Graphical Framework for Action Recognition using Temporally Dense STIPs

Pradeep Natarajan *  
BBN Technologies, 10 Moulton St  
Cambridge MA 02138  
pradeepn@bbn.com

Prithviraj Banerjee  Furqan M Khan  Ramakant Nevatia  
Institute for Robotics and Intelligent Systems  
University of Southern California, Los Angeles CA 90089  
{pbanerje|furqankh|nevatia}@usc.edu

Abstract

Graphical models have been shown to provide a natural framework for modelling high level action transition constraints, and to simultaneously segment and recognize a sequence of actions. More recently, Spatio-temporal Interest Points (STIPs) have been proposed as suitable features for action detection. These interest points are typically mapped to a set of codewords, and actions are detected by accumulating the codeword weights or by learning suitable classifiers. Existing methods for interest point detection provide a sparse representation of actions and require a costly exhaustive search over the entire spatio-temporal volume for action classification. Our contribution here is two-fold - first, we combine the interest point models of actions with pedestrian detection and tracking using a Conditional Random Field (CRF); second, we extend existing interest point detectors to provide a dense action representation while minimizing spurious detections. The larger number of interest points and the high-level reasoning provided by the CRF allows us to automatically recognize action sequences from an unsegmented stream, at real time speed. We demonstrate our approach by showing results comparable to state-of-the-art for action classification on the standard KTH-action set, and also on more challenging cluttered videos.

1. Introduction

Ability to recognize actions in video has several compelling applications including surveillance, human-computer interaction (HCI), search and retrieval among others. There has been two broad threads in current research on this topic - the first focuses on modelling high-level constraints and structure, typically using graphical models, while the second focuses on extracting suitable low-level features from video for subsequent classification. While significant progress has been made, the capability of current approaches remain limited to segmented action clips in clean backgrounds. Here, we bring these two threads together to combine their advantages for continuous recognition in cluttered scenes.

Graphical models, in particular Hidden Markov Models (HMM) and more recently Conditional Random Fields (CRF) provide a natural framework for modelling high level action transition constraints. The best action sequence is computed by evaluating suitable observation potentials which map the high-level state to image features. On the other hand, Bag-of-words approaches using Spatio-temporal Interest Points (STIP) as the basic features are being increasingly used for action recognition in recent work [21, 11, 14]. Such methods provide a compact representation of the shape and motion in actions but have two key deficiencies - first, the interest point detectors used in [9, 11, 10] produce a very sparse action representation. Second, they either require segmented action clips or an exhaustive search of the entire video volume.

The first issue is typically addressed by using Harris or other spatial corner detectors (e.g. [2, 14]) or by combining spatial corners with the spatio-temporal corners (e.g. [18]). Such features are however too dense and also require bounding box annotations during training to avoid learning on spurious background features. The second issue is best addressed by incorporating the interest point features in a graphical model framework.

In our work, we map unstructured STIP-based features to a structured graphical model representation that allows continuous action recognition in cluttered scenes with scale and viewpoint variations. Graphical models require observations in every frame for inference, while the STIPs are temporally sparse. Hence we introduce a Temporal Dense-STIP (TD-STIP) for feature extraction. Further, HMMs require the observations and transitions to be modeled with constant probabilities, which makes it hard to use the unstructured interest point features. CRFs on the other hand allow free form potential functions, which has been taken advantage of in earlier works in pose tracking [24] and action recognition [17]. Hence we map our TD-STIP features to a CRF. This is in contrast to [23] and [15] which use foreground and track based features respectively.

*This work was done when the author was at USC
In the rest of the paper - section 2 discusses related work, section 3 describes our graphical model, section 4 describes our feature detector, section 5 describes our descriptor, section 6 presents results and conclusions are in section 7.

2. Related Work

HMMs have been explored extensively to model high-level transition and spatio-temporal constraints. [3] simultaneously models both the natural hierarchical structure as well as durations of events using the switching hidden semi-Markov model (SHSMM). [13] represents key poses of actions rendered from multiple viewpoints as states of an ActionNet for view-invariant action recognition. More recently, Discriminative models like CRFs have been used. [23] applied CRFs for contextual motion recognition and [15] introduced a 2-layer extension (LDCRF) to the CRF framework for continuous gesture recognition. Recently, [17] maps a high-level CRF representation of actions, to low level optical flow and edge based shape features.

