Action Recognition by Time Series of Retinotopic Appearance and Motion Features

Daniel Paul Barrett, *Student Member, IEEE*, and Jeffrey Mark Siskind, *Senior Member, IEEE*

***Abstract*— We present a method for recognizing and localizing actions in video by the sequence of changing appearance and motion of the participants. Appearance is modeled by histogram of oriented gradients object detectors, while motion is modeled by optical-flow motion-pattern detectors. Sequencing is modeled by a hidden Markov model (HMM) whose output models are these appearance and motion detectors. The HMM and associated detectors are simultaneously trained, learning the sequence of detectors that match the most distinctive temporal subsequences of the action represented in the training data. Training uses both positive and negative samples of a given action class and is accomplished without the need for annotation of the correspondence between training video frames and the state-conditioned detectors, by minimizing a discriminative cost function through gradient descent. Trained models are used to perform recognition and localization by simultaneous detection, tracking, and action recognition. In contrast to many prior methods, our approach learns intuitively meaningful models that represent action as a sequence of retinotopic models. We demonstrate such by rendering these models on unseen test video. This method was found to perform competitively on three standard datasets, Weizmann, KTH, and UCF Sports, as well as on the video from the Defence Advanced Research Project Agency (DARPA) Mind's Eye program and a newly filmed dataset.**

***Index Terms*— Hidden Markov model (HMM), object detection, tracking, video action recognition.**

I. Introduction

ACTION recognition in video is a growing field applicable to areas such as surveillance, robotics, and video retrieval. This field largely focuses on forced choice one-out-of-K classification of video clips, rather than binary classification or detection in long streaming videos, which can, in theory, be done by the extension of a classification system. The most prominent datasets [1]–[6] reflect this focus. Every video clip in these datasets contains a single instance of an action; no clip contains multiple actions or fails to depict any action. The most prominent current methods [1], [6]–[15] generally employ a bag of spatio-temporal visual-words approach (BOW). They generally extract feature vectors, such as spatiotemporal interest points (STIP) [4] or dense trajectories [13], at a subset of space–time points, build a codebook by pooling such vectors, quantize such feature vectors on this codebook, compute a histogram of codebook-entry occurrences on the pooled frames of a video, and classify these histograms with temporally invariant models, such as support vector machines (SVMs). However, such methods leave something to be desired, as they do not represent the coarse-grained temporally variant appearance and motion that is characteristic of an action class in a human discernible fashion, relying instead on aggregation of very fine-grained properties. Such methods can learn models for actions based on low-level features that have correlation with the classes in the particular dataset used, but which are unrelated to the meaning of the action class that a human might understand. For example, they might associate *diving* with a blue Olympic swimming pool [16], or *basketball* and *volleyball* with a wooden gym floor [17]. The aggregated properties by which a video is classified, therefore, may have not even come from the part of the video in which the action takes place. Such methods therefore generally do not localize the recognized actions.

These methods are akin to distinguishing lasagna from spaghetti by blending them up and analyzing the ratios of elements with a mass spectrometer. Fundamentally, spaghetti and lasagna differ only in the shape of the pasta, but are usually associated with different sauces, resulting in correlations with different microscopic properties. Thus, such a blender approach would generally have high performance, but completely fail to distinguish lasagna and spaghetti without sauce or fail to distinguish pesto or Alfredo lasagna from spaghetti when trained on tomato lasagna and spaghetti, and would not be well suited to localizing individual pieces of pasta. Similarly, models for *diving* and *basketball* which rely on the backgrounds would have difficulty in recognizing a person shooting a *basketball* in a swimming pool. Perhaps for this reason, many datasets focus on classes which might be better thought of as scenes than actions. For example, the UCF50 [1] dataset depicts classes, such as *military parade* and *horse race*, which are not localizable atomic actions, but rather general scene categories.

One reason that much recent work focuses on BOW-style methods is precisely because they do not involve localization, which itself has proven to be quite difficult. Localization of the action participants generally involves the use of some kind of detector or tracker. However, detection and tracking are themselves unsolved problems, with no totally reliable solution. Background subtraction [18] avoids explicit object detection and aims to identify those pixels that are in the

Manuscript received June 19, 2015; revised September 9, 2015; accepted November 19, 2015. Date of publication November 23, 2015; date of current version December 2, 2016. This work was supported in part by the Army Research Laboratory under Grant W911NF-10-2-0060 and in part by the National Science Foundation under Grant 1522954-IIS. This paper was recommended by Associate Editor S. Ci.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-2035 USA (e-mail: dpbarret@purdue.edu; qobi@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2502839

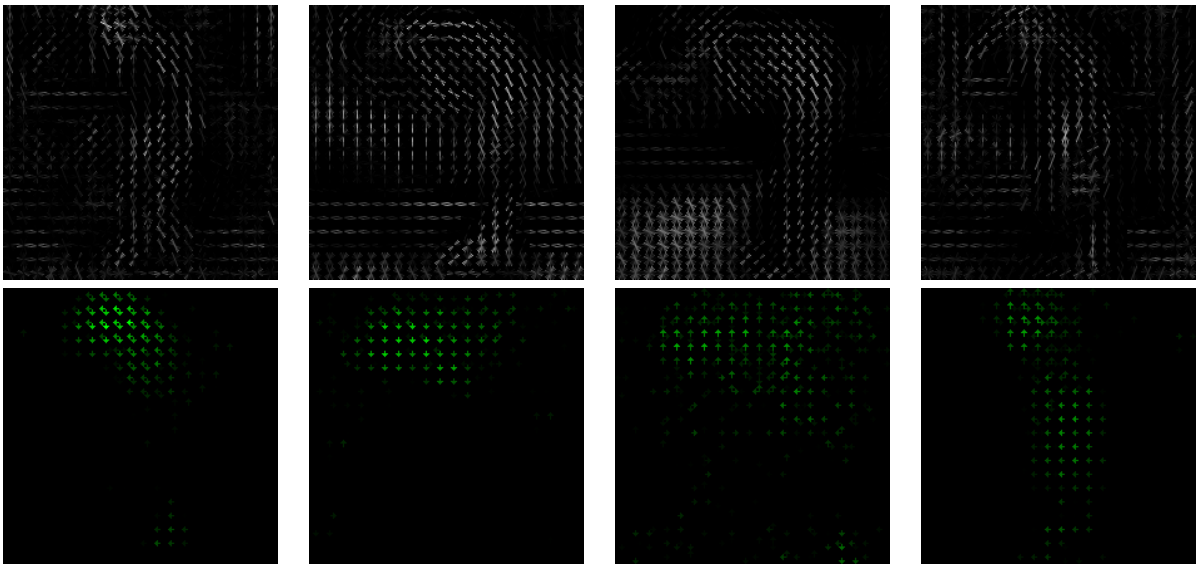


Fig. 1. Visualization of appearance and motion models for the *bend* action. Appearance (top row) and motion (bottom row) are represented as dense grids of edge and HOG and HOF centered on the action. These models depict a person bending over and standing back up.

foreground by building a model of the background and to assign areas that move or otherwise deviate from this model to the foreground. General tracking methods like that of [19] take a bounding box as input and attempt to track the entity bounded by that box through the rest of a video. However, these methods do not have a way to track the particular person performing an action and fail in the presence of occlusion or, in the case of background subtraction, in the presence of several moving objects. Any method that relies on such preprocessing steps to provide the locations of the action participants is subject to cascading failure when the detector or tracker fails to track the relevant participants. The localization performance therefore sets an upper bound on recognition performance.

However, the benefits of localizing the action participants in addition to recognizing the action class have been shown [20] and [21]. They track action participants and use them to recognize and generate natural language sentences complete with adjectives, adverbs, and prepositional phrases describing the properties of and interactions between multiple participants. This kind of deep analysis is impossible without localization. The key to both of these methods is the Event Tracker of Barbu *et al.* [22], which is a framework for performing simultaneous tracking and action recognition. It allows a high-level action model to influence the tracking process, preventing the issue of cascading failure and making it more reliable than two-step independent tracking and action recognition methods.

The Event Tracker is not an action recognition model nor a learning method, but a way to combine an existing action recognizer which satisfies certain properties with existing object-detectors to perform simultaneous tracking and action recognition. The action models used by Yu and Siskind [20], Siddharth *et al.* [21], and Barbu *et al.* [22] represent actions only through simple features such as the gross motion and relative position of the action participants. They therefore can support only rather simple action classes, such as *pick up* and *approach*, which can be represented by such features. They rely on pretrained generic person and object detectors to provide detections to the simultaneous tracking and action recognition process. However, the actions in many standard action recognition datasets, like UCF Sports [2], depict people in rather unusual poses, which result in a failure of generic person detectors. For these reasons, the Event Tracker framework has not been shown to work on standard action recognition datasets.

