Trajectons: Action Recognition Through the Motion Analysis of Tracked Features

Pyry Matikainen[1], Martial Hebert[1], Rahul Sukthankar[2,1]
[1] Robotics Institute, Carnegie Mellon University [2] Intel Labs Pittsburgh
pmatikai@cs.cmu.edu, hebert@ri.cmu.edu, rahuls@cs.cmu.edu

Abstract

The defining feature of video compared to still images is motion, and as such the selection of good motion features for action recognition is crucial, especially for bag of words techniques that rely heavily on their features. Existing motion techniques either assume that a difficult problem like background/foreground segmentation has already been solved (contour/silhouette based techniques) or are computationally expensive and prone to noise (optical flow). We present a technique for motion based on quantized trajectory snippets of tracked features. These quantized snippets, or trajectons*, rely only on simple feature tracking and are computationally efficient. We demonstrate that within a bag of words framework trajectons can match state of the art results, slightly outperforming histogram of optical flow features on the Hollywood Actions dataset. Additionally, we present qualitative results in a video search task on a custom dataset of challenging YouTube videos.*

1. Introduction

The recent rise in popularity of the bag-of-words paradigm for action recognition in video has led to significant gains in performance and the introduction of more challenging datasets to tackle. Indeed, some of these techniques can achieve near perfect performance where more principled approaches only produce mediocre results. Nevertheless, their success has highlighted the fact that significant work remains to be done in the area of features for video, particularly with regards to motion.

Coming from work with static images, it is no surprise that the familiar techniques of that area have come to be applied here, such as the ever-popular histograms of oriented gradients. When dealing with the appearance of frames in video, these types of features are entirely appropriate. But actions in video comprise both an appearance aspect and a motion aspect, and motion features have not had the benefit inheriting a set of canonical techniques honed over decades.

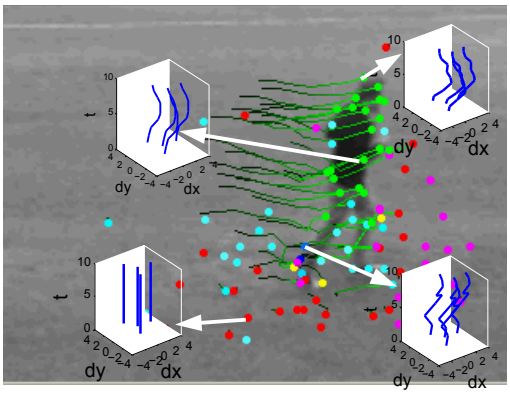

Figure 1. Simple KLT feature tracking is used to track as many features as possible within a video. Each tracked point produces a fixed length trajectory snippet every frame consisting of the last L (usually ten) positions in its trajectory. These snippets are quantized to a library of trajectons.

This leads us to believe that there are still significant gains to be captured with motion features, and since it seems that actions are defined more by their motion than by their incidental appearance, this problem is doubly important.

Present bag-of-words techniques for motion are largely extensions of appearance based techniques. Since motion information cannot be read directly from the video as in the case of appearance, it is necessary to either explicitly estimate this latent information or to implicitly encapsulate it within an appearance measure. In the latter case, image derivatives treating time as simply another dimension (producing an image cube) implicitly encode some amount of motion information; this approach amounts to augmenting 2D to 3D filters. However, as the framerate (that is, the sampling frequency in the time dimension) is rather coarse compared to the image resolution, any significant motion will be poorly represented through 3D image gradients.

The former approach, explicit estimation of motion, has proved to be more popular, most often in the form of optical flow. However, optical flow is itself a difficult problem to

tackle, and even the best algorithms are noisy and computationally expensive. While in principle it should be possible to track a feature using optical flow by simply integrating a trajectory over the time varying flow field, in practice the significant noise will quickly overwhelm any estimated trajectory. In contrast, even simple KLT trackers [2] can, with relative robustness, track a sparse set of features over relatively large periods of time at only a fraction of the computational cost of optical flow.

Compared with a cube of optical flow vectors, the key advantage of a trajectory is that it is attached to a particular moving feature. That is, in video deriving from the movement of physical bodies through space, a properly tracked feature (and hence trajectory) automatically gains foreground-background separation. In contrast, histogramming over a cube of optical flow vectors will blend the various sources of motion within that cube. For simple videos, such as a single person against a static background, this conflation of foreground and background may not matter. For more complicated videos with background motion and multiple occluding actors, this conflation comes at a cost.