Several low-level features have been used to model the appearance and motion of actions. Optical flow templates were used in [4], while shape based templates were used in [5] for recognizing arm gestures. In recent work, [8, 17] combine shape and flow features for event detection in several cluttered scenes. In contrast to these approaches which correlate entire spatio-temporal volumes, several approaches have been proposed which match features around a sparse set of interest points. [21] presents a temporal extension to the 2D Harris corner detector for representing actions with a set of STIPs. [11] extracts Histogram-of-Oriented-Flows (HOF) around the STIPs and train action classifiers using discrete Adaboost, while [10] also uses HOG and HOF descriptors for training SVM classifiers. It has been argued in several other works such as [14, 2] that the STIPs extracted are too sparse for action recognition in cluttered videos, and instead use corner detectors on each frame. [2] uses a combination of Harris interest point detector and Gabor filters, while [14] uses a combination of MSER, Harris-Laplace and Hessian-Laplace interest point detectors. [18] proposes a hybrid feature detector by combining spatio-temporal and spatial interest point detectors.

3. Action Representation and Recognition

We automatically segment and recognize a sequence of actions in video using a Conditional Random Field (CRF), by taking advantage of the fact that actions have typical durations. We also include additional transition constraints in the CRF where appropriate - for example, a stand action can take place only after a sit action.

The state of the CRF at each frame $t$ is represented by the tuple $\theta=[a, d]$, where $a$ denotes the action, and $d$ denotes the duration for which action $a$ has been occurring. Figure 1 illustrates the CRF.

Let $\mathbf{I} = \{I_1, I_2, ..., I_T\}$ denote the sequence of frames in the video. Then, the probability of the state sequence $\theta = \{\theta_1, \theta_2, ..., \theta_T\}$ given the observation sequence $\mathbf{I}$ is given by the standard CRF formulation

$$P(\theta|\mathbf{I}) = \frac{1}{Z} \prod_{t=2}^{T} \phi(\theta_t, I_t)$$

where, $\phi(\theta_t, I_t)$ is the observation potential, $\psi(\theta_{t-1}, \theta_t, I_{t-1}, I_t)$ is the transition potential and $Z(\mathbf{I}) = \sum_{\theta} P(\theta|\mathbf{I})$ is a normalization factor. With these potentials, the best state sequence can be inferred by computing the maximum probability path-

$$p^* = \arg \max_{\theta} P(\theta|\mathbf{I})$$

Equation (2) can be solved efficiently using Viterbi search and automatically provides segmentation of actions in a continuous stream. The inference algorithm is similar to that of VTHMM [16], and hence takes $O(T)$ computation where $T = \text{No. of frames}$. We train the observation and transition potential functions independently, as in [24, 17].

The transition potential is a function of the previous and current state alone and models typical action durations with a $\text{signum}$ function as follows-

$$\psi(\theta_{t-1}, \theta_t, I_{t-1}, I_t) = \psi([a_{t-1}, d_{t-1}], [a_t, d_t]) = \begin{cases} 
\frac{1}{1+e^{-\frac{d_t - \mu(a_t)}{\sigma(a_t)}}} & \text{if } a_t = a_{t-1}, d_t = d_{t-1} + 1 \\
\frac{1}{1+e^{-\frac{d_t - \mu(a_t) + \sigma(a_t)}{\sigma(a_t)}}} & \forall a_t, d_t = 1 \\
0 & \text{Otherwise}
\end{cases}$$

Here $\mu(a_t)$ and $\sigma(a_t)$ are the mean and variance of the duration of one action cycle and is computed from the training data; and $0 \leq d_t \leq D, \forall t$. The intuition behind the choice of the transition potential in equation (3) is that the weight of staying in the current action decreases beyond the mean,
and the weight for transition to another action or a new cycle of the same action increases.

The observation potential of the CRF at each frame is computed by accumulating the codeword weights learned from the training data (as described in Section 5.2).

\[
\phi(\theta_i, I_t) = \phi([a, d], \{l_i\}^n) = \exp \left( \sum_{i=1}^{n} \alpha(l_i)w_{i,a} \right)
\]

where \(\alpha(l_i)\) is a weighting function that reduces the influence of codeword \(w_{i,a}\) based on its location \(l_i\); and \(n\) is the total number of interest points detected in the \(t^{th}\) frame.

For actions which take place in an upright position like walking, we can localize the person using pedestrian detection and tracking into our observations. Further, since our interest point detector produces very few false alarms, the actors could be localized by spatially clustering the interest points, even when pedestrian detections are not available. In our implementation, we accumulated codeword weights in region that is twice the detection window width, when the detections were available. In other cases we accumulated weights over the entire frame. Figure 2 illustrates the computation of the observation potential.

