Volumetric Features for Video Event Detection

Yan Ke[1], Rahul Sukthankar[2,1], Martial Hebert[1]
[1]School of Computer Science, Carnegie Mellon; [2]Intel Labs Pittsburgh
{yke, rahuls, hebert}@cs.cmu.edu

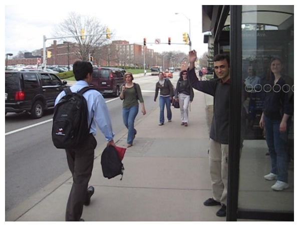
Abstract

Real-world actions occur often in crowded, dynamic environments. This poses a difficult challenge for current approaches to video event detection because it is difficult to segment the actor from the background due to distracting motion from other objects in the scene. We propose a technique for event recognition in crowded videos that reliably identifies actions in the presence of partial occlusion and background clutter. Our approach is based on three key ideas: (1) we efficiently match the volumetric representation of an event against oversegmented spatio-temporal video volumes; (2) we augment our shape-based features using flow; (3) rather than treating an event template as an atomic entity, we separately match by parts (both in space and time), enabling robustness against occlusions and actor variability. Our experiments on human actions, such as picking up a dropped object or waving in a crowd show reliable detection with few false positives.

1. Introduction

Event detection is an important component of automatic human activity understanding. The goal of event detection is to identify and localize specified spatio-temporal patterns in video, such as a person waving his or her hand. As we and Shechtman & Irani previously observed [33, 54], the task is similar to object detection in many respects since the pattern can be located anywhere in the scene (in both space and time) and requires reliable detection in the presence of significant background clutter. Event detection is thus distinct from the problem of human action recognition, where the primary goal is to classify a short video sequence of an actor performing an unknown action into one of several classes [10, 53, 71].

Our goal is to perform event detection in challenging real-world conditions where the action of interest is masked by the activity of a dynamic and crowded environment. Consider the examples shown in Figure 1. In Figure 1(a), the person waving his hand to flag down a bus is partially occluded, and his arm motion occurs near pedestrians that


(a)

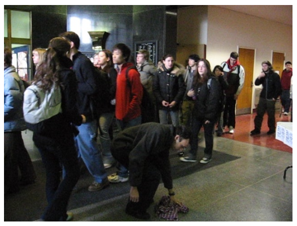

(b)

Figure 1. Examples of successful event detection in crowded settings. (a) The hand wave is detected despite the partial occlusion and moving objects near the actor's hand; (b) The person picking up the dropped object is matched even though the scene is very cluttered and the dominant motion is that of the crowd in the background.

generate optical flow in the image. The scene also contains multiple moving objects and significant clutter that make it difficult to cleanly segment the actor from the background. In Figure 1(b), the goal is to detect the person picking up an object from the floor. In this case, the image flow is dominated by the motion of the crowd surrounding the actor, and the actor's clothing blends into the scene given the poor lighting conditions.

A recent trend in action recognition has been the emergence of techniques based on the *volumetric analysis* of video, where a sequence of images is treated as a three-dimensional space-time volume. Eschewing the building of explicit models of the actor or environment (*e.g.*, kinematic models of humans), these approaches attempt to perform recognition directly on the raw video. This paper focuses primarily on two topics: (1) effective representations of shape and motion for event detection, and (2) efficient matching of event models to over-segmented spatio-temporal volumes. The models that we match are derived from single examples and are manually constructed; automatic generation of event models from weakly-labeled observations is a related interesting problem and is not covered in this work.

Since we used a view-based approach to event detection, the system is sensitive to variations such as camera view-

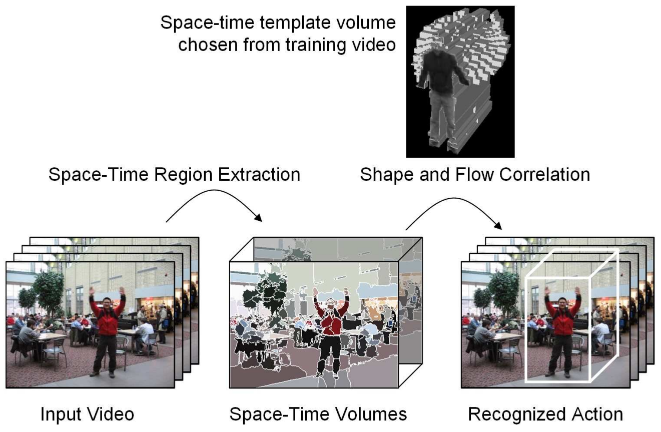

Space-time template volume
chosen from training video
Space-Time Region Extraction
Shape and Flow Correlation
Input Video
Space-Time Volumes
Recognized Action

Figure 2. Our goal is to detect specific actions in realisitic videos with cluttered environments. First, we segment input video into space-time volumes. Then, we correlate action templates with the volumes using shape and flow features. We are able to localize events in space-time without the need for background-subtracted videos.

point, scale, speed, and differences in how actions are performed across people. Our baseline method is not invariant to these changes; however, we address these issues and show how various extensions to our baseline algorithm can cope with these variations. Using a parts-based model and training from multiple examples, we also demonstrate experimentally the robustness of our algorithm to these variations.

This paper is organized as follows. We summarize the related work in Section 2 and give an overview of our experimental datasets in Section 3. We first show that spatio-temporal shapes are useful features for event detection (Section 4). Where the previous work is typically limited to scenes with static backgrounds, we demonstrate shape matching in cluttered scenes with dynamic backgrounds. We then combine our shape descriptor with Shechtman and Irani's flow descriptor, which is a complementary feature that can be computed in cluttered environments without figure/ground separation (Section 4.4). Recognizing the value of a parts-based representation, which is explicitly modeled by the human tracking approaches, and implicitly modeled by the interest-point approaches, we break our action templates into parts and extend the pictorial structures algorithm [24, 25] to 3D parts for recognition (Section 5). Finally, we discuss important issues in using view-based volumetric features in Section 6. Figure 2 presents an overview of the approach.

2. Related Work

Earlier work has identified several promising strategies that could be employed for event detection. These can be broadly categorized into approaches based on tracking [49, 57], flow [22,33,54], spatio-temporal shapes [10,11,70,71], and interest points [21, 47, 53]. A more comprehensive review of historical work is presented by Aggarwal and Cai [4]. More recent work is surveyed by Wang *et al.* [67]. Our work is based on volumetric flow and shape matching and thus is most related to works by Shechtman, Blank, and Irani [10, 54]. This paper extends our previously published work by introducing several methods to increase robustness in real-world scenarios, more in-depth experiments, and a method for automatically generating part configurations [34]. We now first review the related work in each of these areas and then we describe in detail the baseline technique that we use for comparison.

2.1. Shape Matching

Shape-based methods treat the spatio-temporal volume of a video sequence as a 3D object. Different events in videos generate distinctive shapes, and the goal of such methods is to recognize an event by recognizing its shape. Shape-based methods employ a variety of techniques to characterize the shape of an event, such as shape invariants [10, 71]. For computational efficiency and greater robustness to action variations, Bobick and Davis [11] project the spatio-temporal volume down to motion-history images, which Weinland *et al.* extend to motion-history volumes [70]. These techniques work best when the action of interest is performed in a setting that enables reliable segmentation [63, 69]. In particular, for static scenes, techniques such as background subtraction can generate high-quality spatio-temporal volumes that are amenable to this analysis. Unfortunately, these conditions do not hold in typical real-world videos due to the presence of multiple moving objects and scene clutter. Similarly, the extensive research on generalizing shape matching (2D [27, 41] and 3D [5, 5, 18, 26, 32]) requires reliable figure/ground separation, which is infeasible in crowded scenes using current segmentation techniques. We will show how ideas from shape-based event detection can be extended to operate on over-segmented spatio-temporal volumes and work in challenging conditions.

2.2. Flow Matching

Flow-based methods for event detection operate directly on the spatio-temporal sequence, attempting to recognize the specified pattern by brute-force correlation without segmentation. Efros *et al.* correlate flow templates with videos to recognize actions at a distance [22]. Shechtman and Irani propose an algorithm for correlating spatio-temporal event templates against videos without explicitly computing the optical flow, which can be noisy on object boundaries [54]. More recently, Zhu *et al.* uses an SVM classifier trained on histograms of optical flow to recognize tennis actions [72, 73]. Jhuang *et al.* uses biologically-inspired features, which includes optical flow, for action recognition [30]. We observe that flow has been successfully used in many settings, and we extend the previous work by building a real-time system for event detection based on Viola and Jones' framework [64]. Our work has recently been extended by Laptev and Perez [38].

2.3. Space-time Interest Points

Recently, space-time interest points [36] have become popular in the action recognition community [21, 47, 53], with many parallels to how traditional interest points [42] have been applied for object recognition. While the sparsity of interest points and their resulting computational efficiency are appealing, space-time interest points suffer the same drawbacks as their 2D analogues, such as failure to capture smooth motions and tendency to generate spurious detections at object boundaries. They rely on expressing the local region around an area of interest using representations that are robust to geometric perturbations and noise, yet distinctive enough to reliably identify the local region. However, these techniques rely on the assumption that one can reliably detect a sufficient number of stable interest points in the video sequence. For space-time interest points this means that the video sequence must contain several instances of motion critical events — regions where an object rapidly changes its direction of motion — such as the reciprocating path traced by a walking person's shoe. Unfortunately, these techniques fail to detect useful interest points in many common situations where the motions contain no sharp extrema, such as those illustrated in Figure 3. Space-time interest points are also frequently triggered by the appearance of shadows and highlights in the video sequence, as shown in Figure 4. These unstable "events" are sensitive to lighting conditions and can reduce recognition accuracy for the action of interest.

2.4. Pose Tracking

Methods based on tracking process the video frame-by-frame and segment an object of interest from background clutter, typically by matching the current frame against a model. By following the object's motion through time, a trace of model parameters is generated; this trace can be compared with that of the target spatio-temporal pattern to determine whether the observed event is of interest. We use a view-based approach that does not explicitly track these model parameters. Therefore, we are able to generalize to human, animal, or mechanical actions without prior models of these objects. Tracking-based approaches can incorporate existing domain knowledge about the target event in the model (e.g., joint angle limits in human kinematic models) and the system can support online queries since the video is processed a single frame at a time. However, initializing tracking models can be difficult, particularly when the scene contains distracting objects. And while recent work has demonstrated significant progress in cluttered environments [51], tracking remains challenging in such environments, and the tracker output tends to be noisy. An alternate approach to tracking-based event detection focuses on multi-agent activities, where each actor is tracked as a blob and activities are classified based on observed locations and spatial interactions between blobs [6, 28, 29, 39, 62]. These models are well-suited for expressing activities such as loitering, meeting, arrival and departure; the focus of our work is on finer-grained events where the body pose of the actor is critical to recognition.

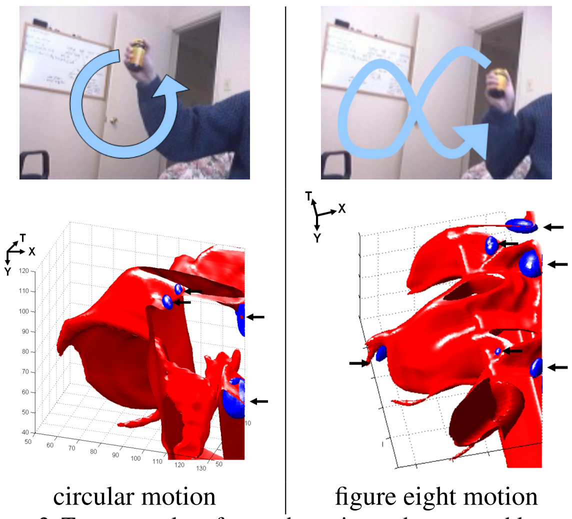


Figure 3. Two examples of smooth motions where no stable space-time interest points are detected. The 3D plots of motion through time were generated using software from [36]. The highlighted ellipsoids show the detected interest points. All of these detections are non-informative, caused by boundary interactions between the arm and the edge of the frame. By contrast, our volumetric features are scanned over the video sequence through space and time, and can accurately recognize such motions.

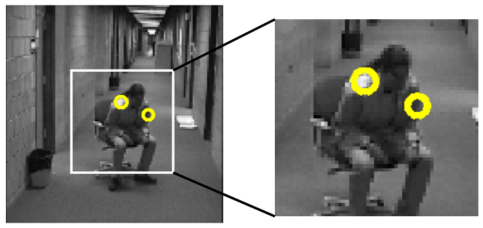


Figure 4. Space-time interest points are often found on highlights and shadows. These points are sensitive to lighting conditions and reduce recognition accuracy. This observation motivates our decision to apply volumetric features to the motion vectors rather than to the raw pixels.

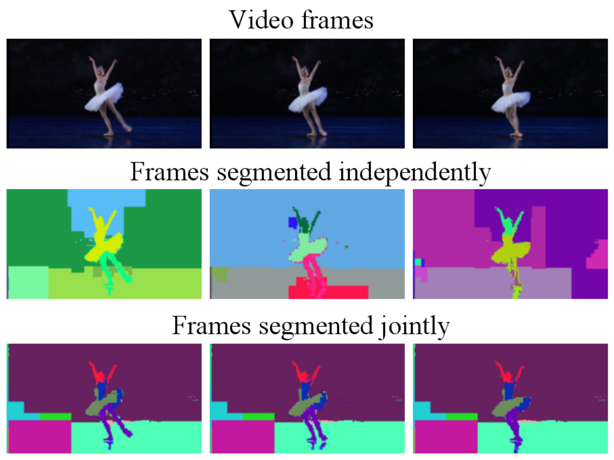


Figure 5. Independently segmenting each frame in the video yields varying and inconsistent segmentations across frames (middle). Jointly segmenting several frames across space-time yields consistent segmentations across adjacent frames (bottom).

