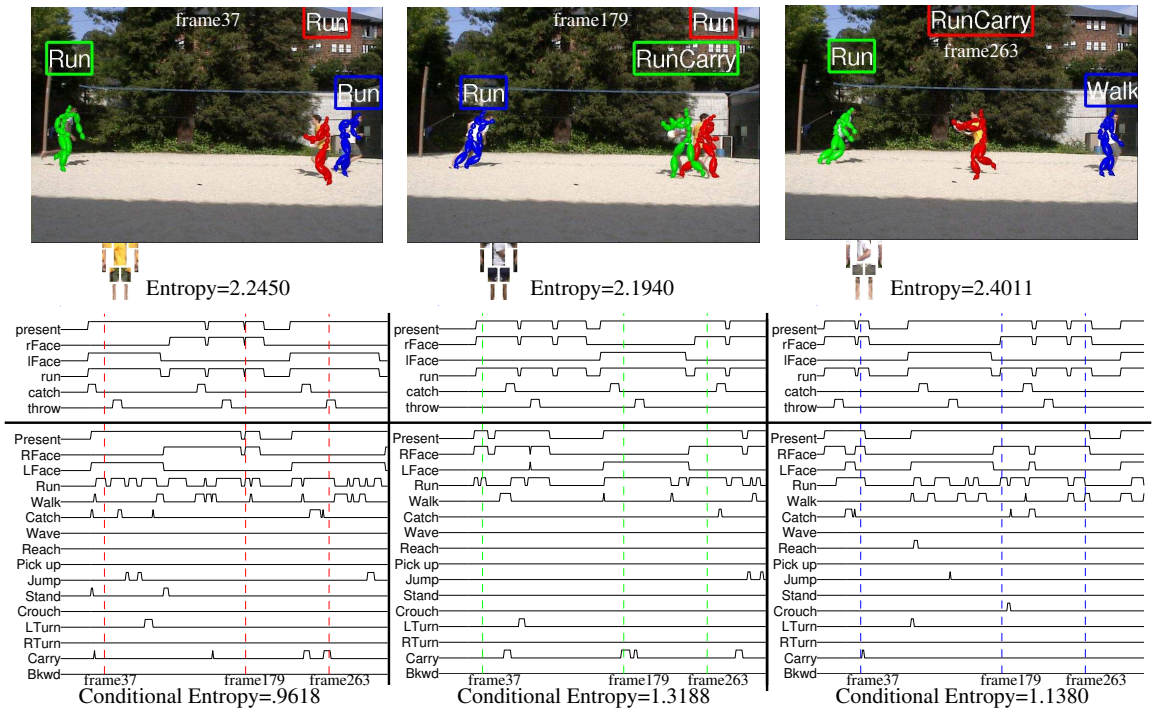

Fig. 6. Annotations of 3 figures from a video sequence of the three passing a ball back and forth using the conventions of figure 3. The dashed vertical lines indicate annotations corresponding to the frames shown **above**. Null matches are allowed. The automatic annotations are largely accurate: the figures are correctly identified, and the direction in which the figures are facing are largely correct. Most of the time, people are `run`ning, but slow down to `walk` when turning or passing the ball. Throws appear to be mislabeled as `catch`es. Generally, when the figure has the ball (after `catch`ing and before `throw`ing, as denoted in the manual annotations), he is annotated as `carry`ing, though there are some missed detections. There are no spurious `crouch`es, `wave`s, etc.

signals agree (without explicit correspondence) by computing the *mutual information*, or reduction in entropy. Intuitively, mutual information measures our ability to build a classifier for user annotations using the automatic annotations as features. We interpret the user annotation signal as a bit-vector defined at every frame t. On any given video sequence, only a small number of unique bit combinations are encountered (we typically see $k \leq 10$ combinations). We define U to be a multinomial random variable taking on one of the k user-labeled annotations. We likewise define A to be a multinomial capturing the unique automatic annotations obtained for that video sequence. Mutual information captures the reduction in uncertainty of U given A, or

Entropy $H(U)$ and Mutual Information $(H(U) - H(U|A))$

Sequence	$H(U)$	$H(U\|A)$	$H(U) - H(U\|A)$
Jumping Jacks	.5862	.4800	.1062
Walk	2.7358	1.6336	1.1022
Weave	2.2800	1.1405	1.1395
Lola	1.859	.9024	.9566