4. Feature Detection and Description

In the following sub-sections we introduce the Temporally Dense Spatio-Temporal Interest Points and provide a comparative analysis of the accuracy of the feature detector.

4.1. Spatial Interest Point Detection

The Harris corner detector detects points in an image that have significant variations in both the \(x\) and \(y\) directions. Let \(I(x, y)\) denote the image and \(g(x, y, \sigma^2)\) a Gaussian kernel-

\[
g(x, y, \sigma^2) = \frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2}
\]

Let \(L\) denote the convolution-

\[
L(x, y, \sigma^2) = g(x, y, \sigma^2) * I(x, y)
\]

\[
\begin{pmatrix}
L_x^2 & L_xL_y & L_xL_t \\
L_xL_y & L_y^2 & L_yL_t \\
L_xL_t & L_yL_t & L_t^2
\end{pmatrix}
\]

and let \(L_x = \partial_x L\) and \(L_y = \partial_y L\) denote the partial derivatives of \(L\) in the spatial dimensions. At a given scale \(\sigma^2\) the interest points can be detected based on the eigenvalues of the second moment matrix-

\[
\mu = g(x, y, \sigma^2) * \begin{pmatrix}
L_x^2 & L_xL_y & L_xL_t \\
L_xL_y & L_y^2 & L_yL_t \\
L_xL_t & L_yL_t & L_t^2
\end{pmatrix}
\]

Significant values for the eigenvalues \(\lambda_1, \lambda_2\) of \(\mu\) indicate significant variations in both the \(x\) and \(y\) directions. \[6\] detects interest points as the positive maxima of the corner function-

\[
H = \text{det}(\mu) - k \cdot \text{trace}^2(\mu) = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2
\]

4.2. Spatio-Temporal Interest Point Detection

\[9\] presents a natural extension of the 2D Harris corner detector to 3D. Let \(V(x, y, t)\) denote the video and \(g(x, y, t, \sigma^2, \tau^2)\) denote a 3D Gaussian kernel-

\[
g(x, y, t, \sigma^2, \tau^2) = \frac{1}{\sqrt{(2\pi)^3\sigma^2\tau^2}}e^{-(x^2+y^2)/2\sigma^2 - t^2/\tau^2}
\]

where \(\sigma^2\) and \(\tau^2\) are the spatial and temporal scales respectively. Let \(L_x = \partial_x L\) and \(L_t = \partial_t L\) denote the partial derivatives of \(L\) in the spatial and temporal dimensions. We can then define a \(3 \times 3\) second moment matrix-

\[
\mu = g(x, y, \sigma^2, \tau^2) * \begin{pmatrix}
L_x^2 & L_xL_y & L_xL_t \\
L_xL_y & L_y^2 & L_yL_t \\
L_xL_t & L_yL_t & L_t^2
\end{pmatrix}
\]

Spatio-temporal interest points are those \((x, y, t)\) locations which have significant eigenvalues \(\lambda_1, \lambda_2, \lambda_3\) for \(\mu\) in all 3 dimensions. \[9\] extends the corner function in equation (8) to 3D-

\[
H = \text{det}(\mu) - k \cdot \text{trace}^3(\mu) = \lambda_1 \lambda_2 \lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3
\]

4.3. Temporally Dense Spatio-Temporal Interest Points (TD-STIP)

While the STIPs extracted using the method described in section 4.2 provide a compact and intuitive representation of actions, they are typically too sparse to be suitable for recognizing actions involving small movements and in cluttered environments. Several existing approaches like \[14, 2\] address this issue by using one or more spatial interest point detectors similar to the Harris detector described in 4.1 or a combination of spatial and spatio-temporal detectors like in \[18\]. Such methods however can result in a large number of points that only correspond to spatial corners with no significant action. These spurious interest points can distract
the recognition algorithm and the training algorithms might require bounding box annotations for effective learning.