2.5. Volumetric Features vs. 2D Features

Traditional methods for processing video have focused on analyzing individual frames independently, or possibly adjacent frames such as optical flow [43]. Features are typically extracted for each frame independently, and then subsequently linked together temporally, *e.g.*, [59]. The main limitation of these type of approaches is that spatial and temporal analyses are done separately. It is difficult to find stable regions that are consistent in both dimensions, as shown in Figure 5. Doing analysis jointly in space-time offers three advantages. First, the region boundaries are stable. Second, the regions are connected across frames. In other words, we know that a region in one frame corresponds to a specific region in the next frame, and no further region matching needs to be done. Third, region growth and death are accounted for automatically. This would be particularly difficult to analyze if the segmentation were to be done independently.

We argue that video should be thought of as three-dimensional volumes, and thus the fundamental processing unit should be 3D blocks consisting of many frames, instead of on a frame by frame basis, as shown in Figure 2. Only recently have researchers begun to simultaneously process blocks of frames of video [20, 54, 66]. Just as researchers have decomposed images into their constituent shapes and used 2D shape descriptors for analysis [9, 27, 58], video can be thought as a group of 3D volumes. There are several advantages to jointly analyzing a video's space and time dimensions. First, spatial and temporal consistency can be easily maintained. Second, instead of analyzing pixels over many frames, higher-level algorithms can focus on large, sparse regions for improved efficiency. Finally, the appearance and motion of objects in the scene can be jointly modeled, which can potentially lead to better recognition results.

2.6. Parts-Based Matching

Parts-based object and action recognition has been studied extensively in the past. The fundamental idea is to break an object into parts to increase the model's generalization power while keeping the computational efficiency tractable. Notable examples in object recognition include Weber *et al.*'s unsupervised object recognition [68], Lowe's keypoints [42], Sali and Ullman's fragments [52], and Fei-fei *et al.*'s one shot object category recognition [23]. This work is closely related and could be thought of as an spatio-temporal extension to the image segmentation and bottom-up recognition work by Cour and Shi [17] a Srinivasan and Shi [60]. Recognition in video include Ramanan *et al.*'s models of animals [50] and Boiman, Shechtman, and Irani's work [12, 55]. We follow the same principles and apply the pictorial structures framework [24, 25] to our task of event detection.

2.7. Flow Consistency Matching

We use Shechtman and Irani's flow-based method for matching actions in videos [56] as a baseline method for comparison. Instead of explicitly computing the optical flow of a pixel, they showed a way to calculate the flow correlation between two video volumes. Given a single video template, they can find actions such as spinning, diving, or clapping in real-world videos. By scanning the template across all locations in space and time and thresholding on the correlation distance, we can detect all instances of the action in the video. Using a similar notation as Shechtman and Irani, we review the details of their algorithm. Let P be a small, *e.g.*, $7 \times 7 \times 3$ space-time patch in the video. We define the space-time gradient as $\Delta P_i = (P_{x_i}, P_{y_i}, P_{t_i})$ for each point in $P(i = 1 \ldots n)$. We define the space-time Harris matrix M as follows (see Shechtman and Irani [56] for further details):

$$M = \begin{bmatrix} \sum P_x^2 & \sum P_x P_y & \sum P_x P_t \\ \sum P_y P_x & \sum P_y^2 & \sum P_y P_t \\ \sum P_t P_x & \sum P_t P_y & \sum P_t^2 \end{bmatrix}. \quad (1)$$

We further define $M^\diamondsuit$ to be the upper left minor on M:

$$M^\diamondsuit = \begin{bmatrix} \sum P_x^2 & \sum P_x P_y \\ \sum P_y P_x & \sum P_y^2 \end{bmatrix}. \quad (2)$$

A space-time patch P contains multiple motions if there is a rank increase between $M^\diamondsuit$ and M, or a single motion if there is no rank increase. Because the local space-time patches are small, we can assume that most patches have only one motion. Suppose there are two space-time patches P_1 and P_2 where M_1 and M_2 are the space-time Harris matrices of the two patches, respectively. Determining whether the two patches have inconsistent motion is equivalent to determining whether the concatenated patch P_{12} has multiple motions. This is straightforward since $M_{12} = M_1 + M_2$. Ideally, one would like to calculate the rank-increase measure Δr, where

$$\Delta r = \text{rank}(M) - \text{rank}(M^{\diamondsuit}), \tag{3}$$

which one can do by calculating the number of non-zero eigenvalues of the matrices. Due to noise, the eigenvalues are never exactly zero, and therefore Shechtman and Irani defined a continuous rank-increase measure $\Delta\tilde{r}$, where

$$\Delta\tilde{r} = \frac{\lambda_2 \cdot \lambda_3}{\lambda_1^{\diamondsuit} \cdot \lambda_2^{\diamondsuit}} = \frac{\det(M)}{\det(M^{\diamondsuit}) \cdot \lambda_1} \tag{4}$$

and λ_i is the i^{th} largest eigenvalue of M. Since finding the eigenvalues of a matrix is time consuming, Shechtman and Irani approximated λ_1 by the Frobenius norm of M. The approximate continuous rank-increase measure $\Delta\hat{r}$ is therefore

$$\Delta\hat{r} = \frac{\det(M)}{\det(M^{\diamondsuit}) \cdot \|M\|_F}, \tag{5}$$

where $\|M\|_F = \sqrt{\sum M(i,j)^2}$. The local *in*consistency measure is defined as

$$m_{12} = \frac{\Delta r_{12}}{\min(\Delta r_1, \Delta r_2) + \epsilon}. \tag{6}$$

To calculate the matching distance between the template T and a video V at a particular location $l = (x, y, t)$, we sum m_{12} at all locations where the template and the video overlap. We define the flow matching distance as

$$d_F(T, V; l) = \frac{\sum_{i \in T, j \in (T \cap V)} m_{ij}}{|T|}. \tag{7}$$

Because we must sum over the entire template volume at every location, this is a very time-consuming process. Shechtman and Irani optimized the running time by doing a hierarchical search and using other techniques to avoid computation. As an optimization, we use quarter-sized templates on quarter-sized videos, leading to a sixteen-times speedup. Our baseline implementation does not do any other optimizations and searches over all pixels. While this is slow to run in practice, it gives us the highest possible accuracy for this algorithm.

3. Datasets

The datasets used in our experiments cover a wide range of actions and vary in difficulty. Some of the publicly available datasets such as the KTH dataset [53] and the Weizmann dataset [10] were initially collected for action classification, where the entire video clip is classified as one of n actions. Most of them have static backgrounds and contain only one actor. Therefore, we collected more challenging datasets with dynamic backgrounds and multiple actors in the field of view. We apply our method to published videos such as tennis matches, aerobics training videos, and videos uploaded by users on YouTube. The YouTube videos are typically very low quality with lots of camera movement, as shown in Figure 9. We also apply our method to standard action classification datasets for comparison purposes even though our algorithm is designed for event detection. Table 1 lists all of the datasets we used and Figures 6, 7, 8 illustrates some of the actions in the various datasets. Additional figures of the other datasets are included with the results.

Figure 6. Example actions from the KTH [53] dataset.

Figure 7. Example actions from the Weizmann [10] dataset.

4. Volumetric Region Matching

4.1. Introduction

We first introduce our approach for event detection by matching volumetric shapes. The target events that we wish to recognize are typically one second long, and represent actions such as picking up an object from the ground, or a hand-wave. We first extract spatio-temporal shape contours in the video using an unsupervised clustering technique. Next, we match the event templates to the extracted shapes to detect the events. The templates are manually generated using interactive segmentation and labeling (see Figure 24). Denoting the template as T and the video volume as V, detecting the event involves sliding the template across all possible locations l in V and measuring the shape matching distance between T and V. An event is detected when the distance falls below a specified threshold. Similar to other sliding-window detection techniques, this is a rare-event detection task and therefore keeping the false-positive rate low is extremely important.

Table 1. Datasets used in our experiments.

Dataset	# actions	# actors	Length	Description
KTH [53]	6	25	$\tilde{2}$ hours	Periodic motion on static background. Standard dataset.
Weizmann [10]	4	9	5 min.	Standard dataset. Actions on static background. Subset of actions used.
Wimbledon [2]	1	1	30 min.	Broadcast sports videos.
Cluttered	5	6	20 min.	Actions in cluttered environment and dynamic background.
Multiview	3	3	30 min.	Four cameras at different viewpoints capturing events simultaneously. Used to test robustness to viewpoint changes.
Moving Camera	4	2	6 min.	Videos captured using shaky and panning cameras. Used to test robustness.
YouTube [3]	4	20+	Var.	Unscripted real world videos. Low quality videos with lots of camera movement.
Aerobics [1]	1	3	1 min.	Three people performing actions simultaneously. Used to test multi-instance event detection.
TV Gestures	4	1	2 min.	A person demonstrating TV remote control using gestures. Used to test multi-instance event detection.

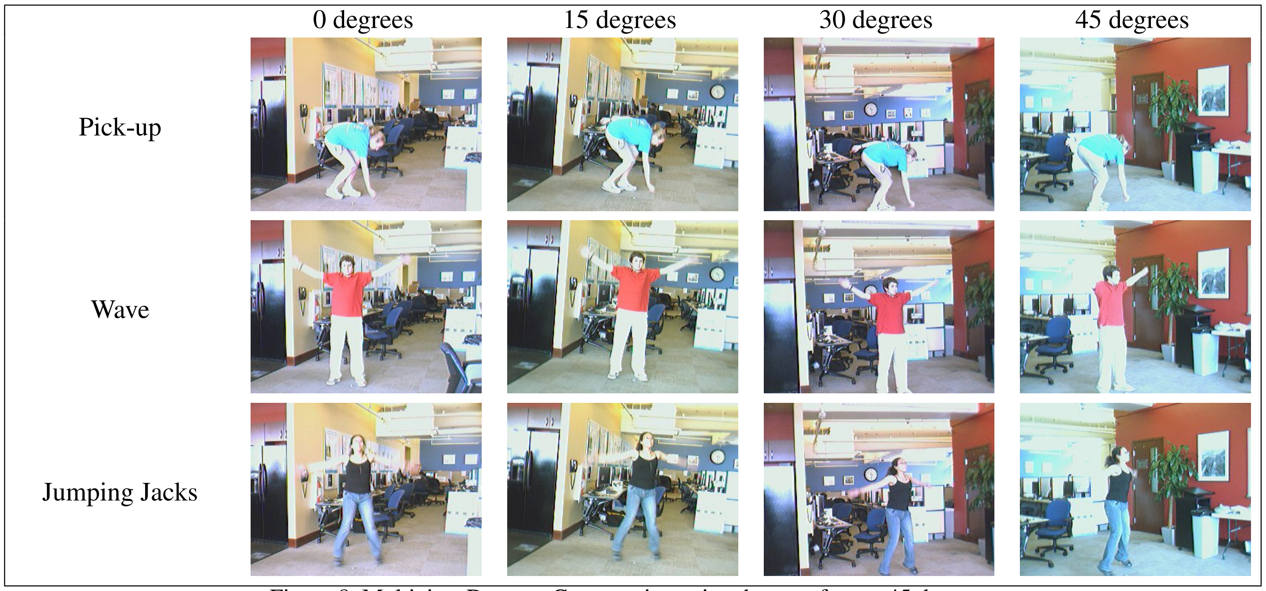


Figure 8. Multiview Dataset. Camera viewpoint change of up to 45 degrees.

4.2. Spatio-Temporal Region Extraction

Using an unsupervised clustering technique, we extract spatio-temporal shape contours in the video. This enables us to ignore highly variable and potentially irrelevant features of the video such as color and texture, while preserving the object boundaries needed for shape classification. As a preprocessing step, the video is automatically segmented into regions in space-time using mean shift, with color and location as the input features [15, 16, 19, 40, 66]. This is the spatio-temporal equivalent of the concept of superpixels [46]. Figure 5 shows an example video sequence and the resulting segmentation. Note that there is no explicit figure/ground separation in the segmentation and that the objects are over-segmented. The degree to which the video is over-segmented can be adjusted by changing the kernel bandwidth. However, since finding an "optimal" bandwidth is difficult and not very meaningful, we use a single value of the bandwidth in all of our experiments, which errs on the side of over- rather than under-segmentation. Processing the video as a spatio-temporal volume (rather than frame-by-frame) results in better segmentations by preserving temporal continuity. We have found mean shift to work well in our task, but in general, any segmentation algorithm could be used as long as it produces an over-segmentation that tends to preserve object boundaries.

Figure 9. YouTube dataset. Notice the poor quality of the videos. They have low frame rate, low resolution, motion blur, poor lighting, and blockiness due to compression artifacts.

Figure 10. Example detections from the TV Gestures dataset. Multiple gestures are are detected in a sequence to control the TV.

4.3. Volumetric Shape Matching

We now present a novel method for matching action templates to over-segmented video that accomplishes three goals. First, the algorithm matches on the shape of the spatio-temporal volume, rather than the pixels in the volume. This is motivated by the fact that the spatio-temporal "shape" of an action is robust to variations in an object's appearance (*e.g.*, an actor's clothing). Second, the algorithm robustly matches over-segmented spatio-temporal volumes. In other words, it identifies the set of supervoxel regions that, when aggregated, best match the given template. Finally, the method must be computationally-efficient because video data is extremely large. Because our action representation is composed of three-dimensional shapes, it would seem straightforward to directly apply algorithms from the 3D shape matching literature to this task. Unfortunately, most of the existing algorithms cannot efficiently cope with over-segmented regions.