A. Contribution

We present a new method that performs action recognition and localization using intuitively meaningful models, while still achieving competitive performance on complex actions and on standard action recognition datasets. This is made possible by a novel learning method, which integrates detection, localization, and action recognition, training a combined model to maximize action classification performance. Further, it does so in a way compatible with the Event Tracker, allowing it to be used as a tool, and showing that it can be extended to complex actions. This new method models the time series of the appearance and motion of the people and/or objects that participate in an action. This is done by learning, for each action class, a temporal sequence of actor-centered histogram of oriented gradients (HOG) object detectors [23] and optical-flow-based motion-pattern detectors. These models are retinotopic. That is, like the receptors of the retina, they are spatially organized in a dense grid. They are therefore able to represent coarse spatial organization, which is lost in sparse interest-point-based methods, and can be meaningfully visualized and understood by humans as we demonstrate in Fig. 1.

Each of these models represents the motion or appearance of the actor during a highly distinguishing temporal subsequence of an action. A sequence of such models represents the changing characteristics of an actor in a video as they evolve

over time during that action. Each action class is modeled with a hidden Markov model (HMM) [24] whose state-conditioned output models are the appearance and motion detectors. Thus, each HMM state represents one of the characteristic appearances and/or motion patterns, and the sequence of HMM states models the changing appearance and motion over time. Such an HMM is learned automatically through a novel discriminative training procedure on both positive and negative samples of an action class. This results in simultaneously learning which subsequences of the action are useful for recognition, the order in which they take place, and the appearance and motion detectors associated with each, without the need for manual key framing or other extensive manual annotations. We require only bounding boxes around the actor in the training videos. We further show how our learned models allow for improvements to the Event Tracker to perform simultaneous object detection, tracking, and recognition of actions.

II. Related Work

There is recent prior work that also attempts to model time series in general [25], which models actions in video as temporal sequences, or which attempts to model the appearance or pose of the participants. Tang *et al.* [26] break videos into fixed-length segments, on which BOW features are computed, and then use an HMM to model the sequence of segments. However, the segment length was manually specified, varying by dataset, and was very long, reducing each video to a very small number of segments, and largely providing the HMM with the latent state assignment information. Niebles *et al.* [5] model actions as sequences of motion segments at varying time scales. However, they represent each segment as a histogram of sparse STIP features, whereas our method uses a dense sampling grid to represent the coarse-scale appearance and motion. Like Tang *et al.* [26], Liang *et al.* [27] also break videos into fixed-length segments. They employ a spatiotemporal AND–OR graph, which models actions as a sequence of disjunctions of local BOW models over HOG features and histogram of oriented flow (HOF) [28] features computed at STIP interest points. They attempt to account for spatial structure by explicitly modeling the relative positions of these local models. Wang *et al.* [8] explicitly model human joint pose, mine for clusters of pose sequences, and then reduce videos to BOW histograms for classification with a temporally invariant SVM. Wu and Shao [29] also classify video with explicitly modeled human pose. They model the sequence of human skeleton estimates extracted from 3-D Red Green Blue Depth sensor data through an HMM with a series of neural networks. In contrast, our method implicitly models pose through appearance and motion models, allowing application to appearance change of nonhuman objects, and models the temporal sequence of changing appearance rather than reducing it to a histogram like the one in [8]. Barbu *et al.* [22] also employ an HMM model, but do not represent the appearance or motion patterns of the action. They instead rely on features computed directly from bounding boxes themselves, ignoring the content of the video inside these boxes. They obtain such boxes from pretrained object models that are used only for detection and are not state conditioned. Thus, while they do model the sequence of changing gross object motion and of spatiotemporal relationships among action participants, they do not model the sequence of changing object appearances or motion patterns which cannot be captured by the gross movement of the bounding detection boxes. Banerjee and Nevatia [30] present a method based on key-pose filters that model both appearance and motion. Whereas we model the temporal sequence of action state and infer the state at each frame with an HMM, they explicitly model and infer the position of the key frames with a hidden conditional random field. This model cannot localize the action and instead depends on a separate pedestrian tracker, whereas we learn models that enable simultaneous detection, tracking, and action recognition.

The two methods that are most similar in spirit to our work are [31] and [32]. Tian *et al.* [31] present a spatiotemporal extension to the deformable part model (DPM) [33] intended to serve as an action detector. They train a template for each action that includes a cuboid root filter and a number of displaceable cuboid part filters. Because the filters are cuboids, they must be large enough to encompass the entire space traversed by a moving object over the course of the entire action, whereas our models track the object through time, allowing the filters to be centered on the image region where an object is present in a specific frame in the video. Their method must also be provided with an annotation of the temporal extent of a single cycle of each training action, such as a single *jumping jack*, whereas our method can automatically handle such repetitions without any annotation. Yao *et al.* [32] present another recent method that extends the ideas of the DPM object detector to action recognition. They use an HMM to model the sequence of such models, each of which contains a HOG root filter and a set of explicitly labeled body-part models, which each consist of a HOG model and an HOF model. For training, they require manual annotation and labeling of the body parts. We require no such part annotations. Whereas our training procedure can automatically cluster the training video frames into a sequence of discriminative poses to be modeled, they must do so by performing clustering on these manual part annotations.

III. Action Model

The sequence of each action class is modeled by an HMM with N states. An HMM assumes the existence of a hidden-state sequence X_t, for $t = 1, \ldots, T$, and that a series of observations D_t was sampled from state-conditioned output distributions $b_i(D_t) = P(D_t|X_t = i)$. An HMM's parameter set λ consists of an initial state distribution π, specifying the probability of beginning in each state, an N by N transition matrix A, whose elements a_{ij} specify the probability of transitioning from state i at time t to state j at time $t + 1$, and the parameters of the set of state output models b_i, for $i = 0, \ldots, N$, which depend on the particular output models used.

The likelihood $l = P(D_1, \ldots, D_T|\lambda)$ of the data for the given HMM can be computed efficiently via the forward algorithm [24]. It is computed as $l = \sum_{i=1}^{N} \alpha_T(i)$, where

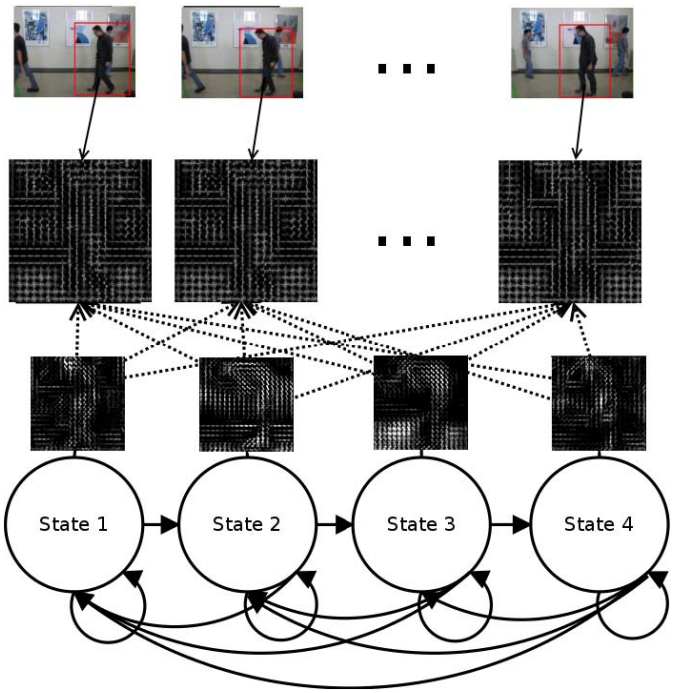


Fig. 2. Diagram illustrating the presented method. Appearance (and motion, not shown) features (second row) are extracted from the images within bounding boxes in the video frames (top row). These features are matched against the output models (third row) associated with each state (bottom row) of an HMM, which models a particular action class. The sequence of states and thus the sequence of appearances and motions associated with that action class is modeled with a transition distribution.

$\alpha_t(i) = P(D_1, \ldots, D_t, X_t = i|\lambda)$, and is computed recursively via $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1} a_{ij} b_j(D_t)$. Given a uniform prior over action classes, a video can be classified by computing the likelihood of the HMM for each class and choosing the class with the highest likelihood.

In the presented method, as shown in Fig. 2, the observations D_t of a video are a sequence of dense $n \times n$ grids of HOG and/or HOF feature vectors extracted from each frame of the video along a sequence of bounding boxes around the actor. For training, we assume that such boxes are available by some means, either manually specified or produced automatically. Such a sequence can be produced automatically by a variety of methods: background subtraction, adjacent-frame image differencing, or more powerful tracking methods. For simple datasets such as Weizmann [3], background subtraction is sufficient to find the actor and such background masks are provided with the dataset. We automatically extract bounding boxes from these masks for use in training. Other datasets, such as UCF Sports, UCF11 [16], and UT-interaction [34], provide such boxes directly. At test time, we obtain such bounding boxes automatically. Our trained models allow us to employ a modified version of the Event Tracker [22] to perform simultaneous detection-based tracking and action recognition, which is described further in Section IV.