A simple way to increase the number of temporally significant interest points would be to compute the second moment matrix and corner function as in equations (10) and (11), and choose points that are local maxima in just the spatial dimensions instead of all 3 dimensions. The extracted points are spatial corners that are also edges in the (x,y,t) volume. This would still result in a large number of spurious points due to small changes in illumination and other noise conditions. Hence, in addition to the corner function in equation (11) we also compute the rank increase measure from [22] that measures the local motion inconsistency in a spatio-temporal patch. We compute the following two matrices in the space-time neighborhood of each interest point

\[
M = \begin{pmatrix}
\Sigma L_x^2 & \Sigma L_x L_y & \Sigma L_x L_t \\
\Sigma L_y L_x & \Sigma L_y^2 & \Sigma L_y L_t \\
\Sigma L_t L_x & \Sigma L_t L_y & \Sigma L_t^2
\end{pmatrix}
\]

(12)

\[
M^\diamond = \begin{pmatrix}
\Sigma L_x^2 & \Sigma L_x L_y \\
\Sigma L_y L_x & \Sigma L_y^2
\end{pmatrix}
\]

(13)

These are similar to second moment matrices in equations (7) and (10) except that the summations are over a spatio-temporal volume. In our experiments we used a 5 × 5 × 5 neighborhood around each point. [22] argues that at spatio-temporal corners corresponding to multiple local motions, the rank increases from \(M^\diamond\) to \(M\), while \(M\) is rank-deficient at other points. Thus we have,

\[
\Delta r = \text{rank}(M) - \text{rank}(M^\diamond) = \begin{cases}
0 & \text{single motion} \\
1 & \text{multiple motion}
\end{cases}
\]

(14)

While theoretically sound, the rank increase measure requires computing eigenvalues of \(M\) and \(M^\diamond\) and is also susceptible to noise. To handle this, [22] introduced a continuous rank increase measure-

\[
\Delta \hat{r} = \frac{\text{det}(M)}{\text{det}(M^\diamond)||M||_F}
\]

(15)

where, ||\(M||_F = \sqrt{\sum (M(i,j))^2}\) is the Frobenius norm of \(M\). Spatio-temporal corners have high \(\Delta \hat{r}\), but this method can generate several interest points in uniform regions which have indeterminate flow. A similar phenomenon was also observed in [8]. Thus we choose only those points that simultaneously maximize (11) in the spatial dimensions and also have high \(\Delta \hat{r}\). Figure 3 compares the interest points produced by our approach, with the others.

4.4. Evaluation of Feature Detection

We evaluated the accuracy of the different feature detectors in a set of indoor and outdoor images collected in a variety of cluttered backgrounds from the dataset used in [17]. We manually annotated bounding boxes around the actors, and define a detected interest point to be correct if it falls inside the bounding box, and define accuracy as the fraction of interest points that are correct.
We need high accuracy and also a good number of correct interest points in every frame as observations for recognition in a graphical model. Hence, we measure the % of frames which have at least $X$ correct interest points at different accuracies. Figure 4 presents the results of our evaluation. As can be seen TD-STIPs produce a large number of good interest points in most frames. Further, while the STIPs have good accuracy, they produce far fewer interest points. The Harris Corner and Rank Increase Measure based detectors produce large number of interest points but with low accuracies. Thus, our detector effectively increases correct interest point detection, without affecting accuracy.

5. Feature Descriptors

We describe the interest points using SIFT-like [12] descriptors in both image gradients and optical flow. The intuition behind this is that SIFT provides a local description of image features suitable for frame-by-frame processing while HoG and HoF are computed over larger regions or volumes. The local shape information is represented by computing a set of orientation histograms on (4×4) pixel neighborhoods in the gradient image. The orientations are assigned to one of 8 bins, and each descriptor contains a (4×4) array of 16 histograms around the interest point. We compute a similar 128 dimensional vector for the optical flow to give a descriptor with 256 dimensions. Figure 5 illustrates the shape and flow descriptors computed for a sample action.

5.1. Spatio-Temporal Codebook Generation

During training, we extract interest points from the training videos and then learn a set of codewords as in the traditional Bag-of-Features (BoF) approaches (e.g., [19]). Further, to handle symmetric action instances, we also include descriptors obtained by flipping the interest point descriptors about the x-axis. We then cluster these descriptors using

$K$-Means++ [1]. Let $\chi$ denote the set of points being clustered and let $D(x)$ denote the minimum distance of point $x$ to the cluster means already chosen. $K$-Means++ improves the traditional $K$-Means clustering by carefully seeding the initial cluster locations as follows-

1a. Choose an initial center $c_1$ uniformly at random from $\chi$.
1b. Choose cluster $c_i$, by selecting $c_i = x' \in \chi$ with probability $\frac{D(x')^2}{\sum_{x \in \chi} D(x)^2}$.
1c. Repeat Step 1b until all $k$ cluster means are initialized.
2. For each $x \in \chi$, compute cluster membership based on the nearest cluster mean $c_i$.
3. Recompute cluster means based on cluster membership: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ where $C_i$ is the set of points that belong to cluster $i$.
4. Repeat steps 2 and 3 until convergence.