To this end we introduce new trajectory-based motion features, which we call *trajectons*. We demonstrate that within a bag-of-words framework our simple and computationally efficient trajecton features are competitive with state-of-the-art motion features on simple datasets and outperform them on complicated ones.

2. Related Work

The idea of textons, or quantized filter bank responses, originated with texture classification but quickly grew to be applied to object and scene recognition in static images [21, 6]. These approaches are backed by some degree of psychological research suggesting that even human vision may employ unstructured statistical approaches at early stages [15], and can scale well even to large datasets when used with sparse features [5]. By analogy to the idea that textual works are composed from a set of discrete repeating elements (words), techniques that model data using a library of quantized features are generally known as bags-of-words approaches.

Recently bag of words techniques have gained significant popularity in video interpretation [11]. In the case of video appearance, the same features that work for static images still apply to video (e.g., histograms of oriented gradients, filter responses). However, in the case of motion it is not possible to directly read off the motion of the video in the same way as appearance can be directly read from pixels, and so the choice of motion features is complicated significantly.

The philosophically closest measure to motion pixels is dense optical flow, which has been a very common representation of motion [11, 8, 10, 18]. Dense optical flow has the benefit of intuitively representing our notion of motion in a directly analogous way to pixels and appearance, but the actual calculation of optical flow from sequential frames is itself a difficult problem and even the best algorithms are plagued by severe artifacts and noise. In an attempt to sidestep this problem, many techniques avoid the actual calculation of optical flow, either by implicitly encoding motion through temporal derivatives [16, 12, 7], or by producing the information that would be required to compute optical flow but refraining from the final step [18].

A more radical approach is to discard the notion of a dense motion field computed over the entire video and only compute motion at locations where it can be effectively estimated. Since most applications are interested in classifying the motion of actors distinct from their backgrounds, a natural desire is to only compute that motion which is relevant to those actors. Due to the intuitive appeal of the idea that the evolution of a silhouette through time is enough to capture the motion of an action, silhouette based features have been common as well [23, 3, 10, 19]. However, silhouettes cannot represent motion that occurs within the silhouette boundary (like a person clapping with both hands in front of her body), so the natural extensions has been from silhouettes to visual hulls [22, 20].

As silhouette extraction is not trivial either, the next step is to discard both density and semantic meaning to simply find special locations of interest (wherever they may be, on a person or not) for which the motion can be effectively computed. When these locations are pointlike, that is, occuring at a single time, the result is space time interest points around which various measures can be calculated, such as optical flow [7, 23].

When these locations extend in time they become trajectories, most frequently arising from tracked features. Often it is assumed that these trajectories are from known, fixed features, such as particular landmarks (*e.g.*, elbows, hands) on a human body [1, 14, 9]. If the trajectories are not on known features, then if they are very long and robust, it is potentially possible to extract full 3D information even from a single view [13]. The duration and coherence of trajectories means that each potentially contains a great deal of information.

Our contribution is to bring trajectories into a bag of words framework while avoiding the pitfalls of existing trajectory-based motion features: the assumption that trajectories are long, noise free, or tracking known body landmarks; at the same time, we do not discard the fundamental time-series nature of trajectories by treating them as merely as unrelated series of derivatives to be binned. Our method deals with short (< 20 frames) and inconsistent trajectories, and we are able to use computationally-efficient stock feature tracking methods such as KLT even on complex video. Since we are able to match the performance of optical flow

based methods even with our naive system, we believe there are still significant gains to be made with the combination of tracked feature point trajectories and bag of words techniques.

3. Method

Our method proceeds according to the standard bag-of-words approach: first, features are tracked over the video using a KLT tracker [2] to produce feature trajectories (x and y positions over time) for a number of features. These trajectories are slightly transformed (cropped and filtered, as described later) to produce a number of trajectory snippets for each video. Given a training set of videos, first a dictionary of trajectory words or trajectons is produced by clustering a sample set of trajectory snippets into a specified number (k) of clusters, the centers of which are retained as the trajecton library. Next, for each video, either training or test, its trajectory snippets are assigned the label of the nearest center in the trajecton library, and these labels are accumulated over the video to produce a histogram with k bins. This k length vector is normalized to sum to one for each video, and the training set of histograms along with training class labels is used to train a support vector machine (SVM) to classify videos into action categories. This SVM is used to classify the test set. The experiments shown below employ the standard LIBSVM [4] implementation of support vector machines.