TABLE I

WE SCORE OUR PERFORMANCE USING THE REDUCTION IN ENTROPY OF (MANUALLY-LABELED) USER ANNOTATIONS U GIVEN THE OUTPUT OF OUR AUTOMATIC SYSTEM A. FOR THE 'JUMPING JACK' SEQUENCE, WE DID NOT ALLOW `null` ANNOTATIONS. FOR THE 'WEAVE' SEQUENCE, WE AVERAGE RESULTS OVER THE THREE TRACKED FIGURES. FOR 'LOLA', WE AVERAGE RESULTS OVER TWO SHOTS OF LOLA PERFORMING INTERESTING MOTION (FIGURE 8 AND 9).IN ALMOST ALL CASES, THE AUTOMATIC SYSTEM REDUCES THE ENTROPY BY A FACTOR OF 2. THE USER ANNOTATIONS FOR THE JUMPING JACK SEQUENCE PROVE TOO SIMPLE; THERE IS LESS THAN A BIT OF UNCERTAINTY IN THEM (SINCE THEY ARE NEARLY CONSTANT), AND SO OUR SYSTEM PROVIDES LITTLE IMPROVEMENT.

$M(U, A) = H(U) - H(U|A)$, where

$$\begin{aligned} H(U) &= \sum_i \Pr(U = i) \log \Pr(U = i) \\ H(U|A) &= \sum_{i,j} \Pr(U = i, A = j) \log \Pr(U = i|A = j), \end{aligned}$$

where the probabilities are computed by counting co-occurrences throughout a given video sequence. Note that in general, computing these co-occurrences is just as difficult as learning large state-space HMMs. The classic problem of not-enough training data manifests itself in a sparse co-occurrence matrix. We avoid this pitfall by computing co-occurrences for only the k observed combinations, as opposed to all 2^{13} possibilities.

Given a test video with user annotation U, the entropy $H(U)$ is equivalent to the number of bits required to encode U. Given knowledge of some other signal A, the entropy (or uncertainty) in U can only decrease. We evaluate different automatic annotations A for a given U by computing which annotations reduce the entropy $H(U|A)$ the most. We tabulate results in Table I.

Is temporal consistency required? Figure 4 compares two versions of our system on a 288 frame sequence of a figure walking back and forth. The annotations on the left were obtained by matching poses M_i independently for each frame t. Note that the automatic `LFace` and `RFace`

signals flip back and forth – it is difficult to estimate the orientation of the figure when he is in the stance phase of a lateral walk. By performing explicit motion synthesis (ensuring the poses M_i are smooth over time, as in Section IV-B), orientation is estimated much more reliably. We quantify this by computing the conditional entropy of each set of automatic annotation signals. The smoothed annotation signal correlates better with the ground-truth user signal.

Annotating multiple people interacting: In Figure 6, we show annotations for three figures from one sequence passing a ball back and forth. Each actor is correctly detected, and the system produces largely correct descriptions of the actor's orientation and actions. The inference procedure interprets a run as a combination of `run` and `walk`. Quite often, the `walk` annotation will fire as the figure slows down to turn from `face right` to `face left` or vice versa. When the figures use their arms to catch or throw, we see increased activity for the similar annotations of `catch`, `wave`, and `reach`.

Annotating novel motions: When a novel motion is encountered, we want the system to either respond by (1) recognizing that it cannot annotate this sequence, or (2) annotating it with the best possible label. We can implement (2) by relabeling those poses M_i that are annotated with a 'null' bit vector (all flags are off). Interestingly, almost $\frac{1}{4}$ of the 3D poses in our library have a null label. This implies our original annotation vocabulary is still too small for our motion library. Off-line, we "force" a label for all null poses by matching them to the closest labeled pose (1-NN classification). Given a synthesized motion, we can report either the user-specified labels or the forced non-null labels. In Figure 3, we use our football library to annotate a jumping jack sequence. Our library does not contain any jumping jack motions, and lacks a `jumping jack` annotation label. System (1) responds with a `stand` label when the figure is near a `closed` stance, and reports a null label for all other frames. By forcing non-null annotations, we see an additional `walkwave` label toward the `extend` phase, with the occasional `turn`. Quantitatively, (2) results in better annotations as seen by a lower conditional entropy score. Note the entropy estimates for this sequence are quite small; this is because the sequence is short and the user signal U is almost constant.