4.3.1 Proposed Algorithm

Our shape matching metric is based on the region intersection distance between the template volume and the set of over-segmented volumes in the video. Given two binary shapes, A and B, a natural distance metric between them is the symmetric difference between the two regions, *i.e.*, $|A \cup B \setminus A \cap B|$.[1] We adapt this distance metric to work with over-segmented regions as follows. Given a template T, we slide the template along the x, y, and t dimensions of the video. Consider a candidate volume V with the template at some location $l = (x, y, t)$. Because the video is over segmented, V could be composed of k regions V_i such that $V = \cup_{i=1}^{k} V_i$. Consider how one might calculate the voxel intersection distance between the template T and a subset of regions of V. Since every region V_i is either selected or not selected, a naive approach would enumerate all possible 2^k subsets of V, calculate the voxel intersection between the template T and each subset, and choose the minimum. We propose a fast method for both identifying the subset of V that minimizes the distance and for calculating this distance.

[1]The distance metric $D = |A \cup B \setminus A \cap B|$ is related to the Jaccard similarity coefficient $S = \frac{|A \cap B|}{|A \cup B|}$ with $D = (1 - S) \cdot |A \cup B|$.

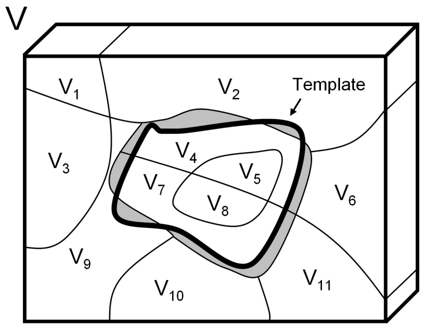


Figure 11. Example showing how a template is matched to an over-segmented volume using the Region Intersection method. The template is drawn in bold, and the distance (mismatch) is the area of the shaded region.

There are four cases that we must consider when deciding whether a region V_i belongs in the minimum set, where the minimum set $\hat{S}$ is defined as

$$\hat{S} = \bigcup_{i \in S} V_i, \qquad (8)$$

and S is defined by

$$\underset{S \subset \{1,\ldots,k\}}{\operatorname{argmin}} \; |((\cup_{i \in S} V_i) \cup T) \setminus ((\cup_{i \in S} V_i) \cap T)|. \qquad (9)$$

In Figure 11, we have drawn the template T in bold and overlaid onto the candidate volume V, which is segmented into 11 regions $V_1 \ldots V_{11}$. The set of regions that minimizes the distance to the template is $\{V_4, V_5, V_7, V_8\}$, and the actual distance is the area occupied by the shaded regions. By inspection, it is obvious that removing any region from the minimal set or adding any region not already in the minimal set, will increase the distance. The four cases of region intersections that we must consider are as follows. If a region V_i is completely enclosed by the template, such as V_5, then it is always contained in the minimal set. Similarly, if a region V_i does not intersect with the template, such as V_{11}, then it is never contained in the minimal set. The two interesting cases are when V_i intersects the template, such as V_2 and V_4. Let us consider V_2; it is obvious that excluding V_2 minimizes the distance between the template and the minimal set. Similarly, including V_4 in the minimal set minimizes the distance. Intuitively, we should include a region if there is a large overlap between the region and the template. More formally, the distance between the template T and the volume V at location l is defined as

$$d(T, V; l) = \sum_{i=1}^{k} d(T, V_i; l), \tag{10}$$

where

$$d(T, V_i; l) = \begin{cases} |T(l) \cap V_i| & \text{if } |T(l) \cap V_i| < |V_i|/2 \\ |V_i - T(l) \cap V_i| & \text{otherwise,} \end{cases} , \tag{11}$$

and $T(l)$ denotes the template T placed at location l. This distance metric is equivalent to choosing the optimal set of over-segmented regions and computing the region intersection distance. It is important to note that once the relative positions of the template T and the candidate volume V are specified, each of the regions V_i can be considered independently. In other words, whether V_i is in the minimal set is independent of any of the other regions $V_{\{1...k\}\setminus i}$. The implementation details of the shape matching algorithm are summarized in Algorithm 1.

As we slide the window across the video, we mark all locations with a distance less than some threshold θ as a match. We show next that the distance computations at adjacent locations can be updated with only a small update cost.

4.3.2 Speed Optimizations

Because we slide the template over the video volume in small increments, there is significant overlap and redundant computation in successive calculations of the distance function. A naive implementation would require $O(|T|)$ time to calculate the distance function. Figure 12 illustrates the template at two adjacent horizontal positions. Once we have computed the distance at one location, we only need to examine the shaded region to update the distance at the next location. The number of voxels that need to be updated is proportional to the surface area of the template, rather than the volume of the template. In practice, this optimization results in approximately an order of magnitude speedup for the matching algorithm.

```
Input: C: 3D array of region labels for each voxel.
       S: Array containing the size of each region.
       N: Number of regions.
       L: Location to calculate matching distance.
       T: Array of points corresponding to the
       template shape.
Output: Matching distance

// overlap[i]: amount of overlap between $T$ and region $V_i$.
overlap = $\{0\}^N$ ;
foreach $p$ in $T$ do
    increment(overlap[C[$\vec{L} + \vec{p}$]]) ;
end
dist = 0 ;
for $i = 1 \ldots N$ do
    dist += min(overlap[i], S[i] - overlap[i]) ;
end
return dist ;
```

Algorithm 1: Shape matching algorithm.

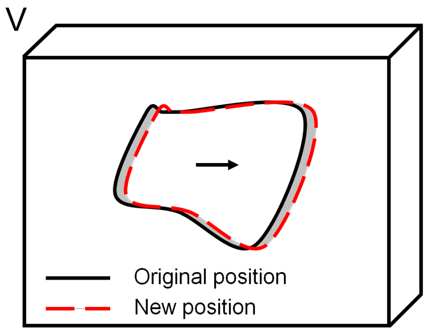


Figure 12. When we shift the template to a nearby location, only the shaded area changes. Therefore, we only need to update an area that is proportional to the surface area of the template, rather than the volume of the template. This dramatically decreases the running time during retrieval.

4.3.3 Modeling Segmentation Granularity

A potential problem with our method is that highly-textured regions of the video can generate many false positives. Figure 13 shows an example false positive of a tennis serve in a crowded region of a video. This is because such volumes consist of many tiny supervoxels that can be appropriately aggregated to match the given template. More formally, recall that the maximum error that a region V_i can contribute to the distance between the template and the volume is $|V_i|/2$. Therefore, as V is segmented into more regions, the smaller the size of each region, and therefore the more likely that some portion of V will match *any* template. In the limiting case, when V is segmented into $|V|$

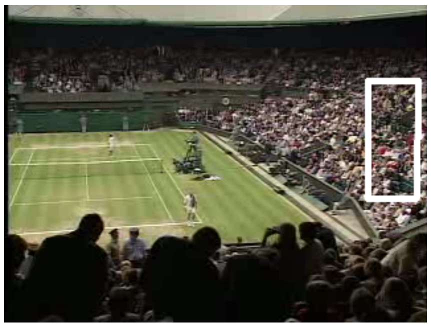


Figure 13. False positive in cluttered video regions. Because of the extreme over-segmentation of these regions, the tiny regions can be arbitrarily grouped to match any template.

unit-sized supervoxels, then the distance between V and any template is 0, since any volume can be trivially constructed from $1 \times 1 \times 1$ voxels. This motivates the need for a normalization term that balances the template match by the target volume's inherent flexibility. This is illustrated in Figure 14, where we match an arbitrary template (a) to two video volumes with a normal segmentation (b) and an extreme over-segmentation (c). The expected distance between the template and the extreme over-segmented video is much smaller than the distance to the typically-segmented video, as illustrated in (d) and (e). While the shape matching would fail in extreme over-segmentations, we propose a normalization model to compensate for a moderate amount of over-segmentation as follows.

We divide the distance metric $d(T, V)$ by a normalization term so that it becomes

$$d_N(T, V; l) = \frac{d(T, V; l)}{E_{\mathcal{T}}[d(\cdot, V; l)]}, \tag{12}$$

where the denominator is the expected distance of a template to volume V, averaged over $\mathcal{T}$, the set of all possible templates that fit within V. Essentially, the normalization term $E_{\mathcal{T}}[d(\cdot, V; l)]$ is measure of of the match confidence. Enumerating through all possible templates to compute the expected value may seem intractable at first, but we show that it is possible to compute this efficiently. Writing out the definition of the expectation, we have

$$\begin{aligned} E_{\mathcal{T}}[d(\cdot, V)] &= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}} d(\tau, V) & (13)\\ &= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{i=1}^{k} d(\tau, V_i), \text{ by Eqn. 10} & (14)\\ &= \frac{1}{|\mathcal{T}|}\sum_{i=1}^{k}\sum_{\tau\in\mathcal{T}} d(\tau, V_i), \text{ by indep.} & (15)\end{aligned}$$

Since practical templates represent one solid object (*e.g.*, a person) or a small set of solid objects (*e.g.*, a few people

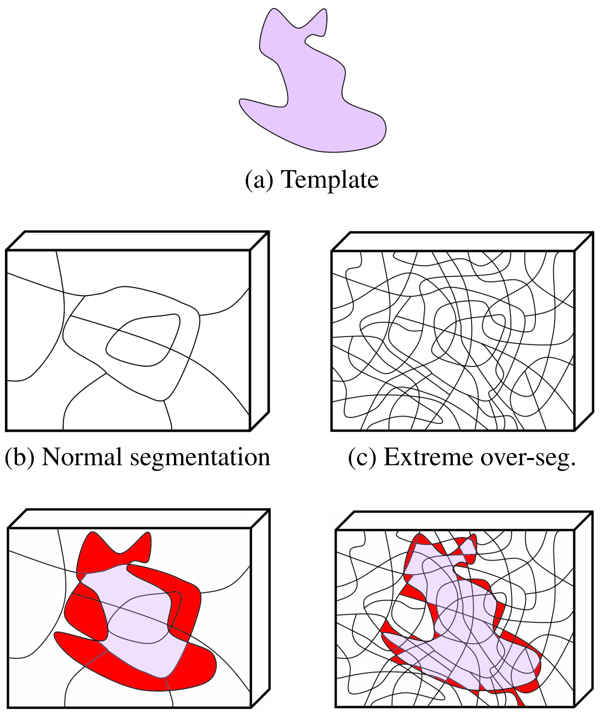


Figure 14. Illustration of how the expected distance between an arbitrary template (a) to a typically-segmented video (b) is much larger than the distance to an extreme over-segmented video (c). This motivates the need for a regulization term in the distance metric.

interacting), the set of templates T should include only 3D templates composed of a small number of connected pieces. However, constraining the enumeration of T to the connected templates is combinatorially hard to compute. As an approximation, we estimate the sum by considering *all* templates that fit within the volume V whether or not they are connected. For each region V_i, we enumerate all possible templates that have j pixels intersecting the region, which is $2^{|V|-|V_i|}\binom{|V_i|}{j}$. Then, we calculate the distance between the region and the template which is either the area of the intersecting region or the non-intersecting region, whichever is smaller. Therefore, the expected distance is equal to

$$\begin{aligned} &= \frac{1}{2^{|V|}}\sum_{i=1}^{k}\sum_{j=1}^{|V_i|-1} 2^{|V|-|V_i|}\binom{|V_i|}{j}\min(j, |V_i| - j)\\ &= \sum_{i=1}^{k}\frac{1}{2^{|V_i|}}\sum_{j=1}^{|V_i|-1}\binom{|V_i|}{j}\min(j, |V_i| - j). & (16)\end{aligned}$$

This can be simplified to:

$$= \sum_{i=1}^{k} f(|V_i|), \text{ where} \tag{17}$$

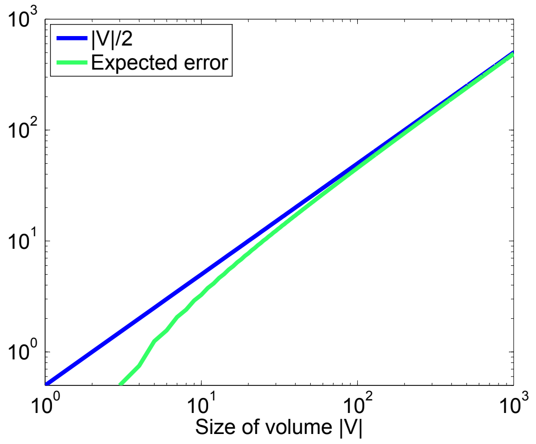


Figure 15. Illustration of Equation 18 on the size of the volume $|V|$ in a log-log plot. The expected error contribution, $f(n)$, approaches $n/2$ as n increases.

$$f(n) = \begin{cases} \frac{n}{2} - \frac{1}{2^n}\binom{n}{n/2}(n/2), & n \text{ even}, \\ \frac{n}{2} - \frac{1}{2^n}\binom{n-1}{(n-1)/2}n, & n \text{ odd}. \end{cases} \tag{18}$$

There exists a simple recurrence for computing $f(n)$ (Eqn. 18) exactly. Let $f(n) = n/2 - T(n)$. If n is even, then $T(n)$ is defined as

$$T(n) = \frac{1}{2^n}\binom{n}{n/2}(n/2). \tag{19}$$

The recurrence can be calculated as

$$\begin{aligned} T(n+2) &= \frac{n+1}{n}T(n) && (20) \\ T(2) &= 1/2. && (21) \end{aligned}$$

If n is odd, then $T(n)$ is defined as

$$T(n) = \frac{1}{2^n}\binom{n-1}{(n-1)/2}n. \tag{22}$$

The recurrence can be calculated as

$$\begin{aligned} T(n+2) &= \frac{n+2}{n+1}T(n) && (23) \\ T(1) &= 1/2. && (24) \end{aligned}$$

To get an intuition of how this normalization function behaves, $f(n)$ is illustrated in Figure 15 as a log-log plot. As n increases, $f(n)$ approaches $n/2$ as expected. Note that Equation 17 depends only on the size of the regions V_i and therefore can be pre-computed. At run-time, we only need to perform one table look-up for each supervoxel in the volume.

