VOLUMETRIC FEATURES FOR VIDEO EVENT DETECTION

Yan Ke

March 2008

CMU-CS-08-113

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Martial Hebert (co-chair)
Rahul Sukthankar (co-chair)
Tsuhan Chen
Alexei A. Efros
Larry Davis, University of Maryland

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

This research was sponsored by the National Science Foundation under contract no. IIS-0534962, National Science Foundation under contract no. EEC-0540865, DARPA under contract no. NBCH1030013, Intel Corporation, and National Science Foundation under contract no. DGE-0333420.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: event detection, volumetric features, action recognition, shape matching, video

ABSTRACT

THE amount of digital video has grown exponentially in recent years. We are at a nexus in time where video capture technology, computing power, storage capacity, and broadband networking have matured sufficiently to fuel an explosion in consumer videos. A key part of this ecosystem is the ability to search over vasts amounts of video data. While traditional methods have relied on text, such as those extracted from closed captioning, speech analysis, or manual annotation, we would to like search based on the automated recognition of the visual events in the video. This would enable more general searches to be performed without relying on previously labeled data. We propose a method for visual event detection of human actions that occur in crowded, dynamic environments. Crowded scenes pose 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 over-segmented 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.

ACKNOWLEDGEMENTS

I would like to thank my advisors Rahul Sukthankar and Martial Hebert for their guidance and support in developing this thesis. The knowledge and skills that I have learned from them will continue to shape me and I hope to pass their wisdom to future generations.

I also thank my thesis committee members Alyosha Efros, Tsuhan Chen, and Larry Davis not only for their advice on the thesis, but also as great role models on how to be successful in the field.

Throughout my graduate program, I have collaborated and subsequently became friends with many people at Carnegie Mellon, Intel Research Pittsburgh, and Microsoft Research Asia. I am proud to have been a part of these institutions. I would like to especially thank the researchers and interns at Intel Research Pittsburgh, whom have built a great collaborative research environment on the university campus.

I thank my wife Claire for her unconditional love and unending patience. Finally, I owe everything to my parents for placing me on this path in life. It is only through their dedication, hard work, and sacrifices, that I have come this far.

TABLE OF CONTENTS

CHAPTER 1

Introduction

DIGITAL video data has been growing rapidly in recent years. For example, as of 2003, there are over 2.4 million CCTV cameras in Britain [**96**]. Most digital cameras and many cell phones can record video, and consumers are recording and uploading their videos online, as evidenced by the popularity of video sharing sites such as YouTube. In August 2006, YouTube had over 6 million videos [**65**] and in January 2008, this has grown to over 60 million [**4**].[1] Unfortunately, the technology for making intelligent searches on video has failed to keep pace. Because of the large amount of data inherent in videos, it is extremely computationally intensive to do searches on them. Current production systems can only search videos based on text summarized from keywords and metadata. The keywords are typically extracted from closed captioning, speech analysis, manual annotation, and optical character recognition [**27**, **126**]. While text-based search is very efficient, annotating the videos is labor-intensive. Keyword searches are limited by existing annotations and thus one can not search for unlabeled events. Finally, manual annotation does not scale to large amounts of data and it is difficult to do in real time. Many research video retrieval systems that analyze visual information only process the key-frames in the video [**107**]. Ideally, we would like to automatically recognize and annotate all objects and events that occur in a video. For example, in a video of a football game, we would like to automatically recognize all of the players, their actions, and the overall strategy of the teams. While

[1] Typical videos are between 5 and 10 minutes long.

the overall goal of general video understanding is beyond the scope of this thesis, we propose fundamental building blocks that are instrumental towards this goal.

We propose the use of volumetric features for representing and searching over video. We make the key observation that objects in video span both space and time, and therefore three-dimensional spatio-temporal volumetric features are natural representations for them. The goal of this thesis is to propose efficient volumetric representations and to evaluate how well these representations perform in visual event detection. There are already many promising techniques in the literature for recognizing objects in static images. We borrow ideas from them while at the same time keeping in mind that many techniques when directly applied to video are often computationally intractable.