Following [23], our HOG features use nine orientation bins. However, rather than using four such histograms normalized according to different blocks, which would result in a 36-element vector for each block, we only normalize each histogram once in each block in order to reduce the feature

Fig. 3. Visualization of the learned HOF models from *jumping jack* (Weizmann) and learned HOG models from *swing* (UCF Sports), and *open drawer* (Office11). The *jumping jack* models closely follow the person's sequence of body and limb movements, while the *swing* and *open drawer* models closely follow the sequence of edge structure characteristic of each action. Each visualization was produced on an unseen test video using the forward–backward algorithm to produce a weighted assignment of each frame to each HMM state and by rendering on each frame the model that belongs to the HMM state with the highest weight.

dimensionality. Thus, each HOG grid position consists of a nine-element vector. Our flow feature is similar to HOF. While HOF involves computing differential flow, breaking the image region into overlapping patches, binning the differential flow by orientation in each patch, and then normalizing each patch, our feature, in contrast, is computed for each patch by taking the average flow and then binning its horizontal and vertical components each into three bins, resulting in a six-element vector for each grid position. The feature vectors at the grid positions are concatenated into a single HOG vector and single HOF vector per frame. Thus, the feature vector D_t for each frame is of length $9n_h^2$ for an $n_h \times n_h$ HOG grid and is of length $6n_f^2$ for an $n_f \times n_f$ HOF grid.

The HMM state output model $b_j(D_t)$ for a particular state j and the normalized HOG or HOF feature vector for a particular frame t is computed with a sigmoided dot product

$$b_j(D_t) = \frac{1}{1 - \exp\ (-h \sum_i w_{ji} D_{ti})}$$

where h is a smoothing parameter of the sigmoid and w_j is a weight vector of length $9n_h^2$ for a HOG model and $6n_f^2$ for an HOF model.

We further take symmetry into account when computing $b_j(D_t)$. This is done by computing both $b_j(D_t)$ using the original features D_t and using those features D'_t, which were computed from the video after reflecting it along the vertical axis. For each state j and for each frame t, the maximum of these two scores is taken to be $b_j(D_t)$. This provides a degree of viewpoint invariance, allowing a single model to match an action when viewed from either side, or to switch from one orientation to another, as in row two of Fig. 3.

These output models do not satisfy the sum-to-one constraint, and therefore they are not probability distributions, but rather single-layer neural networks. Niles and Silverman [35]

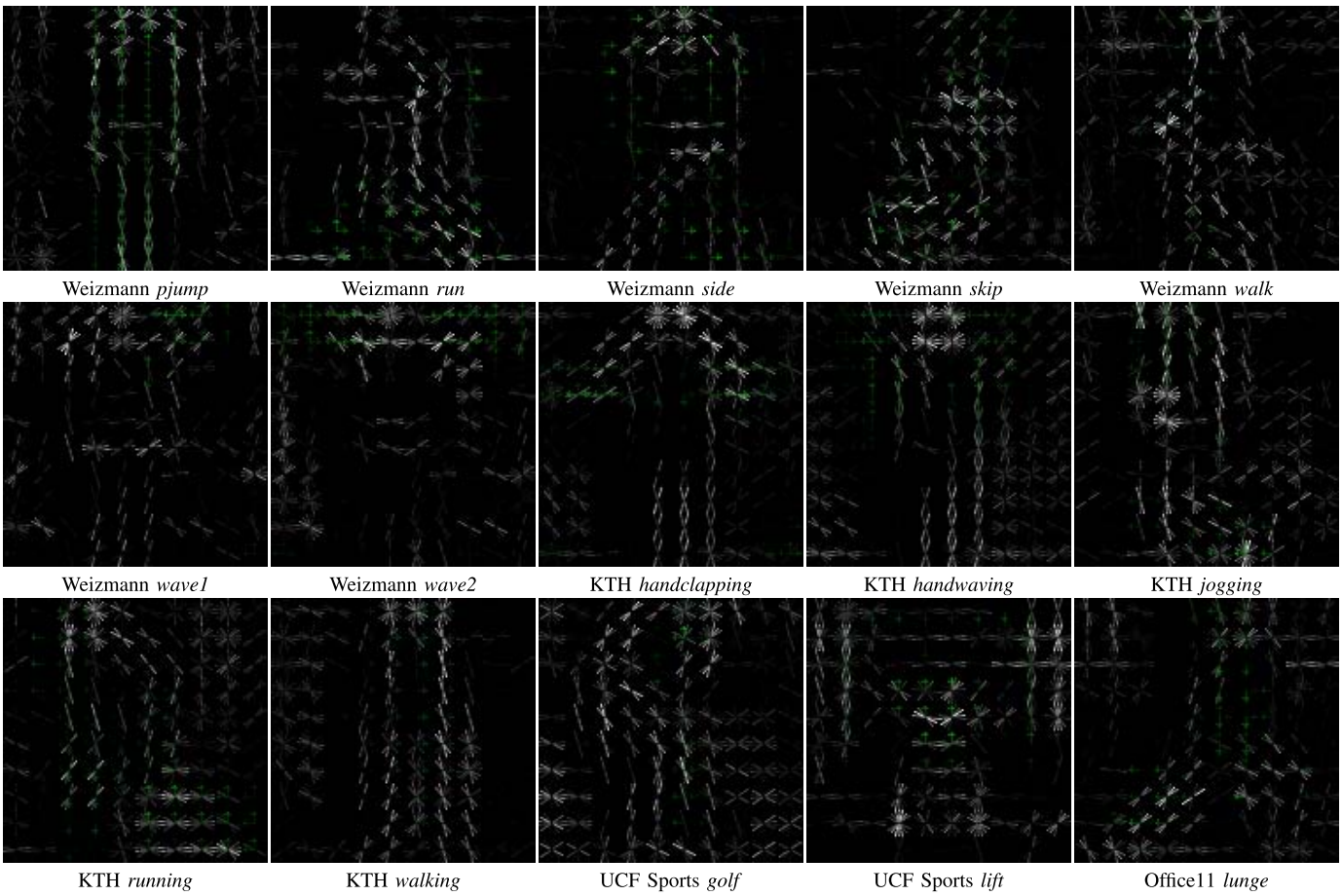


Fig. 4. Visualization of the learned appearance and motion models from example states from a variety of action models. These each depict the edge orientations (white lines) and motion orientations (green arrows) associated with a particular state of a particular action. These examples all show the general form of a person, but in different postures, and with motion in different places and orientations. For example, the UCF Sports *golf* model (bottom row, third column) shows the form of a person bent over while putting, with horizontal motion at the position of the club, while the Weizmann *wave1* model (second row far left) shows a person standing up straight, with motion corresponding to waving a single hand in the air.

showed that HMMs can be viewed as a special case of neural networks and that, in this context, relaxing the sum-to-one constraints is natural, and may result in a better classifier. However, doing so does remove the probabilistic interpretation of the HMM. Therefore, we refer to the output of the HMM forward algorithm as an HMM score rather than a likelihood.

HOG object detectors generally also use a dot product model. Thus, each state's HOG output model can be thought of as an object detector, and its HOF output model as a motion-pattern detector. The use of object and motion detectors as HMM output models allows an action to be recognized by a sequence of appearances, by a sequence of motion patterns, or by a combination of the two. For example, an *open drawer* action could be recognized by the appearance of a closed drawer, as determined by a high scoring closed-drawer detector in one state, followed by a transition to the appearance of an open drawer, as determined by a high scoring open-drawer detector in another state. Fig. 3 shows a visualization of such a model on an unseen test video. Fig. 3 was generated automatically by rendering, on a given frame t, the HOG output model from the state j with the highest $P(X_t = j|D_1, \ldots, D_T, \lambda)$ as computed by the forward–backward algorithm [24]. The white lines indicate HOG model weight at an orientation and location. Brighter lines indicate more weight, while those with nonpositive weight are not drawn. Fig. 3 also shows a similar visualization of the HOF model for *jumping jack* on an unseen test video. A green arrow indicates the weight of the model at a particular position and orientation. The brighter arrows have higher weight, while those with negative or zero weight are not drawn. Fig. 4 shows visualizations of a number of example output models showing both HOG and flow.

Because the opening of a drawer involves simple linear motion that depends on the viewpoint, such an action might not be easily recognizable with motion features. Other actions, such as the *skateboarding* and *walking* classes in UCF Sports, which have very similar appearances, might be more easily distinguished by the difference in motion of the person's legs. The use of both appearance and motion detectors allows such differences to be modeled. Other actions might involve the same appearance or motion, but in a different sequence, such as a *close drawer* action, which involves the same appearances as an *open drawer* action, but in the opposite order. The combination of the HMM's state transition matrix with the

detectors in its output models allows such actions to be distinguished by the order in which these appearances take place. BOW approaches would not be able to make such a distinction.