4.4. Complementary Nature of Shape and Flow

We now highlight some fundamental limitations of shape- and flow-based features and how these can be overcome when the two feature types are combined. Previous work that employs shape features, whether in images or video, typically extracts the outline or silhouette of the object. This raw shape is then frequently represented as a binary image. Since silhouettes are robust to appearance variations due to internal texture and illumination, they are unable to represent the internal motion of an object. For example, a textured rolling ball is indistinguishable from a static ball based on shape alone — yet could easily be recognized based on flow. Figure 16 shows a portion of a hand-clap action sequence. When viewed from the front, the silhouette changes very little, although there is a distinctive change of flow at the hands. Therefore, one would expect the addition of flow features to help particularly in cases where an action cannot be distinguished from its silhouette alone.

Conversely, some actions cannot be distinguished using flow-based features alone. While such features explicitly model the motion of an object, they only implicitly model the object shape; more importantly, the shape of stationary parts of the object are ignored. For example, as we have previous observed in the KTH action recognition database, the flow of the boxing action looks very similar to that of the hand-clap (see Figure 17) [33]. This is because the horizontal trajectories of the arms is similar and the (stationary) body of the actor is invisible; thus the outward motion of the punch matches the inward motion of the clap. However, a shape-based feature could trivially distinguish between the person and the grassy background and disambiguate these actions. Therefore, we argue that shape- and flow-based features are complementary and should be used in conjunction for action recognition. We believe that we are the first to propose a volumetric approach that combines these two feature types and show their effectiveness on non-background subtracted videos. While the idea of combining shape and flow is well known, we show how they complement each other well in our application.

Despite the normalization, our shape-based correlation algorithm can sometimes generate false positives on highly-textured regions, which are finely segmented (Figure 18a). However, we can obtain accurate flow measurements on these regions and a flow-based algorithm such as Shechtman and Irani's flow consistency [54] can filter out these false positives. Similarly, uniform regions pose an analogous problem for flow-based algorithms because these regions have *indeterminate* flow, and therefore can match *all* possible templates. Consequently, we add a pre-filtering step to Shechtman and Irani's technique to discard uniform regions by thresholding on the Harris score of the region. Even with this filtering, we observe that the majority of false-positives occur in low-textured regions (Figure 18b). Fortunately, our shape-based correlation works well on those regions and can be used to filter out the false positives. We quantify the benefits of combining shape and flow in Section 4.6.

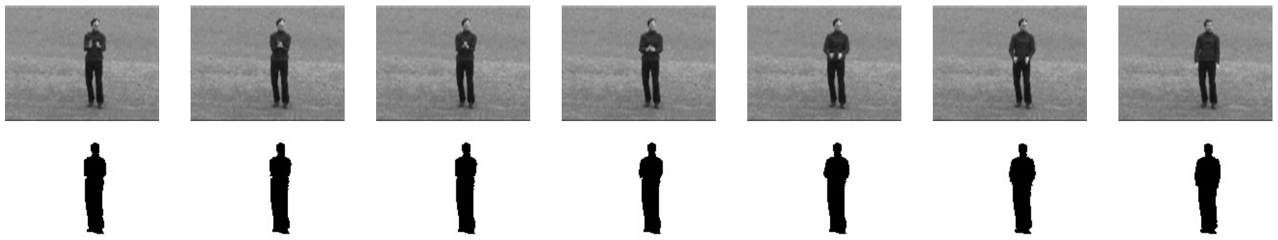


Figure 16. Notice how the silhouette stays constant during this part of the hand-clapping event. More generally, a fundamental limitation of such shape features is that they cannot represent motion inside the silhouette.

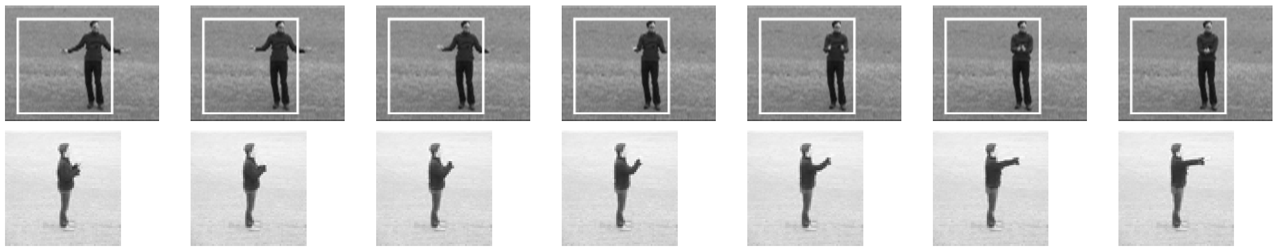


Figure 17. Although clapping (top) and boxing (bottom) are distinct actions, they are very similar in terms of local flow alone. This is because the motion in the right half of the white window (generated by the *inward* movement of the clap) is similar to the flow caused by the *outward* extension of the punching arm. However, the shape feature can easily determine that the center of the white window contains background in the former and a person in the latter, and thus eliminate this false positive.

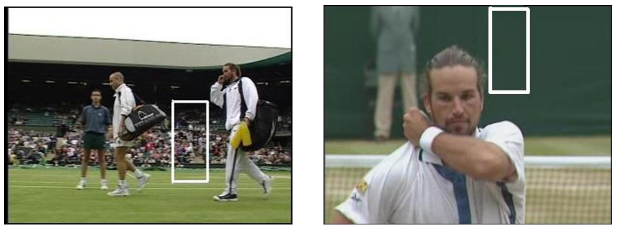

(a) shape fp (b) flow fp

Figure 18. False positives found using a) shape correlation and b) flow consistency correlation. The false positives using shape features occur on highly textured regions, whereas the false positives using flow features occur on uniform regions. Using both features filters out each other's false positives.

4.5. Baseline Detection Algorithm

This section describes a baseline technique for detecting events using spatio-temporal shape and flow correlation. This technique is not scale invariant; standard techniques such as scanning a pyramid of scales can be applied. A more powerful parts-based method for detection is described in Section 5. Suppose first (for now) that we have a template of a single instance of an action of interest. To find other instances of this action in a video clip, we can slide the template over the entire video and measure the correlation distance at all locations in space and time. We use Shechtman and Irani's flow correlation method (described in Section 2.7) to measure the flow matching distance [56]. More formally, for each location $l = (x, y, t)$ in the video, we position the template T at l and measure the shape and flow correlation between T and the video. We use a linear combination of the shape and flow correlation distance as follows:

$$d(T, V; l) = d_N(T, V; l) + \alpha d_F(T, V; l) \text{ from Eqns. 12, 7,} \tag{25}$$

where α is the relative weight between the flow and shape distances. We use $\alpha = 0.2$ for all experiments. Thresholding the correlation distance and finding the peaks give us locations of potential matches. Figure 19 shows the minimum correlation distance of a hand-wave action projected on a time axis. Note that the cyclic nature of the action and the distance is minimized when the phases of the template and the action match. Although the action in the video is periodic, our algorithm does not assume periodic motion and thus we can detect all instances of the event and localize them in both space and time. The advantage of using single templates for matching is that minimal human effort is required to bootstrap the system. This works well in scenarios where we have not trained the system on a large collection of template actions or where a human operator is interactively searching for novel events in large video databases. In such scenarios, the user can manually adjust the threshold to balance the detection and the false positive rates. We will show in Section 6 on how to generalize to multiple templates.

4.6. Evaluation of Baseline Method

We now illustrate how our baseline algorithm performs on an event detection task, where we try to detect and lo-

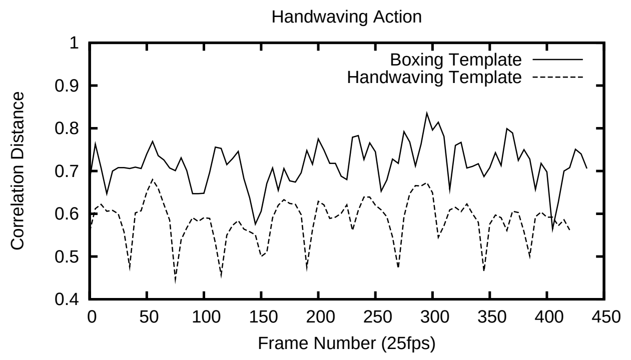


Figure 19. The minimum correlation distance of two templates on a hand-waving action. Notice the cycles in the action, where the distance is minimized when the phase of the template matches that of the action.

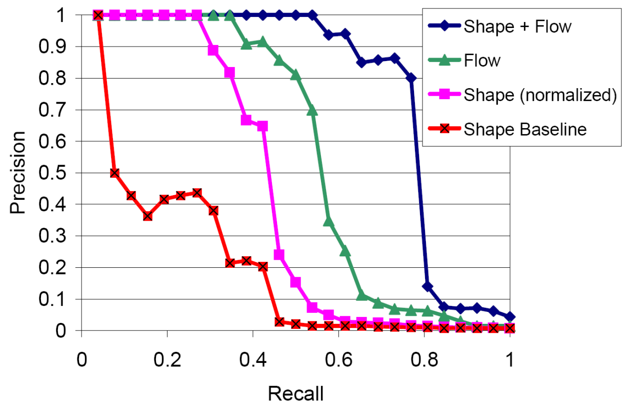


Figure 20. Comparison of detection performance using various features on 30 minutes of a tennis video.

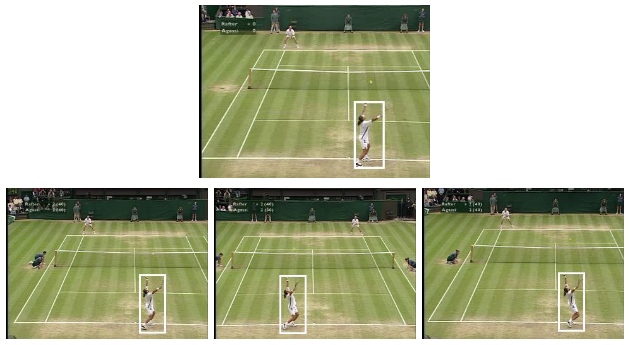


Figure 21. Illustration of detected events in the tennis sequence. Top: template; Bottom row: example detections.

calize an event in space-time. For our first experiment, we used a real life video — a Wimbledon 2000 match between Agassi and Rafter [2]. This experiment is difficult because the video contains a lot of clutter (*e.g.*, Figure 18a) and only a few instances of the actions are present in the video. We manually selected an example of Rafter serving (Figure 21a) and used it as a template to find all other instances of him serving in the first 30 minutes of the video. The template was scanned over all spatio-temporal locations and we kept the best match for each frame, assuming the action only occurs at most once per frame. There were 28 instances of the serve and we considered a detection to be a positive match if there was at least 75% overlap between the detection bounding volume and the manually-labeled event volume. Figure 20 shows the results of using various matching methods, where we varied the matching distance threshold to generate the precision-recall curve. “Shape Baseline” is the performance of our shape-based region intersection algorithm without normalizing for segmentation granularity. “Shape (normalized)” normalizes for the segmentation granularity and performs markedly better. “Flow” represents the results from our implementation of Shechtman and Irani’s algorithm. In this experiment, flow-based correlation performs better than shape-based correlation. This is partly due to false positives matching on finely-segmented crowd scenes, despite the normalization. Combining both features using a weighted sum of the detection scores, we achieve the best results with 80% recall at 80% precision. The two features remove each other’s false positives and results in a much higher precision. The dataset and results are illustrated in Figure 21.

5. Parts Based Recognition

The previous section describes a method for matching volumetric shape features on automatically-segmented video. The main strength of the baseline algorithm is that it can perform shape matching without precise object masks in the input video [10, 11, 71]. Therefore, we can use shape information extracted from automatic volumetric segmentation instead of relying on background subtracted foreground masks. Further, using template-based matching enables search with only one training example. However, like all template-based matching techniques [11, 54], it suffers from limited generalization power due to the variability in how different people perform the same action. A standard approach to improve generalization is to break the model into parts, allowing the parts to move independently, and to measure the joint appearance and geometric matching score of the parts. Allowing the parts to move makes the template more robust to the spatial and temporal variability of actions. A parts-based model allows us to improve generalization even with a single training template. While we do not have a fully and arbitrarily deformable model, the parts-based model is one step in making the model more deformable. This idea has been studied extensively in recognition in both images [68] and video [12, 55]. Therefore, we extend our baseline matching algorithm by introducing a parts-based volumetric shape-matching model. Specifically, we extend the pictorial structures framework [24, 25] to video volumes to model the geometric configuration of

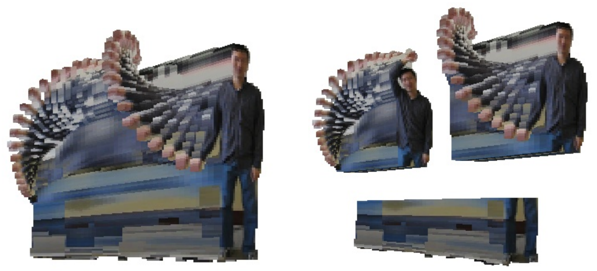


Figure 22. To generalize the model and allow for more variability in the action, we break the action template (a) into parts (b). The model can be split in both space or time to generate the parts.