1. Thesis Contributions

This dissertation makes several contributions showing the usefulness of volumetric features for visual event detection.

- We use a discriminative classifier on volumetric features to build an efficient, real-time event detector.
- We propose a method for matching volumetric shapes in automatically segmented videos without requiring background subtraction.
- We use pictorial structures to match parts-based event models and combine shape and flow features.
- We study the robustness of our system to challenges such as camera movement, viewpoint changes, and other variations.

2. Visual Event Detection

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 **[79]** and Shechtman & Irani **[119]** have previously observed, the task is similar to object detection in many respects since the pattern can be located anywhere in the scene

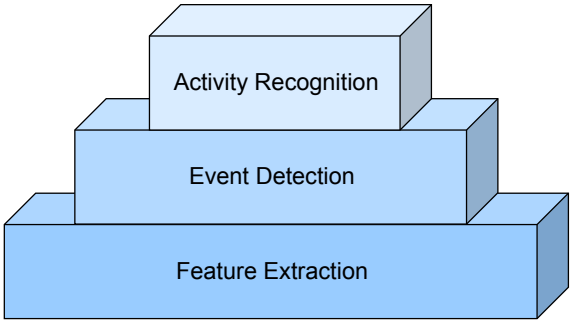


FIGURE 1.1. Event detection is one key component of the overall recognition pyramid.

(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 video classification, where the primary goal is to classify a short video sequence of an actor performing an unknown action into one of several classes [**16**, **117**, **152**]. Typical events that we wish to detect are around one second long. This is how long it takes someone to perform one distinct action, such as getting up from a chair, serving a tennis ball, or waving to someone. If the time scale is much smaller, *e.g.*, one tenth of a second, then there is insufficient movement in the video data. Longer time scales typically capture a series of events and thus should be decomposed into smaller atomic events. The eventual goal is to recognize a series of atomic events and group them together for generalized activity recognition. Recognizing events is one important part of the overall hierarchy in the recognition framework, as illustrated in Figure 1.1. This thesis focuses on exploring volumetric features for event detection.

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 1.2. In Figure 1.2(a), the person waving his hand to flag down a bus is partially occluded, and his arm motion occurs near pedestrians that 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 1.2(b), the goal is to detect the

(a) (b)

FIGURE 1.2. 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.

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.

Similar to many image object detection systems, we use a sliding window approach to event detection in video, as shown in Figure 1.3. First, we specify a model of the event that we are interested in detecting. The left part of Figure 1.3 shows an example *grab-cup* event. We scan this model across all locations in the video in space and time. When the classifier decides that we have a match, we label the event at that particular location and time, as shown by the white box in Figure 1.3. We will propose a number of features, models for representing events and classifiers. However, we will use the sliding window framework throughout the entire work.

Since we used a view-based approach to event detection, the system is sensitive to variations such as camera viewpoint, scale, speed, and differences in how actions are performed across people. Our baseline method is not invariant to these changes. However, we will address these issues and show how improvements to our baseline algorithm can cope with these variations. Using a parts-based model and training from multiple examples, we will demonstrate experimentally the robustness of our algorithm to these variations.

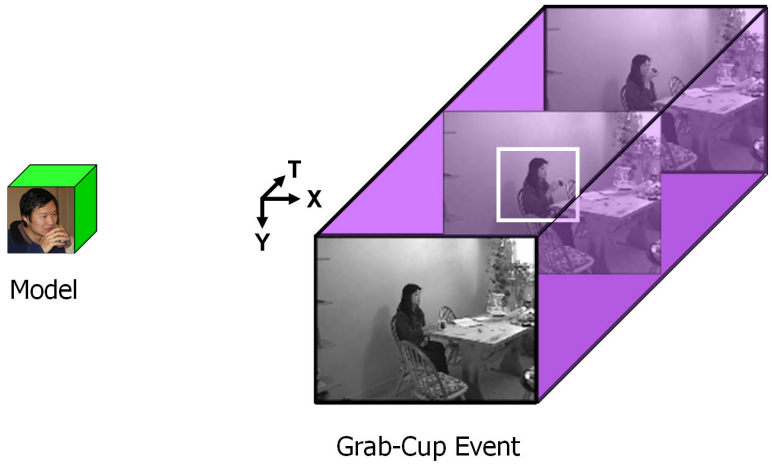


FIGURE 1.3. We use a sliding window approach to event detection. The model (left) is scanned at all spatio-temporal locations in the video (right). We are able to localize the event detection in both space and time.