We propose two variants of trajectons, which differ in their construction of trajectory snippets. For vanilla trajectons, each trajectory snippet is simply a concatenated vector of (dx, dy) derivatives for the trajectory, whereas in Affine-Augmented (AA) trajectons, this vector of derivatives is concatenated with a vector of local affine transforms to describe the motion around each trajectory point.

3.1. Vanilla trajectons

3.1.1 Feature Tracking

A standard KLT tracker is used to track features (using "good features to track") over a video. In our implementation, we track a fixed number of features (typically 100), with features replaced as necessary when tracks are lost. The output of this tracking is a trace of (x, y) pairs for each feature. For convenience of notation, we can assume that feature indices are never reused; then we can express a feature i's position at time t as $X_i^t = (x_i^t, y_i^t)$.

3.1.2 Trajectory Snippet Production

Then, for each frame, each feature that exists during this frame produces a trajectory snippet that consists of the discrete derivatives of the feature point locations in time. In other words, given a frame time t and feature i, and a maximum snippet length L, the trajectory snippet produced is:

$$T_i^t = \{X_i^t - X_i^{t-1}, X_i^{t-1} - X_i^{t-2}, \ldots, X_i^{t-L+1} - X_i^{t-L}\},$$

where if X_i^j does not exist for a given time any terms containing it are set to zero.

Since X_i includes both x and y position, the full flattened vector will be of length $2L$. If the number of tracked features is fixed at n, and a video has f frames, this means that the total number of trajectory snippets (and hence eventually trajectons) will be nf. Also, note that if a feature is tracked for longer than L frames, every window of size L in that trajectory produces its own snippet.

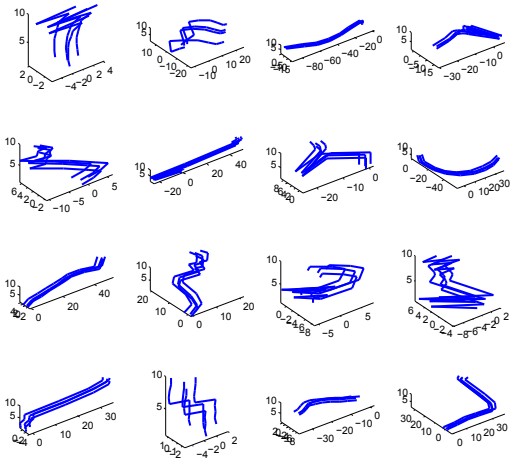

Figure 2. Example trajectons in the trajecton library computed from the KTH dataset. Many, such as the long and straight trajectories and the curving arcs, correspond to stereotypical portions of specific actions in the KTH dataset (running, waving arms).

3.1.3 Trajectory Snippet Clustering and Quantization

Next, these trajectory snippets are clustered into a library and that library is used to quantize snippets to a set of labels. Examples, selected at random, from the trajecton library computed for the KTH dataset [17] are shown in Fig. 2.

Given a sample set of trajectory snippets (vectors of length $2L$), these snippets are clustered using k-means with the standard Euclidean distance metric into k clusters and the cluster centers stored in a library of k trajectons. These trajectons represent archtypical trajectories within the video set.

The trajectory snippets of each video (both training and test) are quantized using the trajecton library by assigning each trajectory snippet the index of the trajecton to which it has the smallest Euclidean distance.