[1] shows that $K$-Means++ is $O(\log k)$ competitive after just the initial seeding step. In our experiments, we found that the clusters from just the initial step produce results comparable to the clusters from the full $K$-Means algorithm. Figure 6 illustrates the codewords produced by clustering our interest point descriptors. Note that the clusters typically contain points that are close spatially and are also of interest semantically.

5.2. Learning Codeword Weights

Once we learn the codebook by clustering interest points in the training data, we learn weights $w_{c,a}$ for each codeword $c$ for each action $a$. Let $F^+_c,a$ denote the number of times codeword $c$ occurs in action $a$, and $F^+_a$ the total number of codewords in action $a$’s training set. We estimate the probability to match code word $c$ for action $a$ as

Figure 4. % of frames with atleast $X$ correct interest points for a given accuracy for different detectors

Figure 5. Shape and Flow Descriptors extracted at an Interest Point

![Figure 4](image-url)
$p_{c,a}^+ = \frac{F_{c,a}^+}{F_a^+}$. We set $F_{c,a}^- = 1$ if no feature in action $a$’s training samples match code word $c$. We treat the samples for all other actions as negative examples for action $a$ and estimate the probability to match codeword $c$ on a negative example for action $a$ as $p_{c,a}^- = \frac{F_{c,a}^-}{F_a^-}$, where $F_{c,a}^-$ is the number of times codeword $c$ occurs in all other actions and $F_a^-$ is the number of codewords in all other actions. We then estimate the weight $w_{c,a}$ as:

$$w_{c,a} = \log \left( \frac{p_{c,a}^+}{p_{c,a}^-} \right)$$

This definition of codeword weights is similar to the one used in [14], and assigns high weights to those codewords $c$ that are most discriminative for action $a$.

### 6. Experimental Evaluation

We rigorously evaluated different aspects of our approach including segmented action recognition, feature descriptors (shape Vs flow) and continuous action recognition. We describe these in detail.

#### 6.1. Segmented Recognition on KTH Dataset

We tested our approach on the KTH set from [21], which is one of the standard datasets for evaluating action recognition algorithms. The dataset consists of 25 persons performing 6 different actions, namely: boxing, handclapping, handwaving, jogging, running and walking. The dataset was recorded under 4 different conditions: outdoors (s1), outdoors with scale variations (s2), outdoors with clothing variations (s3) and indoors under lighting variations (s4).

We trained our models in each of the four conditions separately, and also with full dataset with varying train:test ratios. We repeated our experiments 5 times (each time varying the training samples), and computed the average accuracy for each train:test ratio. Figure 7 and Table 1 summarize the results of our approach. Most of the errors are due to confusion between similar actions like running, jogging and handclapping, handwaving similar to other works.

Our approach produces results comparable to the state-of-the-art, though the accuracy is slightly lower than some recent results [7, 20, 10]. Further, even with small train:test ratios of 6:19 and 10:15, our approach produces results comparable to earlier approaches that use leave-one-out validation like in [19, 25], demonstrating the advantage of using dense representations. Figure 8 illustrate the results for action recognition and localization obtained by our method under the various test conditions.

Another key advantage of our approach is that the runtime speed is real time at $\approx$30fps on a 3GHz Pentium IV running Windows C++ programs. In contrast [7] reports a runtime speed of $\approx$0.5fps and our implementation of [10] on the same platform had a speed of $\approx$4-5fps on the KTH

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours</th>
<th>State-of-Art</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTH-s1</td>
<td>96.67%</td>
<td>96.0%[7]</td>
</tr>
<tr>
<td>KTH-s2</td>
<td>86.67%</td>
<td>86.1%[7]</td>
</tr>
<tr>
<td>KTH-s3</td>
<td>88.10%</td>
<td>92.1%[20]</td>
</tr>
<tr>
<td>KTH-s4</td>
<td>95.24%</td>
<td>96.7%[20]</td>
</tr>
<tr>
<td>KTH-s1+s2+s3+s4</td>
<td>91.08%</td>
<td>91.8%[10]</td>
</tr>
</tbody>
</table>

Table 1. Accuracy on KTH Dataset
videos and even slower on videos with higher resolutions. Our method is faster because we only need to compute features near the interest points, rather than over the entire volume as for the HOG/HOF features in [10]. Further the CRF formulation avoids searching over the space of spatio-temporal volume for detecting actions.