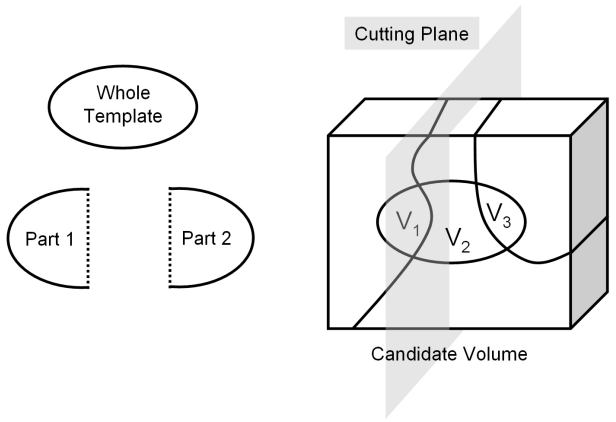


Figure 23. Illustration of how we artificially cut the candidate volume to match how the whole template is split into its constituent parts. The candidate volume is dynamically cut as we slide the template parts along it. The cutting process is very efficient.

the parts and to find the optimal match in both appearance and configuration in the video.

5.1. Parts Based Shape Descriptor

A key feature of our baseline algorithm is that it can perform shape matching with over-segmented regions. However, it assumes that the template consists of a single region, and that only the video is over-segmented. Given a single template, one must use prior knowledge to break the template into parts. For events that consist of human actions, these parts typically correspond to the rigid sections of the human body, and therefore the process is straightforward. We illustrate how one might manually break the handwave template into parts, as shown in Figure 22. We note that, for this action, only the upper body moves while the legs remain stationary. Therefore, a natural break should be at the actor's waist. Such a break would allow the template parts to match people with different heights. Another natural break would be to split the top half of the action temporally, thus producing two parts that correspond to the upward and downward swing of the handwave action. This allows for some variation in the speed with which people swing their arms. It is important to note that, just like the whole template, the parts are also spatio-temporal volumes and could represent a body part in motion.

We now generalize our baseline algorithm (in Section 4) and describe how we match template parts to over-segmented regions. Consider the oval template that has been split into two parts in the toy example illustrated in Figure 23. Although the whole template matches the oval ($V_1 \cup V_2 \cup V_3$) in the candidate volume, the parts would match poorly because the over-segmentation is inconsistent with the boundaries between the two parts. For example, our baseline algorithm would not match Part 1 to V_1, nor Part 2 to V_3. In general, there is no reason to believe that they should match because some of the part boundaries are artificially created (as shown by the dashed lines) and do not necessarily correspond to any real object boundaries. Our solution is to introduce additional cuts using a virtual plane that is aligned to and moves with the template part. For example, as we slide Part 1 across the video, we subdivide all the regions that intersect with the cutting plane placed on the right edge of the Part 1. V_2 is divided correctly, and Part 1 now matches the union of V_1 and the shaded region of V_2. For convenience, we only use cutting planes that are aligned with the principal axes, but in general the plane can be oriented in any direction. By pre-computing the cuts and with judicious bookkeeping, the parts-based matching can be performed with the same computational efficiency as our baseline shape-based matching algorithm.

5.2. 3D Pictorial Structures

We now describe how the framework of pictorial structures [24,25] can be extended to parts-based event detection in video. Intuitively, each part in the template should match the video well, and the relative locations of parts should be in a valid geometric configuration. More formally, consider a set of n parts that form a tree in a graph. Adopting a notation based on Felzenszwalb and Huttenlocher [24], let the part model be specified by a graph $G = (P, E)$. Template part T_i is represented as a vertex $p_i \in P$ and the connection between parts p_i and p_j is represented as an edge $(p_i, p_j) \in E$. The configuration of the parts is specified by $L = (l_1, \ldots, l_n)$, where $l_i = (x_i, y_i, t_i)$ is the location of part T_i in the candidate volume V. Let $a_i(l_i)$ be the correlation distance between the template part T_i and the video at location l_i. Let $d_{ij}(l_i, l_j)$ be the distance in configuration between parts T_i and T_j when they are placed at locations l_i and l_j, respectively. We want to find the optimal location of all of the parts, L^*, by minimizing the energy function:

$$L^* = \operatorname*{argmin}_L \left(\sum_{i=1}^{n} a_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \quad (26)$$

The correlation distance $a()$ is a linear combination of our normalized distance metric (Equation 12) and Shechtman and Irani's flow-based correlation distance (Equation 7):

$$a_i(l_i) = d_N(T_i, V; l_i) + \alpha d_F(T_i, V; l_i), \tag{27}$$

where $\alpha = 0.2$ and we use the same weight for all experiments. For matching efficiency, our parts model is organized in a tree structure and we model the relative position of each part as a Gaussian with a diagonal covariance matrix. Therefore,

$$d_{ij}(l_i, l_j) = \beta \mathcal{N}(l_i - l_j, s_{ij}, \Sigma_{ij}), \tag{28}$$

where $l_i - l_j$ represents the offset in (x, y, t) between part T_i and T_j, s_{ij} is the mean offset, and Σ_{ij} is the diagonal covariance. β adjusts the relative weight of the configuration vs. appearance terms and for all of our experiments we use $\beta = 0.02$. The mean offset is taken from the location where we cut the parts, and the covariance is set manually, typically around 10% of the template size. As described by Felzenszwalb and Huttenlocher [24], the minimization can be efficiently solved using distance transforms and dynamic programming. Because we use a sliding window approach to event detection, we also record the actual distance solved in the minimization and threshold on the distance. Only those below a specified threshold are considered as detections. As we pointed out earlier, the key feature of the algorithm is that it requires a segmented event template as a model, but it does not require an exact segmentation of the input video, thus making detection possible in cases, such as crowded scenes, in which reliable segmentation would be difficult.

5.3. Evaluation of Parts-Based Matching

To evaluate the effectiveness of our parts-based matching algorithm, we selected events that represent real world actions such as picking up an object from the ground, waving for a bus, or pushing an elevator button (Figure 25). We acquired videos by using a hand-held camera in cluttered environments with moving people or cars in the background. This data set is designed to evaluate the performance of the algorithm in crowded scenes. We study the effects of using different combinations of shape and flow descriptors, and parts-based versus whole shape models. One subject performed one instance of each action for training.[2] Between three to six other subjects performed multiple instances of the actions for testing. We collected approximately twenty minutes of video containing 110 events of interest. The videos were down-scaled to 160×120 in resolution. There is high variability in both how the subjects performed the actions and in the background clutter. There are also significant spatial and temporal scale differences in the actions as well.

[2]The two-handed wave action template was taken from the KTH videos.

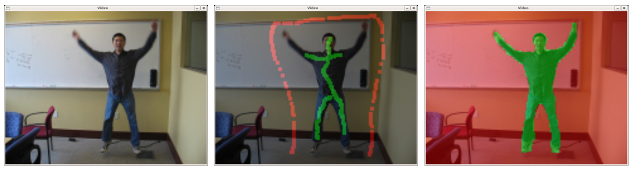


Figure 24. Illustration of how we generate model templates using an interactive video segmentation process similar to Wang *et al.* [65]. Given the training video (left), we draw a few strokes to label the foreground and background regions (middle), and graph-cut generates the complete segmentation mask (right).

For each event, we create the model from a single instance by interactively segmenting the spatio-temporal volume using a graph-cut tool similar to [65]. Note that we do not require the entire person to be observed and segmented to detect an event. For example, in the hand-wave event in Figure 25, only the upper torso is labeled and used for training. The templates are typically $60 \times 80 \times 30$ in size and range from 20,000–80,000 voxels. The whole template is then manually broken into parts, as shown in Figure 25. The video is automatically segmented using mean shift; the average segment size is approximately 100 voxels. We scan the event template over the videos using the shape and flow distance metrics described earlier, and combine them using pictorial structures. There are approximately 120,000 possible locations to be scanned per second of video for a typical template. In these experiments, to evaluate the robustness of our matching algorithm to variations in observed scale, we match only at a single fixed scale; in practice, one could match over multiple scales. The algorithm returns a three-dimensional distance map representing the matching distance between the model and the video at every location in the video. For efficiency, we project the map to a one-dimensional vector of scores, keeping only the best detection for each frame, as shown in Figure 26(a). Since it is rare for two instances of an action to start and end at exactly the same time instant, this is a reasonable simplification. An event is detected when the matching distance falls below a specified threshold. We vary this threshold and count the number of true positives and false positives to generate the Precision-Recall graphs. A detected event is considered a true positive if it has greater than 50% overlap (in space-time) with the labeled event.

We now analyze the performance of our algorithm and compare it to baseline methods. Figure 25 shows example detections using our algorithm with the parts-based shape and flow descriptor in crowded scenes. Note the amount of clutter and movement from other people near the event. The precision-recall graphs for all of the actions are shown in Figures 26(b)–(f). We compare our results to Shecht-

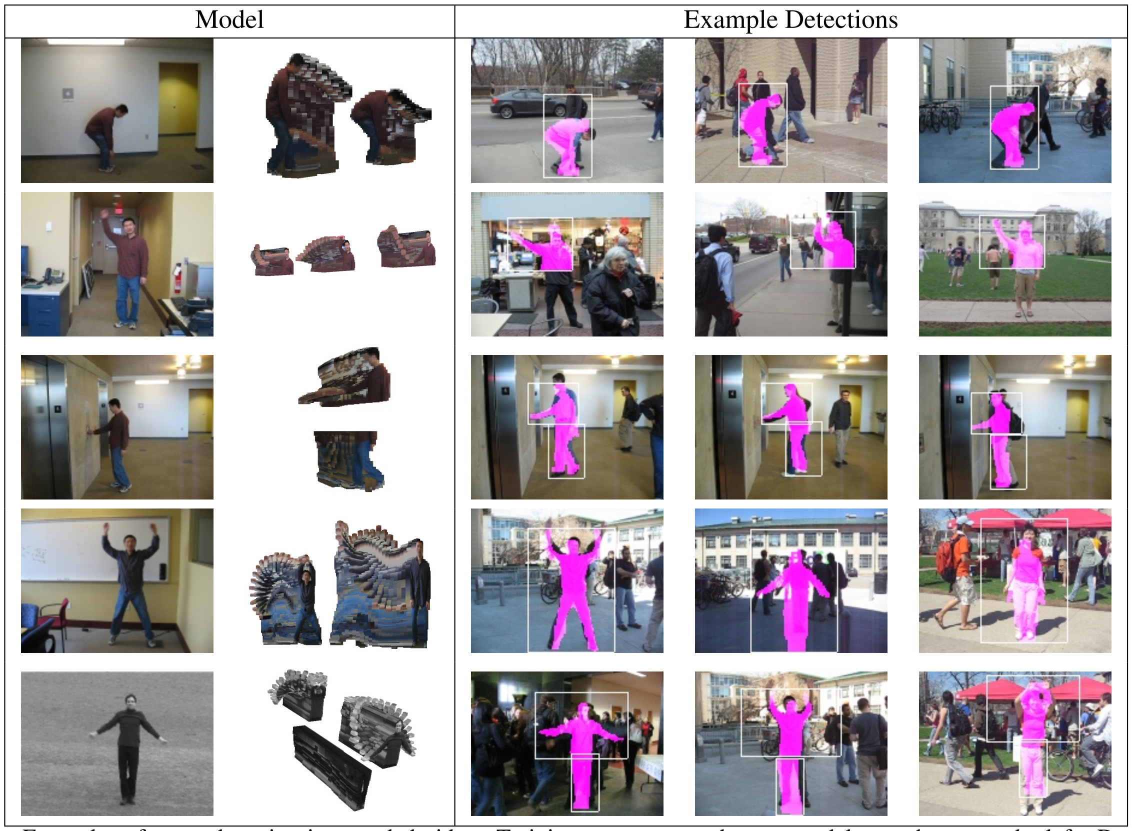


Figure 25. Examples of event detection in crowded video. Training sequences and event models are shown on the left. Detections in challenging test sequences are shown on the right. The action mask from the appropriate time in the event model is overlayed on the test sequence, and a bounding box drawn around each part.

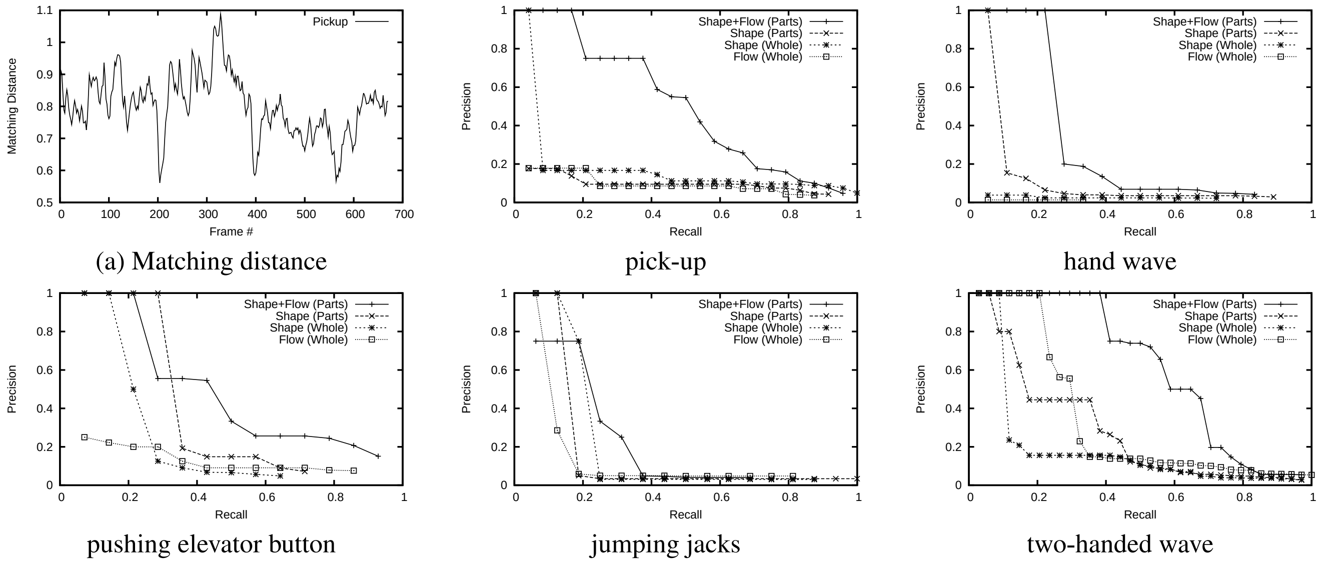


Figure 26. (a) Projected matching distance on video with three pick-up events. A threshold of 0.6 successfully detects all of them. (b)–(f) Precision/recall curves for a variety of events. Our parts-based shape and flow descriptor significantly out-performs all other descriptors. The baseline method [54], labeled as "Flow (Whole)", achieves low precision and recall in most actions, demonstrating the difficulty of our dataset.