Annotating commercial footage: We have applied our system to the feature film 'Run Lola Run'. We show results on various shots in Figure 7-9 (these shots were identified automatically using the kinematic tracker of [5]). In Figure 7, our annotation system correctly labels Lola as `run`ning for the duration of the sequence. In Figure 8, Lola initially runs, and then stops in

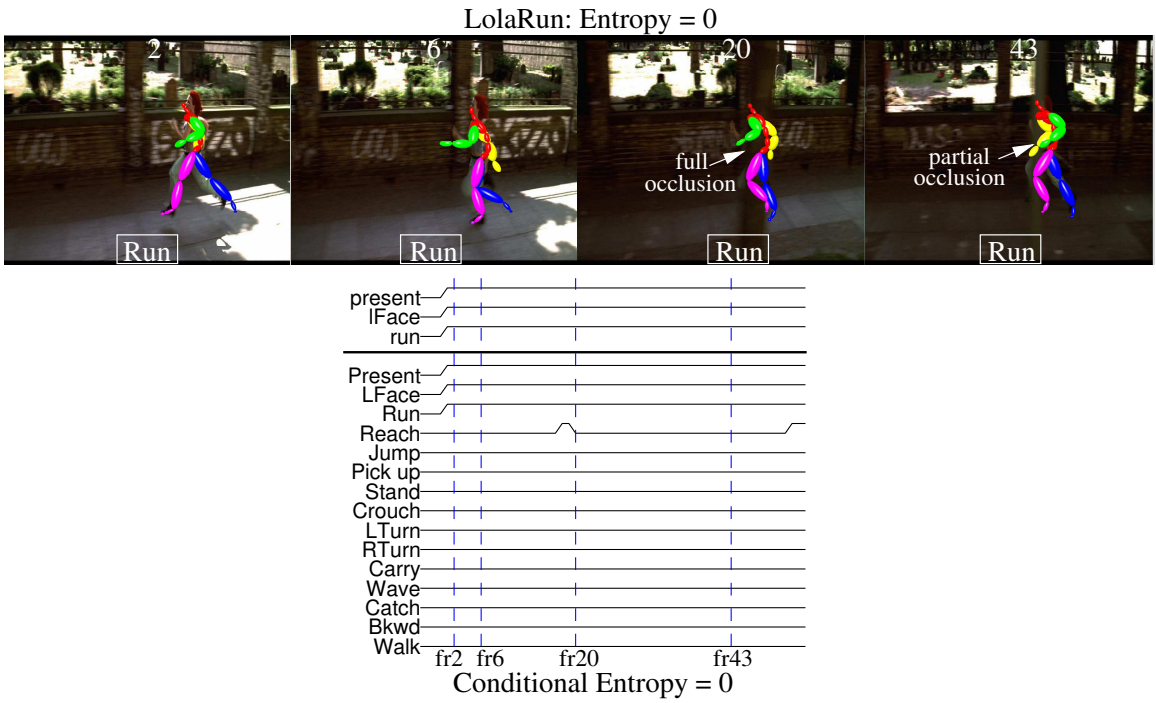

Fig. 7. Annotation of a shot of 'Lola' running (using the conventions of Figure 6). Because she is running throughout this clip, the entropy of the user-defined annotations is 0 (there is no uncertainty). The automatic annotation results are quite successful (the figure is `present`, facing `left`, `run`ing, not `jump`ing, etc), although there is a brief `reach` misclassification due to a partial occlusion by a telephone pole.

front of a truck. Our system responds with a reasonable description of Lola initially `run`ning, then slowing down to `walk` before `stand`ing. The final shot in Figure 9 is quite challenging. Here, Lola runs toward the camera, runs around a corner, collides with another person, spins around, and then continues running around the corner. The synthesized motion labels Lola as initially `run`ning toward the camera, `turn`ing, and then `run`ning away from the camera. When Lola is small, it is difficult to estimate her kinematics and so our system reports some spurious annotations. However, our annotation results are impressive given the difficulty of the sequence. On average, we reduce one bit of uncertainty in the manual annotations of our commercial sequences.