IV. Simultaneous Tracking and Action Recognition

At test time, the goal of our system is to both recognize and localize the actions occurring in video. We use a modification to the Event Tracker [22] to combine our action model and its internal object detectors into an integrated simultaneous detection, tracking, and action recognition system. Given a video, a set of scored object-detection boxes in each video frame, and an HMM action model, the Event Tracker produces a track composed of a single detection in each frame along with a score indicating how well the track depicts the action based on the HMM action model.

The Event Tracker operates by globally optimizing a joint tracking and action recognition objective function

$$\max_{\substack{q^1,\ldots,q^T \\ j^1,\ldots,j^T}} \sum_{t=1}^{T} f\left(p^t_{q^t}\right) + \sum_{t=2}^{T} g\left(p^{t-1}_{q^{t-1}}, p^t_{q^t}\right) + \sum_{t=1}^{T} e\left(j^t, p^t_{q^t}\right) + \sum_{t=2}^{T} a(j^{t-1}, j^t) \tag{1}$$

where the four terms are $f(p)$, the detection score of each box p in the sequence, $g(p, p')$, the track-coherency score between each pair of boxes p and p' in adjacent frames, $e(j, p)$, the HMM state output model score of the features in box p scored against the state j, and $a(j, j')$, the state-transition score between the states j and j' assigned to each pair of adjacent frames. Thus, it simultaneously finds the best possible sequence of detections and the best possible sequence of HMM states, producing the maximum *a posteriori* estimate of the HMM score for the given optimal track.

The Event Tracker has three steps:

1) obtaining an over-generated set of scored object-detection boxes;
2) constructing a lattice through the cross-product of detections and HMM states;
3) using the Viterbi algorithm [36] to find the optimal path through the lattice.

The Viterbi algorithm uses dynamic programming to efficiently find the optimal path through a lattice with unary vertex costs and binary edge costs. Because the HMM and tracking cost functions each consist of unary and binary terms, a combined lattice with unary and binary costs acting on the cross product of all box state pairs can be generated. The Viterbi algorithm then finds the optimal combined sequence of detections and HMM states, along with the corresponding cost.

Joint optimization of the combined cost function allows the action model to bias the tracker. This is advantageous when compared with the use of tracking and action recognition in independent steps. For instance, only one of several people in a particular video might be performing a given action. If that person is also difficult to track, perhaps partially occluded or in the background, an independent tracker would have difficulty in finding that person amidst the other, more easily tracked people. However, the Event Tracker allows the high-level information reflected in the action model to bias the low-level tracker toward the person that best exhibits the characteristics of that action.

Our use of HMMs allows us to make use of the Event Tracker. In addition, the availability of HMM state-conditioned object models produced by our method allows us to improve on it.

In Step 1) of the Event Tracker, Barbu *et al.* [22] use generic pretrained DPM object detectors to generate a large number of detection boxes. Then they discard all but the top P detections in each frame based on their detection scores. In contrast, we produce the detections using the action-specific HMM state output models, which have been trained by our system to maximize recognition performance. We do so by running our state-conditioned output models as object detectors, producing, as in the DPM, a score pyramid in each frame corresponding to the scores of boxes at each possible box position at a variety of scales. Thus, for each position in the pyramid, which corresponds to a box with a particular position and scale, we have the output score for each HMM state. Next, the top P boxes in each frame are kept for each HMM state based on this score. This differs from [22], where the top P boxes are kept solely based on a generic object-detector score. Step 2) differs in that our detection boxes were produced using state-conditioned models, and thus each box has an associated HMM state. We therefore add a constraint to the lattice that forces the chosen HMM state in a given frame to match the state associated with the chosen box in that frame.

These modifications allow further influence of the high-level action model on not only the tracker but also the object-detection process. This has two advantages. First, the action-specific HMM state-conditioned object models can detect people or objects in poses or configurations that are generally unusual, and thus score poorly with a generic detector, but might be common or even characteristic of the action in question. Thus, actions that are very difficult to track with generic models because the person takes an unusual pose become straightforward to track with our method. Further, the inclusion of other features, like motion, into the score prior to thresholding the boxes prevents the loss of detections that have a comparatively poor object appearance score, but exhibit the desired motion so well that they are part of the optimal track.

We run our modified Event Tracker using the model for each action class. For each action, this yields the optimal track, along with a score indicating how well each such optimal track depicts the action. If there is no person or other agent performing a given action in the video, the associated score will be low. We perform classification and localization by choosing the class and track from the model with the highest score.

V. Training the Models

The action models are discriminatively trained through gradient descent with a maximum-margin objective function.

Such an objective encourages each model to score highly on training samples of its own class and poorly on others. It also encourages each model to have higher score on training samples with matching class than the other models on those same samples. Thus, optimizing such a function aims to improve classification performance.

The objective function used, O, is a sum over individual training video costs O_v. Each video cost O_v is the square of a soft version of the multiclass hinge loss function used by Crammer and Singer [37] for multiclass SVMs

$$O_v = \max(0, 1 + \textsc{SoftMax}(\{s_{vr} | r \neq k_v\}) - s_{vk_v})^2$$

where s_{vr} is the HMM score (in log space) of video v with HMM r, k_v is the index of the HMM whose class matches the class of video v, and

$$\textsc{SoftMax}(S) = y \log\left(\sum_{i=1}^{U} \exp\left(\frac{S_i}{y}\right)\right)$$

where U is the dimension of S and y is a smoothing constant. Thus, when the correct HMM score s_{vk_v} on a video v is greater than the SoftMax of the incorrect scores s_{vr} by a margin greater than 1, the video is classified correctly with a margin of at least 1 and the cost is zero. If the margin is less than 1, then

$$O_v = \left(1 + y \log\left(\sum_{\substack{r=1 \\ r \neq k_v}}^{C} \exp\left(\frac{s_{vr}}{y}\right)\right) - s_{vk_v}\right)^2$$

where C is the number of classes.

To optimize O, we compute its gradient with respect to the HMM parameters and perform gradient descent. We use the chain rule to compute the gradient as a function of the gradients of the individual video costs with respect to the parameters of individual HMMs. The derivative $(\partial O / \partial x_{rd})$ of O with respect to x_{rd}, the dth parameter of HMM r, is

$$\frac{\partial O}{\partial x_{rd}} = \sum_v \frac{\partial O_v}{\partial x_{rd}}$$

where by the chain rule

$$\frac{\partial O_v}{\partial x_{rd}} = \frac{\partial O_v}{\partial s_{vr}} \frac{\partial s_{vr}}{\partial x_{rd}}$$

and $(\partial O_v / \partial s_{vr})$ depends on whether $r = k_v$, i.e., whether this HMM's class matches that of video v.

If $r = k_v$, then $(\partial O_v / \partial s_{vr}) = -2\sqrt{O_v}$; otherwise $(\partial O_v / \partial s_{vr}) = 2\sqrt{O_v}(\partial \textsc{SoftMax}_u(s_{vu}) / \partial s_{vr})$. After simplifying, we obtain

$$\frac{\partial O}{\partial x_{rd}} = \sum_v 2Z\sqrt{O_v} \frac{\partial s_{vr}}{\partial x_{rd}}$$

where $Z = -1$ if $r = k_v$, otherwise

$$Z = \frac{s_{vr} \exp(s_{vr})}{\sum_{\substack{u=1 \\ u \neq k_v}}^{C} \exp\left(\frac{s_{vu}}{y}\right)}$$

and where $(\partial s_{vr} / \partial x_{rd})$ is the derivative of the score computed by the HMM forward algorithm on video v with HMM r, with respect to parameter d. Using the techniques of reverse-mode automatic differentiation [38], which is a systematic way to apply the chain rule, the entire gradient of s_{vr} with respect to all HMM parameters can be computed efficiently within a small constant factor of the time needed to compute the score s_{vr}. This is done by taking a forward pass through the program and computing the value of the function while storing intermediate values, and then taking a backward pass and computing the derivatives using these stored intermediate values. This process is analogous to back propagation in neural networks [39].