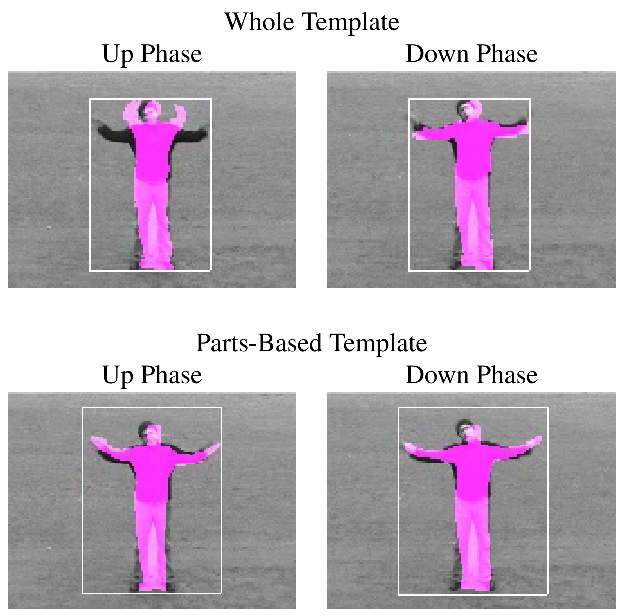


Figure 27. Illustration of how our parts-based model can stretch to accomodate the mismatch in speed between the template and the detected action. While the down-phase of the hand-wave is matched to the whole template, the up-phase is clearly misaligned. The parts-based template matches both phases well and thus enables better detection accuracy.

man and Irani's flow consistency method [54] as a baseline, labeled as Flow (Whole) in our graphs. This state-of-the-art baseline method achieves low precision and recall in nearly all actions, demonstrating the difficulty of our dataset. Our combined parts-based shape and flow descriptor is significantly better and outperforms either descriptor alone [35]. The parts-based shape descriptor is better than the whole shape descriptor in the hand-wave, push button, and two-handed wave actions, while there is little benefit to adding the parts model for the jumping-jacks and pick-up actions. To llustrate qualitatively the benefits of the parts-based model, we applied the two-handed wave template to one of the videos in the KTH dataset. Figure 27 shows the difference between the whole and parts-based templates as they are overlayed (in pink) onto the video. There was a difference in handwaving speed between the template and the target video. Consequently, the whole template was not able to align both phases of the action, while the parts-based template was able to stretch to match the action well.

5.4. Automatic Part Generation

We have previously assumed that the parts were manually cut from a single training template. We would like to automatically cut the whole template in Figure 22a to look like the parts in Figure 22b. Given many training examples, we can automatically learn the optimal place to cut the templates. Intuitively, the cuts divide the templates into parts

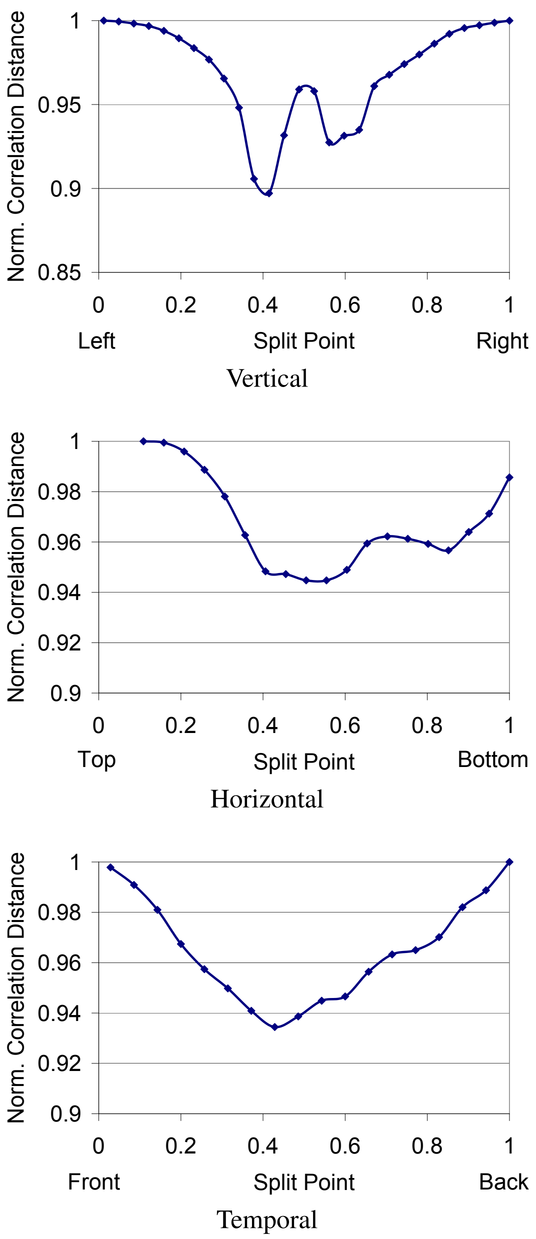


Figure 28. Matching distance at different cut locations along each axis.

that move independently of each other. This is done by trying cuts at various locations and see which cut gives the best performance. Performance is measured by the amount of volumetric overlap between the individual parts.

Suppose we have one training template that we would like to cut and n labeled events in a set of video files. First, we choose a cutting plane, for example the X-T plane, which cuts the templates horizontally. We iterate through all possible locations at which to cut, which in this case is equal to the height of the template. At each cut location, we match the parts to the labeled events and we measure

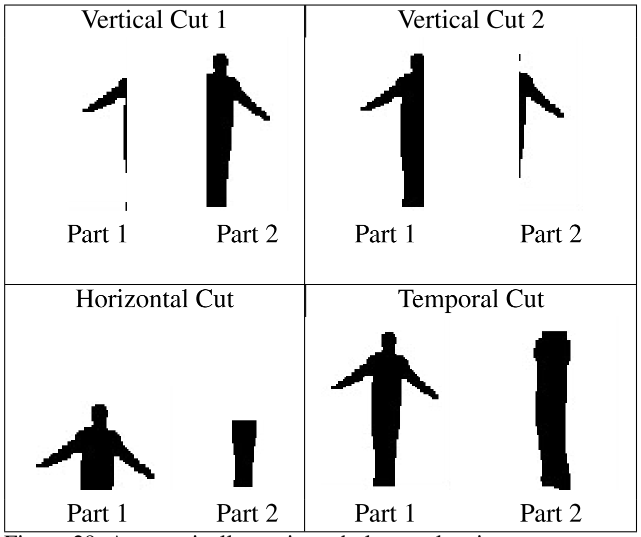


Figure 29. Automatically cutting whole template into parts.

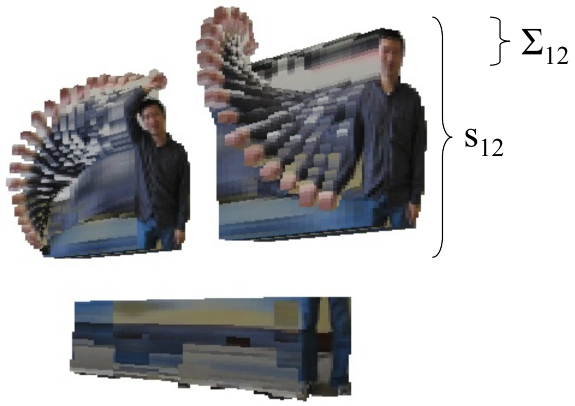


Figure 30. Illustration of how the part offset parameters are initialized.

the shape correlation distance. We scan a small area around the labeled event to find the best matching distance. The cut location that minimizes the distance between the parts and all labeled events is considered the best one. Figure 28 shows the plot of the matching distance between the *two-handed wave* template and the labeled events at various cut locations. We see that there are two vertical cuts that generate good matches. The parts are illustrated in Figure 29 and we see that these two vertical cuts are near the arms, which is quite intuitive. The horizontal cut is around the waist, suggesting that the arms move independently as expected. Finally, the temporal cut splits the up and down phases of the wave. We show only the first division of the templates in Figure 29. Further divisions can be made by recursively dividing the individual parts. One must weigh the trade-off between robustness in using more parts and the specificity of the template with fewer parts. Ideally, the number of parts to generate could be automatically learned as well. Instead of automatic cuts, one could cut the parts manually, but the cuts would arbitrarily placed and likely to be suboptimal. Once we have the parts, we can learn the part configuration parameters, which we describe next.

5.5. Learning Part Configurations

The two main parameters for the parts configuration are the mean and covariance of the part offsets. When we were given only one template, we specified the mean offset, s_{ij}, as the location where we cut the part and the covariance, Σ_{ij}, as 10% of the template size. Ideally, we would want to learn these parameters automatically given training data. If there are multiple labeled examples for an event, we can learn these parameters as follows. We propose two methods for learning the parameters – a completely supervised method and a semi-supervised method.

The supervised method for learning the part parameters is straight-forward. First, we assume that we have already cut the template T into a set of m parts, $T = \{T_1 \ldots T_m\}$. We then manually label the location of each part for all n training examples. We denote l_i^k as the location of part T_i for the k^{th} training example. The mean and covariance of the part offsets, s_{ij} and Σ_{ij}, can be estimated directly.

$$s_{ij} = \frac{1}{n} \sum_{k=1}^{n} (l_i^k - l_j^k). \tag{29}$$

$$\Sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} (l_i^k - l_j^k - s_{ij})^2. \tag{30}$$

However, this is a labor-intensive process and we would like to minimize the amount of required manual labeling. Further, we are constrained by the parts given to us; if the template is divided into another part configuration, we would need to relabel all of the events. Therefore, we propose a semi-supervised method for learning the part parameters.

The overall learning is an EM-like iterative process [44]. First, we label only the bounding box surrounding the event. Because we do not label the individual parts, the manual labeling is much faster. However, this creates additional unknowns that must be estimated. Specifically, we need to estimate the part offsets for each labeled event, $l_i^k - l_j^k$. We initialize our estimate of these parameters using the same method as we used for the single template in Section 5.2. s_{ij} is initialized to the location where we cut the part, and Σ_{ij} is initialized to 10% of the size of the template. Figure 30 shows an illustration of the parameters between two parts. Now, we need to find the location of all the parts for every labeled example, $\{l_i^k\}$. Using the initial guess of s_{ij} and Σ_{ij}, we run the detector in a small area around the labeled event. From the k^{th} event, we use pictorial structures to find the optimal location of the parts, $L^{k*} = \{l_1^k, \ldots, l_m^k\}$ (using

```
Input: $E = \{E_1 \dots E_n\}$: Location of training events.
        $s', \Sigma'$: Initial estimates of $s$ and $\Sigma$.
        $T = \{T_1 \dots T_m\}$: Template parts.
Output: Estimated part configurations.

repeat
    for $i = 1 \dots n$ do
        Estimate part locations $\{l_i^1 \dots l_i^m\}$ for event $E_i$
        using pictorial structures with $T$, $s'$, and $\Sigma'$.
    end
    Estimate $s, \Sigma$ using part locations for all events
    (Equations 29 and 30).
    $s' = s$ ;
    $\Sigma' = \Sigma$ ;
until $s, \Sigma$ converges ;
return $s, \Sigma$ ;
```

Algorithm 2: Estimating part configurations.

Equation 26). Given the set of part locations, we can now re-estimate s_{ij} and Σ_{ij} using Equations 29 and 30. This process is repeated until convergence. Our experiments indicate that they converge relatively quickly, usually after a few iterations. The process is summarized in Algorithm 2.

6. Improving Robustness

There are many important and practical issues that arise when we apply event detection in real-world videos. There are more drastic variations, such as viewpoint, scale, actor variability, and camera movement. We analyze how our system performs in various settings and how robust it is to these kind of variations.

6.1. Robustness to Viewpoint

A common challenge to all view-based recognition algorithms is that they are dependent on the camera view point. While our method is not view invariant, it is robust to small changes in viewpoint. We quantify this using the Multiview dataset, where we used four cameras to simultaneously capture the events. The cameras were arranged in a 45 degree semi-circle around the actors and sample frames are illustrated in Figure 8. We averaged the area under the precision-recall curve (AUPRC) [7, 13] over all actions and plotted them against the camera viewpoint, shown in Figure 31. The models were trained using a camera placed directly in front of the person, so we expect "0 degrees" to give the best performance. The curves show that for all features, there is only a slight drop in performance for a skew of up to 30 degrees. At 45 degrees, the performance drops significantly.