Note that no attempt is made to explicitly induce scale invariance, either spatial or temporal. If a particular action

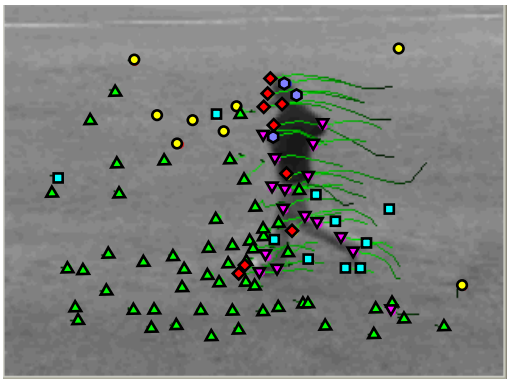

Figure 3. Example computed motion clusters for a video of a man jogging. Point color and shape indicates cluster assignment. The jogging person is oversegmented into four clusters, however as each cluster's points are largely correct this oversegmentation will have no effect on the end result. Note that each frame's motion segmentation is independent.

occurs at different scales or speeds, then instances of that action are initially represented by different sets of trajectons, and it is at the classification stage that these instances are grouped together under a single label. This idea is consistent with typical bag of words approaches and allows the representation to discriminate between similar types of motion when necessary (*e.g.*, running *vs.* jogging).

3.1.4 Video Classification

Following the standard bag of words framework, these trajecton labels are binned to produce a fixed-length histogram that is the final feature vector for a video. Given the nf trajectory snippets and associated trajecton label for each video, the trajecton labels are accumulated over the entire video into a histogram with one bin per label, for k total bins. Each video's histogram is normalized to sum to one.

Finally, videos are classified using support vector machines. A multi-class SVM is trained on the set of training histograms to produce video action classifications.

3.2. Affine-Augmented Trajectons

Vanilla trajectons suffer from the deficit, relative to histograms of optical flow, that each trajecton contains only information about a single point while ignoring the motion of neighboring points. Since we want to preserve the property that a trajecton encodes information that is attached to a particular body, we cannot simply histogram derivatives of nearby trajectories since that would confuse the trajectories of points co-located on the same body and foreign trajectories. Instead, we propose to first cluster the motions within the video into sets of trajectories which can be well described with mutually shared transforms; these motion clusters ideally fall within a single moving body. Each trajecton can then calculate local movement around itself according to the transforms for its motion cluster. Some example motion clusters can be seen in Fig. 3 in which a person moving his head and arm in independent ways has them properly assigned into different clusters.

For k motion clusters, the goal is to produce a set of assignments of trajectory snippets to clusters and cluster transforms such that the error between how a trajectory is expected to evolve (as calculated by successively transforming the first found location of a given feature point according to a cluster center) and its actual historical record. This goal is achieved in a k-means like manner in which trajectory snippets are first assigned to the centers that minimize error, and then center transforms are refined according to their assigned trajectory snippets. These two steps repeat until either convergence is reached or a fixed number of iterations have elapsed (in our implementation we limit to 20 iterations).

3.2.1 Point to Center Assignment

In the center assignment step, each trajectory snippet is assigned to the center which minimizes the error between its predicted trajectory according to those transforms and its actual trajectory.

For a given trajectory X^i, let $X^i_t =< x^i_t, y^i_t >$ be the location of the tracked point at time t, where t_0 is the current frame.

Let $T^j_{a\to b}$ be the transform for center j from time t_a to time t_b. In particular, let $T^j_{a\to 0}$ be the cumulative transform from a given time t_a to the current frame t_0.

Then the error for a trajectory X^i to a center T^j is given by

$$e(X^i, T^j) = \frac{\sum_{t=s^{i,j}+1}^{0} ||T^j_{s^{i,j}\to t} X^i_{s^{i,j}} - X^i_t||}{|s^{i,j}|},$$

where $s^{i,j}$ is the earliest time for which both the trajectory and the center have information. This is simply the average Euclidean distance between where a trajectory's next point is expected to be according to its transforms and its recorded position.

Each trajectory is simply assigned the center to which it has the least error:

$$a_i = \arg\min_j e(X^i, T^j).$$

3.2.2 Center Refinement

In the center refinement step, given a number of assigned trajectory snippets, each center's transforms (a set of affine transforms, one per frame) is re-estimated by solving the least squares minimization for the transforms.

Given a center with transforms T^j and assigned points $X^1, X^2 \dots X^k$, we can refine the transforms by solving for the cumulative transforms $T^j_{t\to 0}$ according to

$$(T^j_{t\to 0}) \begin{pmatrix} X^1_t & X^2_t & \cdots & X^k_t \end{pmatrix} = \begin{pmatrix} X^1_0 & X^2_0 & \cdots & X^k_0 \end{pmatrix}.$$

Any other needed transforms can be calculated from the cumulative transforms.