6.2. Shape Vs Flow Features

Several earlier works [7, 20, 17] show that the combination of shape and flow features produce better results than using either of them alone. We did a similar test, by using only the Shape and the Flow descriptors described in Section 4. Figure 9 illustrates the variation in accuracy with train:test ratios when using Shape, Flow and Shape+Flow features. Combination of shape and flow produces a ≈5-10% improvement over using either alone.

6.3. Continuous Recognition on KTH Dataset

In order to test the ability of our approach to automatically segment and recognize actions from a continuous stream, we created a dataset by concatenating action videos from the KTH dataset. In order to avoid large discontinuities in the video volume at the concatenation points, we only used the actions boxing, handclapping and handwaving which occur in place near the center of the image. We used action videos from the test set and concatenated 3-6 action segments, for a total of ≈20 videos for each of the datasets s1-s4. We then localized the person in the combined videos using low-level pedestrian detection and tracking and then accumulated codeword weights using the CRF as described in section 4. We used the same codewords that we learned during segmented action recognition. In this experiment, none of the actions that were classified correctly during segmented recognition were misclassified. Further, we also had a high accuracy in detecting the action boundaries in the concatenated videos. Table 2 summarizes our results, where N=Total number of frames in the concatenated video dataset, E=Number of frames with erroneous event label from our algorithm.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTH-s1</td>
<td>94.16%(N = 9479, E = 554)</td>
</tr>
<tr>
<td>KTH-s2</td>
<td>85.05%(N = 6672, E = 997)</td>
</tr>
<tr>
<td>KTH-s3</td>
<td>90.34%(N = 6903, E = 667)</td>
</tr>
<tr>
<td>KTH-s4</td>
<td>92.87%(N = 10261, E = 732)</td>
</tr>
</tbody>
</table>

Table 2. Frame-by-frame Accuracy for continuous recognition; N=total no. of frames, E=no. of errors

6.4. Recognition on USC Dataset

To test the effectiveness of our approach in cluttered scenes, we tested our approach on videos of 6 actions from [17] - sit-on-ground(SG), standup-from-ground(StG), sit-on-chair(SC), standup-from-chair(StC), pickup(PK), point(P). We used instances of these actions around 3 different tilt angles - 15°, 30°, 45° from multiple pan angles, typically, around 0°, 45°, 90°, 270°, 315°. These actions were collected in typical indoor office settings and also outdoors with varying zoom. In all we had a dataset of 265 action segments across all viewpoints and background conditions.

To train our models we used a train:test ratio of 9:1. We tested the trained models for segmented action recognition, and also for continuous recognition where we encoded the CRF with high-level constraints like a stand can happen only after a sit action. However, since the low-level pedestrian detection works only in the upright pose we cannot use it for tracking through changing poses in actions like sitting. Hence, we used the weights from all the detected interest points in our recognition. But since our interest point detector produces very few spurious detections, we could localize the actor by spatially clustering the interest points; Figure 10 illustrates this. In addition we also trained SVM classifiers using different HoG and HoF channels, similar to the ones described in [10] for the same train:test separation for comparison. Table 3 summarizes the results of our experiments. Our approach consistently produces best performance at all tilt angles, while the STIP based classifiers’ performance depends on the channels used.

Figure 9. Variation in Accuracy using Shape, Flow and Shape+Flow

Figure 10. Sample results for Action Recognition on USC dataset
[17] reports accuracies of 82.98%, 81.25% and 65.23% for the 15°, 30° and 45° tilts respectively, using Mocap models. These models do not require additional training, but it is cumbersome to collect Mocap data. Our models, on the other hand requires large training sets to generalize well, but can be learned from videos. Also, our inference on the USC dataset runs at \( \approx 6 \text{fps} \) (on 740×480 resolution), while [17] report a speed of 0.37 fps, and our implementation of [10] ran at 0.5 fps.

### 7. Summary and Future Work

We have made two main contributions in our work. First we have introduced a simple interest point detector that detects temporally significant interest points in every frame while minimizing spurious detections; this allows us to process videos online, frame-by-frame. Second, we have presented a method that combines the unstructured interest points with a structured Conditional Random Field representation of actions; this allows high-level reasoning for continuous recognition in an unsegmented video stream. We have presented good results on standard datasets and our method runs at real-time speed. We plan to extend our work by including spatial relationships between the interest points by matching pose templates for events. We also plan to explicitly model view point variations using multi view models.

### References