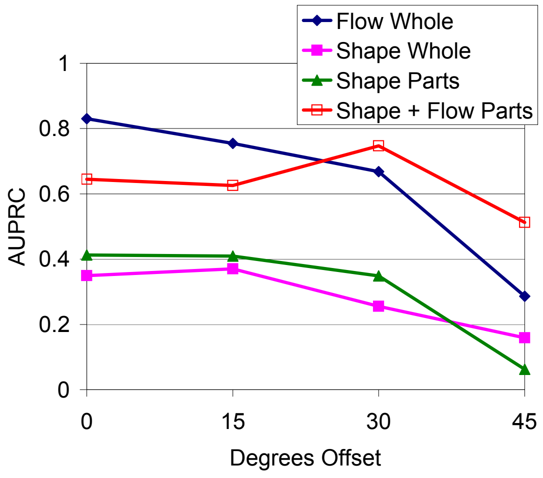


Figure 31. Multiview comparisons. The bump at 30 degrees for the Shape+Flow Parts method is likely due to noise.

6.2. Multi-scale Detection

Our baseline algorithm is not invariant to spatial and temporal scale changes in the events. While our parts-based method is more robust to scale variations, a complete system would need to also explicitly search across scales. We show results that confirm the belief that searching across scales improves performance. We scaled the templates linearly in both space and time and searched for these events in the Multiview dataset. The templates were rescaled using factors of (0.8, 1.0, 1.2) in space and time to generate 9 templates from each of the original templates. The AUPRC averaged overall all actors, actions, and camera view points are plotted in Figure 32. Column (a) shows the detection results using a single template per action. Column (b) shows the results with the multi-scale templates and it clearly improves over the single template.

A potential problem with using multi-scale templates is that since the templates are different sizes, they cannot be directly compared to each other using the same detection threshold. This is evident from our distance metric:

$$d(T, V_i; l) = \begin{cases} |T \cap V_i| & \text{if } |T(l) \cap V_i| < |V_i|/2 \\ |V_i - T(l) \cap V_i| & \text{otherwise.} \end{cases} \tag{31}$$

Larger templates generate larger distances, and therefore we cannot use the same detection threshold for different templates. We introduce a regularization term that normalizes for the size of the template:

$$d_N(T, V; l) = \frac{d(T, V; l)}{E_{\mathcal{V}}[d(T, \cdot)]}. \tag{32}$$

Note that this is different than the regularization term for the segmentation granularity. We calculate $E_{\mathcal{V}}[d(T, \cdot)]$ empirically by averaging the correlation distance between the

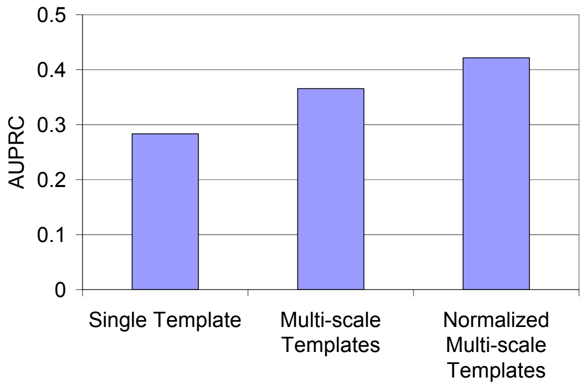


Figure 32. Effectiveness of scanning multiple scales in space-time (on the Multiview dataset). Left: Single template. Middle: in 9 scales (3 in space and 3 in time). Right: Multi-scale normalized templates to account for differences in template size.

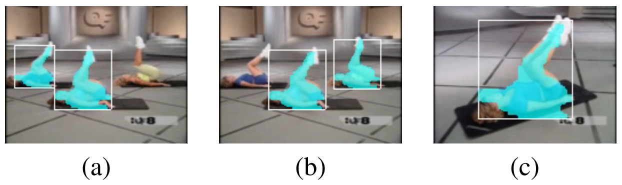


Figure 33. Multiscale on aerobics videos. Successful detections on actions with a 2.3x scale difference. Images in (a) and (b) are from the same sequence a few tenths of a second apart and show that all three events were detected at slightly different temporal offsets. The actors were not completely synchronized.

template T and many segmented videos. Using this normalized distance metric, we achieve the best detection results, as shown in column (c). Figure 33 shows anecdotal results on the Aerobics dataset. The spatial scale difference between the two views is 2.3$\times$ and we successfully detect all of the events. The bounding box and template overlay show that we correctly detected the scale of the events.

Unlike variations in spatial scale, there are much smaller temporal scale variations between actions. Spatial scale variations are determined by the size of the person, the distance between the person and the camera, and the resolution of the camera. The best-known technique for recognizing human actions at low resolutions is limited to a minimum of around 30 pixels [22]. For 640 $\times$ 480 resolution videos, this translates to a maximum of 16x in scale variation. Temporal variations, on the other hand, are much smaller. Table 2 shows the cycle periods of actions in the Weizmann dataset. The maximum scale change is only 1.43 as seen in the *Pjump* action. Therefore, we only need to scan across a small range of temporal scales for event detection.

Table 2. Cycle periods of actions in the Weizmann dataset. There are small temporal scale variations in this dataset, with a maximum of 1.43 on the *Pjump* action.

Action	Avg	Std. Dev.	Max/Min
Bend	2.35	0.13	1.19
Jack	1.11	0.06	1.17
Jump	0.54	0.06	1.36
Pjump	0.67	0.08	1.43
Side	0.63	0.03	1.17
Skip	0.49	0.05	1.36
Run	0.83	0.07	1.31
Walk	1.11	0.04	1.10
Wave1	1.15	0.08	1.26
Wave2	1.16	0.10	1.35

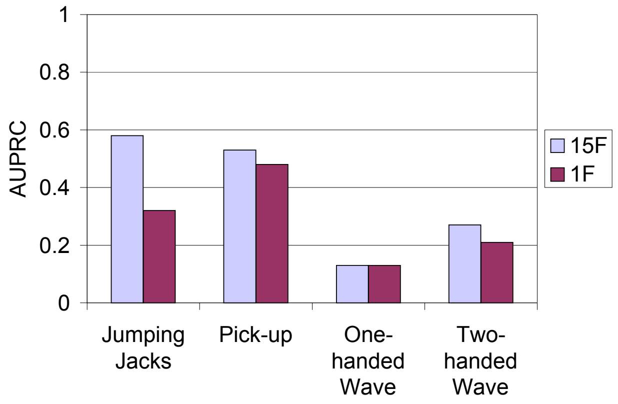


Figure 34. Recognition results comparing volumetric segmentation (15F) versus single-frame (1F) segmentation on 4 actions in the Weizmann dataset. We see that segmentation using blocks of 15 frames gives better results than single-frame segmentation.

6.3. Volumetric Segmentation

Using volumetric segmentation is better than segmenting the video on a frame-by-frame basis. Figure 5 showed anecdotally the improvement in quality between segmenting frame-by-frame versus blocks of frames. We now demonstrate quantitatively the effect of volumetric segmentation on recognition performance. Figure 34 shows the results of detecting various actions on the Weizmann dataset. "15F" denotes the results from segmentation on blocks of 15 frames and "1F" corresponds to the results from 1-frame segmentation. Only a single, whole shape template is used per action to isolate the effects of segmentation. All mean-shift segmentation parameters were fixed and we adjusted average number of segmentations per frame to be the same for both method. We see that segmentation using blocks of 15 frames leads to better recognition results. This is because volumetric segmentation is able capture the object boundaries more consistently through time. Further, given the same number of regions, volumetric segmentation can more accurately represent the boundaries.

6.4. Real-world Videos

We now show anecdotal results on real-world videos download from YouTube. The videos were found using the search terms “pick up coin” and “jumping jacks”. Figure 35 shows example detections on these actions. We used the same templates as in prior experiments and used the parts-based shape and flow features for detection. The videos are all unscripted and represent the diversity of how actions can be performed. Many of the videos are low quality due to high compression, noise, poor lighting, and have low frame rate. Since the videos are mostly taken with hand-held cameras, we do camera stabilization as a pre-processing step on the videos. Figure 36 shows how much camera movement there is in a typical video.

6.5. Robustness to Camera Movement

While our features are robust to small amounts of camera motion such as those from a hand-held camera, it is not invariant to large camera movements. These movements distort both the volumetric shape and flow features. We use an off-the-shelf camera motion estimation software, Motion2D [48], to compensate for these movements. A simple translation and rotation motion model is sufficient to stabilize the camera motion. Prior approaches have encountered similar problems and have also used camera stabilization to as a preprocessing step [6, 8, 45, 61]. To measure the effect of camera movement, we captured a sequence of videos that have large amounts of camera movement in them. There are two types of motion – camera shake (hand-held camera) and panning (simulating a pan-tilt camera). Figure 37 shows an example of camera shake and Figure 38 shows an example of a camera panning. The average displacement in this dataset is 2.73 pixels per frame, compared to a displacement of less than 0.1 pixels per frame in a typical hand-held camera. Panning at 2.73 pixels per frame gives a 82-pixel offset in one second of video, or roughly half the video frame since the videos are 160×120. It would be very difficult to reliably detect events with this amount of movement and without motion compensation. Figure 37 shows example frames from an original unstabilized and the corresponding stabilized video. Events that were undetectable in the original videos are easily detected in the stabilized videos. Quantitative results on the entire dataset is shown in Figure 39. The results for all actions improve when we add camera stabilization.

6.6. Comparison on standard datasets

Although our goal is to detect and locate events, we adapted our algorithm to perform video classification on the KTH action database to compare against other algorithms. The KTH actions database contains 25 people performing six actions in four different scenarios [53]. Each video clip contains one person performing an action multiple times under significant lighting, clothing, and scale changes (Figure 40). Different people also perform the actions at different speeds and orientations. The videos were recorded using a handheld camera which prevent simple background subtraction techniques from reliably extracting the person. The goal of the experiment is to classify the video clips into one of the six actions — walking, jogging, running, boxing, clapping, and waving. Classifying the entire video simplifies the training and recognition process because we do not have to label each instance of the action; we only need to label the sequence as a whole.

To classify the video sequences, we first classify the individual frames and assign the class with the highest frame count to the video clips. While there are a number of classifiers that one could use to classify the frames, we chose to use SVM because of its reported success in a wide range of applications. We train the SVM (using LIBSVM [14] and an RBF kernel) as follows. Given a candidate video at some space-time location, we correlate it with a database of n template actions. This gives us a feature vector of size n if use our region intersection algorithm, or $2n$ if we also include flow features. Each dimension of the feature vector corresponds to a distance between the candidate video location and a template action.

Following the methodology of Niebles *et al.* [47], we use leave-one-out cross-validation grouped by person to measure the classification accuracy. To generate the trianing data, we manually label 4 templates for each action. Each template contains one cycle of the action, typically 15 to 30 frames long. The videos used to extract the templates are removed from the cross-validation set. For each template t_i, we scan over all space-time locations in a video clip. For each frame of the video, we extract the best correlation score for each template. There is one feature f per frame, where f_i is the best correlation score to template t_i. During classification, each frame in a video clip is classified as one of the six actions and votes for the label of the entire video clip.

Figure 42 shows the confusion matrix on the KTH dataset. The result is generated using both shape and flow features and correlated against two templates per action. We achieve an accuracy of 80.9%, which is comparable to the most recent studies on the same dataset (Table 3). Unfortunately, we can only loosely compare the results in Table 3 because different groups employed different experimental methodologies. Like the other studies, we find that there is confusion mainly between walk-jog-run and box-clap-wave. As expected, running is more easily confused with jogging than with walking. Boxing is also more easily confused with clapping (horizontal motion) than waving (vertical motion). Figure 42 shows the effect of using different features and training on different number of templates. We

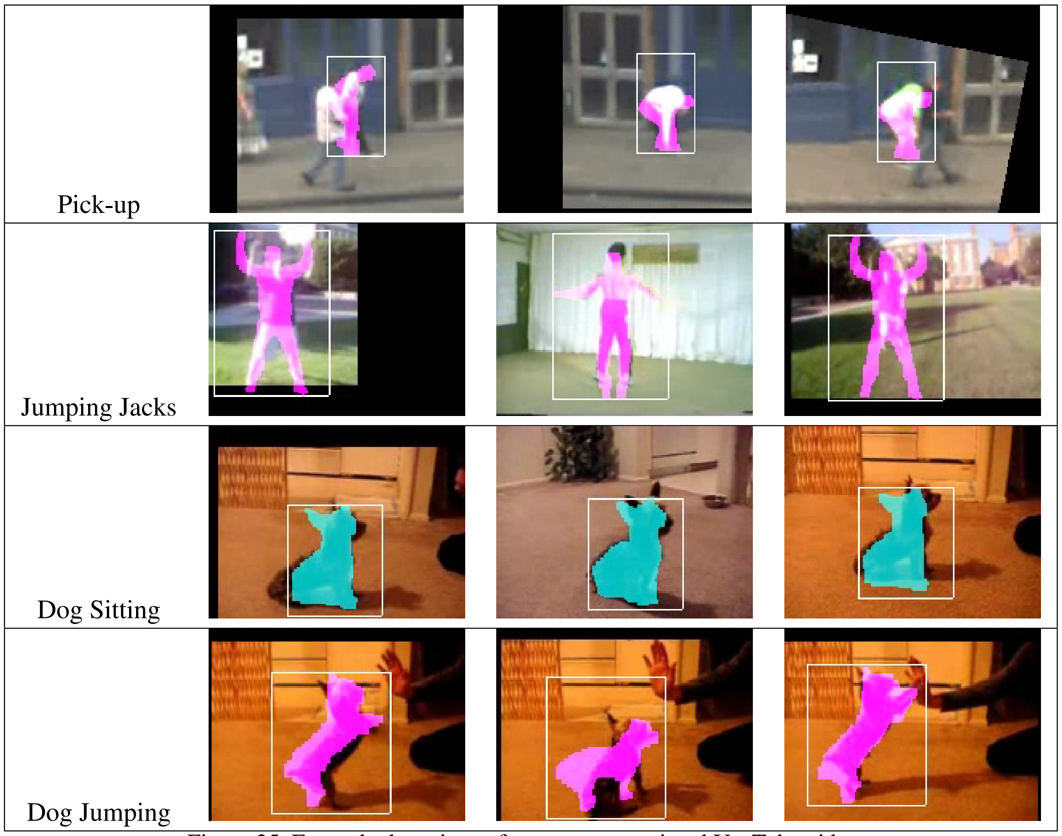


Figure 35. Example detections of events on unscripted YouTube videos.