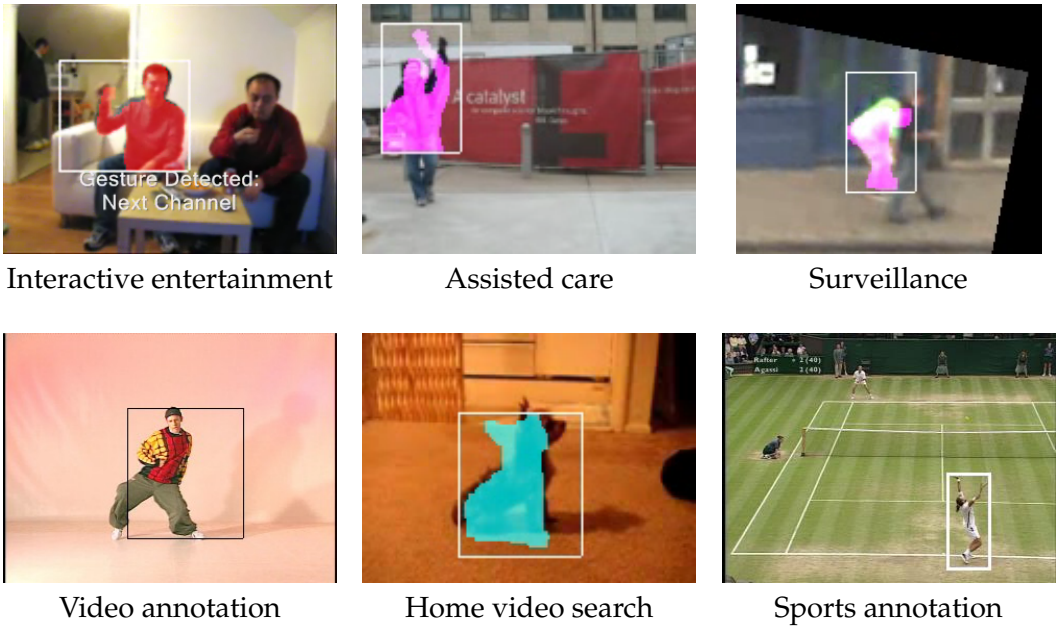


FIGURE 1.4. Example applications.

3. Applications

Visual event detection is a fundamental building block for many applications. If we can recognize all of the events that occur in the videos, then higher-level recognition can be performed using the atomic events. Table 1.1 summarizes some

TABLE 1.1. Example applications of event detection.

Application Domain	Example Events and Uses
Assisted Care	Falling down, waving for help.
Sports Annotation	Tennis serve, kicking a goal.
Sports Training	Determining whether an action is performed correctly.
Human-Computer Interaction	Gestures for controlling games or TV.
Video Search	Searching for actions in home videos, movie database, or online video sites.
Surveillance	Picking up and dropping off packages, opening doors.

of the application domains that can benefit from visual event detection and Figure 1.4 illustrates some examples. One example application is assisted care, where it would be useful to recognize elderly people falling down or having trouble getting up from a chair. An announcer would benefit from automatic recognition of actions that occur in a sports game. Users could also search for the most interesting parts of a game, for example when a goal is scored. If we compile statistics on the detected events, one could use it for sports training scenarios. We can recognize whether certain actions are performed correctly to improve athlete performance. Event detection can also be used in a gesture recognition system for human-computer interaction. The user can wave to the TV to change channels or to control a game. Finally, event detection can help in searching through large online video or movie databases. A user might want to search for certain events in their home videos or those uploaded by others on YouTube [**26**, **57**]. Figure 1.5 shows example event detections of a dog sitting and jumping. We are able to generalize to non-human actions because our models are appearance-based. Another obvious area is security and surveillance applications. While most of the work in this area have focused on long-term activity such as movement and location patterns, being able to recognize specific events such as picking up or dropping a package from the ground or opening doors and windows would be very useful. It is clear that event detection is useful in a wide range of applications.