The parameters of an HMM are composed of three parts, π, a, and w. The derivative $\bar{a}_{ij}$ of an HMM's score on a video with features D_t with respect to the elements of its transition matrix a_{ij}, and $\bar{\pi}_i$, the derivative with respect to the elements of its initial state distribution π_i, are given by

$$\bar{a}_{ij} = \sum_t \bar{\alpha}_{tj} \alpha_{(t-1)i}$$
$$\bar{\pi}_i = \bar{\alpha}_{0i} b_i(D_0)$$

where $\bar{\alpha}_{ti}$, the derivative of the HMM score with respect to α_{ti}, is given by

$$\bar{\alpha}_{ti} = \sum_j \bar{\alpha}_{(t+1)j} b_j(D_t) a_{ij}.$$

The derivative $\bar{w}_{jm}$ of the HMM score with respect to w_{jm}, the mth weight parameter of the output model of state j, is given by

$$\bar{w}_{jm} = \sum_t \frac{h D_{tm} \bar{b}_{tj} \exp(-h\gamma_{tj})}{(1 + \exp(-h\gamma_{tj}))^2}$$

where h is the smoothing parameter of the sigmoid function, D_{tm} is the value of the mth element of the feature vector D_t at frame t, γ_{tj} is the value of the dot product between D_t and w_j, and where $\bar{b}_{jt}$, the derivative of the HMM score with respect to $b_j(D_t)$ is given by

$$\bar{b}_{jt} = \begin{cases} \bar{\alpha}_{tj} \sum_i a_{ij} \alpha_{(t-1)i}, & t > 0 \\ \bar{\alpha}_{0j} \pi_j, & t = 0. \end{cases}$$

During optimization, the output models are constrained to have unit magnitude. This is accomplished in two ways. First, the gradient of the objective function is made to take the normalization into account. This is done by creating an augmented objective function that normalizes the output models before passing them into the original objective function. We then take the gradient of this augmented objective function. We let

$$w_{jm} = \frac{w'_{jm}}{\sum_z w'_{jz}}.$$

Then, the desired derivative becomes

$$\bar{w}'_{jm} = \frac{\bar{w}_{jm}}{\sum_z w'_{jz}} - \frac{\sum_z \bar{w}_{jz} w_{jm}}{(\sum_z w_{jz})^2}.$$

The optimization procedure breaks the parameter space into two subspaces, the space of initial-state parameters π and

transition parameters a_{ij} and the space of output-model parameters w_{jm}. It alternates between taking steps in each space. The initial-state parameters π and transition parameters a_{ij} are updated using the growth transform [40], while the output models are updated using gradient descent with a dynamic step size. While the above gradient points in a direction for which an infinitesimally small step will maintain the magnitude of the output models, the finite step size taken by gradient descent results in small changes to the magnitude. To compensate, the output models are also renormalized after each gradient-descent step.

The effect of this optimization process is that the HMM parameters are updated in a way that maximally improves the discriminative power of the HMM score. The state-conditioned output models each gradually become more distinct and specialized toward a particular temporal subsequence of the action as the weighted assignment of frames to states becomes less uniform. This results in the HMM states for a given action model matching the sequence of the most distinct appearance and/or motion that occurs in that action class, and rejecting the appearance and motion that occurs in other actions as encoded in negative training samples. The latent HMM states are automatically assigned to these distinctive moments in the video without the need for manual annotation of such. Fig. 3 shows visualizations of the models learned for *jumping jack* from Weizmann, for *swing* from UCF Sports, and for *open drawer* from Office11, a new dataset that we have filmed.

In contrast to pretraining object models and treating the detector output scores as features, which would require manual assignment of training frames to each model, the use of detectors as HMM state-conditioned output models allows the sequence of detectors to be trained automatically as part of the action model. Thus, the latent state assignment is learned without supervision in conjunction with the detectors themselves.

Training the detectors in conjunction with an HMM also makes it possible for the detectors to be trained in a discriminative fashion specifically to maximize action class discrimination. In contrast, an object detector is generally trained to maximize detection of that object, and therefore seeks to score highly on all positive training frames. This is achieved in methods such as the DPM by the use of a disjunction of root filters. However, some poses are not useful for discriminating between actions, such as a person standing upright, who might be preparing to *bend*, *kick*, or *jump*. Training a DPM model for each action would result in each model including a root filter corresponding to a person standing before the action. In contrast, the simultaneous training of the detectors and the HMM to maximize discrimination between the actions allows nondiscriminative poses to be ignored automatically as the detectors learn to model the most discriminative sequence.

A. Mining for Difficult Negatives

The HMM scores s_{vr} depend on the sequence of boxes on which the features have been computed. For the positive samples, i.e., when $r = k_v$, the provided track of bounding boxes around the actors are initially used to compute s_{vr}. After the models have been partially trained, the Event Tracker is used to find the highest scoring track through each training video with the model whose class matches that video. The score s_{vr} for $r = k_v$ is then taken to be a weighted mean between that computed on the provided track and that computed on the automatically determined track. This allows the system to handle errors and misalignments in the tracks provided for training.

We obtain additional negative samples by running our partially trained action detector on the training videos similar to the way that Felzenszwalb *et al.* [33] obtain such for training an object detector by running it on training images. To produce difficult negative samples, our modified Event Tracker is used to find the highest scoring tracks in each video v using all nonmatching models of class r such that $r \neq k_v$. These tracks are used to compute the HMM scores s_{vr} for the nonmatching models for each training video. Thus, the objective O pushes each model to score as high as possible on the tracks through the positive videos and as low as possible on all tracks through other videos. The use of such difficult negative tracks is very important. Without them, the system can easily classify the training videos on the provided tracks and will not learn models that can classify videos with automatically produced tracks, as will be shown in Section VII.

B. Implementation Details

In addition to the output parameters w and the transition parameters a, which are fully learned, our system contains eight hyperparameters, which are not learned. There are four hyperparameters in the training procedure: the iteration delay until the onset of retracking the training videos, the frequency of retracking, the automatic track queue size, and the smoothing parameter of the SOFTMAX in the objective function. There are four hyperparameters in the model: the number N of states, the grid sizes n_h and n_f for the HOG and HOF models, and the smoothing parameter h to the output model sigmoid.

SOFTMAX is used in place of MAX in the objective function in order to make the objective smooth. It removes the discontinuities in the gradient that would otherwise be caused by MAX. The smoothing constant y in SOFTMAX, which controls how closely it approximates MAX, was fixed at 0.1. This small value was chosen so that the value of SOFTMAX is close to the true MAX.

At the start of training, the models are initialized in the following way. The output models are initialized randomly with elements sampled uniformly in $[-1, 1]$. These are then shifted so that each output model sums to zero and subsequently normalized to have unit magnitude. The shift is done in order to prevent some output models from having lower mean and thus scoring poorly on all feature vectors. This would be undesirable and potentially cause the model to learn to skip that state before it has a chance to adapt to the data.

The transition matrices are initialized to a chain structure that allows arbitrary backward transitions but only allows transitions forward by a single state, i.e., the states are labeled $0, \ldots, N-1$ and state i can transition to all states $j \leq i+1$, where the probabilities are uniform among allowed transitions. Thus, skipping states is initially disallowed, but arbitrary

looping is possible. The HMMs are constrained to begin in the first state 0 and end in the last state $N-1$. This is done so that the HMM cannot return a high score on a video that depicts only a subsequence of an action, but must find the full sequence. On such a video, the low score of states corresponding to undepicted parts of the action will result in a poor overall score. The number of states N was set to 4 by looking at the actions in the datasets and observing that there are generally fewer than four major human-discernible poses. Because the model can learn that not all states are necessary, we used this upper bound for all datasets and for all classes.

The computation needed to mine a training video for high-scoring negative tracks, $O(TNB(n_f^2 + n_h^2))$ (where B denotes the number of locations in the image pyramid), dominates that needed to compute the gradient and perform a parameter update, $O(TN(n_f^2 + n_h^2) + TN^2)$, because of the comparatively compute-intensive step of running the HMM state-conditioned models as detectors at a large number B of positions and scales in each frame. Therefore, new negative tracks are not produced every iteration of training. This is done with a retracking frequency of once per 1000 iterations instead, so that the models are substantially modified and will produce a new set of negative tracks each time. To avoid cyclic behavior caused by abrupt changes in the set of negatives, we do not discard the previous set of negative tracks each time new tracks are found. Instead, we maintain a queue of three such sets, discarding the oldest set when the queue becomes full. Thus, each set of negative tracks is used for 3000 updates. Positive mined tracks are not used until the model has been substantially trained in order to avoid using garbage tracks as positive samples. In practice, they have been used after an onset delay of 2000 iterations, and the weight given to the mined positives has been half the accuracy on the training set.

As in [23], our HOG features are computed using 8×8 pixel blocks. During training, the image inside each bounding box is scaled so that the number of 8×8 pixel blocks in each box matches the size of the grid of the output models. When performing automatic detection and tracking, the computation of the image scale pyramid controls the size of the boxes. Larger values of the grid sizes n_h and n_f allow the modeling of finer grained spatial information, but increase the runtime, particularly for running all the output models as detectors. The balance between details in the model and computational efficiency is thus determined by the scale of the participants in the dataset. We want the grid size to be large enough to match the scale of the participants in the training videos: if it is smaller, the boxes will be scaled down and details will be lost; if it is larger, the boxes will be scaled up, and the extra detail will be redundant and add unnecessary computation. For all datasets except for the LCA dataset, we used $n_h = 10$ and $n_f = 10$, yielding 10×10 grids of HOG and HOF features. For the Large Continuous Action (LCA) dataset, we used $n_h = 6$ because the participants tend to be at a smaller scale, and the larger dataset makes computational efficiency more important. The smoothing parameter of the output model sigmoid h controls its steepness. It was chosen to be ten so that the output for the dot product of two random unit vectors is usually close to zero. Since our method performs well with these intuitively chosen values on all the datasets, we have not done a search through the space of these parameters to maximize performance. It is possible that some other values produce better results.