3.2.3 Trajectory Snippet Augmentation

Each trajectory snippet now has an associated center assignment. For each frame in the snippet, along with the (dx, dy) information that it already contains, the snippet is augmented with the affine matrix $A^t_{C(i)}$ for the center associated with it at the current frame. Each trajectory snippet is of length $6L$, $2L$ for the derivatives for each of L frames and $4L$ for the parameters of the affine matrix.

4. Experiments

We evaluate our proposed method quantitatively on the Hollywood Actions [11] dataset and qualitatively on a custom YouTube dataset.

4.1. Hollywood Actions

Table 1. Hollywood Actions Results

Action	Ours	Ours (Lax)	Laptev *et al.* [11]
Total	**31.1%**	**27.2%**	**27.1%**
SitDown	4.5%	13.6%	20.7%
StandUp	69.0%	42.9%	40.0%
Kiss	71.4%	42.9%	36.5%
AnswerPhone	0.0%	35.0%	24.6%
HugPerson	0.0%	23.5%	17.4%
HandShake	5.3%	5.3%	12.1%
SitUp	11.1%	11.1%	5.7%
GetOutCar	7.7%	7.7%	14.9%

We evaluate on the Hollywood Actions dataset [11] in order to gauge the performance in a difficult scenario. We train and test on the "clean" (manually annotated) training and testing sets. We track 100 features which are clustered into 1000 trajectons using the AA-trajecton method with six motion clusters per frame. Classification is performed using a SVM with a linear kernel.

Per-class classification accuracies are presented in Table 1 with a comparison to Laptev *et al.*'s HOF features. As can be seen, with aggressive SVM settings we outperform HOF at the cost of concentrating most of the discriminative ability into a few classes (Hollywood Actions is an imbalanced dataset with some actions representing significantly more than 1/8 of the total instances). With less aggressive SVM settings (labeled "lax" in Table 1), we still outperform HOF, and our gains can be seen in five out of the eight classes.

Figure 4. Sample frames from our YouTube dataset

4.2. YouTube Dataset

As an exploration of how our trajectons motion representation can fare in a difficult retrieval task, we evaluate qualitative search results on a custom YouTube dataset. This dataset is composed of 2019 YouTube videos of an average length of approximately 2 minutes each (or approximately 66 hours of video), some frames of which can be seen in Fig. 4. Each raw YouTube video is split into overlapping sequences of 2s such that one sequence starts each second of each video. A library of 400 trajectons was used, using 100 tracked features and 6 motion clusters using the AA-trajecton method. Trajecton histograms are accumulated over these 2s windows, and search is performed over all of these windows, and as a result we are effectively searching over more than 230,000 clips.

4.2.1 Video Similarity with Trajectons

For video search, since we are doing a direct comparison between videos without an intermediate machine learning step, direct trajecton histogram comparisons with a chi-squared distance would be dominated by the more common trajecton bins. To account for this, following Chum *et al.* [5] we downweight trajectons that occur frequently using tf-idf weighting, the weights being placed into the diagonal matrix W (or equivalently, all the trajecton histograms are element wise multiplied with the weighting vector w), and the histogram distance simply given by the chi-squared distance between histograms weighted by w. The distance between two videos is then given by

$$d(m_1, m_2) = chisqr(Wm_1, Wm_2), \qquad (1)$$

where m_i is the trajecton histogram for a video i. The actual search is performed by calculating the motion m_i feature vector for the query video clip and then comparing it to all the dataset video clips using the distance metric in eq. 1. The n video clips with the smallest distance are returned as matches.

We randomly chose 25 windows to act as queries; we have identified a number of common and interesting cases in the clips.

4.2.2 Human Body Motion

Performance on whole-body human motions (such as raising one's hands up) is reasonable. In the pictured example (top row of Fig. 5), the best match is a false positive of people bouncing up and down in a rollercoaster (mimicking the up and down motion of the query video person's arms). However, the following two matches are correct.

4.2.3 Camera Motion (not shown here)

Performance in finding similar camera movements (*e.g.*, panning, zooming) is very good, as expected. However, since camera motions are so simple there is often little semantic agreement between the matches (*e.g.*, a POV shot from a car driving along a road is motion-wise a match for a zooming camera shot of a soccer game, but is semantically unrelated).