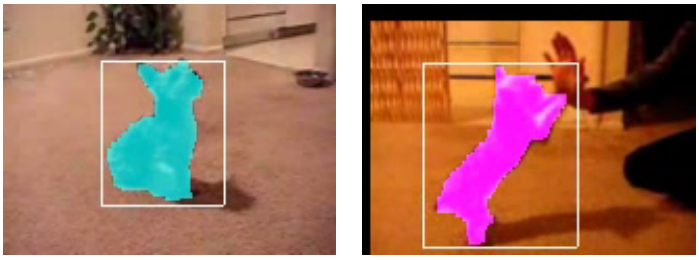


FIGURE 1.5. Example detections of a dog sitting and jumping. Our models are appearance-based, and therefore can generalize to non-human events.

4. Volumetric Features vs. 2D Features

Many features and matching algorithms have been developed for object recognition [**14**,**77**,**135**], and more recently object category recognition in static images [**48**, **52**, **55**, **133**]. Some of those techniques, such as interest points [**91**], have been extended to their 3D analogues and are used activity recognition in videos [**84**, **117**]. For the task of event detection, there are two general features which are both discriminative and robust to variations – shape and flow. Both of them capture how people deform and move through space-time, thus enabling us to recognize their action. We believe that one particular feature that is popular in the image analysis literature, patches of texture, is less relevant to this task because of its high variability. People could wear clothing with many different textures and could also appear in various lighting conditions. The shape (of a person's silhouette) and flow are both invariant to these kinds of changes.

Traditional methods for processing video have focused on analyzing individual frames independently, or possibly adjacent frames such as optical flow [**92**]. Features are typically extracted for each frame independently, and then subsequently linked together temporally [**127**]. 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 1.6. Doing analysis jointly in space-time offers three advantages. First, the region boundaries are stable. Second, the regions are automatically connected and tracked 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

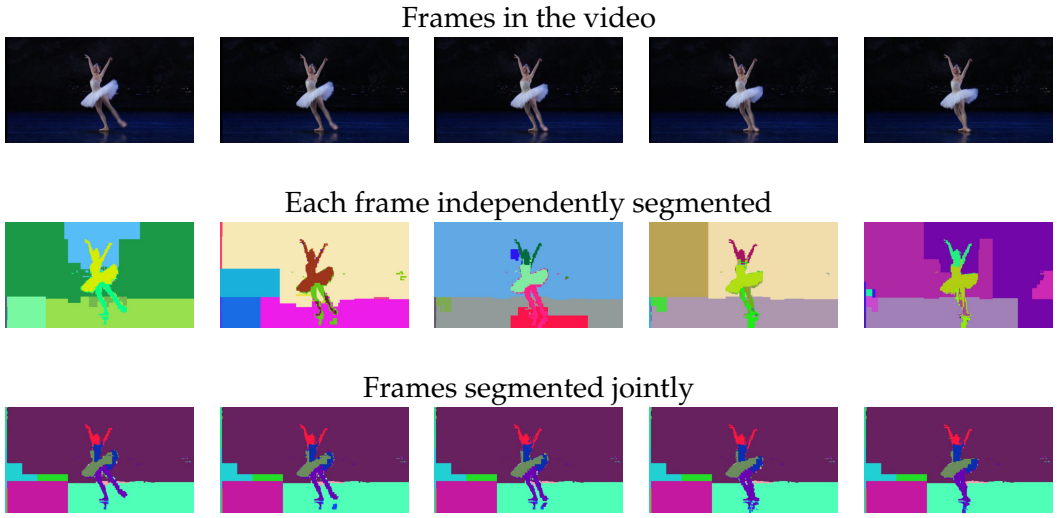


FIGURE 1.6. Independently segmenting each frame in the video yields varying and inconsistent segmentations across frames (middle). Jointly segmenting several frames across space-time yields consistent segmentations across adjacent frames (bottom).