VI. New Dataset

We filmed a new dataset, Office11, to highlight the power of the presented method. We are interested in recognizing and localizing specific actions, such as might be used for a robotics or surveillance application. Many other methods are focused on Web video and general pattern recognition. These different focuses result in different challenges. Videos taken from movies or from Web sites like YouTube were produced for human consumption, and contain scene transitions, highly zoomed in camerawork, and unknown camera movement. However, a surveillance camera or robot would not be presented with these difficulties, but instead have other challenges, like occlusion and multiple simultaneous actions. Localization of detected objects and actions would be also be more important than in a Web-based setting. The types of actions to recognize in these two contexts also differ. Whereas it might be useful to categorize and retrieve YouTube videos based on general scene categories like *military parade*, *wedding*, or *horse race*, specific actions like a person *marching*, *giving* an object, or *riding* a horse would be more relevant to a surveillance system or robot.

Our dataset is meant to better reflect these challenges and attempts to remove two shortcomings of prior standard datasets. First, most action recognition datasets focus on people and do not include actions that involve the manipulation of objects. Second, many action recognition datasets have strong correlation between the background and the action taking place. For example, in UCF Sports, *diving* always occurs in an Olympic swimming pool. No other actions take place in a pool, so identification of the background provides a powerful cue to the action, "for instance, *v_spiking* normally happens in a crowd of people and *diving* happens in a pool. This is common for professional sport actions that take place in highly structured environments" [16]. Similarly, in UCF11, "basketball shooting and volleyball actions are also confused in some cases: this is largely because most of the time, the basketball and volleyball sports use very similar courts" [17]. While some consider the use of contextual information to inform the classification process to be a virtue, we desire to recognize actions in a semantically meaningful way, solely by recognizing and localizing the action itself, which is more true to the goal of video action recognition.

Office11 was filmed to remove these two shortcomings. It avoids spurious correlation between actions and backgrounds and includes actions that involve both manipulation of objects and human body-posture changes. The dataset includes 11 action classes: *bend*, *kick*, *lunge*, *wave*, *open drawer*, *close drawer*, *answer phone*, *hang up phone*, *talk on phone*, *operate hole punch*, and *knock over cup*. Some actions are similar to those in other datasets, such as *bend*, *kick*, and *wave*, but they all take place in the same background, with several people constantly walking around in the field of view, often occluding each other and depicting parts of other

actions as one person performs the entirety of the desired action. The *open drawer* and *close drawer* actions take place in the same cluttered office with seven different drawers, which are opened and closed. The remaining actions also take place in a cluttered office. In this dataset, the background cannot be used to distinguish *bend*, *kick*, *lunge*, and *wave* from each other, nor to distinguish *open drawer* from *close drawer*, or discriminate *answer phone*, *hang up phone*, *talk on phone*, *operate hole punch*, or *knock over cup* from each other. The dataset is slightly bigger than UCF Sports, with 179 videos.

VII. Experiments

The performance of the presented method was compared with state-of-the-art prior methods on three standard datasets (Weizmann, KTH [4], and UCF Sports), as well as on the LCA dataset [41] and Office11, and was found to be competitive with recent methods. We also compare with several baseline methods: the HOG baseline, the HOF baseline, and the 3-Stage baseline. The HOG and HOF baselines use the method described in this paper, but with only a single HMM state and using only the HOG or HOF feature, respectively. Both baselines use the same combined automatic tracking method described here. Their poor relative classification accuracy shows that the performance of our method is not simply due to the strength of the features, but due to their combination together with the sequence model. The 3-Stage baseline classifies each video by detecting the person, tracking them, and running the classifier of the presented method in sequence. The DPM object detector produces the top person detection in the first frame of the video, initializing the tracking–learning–detection (TLD) tracker of Kalal *et al.* [19]. Finally, this track is passed to the same trained action recognition models used to evaluate our method. The poor relative performance of this baseline shows the importance of the integration of the detection, tracking, and classification systems. The tracks produced by this baseline are also used to compare the localization performance.

The classification results are reported in Table I and Fig. 5, and the localization accuracy results are reported in Fig. 6. The standard leave-one-actor-out evaluation protocol from prior publication was used for the Weizmann dataset. The KTH dataset was run with the train–test split associated with the dataset and used by Yao *et al.* [32]. The UCF Sports dataset was run with 16-fold cross validation. To compare with the prior work on the LCA dataset, we used the same 70:30 Train–test split used to produce the results in [41]. For Weizmann, UCF Sports, and Office11, the bounding boxes for training were determined using information provided with the dataset. For KTH, we used the boxes made available by Lin *et al.* [42]. For the LCA dataset, as no such information was provided, the boxes used as input for training were generated automatically. For each dataset, we report the accuracy of our method on the test videos with boxes produced automatically by our learned models through simultaneous tracking and action recognition. We also report classification accuracy with manually annotated boxes on the Weizmann, KTH, UCF Sports, and Office11 datasets, in order to observe performance in the absence of difficulties caused by tracking. As no such manual boxes are available for the LCA dataset, we also report results using boxes

TABLE I

Classification Accuracy of Presented Method Compared With Recent Prior Methods (All Published Since 2009) on Weizmann, KTH, UCF Sports, LCA, and Office11 Datasets. Our Numbers Are Uniformly Reported to One Decimal Place. Prior Results Are Uniformly Reported to Published Precision

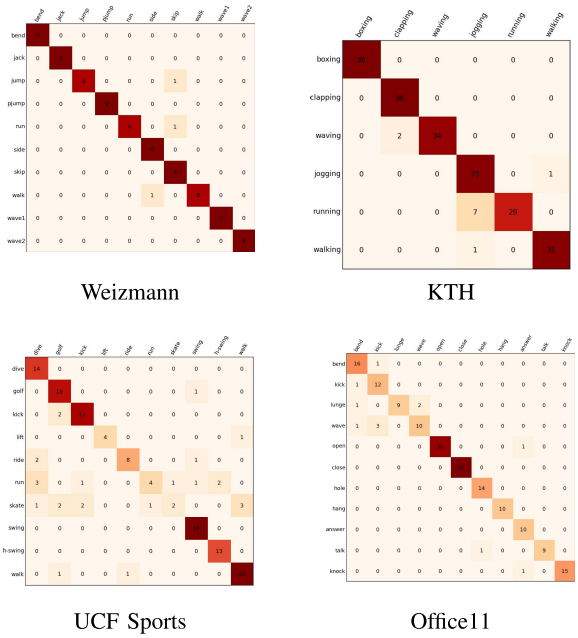
Weizmann	Accuracy (percent)
Tian *et al.* (2013) [31] (with parts)	100
Oreifej & Shah (2014) [12]	92.8
Tian *et al.* (2013) [31] (without parts)	92.4
Nagar & Agrawal (2014) [43]	92
presented method (automatic tracks)	96.7
presented method (manual tracks)	97.8
presented method (without negative mining)	12.2
HOG baseline	75.5
HOF baseline	92.2
3-Stage-Baseline	43.3

KTH	Accuracy (percent)
Yao *et al.* (2014) [32] (with annotated parts)	94.53
Yao *et al.* (2014) [32] (without annotated parts)	84.70
presented method (automatic tracks)	94.9
presented method (manual tracks)	91.2
presented method (without negative mining)	44.9
HOG baseline	62.5
HOF baseline	82.8
3-Stage-Baseline	59.7

UCF Sports	Accuracy (percent)
Sadanand & Corso (2012) [44]	95.0
Yuan *et al.* (2013) [10]	92.67
Wu *et al.* (2011) [7]	91.3
Wang *et al.* (2013) [8]	90.22
Oreifej & Shah (2014) [12]	89.7
Wang *et al.* (2013) [13]	89.1
Everts *et al.* (2013) [9]	85.7
Yuan *et al.* (2013) [11]	87.33
Tian *et al.* (2013) [31] (with parts)	75.2
Tian *et al.* (2013) [31] (without parts)	64.9
presented method (automatic tracks)	82.0
presented method (manual tracks)	94.0
presented method (without negative mining)	38.6
HOG baseline	48.6
HOF baseline	48.6
3-Stage-Baseline	38.3

LCA	Accuracy (percent)
Sadanand & Corso (2012) [44]	16.667
Wang *et al.* (2013) [45]	15.556
Wang *et al.* (2013) [13]	14.074
Kuehne *et al.* (2011) [6]	9.259
Cao *et al.* (2013) [46]	7.592
Le *et al.* (2011) [47]	6.667
Ryoo (2011) [48] as reimplemented by [46]	6.667
Messing *et al.* (2009) [49]	6.296
presented method (automatic tracks)	14.3
presented method (pregenerated tracks)	11.8
presented method (without negative mining)	5.5
HOG baseline	6.3
HOF baseline	12.0
3-Stage-Baseline	7.6