VI. Discussion

We have described a method for automatically annotating a video of everyday movements. We do this by taking an "analysis by synthesis" approach; we first estimate the configurations of

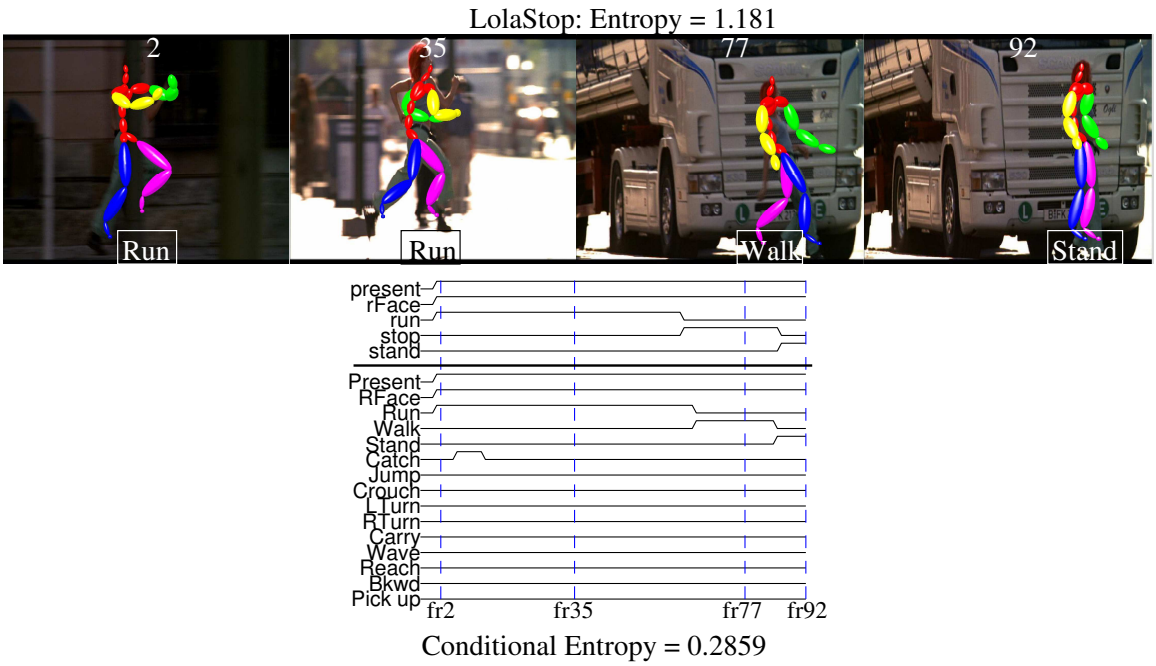

Fig. 8. Annotation of a shot of 'Lola' running and stopping in front of a moving truck (using the conventions of Figure 6). Note that our background stabilization procedure from SectionIV-A is successful even given the large changes in the background (crowds of people and the truck are moving). Other than the a brief `catch` misclassification (due to a telephone-pole occlusion), the automatic reports match the user-obtained reports extremely well. Note the automatic system equates `stop`ing with `walk`ing, a reasonable assignment given its limited vocabulary.

a figure over time, and then we *re-create* that estimate by matching to an existing set of labeled examples. As a result, labels are generated for free from the synthesis process.

This approach requires a method that can reasonably estimate the configurations of the body over time; we demonstrate that a pictorial structure can generate such configuration reports. By using real data as opposed to hand-labeled body estimates [2, 3, 4], we find that our synthesis engine must compensate for ambiguities such as torso orientation, left/right labellings, and missed detections.

We introduce mutual information as a scoring criteria. Evaluation is hard because there is no canonical vocabulary for describing everyday behavior (see Figure 1). Our measure provides a reasonable scheme for ranking different systems against test footage that is labeled with some annotation set.

Our analysis-by-synthesis approach appears to label small-scale activities reasonable well. We