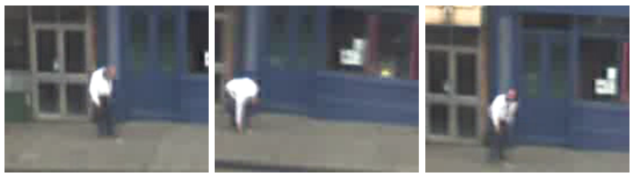
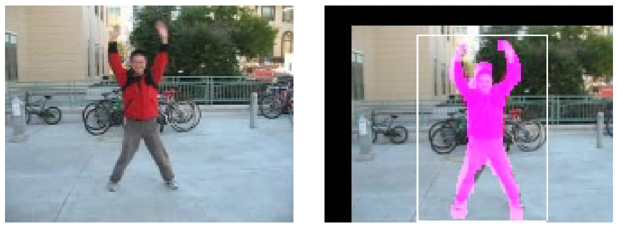


Figure 36. These frames show how much camera movement there is in a typical video sequence. The frames were extracted in a two second window.



(a) Video with Movement (b) Stabilized Video

Figure 37. Example detections from the Moving Camera dataset. The jumping jacks were not detectable in the original video (a). Using an off-the-shelf camera motion estimation algorithm [48], we were able to easily detected events in the stabilized sequence (b).

Figure 38. Example video of a camera panning like that of a surveillance camera. This type of motion can be easily stabilized and we are able to detect the events in these videos.

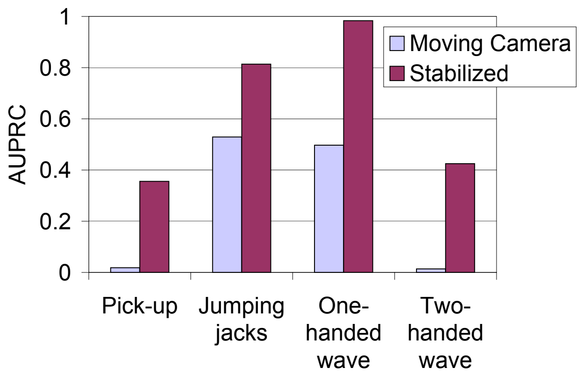


Figure 39. Detection results on the Moving Camera dataset. Using camera stabilization dramatically increases the recognition rate.

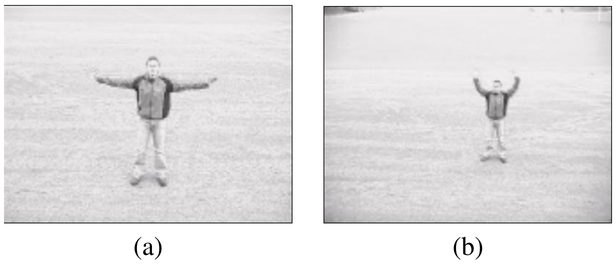


Figure 40. Notice the difference in scale between some videos in the KTH dataset. The contrast is also low making segmentation difficult.

	walk	jog	run	box	clap	wave
walk	.88	.08	.04	.00	.00	.00
jog	.14	.67	.19	.00	.00	.00
run	.04	.15	.81	.00	.00	.00
box	.03	.00	.00	.84	.10	.03
clap	.00	.01	.00	.12	.80	.07
wave	.00	.00	.00	.06	.06	.88

Figure 41. KTH Actions confusion matrix. Walk, jog, and run actions are most easily confused. Box, clap, and wave actions are sometimes confused.

Table 3. Although our method is not specifically designed for whole-video classification, our results on the KTH actions dataset [53] are competitive with recent studies.

Related work	Accuracy
Our Method (shape + flow)	**80.9%**
Ke *et al.* [33]	63.0%
Schuldt *et al.* [53]	71.7%
Dollar *et al.* [21]	81.2%
Niebles *et al.* [47]	81.5%
Jiang *et al.* [31]	84.4%
Laptev *et al.* [37]	91.8%

are able to generalize the actions and increase the classification performance by training on more templates, but with diminishing returns. On this dataset, shape-based correlation performs better than the flow-based correlation, and performance improves slightly when we combine the two features.

7. Conclusion

We presented a method for using volumetric features for event detection in crowded videos. The video is treated as a spatio-temporal volume and events are detected using our

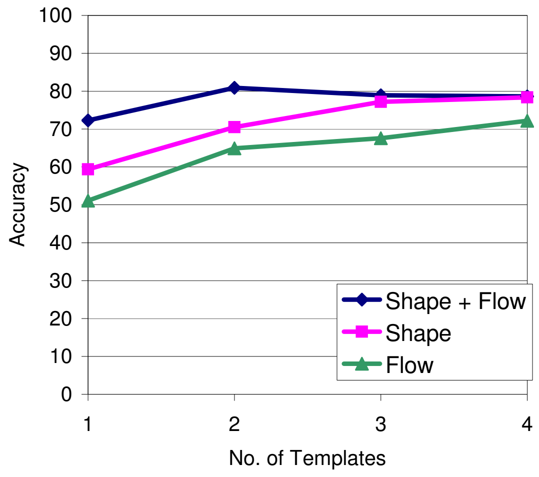


Figure 42. Results on the KTH actions database. Training on more templates improves results with diminishing returns. The combined shape and flow features perform better than either alone, especially with few training examples.

volumetric shape descriptor in combination with Shechtman and Irani's flow descriptor. Unlike existing shapebased methods, our system does not require figure/ground separation, and is thus more applicable to real-world settings. We extend our baseline shape matching algorithm to detect event parts (sliced in both space or time), and generalize the model to recognize actions with higher actor var ability. The parts are combined using pictorial structures to find the optimal configuration. Our approach detects events in difficult situations containing highly-cluttered dynamic backgrounds, and signficantly out-performs the baseline method [54]. This paper emphasizes the matching aspects of event detection and demonstrates robust performance on real-world videos. While the system is view dependent and is not scale invariant, we demonstrate robustness to camera view changes and some scale variations. We also demonstrate the feasibility of training and using multiple templates for greater generalizability.

There are several promising directions of future work. Although this work did not demonstrate real-time event detection, speed optimizations such as course-to-fine search could drastically improve the performance. Another interesting area of work is to automatically generate the event templates in a semi-supervised and ultimately completely unsupervised setup. For example, just by observation an action occur many times in a training sequence, one could automatically segment the event, align the training events spatially and temporally, and finally automatically generate the parts-based templates. In addition to shape and flow features, it would be interesting to explore texture and gradient features for shape matching. Further, better segmentation and more consistent super pixels would improve the shape matching part of this work.

References

[1] QuickFix Tight Abs Workout. Peter Pan Studios. ASIN: B00004Z73V.

[2] Wimbledon 2000 Semi-Final - Agassi vs. Rafter. SRO Sports Entertainment. ISBN: 0-7697-7886-0.

[3] YouTube, 2008. http://www.youtube.com/.

[4] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[5] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proceedings of International Symposium of Advances in Spatial Databases*, 1999.

[6] P. Arambel, J. Silver, J. Krant, M. Antone, and T. Strat. Multiple-hypothesis tracking of multiple ground targets from aerial video with dynamic sensor control. In *In Proceedings of SPIE 5429, (Signal Processing, Sensor Fusion, and Target Recognition XIII)*, 2004.

[7] J. A. Aslam, V. Pavlu, and E. Yilmaz. A geometric interpretation of R-precision and its correlation with average precision. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[8] W. Bell, P. Felzenszwalb, and D. Huttenlocher. Detection and long term tracking of moving objects in aerial video. Technical report, Cornell University, 1999.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(24), 2002.

[10] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Proc. ICCV*, 2005.

[11] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *PAMI*, 23(3), 2001.

[12] O. Boiman and M. Irani. Similarity by composition. In *NIPS*, 2006.

[13] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[14] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at www.csie.ntu.edu.tw/~cjlin/libsvm.

[15] Y. Cheng. Mean shift, mode seeking, and clustering. *PAMI*, 17(8), 1995.

[16] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5), 2002.

[17] T. Cour and J. Shi. Recognizing objects by piecing together the segmentation puzzle. In *Proc. CVPR*, 2007.

[18] C. M. Cyr and B. B. Kimia. 3D object recognition using shape similiarity-based aspect graph. In *Proc. ICCV*, 2001.

[19] D. DeMenthon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Statistical Methods in Video Processing Workshop*, 2002.

[20] D. DeMenthon and D. Doermann. Video retrieval of near-duplicates using k-nearest neighbor retrieval of spatio-temporal descriptors. *MTAP*, 30(3), 2006.

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

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

[23] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *PAMI*, 28(4), 2006.

[24] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005.

[25] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. on Computers*, 22(1), Jan. 1973.

[26] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 2003.

[27] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *PAMI*, 28(12), 2006.

[28] R. Hamid, S. Maddi, A. Johnson, A. Bobick, I. Essa, and C. Isbell. Discovery and characterization of activities from event-streams. In *Proc. UAI*, 2005.

[29] S. Hongeng and R. Nevatia. Multi-agent event recognition. In *Proc. ICCV*, 2001.

[30] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *Proc. ICCV*, 2007.

[31] H. Jiang, M. S. Drew, and Z.-N. Li. Successive convex matching for action detection. In *Proc. CVPR*, 2006.

[32] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, 2003.

[33] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Proc. ICCV*, 2005.

[34] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Proc. ICCV*, 2007.

[35] Y. Ke, R. Sukthankar, and M. Hebert. Spatio-temporal shape and flow correlation for action recognition. In *Workshop on Visual Surveillance*, 2007.

[36] I. Laptev and T. Lindeberg. Space-time interest points. In *Proc. ICCV*, 2003.

[37] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008.

[38] I. Laptev and P. Perez. Retrieving actions in movies. In *Proc. ICCV*, 2007.

[39] B. Leibe, K. Schindler, and L. V. Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Proc. ICCV*, 2007.

[40] Y. Leung, J.-S. Zhang, and Z.-B. Xu. Clustering by scale-space filtering. *PAMI*, 22, 2000.

[41] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *PAMI*, 29(2), 2007.

[42] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[43] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.

[44] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, 1997.

[45] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *PAMI*, 2001.

[46] G. Mori. Guiding model search using segmentation. In *Proc. ICCV*, 2005.

[47] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *Proc. BMVC*, 2006.

[48] J.-M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4), 1995.

[49] D. Ramanan and D. A. Forsyth. Automatic annotation of everyday movements. In *NIPS*, 2003.

[50] D. Ramanan, D. A. Forsyth, and K. Barnard. Building models of animals from video. *PAMI*, 28(8), 2006.

[51] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *PAMI*, 29(1), 2007.

[52] E. Sali and S. Ullman. Combining class-specific fragments for object classification. In *Proc. BMVC*, 1999.

[53] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *Proc. ICPR*, 2004.

[54] E. Shechtman and M. Irani. Space-time behavior based correlation. In *Proc. CVPR*, 2005.

[55] E. Shechtman and M. Irani. Matching local self-similarities across images and video. In *Proc. CVPR*, 2007.

[56] E. Shechtman and M. Irani. Space-time behavior based correlation -OR- How to tell if two underlying motion fields are similar without computing them? *PAMI*, 29(11), Nov. 2007.

[57] Y. Sheikh, M. Sheikh, and M. Shah. Exploring the space of a human action. In *Proc. ICCV*, 2005.

[58] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8), 2000.

[59] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, Oct. 2003.

[60] P. Srinivasan and J. Shi. Bottom-up recognition and parsing of the human body. In *Proc. CVPR*, 2007.

[61] P. B. Thomas Veit, Frederic Cao. Probabilistic parameter-free motion detection. In *Proc. CVPR*, June 2004.

[62] N. Vaswani, A. R. Chowdhury, and R. Chellappa. Activity recognition using the dynamics of the configuration of interacting objects. In *Proc. CVPR*, 2003.

[63] A. Veeraraghavan, R. Chellappa, and A. K. Roy-Chowdhury. The function space of an activity. In *Proc. CVPR*, June 2006.

[64] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2), 2004.

[65] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. In *ACM SIGGRAPH*, 2005.

[66] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *Proc. ECCV*, 2004.

[67] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3), 2003.

[68] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proc. ECCV*, 2000.

[69] D. Weinland, R. Ronfard, and E. Boyer. Automatic discovery of action taxonomies from multiple views. In *Proc. CVPR*, 2006.

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

[71] A. Yilmaz and M. Shah. Actions as objects: A novel action representation. In *Proc. CVPR*, 2005.

[72] G. Zhu, C. Xu, W. Gao, and Q. Huang. Action recognition in broadcast tennis video using optical flow and support vector machine. In *ECCV Workshop in HCI*, 2006.

[73] G. Zhu, C. Xu, Q. Huang, and W. Gao. Action recognition in broadcast tennis video. In *Proc. ICPR*, 2006.