Office11	Accuracy (percent)
Sadanand & Corso (2012) [44]	87.2
Kuehne *et al.* (2011) [6]	73.2
presented method (automatic tracks)	93.3
presented method (manual tracks)	98.3
presented method (without negative mining)	21.8
HOG baseline	33.5
HOF baseline	73.2
3-Stage-Baseline	24.5



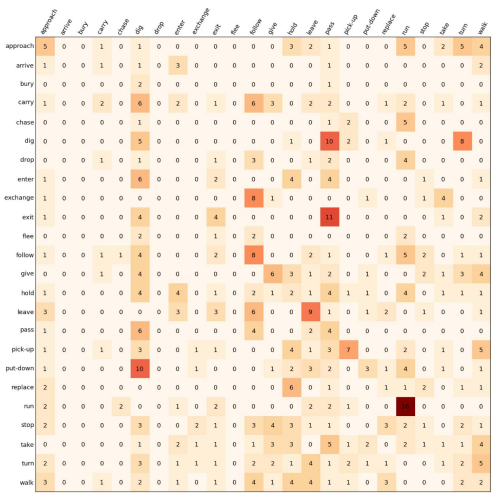


Fig. 5. Confusion matrices for the presented method with automatic tracks on each dataset.

obtained with the same method used to obtain the training boxes. We also report results with models that were trained without the use of additional bounding boxes produced by mining for high scoring tracks with the Event Tracker during training. This last number is included in order to show the importance of such negative training data. Without it, the performance is much worse than any of the other baselines on almost every dataset.

A. Classification Accuracy

1) Weizmann Dataset: The Weizmann dataset consists of 90 low-resolution video clips. There are ten classes, each depicted once by each of nine different actors, wearing a variety of different clothing. Experiments are done with a ninefold leave-one-actor-out cross validation. These videos

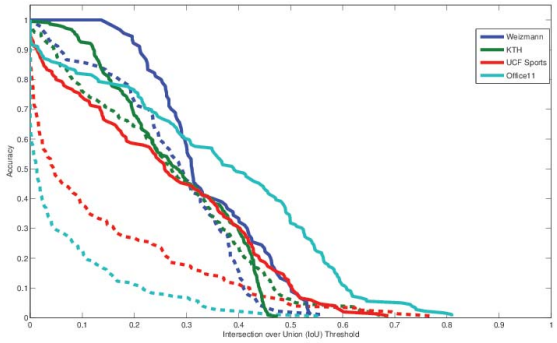


Fig. 6. Localization accuracy as a function of IoU threshold. A comparison between the presented method (solid lines) and the TLD tracker initialized with the DPM object detector (dashed lines).

have a clean white background and a single visible person, who performs the action. As noted in [31], many methods that report results on this dataset depend on this clean background to obtain a clean mask of the actors' silhouette by performing background subtraction as a preprocessing step.

Like [31] we do not rely on such properties. We instead perform simultaneous recognition and localization with our general-purpose system. Table I compares our results against recent methods on this dataset. Our results with automatic tracking are almost identical to those with manual tracks, both making very few mistakes and outperforming several recently published methods. While we do not achieve perfect results like Tian *et al.* [31], we significantly outperform the result they report without deformable parts. As [31] is among the most similar methods to this work, particularly the version without parts, this is an interesting result.

2) KTH Dataset: We evaluate our method on the KTH dataset in order to compare with [32], the most recent and most similar method. The KTH dataset consists of videos depicting six human action classes: *boxing*, *hand clapping*, *jogging*, *running*, *walking*, and *hand waving*. Each action is performed several times by 25 different people each in four scenarios: indoors, outdoors, outdoors while wearing different clothes, and outdoors with scale variation caused by camera zoom. This yields a total of 600 videos and 2391 action instances. The dataset provides a split into training, validation, and testing videos. Like [32], we use the training and validation sets for training, and test on the test videos. While the training videos each contain several instances of each action, they are quite similar, so we need to use only the first from each video.

Table I compares our performance with that of [32]. The method of Yao *et al.* [32] depends on manual annotation of bounding boxes in the training videos, as we do, but also uses manual annotation of part locations. They report results both with manual annotation of part locations in the training videos and without. We outperform both numbers, despite not using all the training data. In particular, we greatly outperform them when they only have bounding box annotations for training, as we do. Since they use clustering on the part locations to determine how the frames of training videos should be broken up into distinct poses, the fact that we greatly outperform them for the given same level of manual supervision shows that

our method is better able to automatically learn the relevant characteristics of the actions.

3) UCF Sports Dataset: The UCF Sports dataset consists of 150 videos obtained from the Web. This dataset also contains ten classes. The videos in UCF Sports are much more difficult than those of Weizmann, containing substantial camera movement, widely variant backgrounds and viewpoints, and occasionally low temporal resolution. It also includes scene changes that result in the actor sometimes jumping significantly in the image from frame to frame. As noted earlier, it contains significant correlation in background between videos of the same class.

The previous work often uses leave-one-out cross validation. It has been shown in [50] that the leave-one-out setting allows methods to take advantage of scene correlation between training videos. We instead use 16-fold cross validation. Doing so reduces the amount of training data available, but our method performs well despite this. Table I compares our results with the recently published work. Our result with manual tracks outperforms all previous works except [44], but the result with automatic tracks outperforms only [31]. The camera movement, scene changes, and sometimes low temporal resolution make tracking in this dataset very difficult. Many of these other methods are BOW based: they do not perform localization and can take advantage of the correlation in background, which we ignore by design. However, our results with automatic tracking are still good and critically outperform [31].

Because Tian *et al.* [31] also attempt to localize actions and recognize them based on the appearance and motion of the actor, this comparison is worth discussing further. As on the Weizmann dataset, they report results both with part models and without, showing that they perform much better with such part models. As our method does not use part models, it is more similar to the version that lacks parts. On this dataset, we outperform both numbers. This suggests that our method of closely tracking the actor so that the models are centered on the action better handles the variation present in UCF Sports than the cuboid models of [31].

4) LCA Dataset: The LCA dataset is designed to simulate a ground-based surveillance task. It consists of a 13-h subset of the video produced by DARPA for year 2 of the Mind's Eye program and is very difficult. It contains 24 classes (*approach*, *arrive*, *bury*, *carry*, *chase*, *dig*, *drop*, *enter*, *exchange*, *exit*, *flee*, *follow*, *give*, *hold*, *leave*, *pass*, *pick up*, *put down*, *replace*, *run*, *stop*, *take*, *turn*, and *walk*) in a number of outdoor environments depicted from a variety of viewpoints and scales, and often includes several people simultaneously moving in the same video, partially occluding one another and performing other actions not in the list of 24 classes. Being designed to emulate surveillance, there is no camera movement. All the action classes occur in each background environment, so that the background gives little to no clue as to the action. This is a very difficult dataset, despite the lack of camera motion. A recent paper [41] compares the results of eight recent methods. No method surpasses 17% accuracy.

As no bounding box information is included with LCA, we used automatically generated boxes to initialize training. This was done using our tracker with a generic object proposal method [51] and a motion prior. To bias the tracker toward moving objects, the mean optical-flow magnitude of each proposal was added to its score to produce the detection score f used by the tracker. The single highest scoring track as determined by the Viterbi algorithm was used as the initial positive track for each video. This simple tracker has no information about the actions and therefore has no way to determine which visible person to track when, as is often the case in the LCA dataset, several people are simultaneously visible and moving. Therefore, these initial tracks are quite poor, often tracking the wrong person. However, our method can handle these noisy training detections. The use of the Event Tracker to repeatedly resample the positive tracks according to the partially learned action models during training allows our method to recover from this, and perform competitively, as can be observed in Table I.

We report results both using tracks for the test videos that were pregenerated in the same manner described for the training videos and using the learned models to automatically localize the action with the Event Tracker. The results using the Event Tracker outperform those with pregenerated tracks, because the learned action models make it possible to determine which person is performing the action. Our method performs better than several methods, including Dense Trajectories [13], and performs at a similar level with Action Bank [44] and Improved Trajectories [45]. These four top methods significantly outperform the rest, with all others besides C2 [6] performing at a near chance level. This shows that our method is able to match the state of the art on this difficult dataset, even with no manual annotation on the training videos, using only noisy automatically generated boxes.

5) Office11 Dataset: As described previously, Office11 consists of 179 videos, depicting 11 classes. It contains both human-pose-related actions and manipulation of objects. We performed a comparison experiment between our method and two top-performing methods for which software is available: C2 [6] and Action Bank [44]. Action Bank performs the best on both the UCF Sports and LCA datasets, and so provides a strong comparison. We performed an identical fivefold cross-validation experiment on this dataset with both these methods and our own. The results are shown in Table I. Our method outperforms both other methods on this dataset.