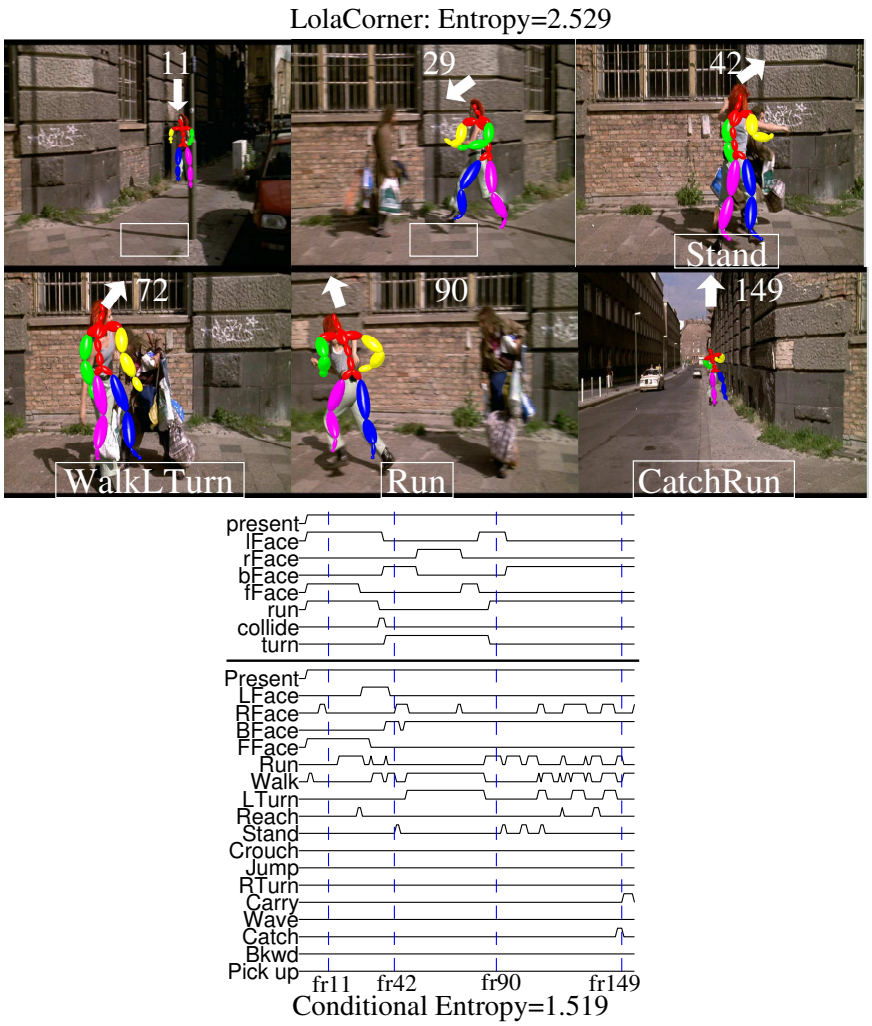

Fig. 9. Annotation of a challenging shot of 'Lola' running around a corner (using the conventions of Figure 6). Lola runs toward the camera, collides with another person, spins around, and then runs away from the camera. We annotate the rendered figures with an arrow pointing in the direction the figure is facing (down represents toward the camera). The automatic annotator realizes the figure is initially `front`ally `face`ing, `turn`s in the middle of the sequence, and is then `back face`ing. The matched motions capture the swinging of her arm during her spin (frame 42), but do not capture her protective steps (frame 72). During the `turn`, the automatic system also labels Lola as `walk`ing. Towards the end of the sequence when Lola becomes quite small, the automatic system reports spurious annotations due to poor kinematic estimates. Quantitatively, the automatic system can be seen as successful because it reduces one bit of uncertainty in the user-defined annotations.

hope the analogy can apply to large-scale actions as well. It suggests one approach for authoring complex models; we find rules that synthesize complex actions realistically, and apply them to analyze such actions from video.

REFERENCES

[1] M.J. Black and Y. Yacoob, "Parametrised modelling and tracking of human activities," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 232–247, 1999.

[2] M.E. Leventon and W.T. Freeman, "Bayesian estimation of 3D human motion from an image sequence," Tech. Rep. TR-98-06, MERL, 1998.

[3] D. Liebowitz and S. Carlsson, "Un-calibrated motion capture exploiting articulated structure constraints," in *ICCV*, July 2001.

[4] D. DiFranco, T.J. Cham, and J.M. Rehg, "Reconstruction of 3-d figure motion from 2-d correspondences," in *CVPR*, December 2001.

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

[6] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding: CVIU*, vol. 73, no. 3, pp. 428–440, 1999.

[7] A. F. Bobick and J.W. Davis, "The recognition of human movement using temporal templates," *IEEE T. Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[8] A.F. Bobick, "Movement, activity, and action: The role of knowledge in the perception of motion," *Philosophical Transactions of Royal Society of London*, vol. B-352, pp. 1257–1265, 1997.

[9] 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.

[10] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 34, no. 3, pp. 334–352, Aug 2004.

[11] R. Polana and R.C. Nelson, "Detecting activities," in *Proceedings, International Conference on Pattern Recognition*, 1994, pp. A:815–818.

[12] A.F. Bobick and J.W. Davis, "Real-time recognition of activity using temporal templates," in *Workshop on Applications of Computer Vision*, 1996.