4.2.4 Small Motion (Shaky Cam Failure Case)

Many scenes are largely static but filmed with handheld cameras which introduces small whole-scene movements. Since these movements are relatively strong and correlated over the entire scene, the whole scene shaking dominates the motion distance. Any semantically important motions are lost in the movement, and the returned matches are scenes where the camera shaking shares similar statistical properties rather than semantically interesting matches. The tf-idf weighting is unable to help, since trajectons produced by strong random motion are uncorrelated with each other and hence individually uncommon, so they will not be downweighted by tf-idf.

In the pictured example (third row of Fig. 5), a video of a baby taken with a handheld camera is incorrectly matched to a number of unrelated scenes also filmed through handheld cameras. The third match of the motion only case is arguably a correct match. These failure cases can largely be avoided by preprocessing the video using a standard stabilization technique that eliminates dominant frame motion.

4.2.5 Static Scenes (not shown here)

The most common types of video in our YouTube sample are those showing static images to accompany music. Such static scenes are trivially matched to other static scenes by our technique, but trajectons alone (since they focus exclusively on motion) are insufficient to determine whether the content present in different static scenes is semantically related.

4.2.6 Interviews

Another common case is a person or group of people talking directly to a camera. This results in a combination of handheld camera movement and person-like movement. In the "Interview 1" section of Fig. 5, a small video of a cowboy musician interview has been correctly matched to two interviews and incorrectly to a basketball game with similar camera movement.

In the "Interview 2 (talking)" example the camera is completely stationary and the person is quite still as well, with mouth motion being the most prominent. This is correctly matched to two scenes of people talking, and surprisingly correctly matched to a cartoon character talking (likely because of the exaggerated mouth movements produced by cartoon characters).

5. Conclusions

We present a novel and concise method for using the trajectories of tracked feature points in a bag of words paradigm for video action recognition. Compared to existing motion features (optical flow, silhouettes, derivatives) our quantized trajectory words or trajectons are able to take advantage of the positive features of each class: the computational efficiency of derivatives and sparse features, the performance of optical flow, and the deep structure of silhouettes. However, on this third point we have barely scratched the surface of using the structured information content of trajectories, and we believe that there are still significant gains to be made in this area. Future work will concentrate on how to use the information within trajectories without making assumptions on trajectory consistency or location.

5.1. Acknowledgements

This work was supported in part by NSF Grant IIS-0534962 and by ERC program under Grant No. EEC-0540865.

References

[1] S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2007.

[2] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, 2007.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Proceedings of International Conference on Computer Vision*, 2005.

[4] C.-C. Chang and C.-J. Lin. LIBSVM – a library for support vector machines, 2001.

Figure 5. Top matches for query videos on the YouTube dataset.

[5] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total Recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings of International Conference on Computer Vision*, 2007.

[6] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[7] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[8] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[9] I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. Cross-view action recognition from temporal self-similarities. In *Proceedings of European Conference on Computer Vision*, 2008.

[10] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Proceedings of International Conference on Computer Vision*, June 2007.

[11] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[12] O. Masoud and N. Papanikolopoulos. A method for human action recognition. *Image and Vision Computing*, 21:729–743, 2003.

[13] V. Rabaud and S. Belongie. Re-thinking non-rigid structure from motion. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[14] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2), 2002.

[15] L. W. Renninger and J. Malik. When is scene identification just texture recognition? *Vision Research*, 44(19), 2004.

[16] M. Rodriguez, J. Ahmed, and M. Shah. Action MACH: a spatio-temporal maximum average correlation height filter for action recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[17] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of International Conference on Pattern Recognition*, 2004.

[18] E. Shechtman and M. Irani. Space-time behavior-based correlation—or—how to tell if two underlying motion fields are similar without computing them? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007.

[19] S. Vitaladevuni, V. Kellokumpu, and L. Davis. Action recognition using ballistic dynamics. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[20] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2), 2006.

[21] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of International Conference on Computer Vision*, 2005.

[22] P. Yan, S. Khan, and M. Shah. Learning 4D action feature models for arbitrary view action recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[23] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.