B. Localization

We evaluated the localization accuracy of our method, separate from classification accuracy. It is evaluated by computing the intersection-over-union (IoU) measure between the track produced by the highest scoring model and the ground-truth track. We report the accuracy of localization as the fraction of test videos with IoU above a certain threshold. Fig. 6 shows the localization accuracy of our method on each of the datasets as a function of this threshold and compares this with that produced by the TLD tracker. Results are not reported for LCA because ground truth is not available. Fig. 7 visualizes localization examples that cannot be produced by low-level tracking systems. Our method outperforms the TLD tracker

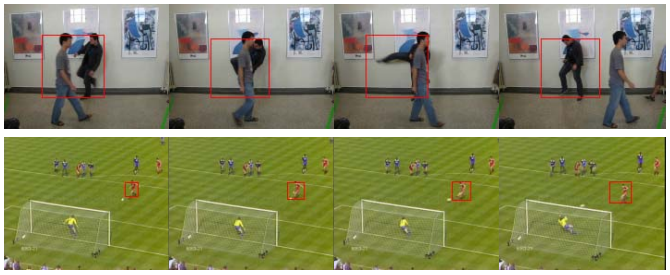


Fig. 7. Example tracks automatically produced by our system on the unseen test video. The top example from Office11 shows successful tracking of a *kick* despite complete occlusion. A tracking system unaware of the action would not produce this track because it is highly suboptimal according to low-level criteria. The bottom example, from UCF Sports, shows successful tracking of the single person performing the *kick* action out of many other visible and moving people. Low-level tracking systems have no mechanism to choose that particular person to track.

on every dataset. The performance of our system is somewhat similar to that of TLD on the easier to track Weizmann and KTH datasets, but is dramatically superior on the more difficult UCF Sports and Office11 datasets.

VIII. Discussion

Our method outperforms both [31] and [32], the two most similar methods, both of which were published recently. It also outperforms two state-of-the-art methods on our new dataset Office11 and performs competitively in the very difficult LCA dataset. It performs well with both manual and automatic tracking. On the KTH dataset, classification performance is actually higher than with manual tracking. This is likely due to the manually annotated tracks being aligned poorly with the person in some videos, a problem that is solved by performing automatic tracking. On the LCA dataset, it performs well despite noisy bounding boxes used for training and also performs better with automatic tracking than with predefined tracks obtained with a tracker unaware of the action class. The evaluation of localization shows that lower classification performance on the UCF Sports dataset with automatic tracks compared with manual tracks may be due to the difficulty in tracking the people performing the actions in that dataset. Our results also show the importance of mining for difficult negative tracks in the training data, rather than using only instances of other actions as negative samples. Without this additional negative training data, our method learns models that perform extremely poorly. This suggests that other methods such as [31] or [32] may benefit from a similar step during training.

IX. Conclusion

We have presented a method for automatically learning the changing appearance and motion patterns of actions in video. It accomplishes this by automatically identifying the most discriminative temporal subsequences of the action classes and training sequences of appearance and motion detectors to maximize the training classification margin. The learned sequences of detectors are shown by their visualizations to be intuitively meaningful representations of the actions. These learned models can then be used to perform simultaneous recognition and localization of actions in a new video. This method has been shown to perform competitively with the state of the art in action recognition on several datasets. In particular, it outperforms two recent methods that also attempt to recognize and localize actions by appearance and motion.

Acknowledgment

The views, opinions, findings, conclusions, and recommendations contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Army Research Laboratory, the National Science Foundation, or the U.S. Government.

References

[1] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of Web videos," *Mach. Vis. Appl.*, vol. 24, no. 5, pp. 971–981, 2013.

[2] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A spatio-temporal maximum average correlation height filter for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, Oct. 2005, pp. 1395–1402.

[4] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. 17th Int. Conf. Pattern Recognit.*, Aug. 2004, pp. 32–36.

[5] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 392–405.

[6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2556–2563.

[7] X. Wu, D. Xu, L. Duan, and J. Luo, "Action recognition using context and appearance distribution features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 489–496.

[8] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 915–922.

[9] I. Everts, J. C. van Gemert, and T. Gevers, "Evaluation of color STIPs for human action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2850–2857.

[10] C. Yuan, W. Hu, G. Tian, S. Yang, and H. Wang, "Multi-task sparse learning with Beta process prior for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 423–429.

[11] C. Yuan, X. Li, W. Hu, H. Ling, and S. Maybank, "3D $\mathcal{R}$ transform on spatio-temporal interest points for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 724–730.

[12] O. Oreifej and M. Shah, *Robust Subspace Estimation Using Low-Rank Optimization*. Berlin, Germany: Springer, 2014, ch. 5, pp. 55–67.

[13] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 60–79, 2013.

[14] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu, "Action recognition with actons," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3559–3566.

[15] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with Fisher vectors on a compact feature set," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1817–1824.

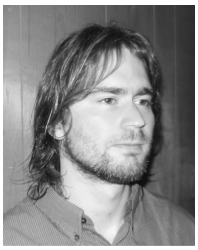
[16] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos 'in the wild,'" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1996–2003.

[17] N. Ikizler-Cinbis and S. Sclaroff, "Object, scene and actions: Combining multiple features for human action recognition," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 494–507.

[18] L. Lin, Y. Xu, X. Liang, and J. Lai, "Complex background subtraction by pursuing dynamic spatio-temporal models," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3191–3202, Jul. 2014.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Jul. 2012.

[20] H. Yu and J. M. Siskind, "Grounded language learning from video described with sentences," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, 2013, pp. 53–63.
[21] N. Siddharth, A. Barbu, and J. M. Siskind, "Seeing what you're told: Sentence-guided activity recognition in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 732–739.
[22] A. Barbu, A. Michaux, N. Siddharth, and J. M. Siskind, "Simultaneous object detection, tracking, and event recognition," *Adv. Cognit. Syst.*, vol. 2, pp. 203–220, Apr. 2012.
[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.
[24] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Statist.*, vol. 37, no. 6, pp. 1554–1563, 1966.
[25] Z. Chen, W. Zuo, Q. Hu, and L. Lin, "Kernel sparse representation for time series classification," *Inf. Sci.*, vol. 292, pp. 15–26, Jan. 2015.
[26] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1250–1257.
[27] X. Liang, L. Lin, and L. Cao, "Learning latent spatio-temporal compositional model for human action recognition," in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 263–272.
[28] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. 9th Eur. Conf. Comput. Vis.*, 2006, pp. 428–441.
[29] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 724–731.
[30] P. Banerjee and R. Nevatia, "Pose filter based hidden-CRF models for activity detection," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 711–726.
[31] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2642–2649.
[32] B. Z. Yao, B. X. Nie, Z. Liu, and S.-C. Zhu, "Animated pose templates for modeling and detecting human actions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 436–452, Mar. 2014.
[33] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[34] M. S. Ryoo and J. K. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 1593–1600.
[35] L. T. Niles and H. F. Silverman, "Combining hidden Markov model and neural network classifiers," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Apr. 1990, pp. 417–420.
[36] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.
[37] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002.
[38] A. Griewank, "On automatic differentiation," in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe, Eds. Boston, MA, USA: Kluwer, 1989, pp. 83–108.
[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[40] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 107–113, Jan. 1991.
[41] D. P. Barrett, R. Xu, H. Yu, and J. M. Siskind. (2015). "Collecting and annotating the large continuous action dataset." [Online]. Available: http://arxiv.org/abs/1511.05914
[42] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape-motion prototype trees," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 444–451.
[43] P. Nagar and A. Agrawal, "Geometric invariant model based human action recognition," in *Proc. 9th Int. Conf. Ind. Inf. Syst.*, Dec. 2014, pp. 1–6.
[44] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1234–1241.
[45] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.
[46] Y. Cao *et al.*, "Recognize human activities from partially observed videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2658–2665.
[47] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 3361–3368.
[48] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1036–1043.
[49] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 104–111.
[50] T. Lan, Y. Wang, and G. Mori, "Discriminative figure-centric models for joint action localization and recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2003–2010.
[51] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.



Daniel Paul Barrett (S'15) received the B.S. degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2011, where he is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering.

His current research interests include computer vision, robotics, and artificial intelligence, particularly their intersection, where a robot perceives, learns about, and acts on the world through noisy real-world camera and sensor input.

Jeffrey Mark Siskind (M'98–SM'06) received the B.A. degree from the Technion–Israel Institute of Technology, Haifa, Israel, in 1979, and the S.M. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1989 and 1992, respectively, all in computer science.

He held a post-doctoral fellowship with the Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA, USA, from 1992 to 1993. He was an Assistant Professor with the Department of Computer Science, University of Toronto, Toronto, ON, Canada, from 1993 to 1995, a Senior Lecturer with the Department of Electrical Engineering, Technion–Israel Institute of Technology, in 1996, a Visiting Assistant Professor with the Department of Computer Science and Electrical Engineering, University of Vermont, Burlington, VT, USA, from 1996 to 1997, and a Research Scientist with the NEC Research Institute, Inc., Princeton, NJ, USA, from 1997 to 2001. He joined the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2002, where he is currently an Associate Professor.