[13] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proc ICCV*, 2003.

[14] J.W. Davis and A.F. Bobick, "The representation and recognition of action using temporal templates," in *CVPR*, 1997.

[15] L. Zelnik-Manor and M. Irani, "Event-based analysis of video," in *CVPR*, 2001.

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

[17] J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, "Human activity recognition using multidimensional indexing," *PAMI*, vol. 24, no. 8, pp. 1091–1104, August 2002.

[18] C.S. Pinhanez and A.F. Bobick, "Human action detection using pnf propagation of temporal constraints," in *IEEE Conference on Computer Vision and Pattern Recognition*. 1998, pp. 898–904, IEEE Press.

[19] C.S. Pinhanez and A.F. Bobick, "Pnf propagation and the detection of actions described by temporal intervals," in *Proc. DARPA IU Workshop*, 1997, pp. 227–234.

[20] James F. Allen, "Towards a general theory of action and time," *Artifical Intelligence*, vol. 23, no. 2, pp. 123–154, 1984.

[21] Jeffrey Mark Siskind, "Grounding language in perception," *Artifical Intelligence Review*, vol. 8, pp. 371–391, 1995.

[22] Jeffrey Mark Siskind, "Reconstructing force-dynamic models from video sequences," *Artifical Intelligence*, vol. 151, pp. 91–154, 1995.

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

[24] M. Brand, N. Oliver, and A.P. Pentland, "Coupled hidden markov models for complex action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*. 1997, pp. 994–999, IEEE Press.

[25] Xiaolin Feng and P. Perona, "Human action recognition by sequence of movelet codewords," in *First International Symposium on 3D Data Processing Visualization and Transmission*, 2002, pp. 717–721.

[26] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *CVPR*, 1992, pp. 379–385.

[27] A.D. Wilson, A.F. Bobick, and J. Cassell, "Recovering the temporal structure of natural gesture," in *Proceedings of the Second Int. Workshop on Automatic Face- and Gesture-Recognition*, 1996.

[28] J. Yangan, Y. Xu, and C. S. Chen, "Human action learning via hidden markov model," *IEEE Transactions on Systems Man and Cybernetics*, vol. 27, pp. 34–44, 1997.

[29] N. Oliver, A. Garg, and E. Horvitz, "Layered representations for learning and inferring office activity from multiple sensory channels," *CVIU*, vol. 96, no. 2, pp. 163–180, November 2004.

[30] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *Fourth IEEE International Conference on Multimodal Interfaces*, 2002, pp. 3–8.

[31] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato, "Hierarchical recognition of daily human actions based on continuous hidden markov models," in *Automatic Face and Gesture Recognition*, 2004, pp. 779–784.

[32] A.D. Wilson and A.F. Bobick, "Parametric hidden markov models for gesture recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 884–900, September 1999.

[33] M. Brand and V. Kettnaker, "Discovery and segmentation of activities in video," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 844–851, August 2000.

[34] A. Galata, N. Johnson, and D. Hogg, "Learning structured behavior models using variable length markov models," in *IEEE International Workshop on Modelling People*, 1999, pp. xx–yy.

[35] A. Galata, N. Johnson, and D. Hogg, "Learning behaviour models of human activities," in *British Machine Vision Conference*, 1999, p. Modelling Human Behaviour.

[36] O. Arikan and D. Forsyth, "Interactive motion generation from examples," in *Proc. ACM SIGGRAPH*, 2002.

[37] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *SIGGRAPH*, 2002.

[38] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data," in *SIGGRAPH*, 2002.

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

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

[41] L. Ren, G. Shakhnarovich, J. Hodgins, H. Pfister, and P. Viola, "Learning silhouette features for control of human motion," *ACM Transactions on Graphics*, October 2004.

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

[43] O. Arikan, D.A. Forsyth, and J. O'Brien, "Motion synthesis from annotations," in *Proc. ACM SIGGRAPH*, 2003.

[44] C. C. Chang and C. J. Lin, "Libsvm: Introduction and benchmarks," Tech. Rep., Department of Computer Science and Information Engineering, National Taiwan University, 2000.

[45] 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.

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

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

[48] B.K.P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society A.*, vol. 4, no. 4, pp. 629–642, April 1987.

[49] B.K.P. Horn, "Relative orientation," *IJCV*, vol. 4, no. 1, pp. 59–78, January 1990.