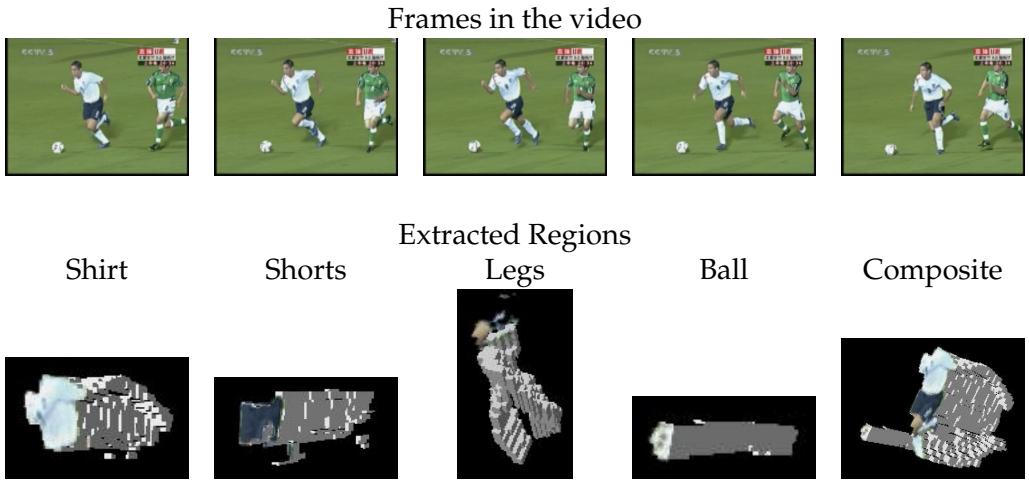


FIGURE 1.7. Just as static images can be decomposed into 2D shapes, video decomposed and represented as 3D volumes. A five-frame sequence (above) is segmented (based on color), into the volumetric parts (below). The parts are pieced back together and a composite is shown on the lower right.

to be done. Finally, region growth and death are accounted for automatically. This would be difficult to analyze if the segmentation were to be done independently for each frame.

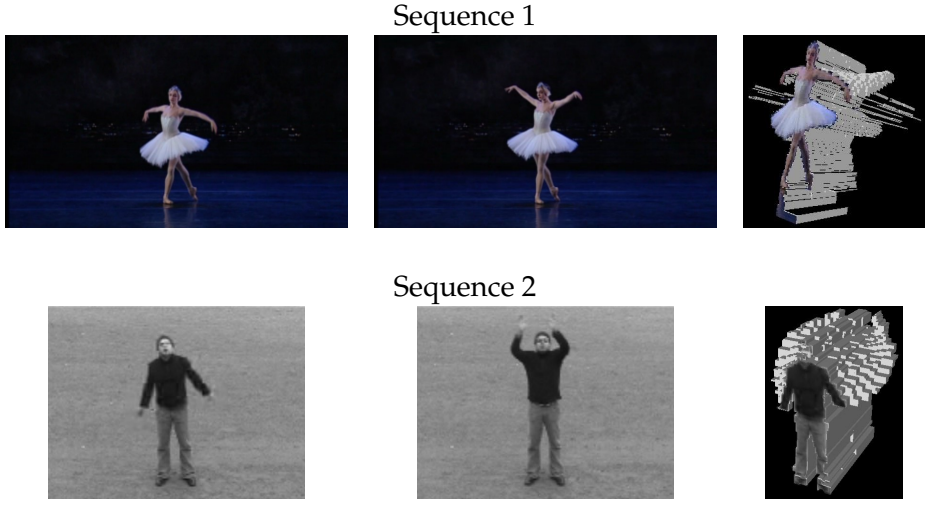


FIGURE 1.8. Objects and actions represented as 3D volumes. Once the volumes are extracted, object and action recognition is reduced to matching 3D shapes.

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 1.7. Only recently have researchers begun to process blocks of frames of video [**45**, **119**, **143**]. Just as previous work has decomposed images into their constituent shapes and used 2D shape descriptors for analysis [**14**, **66**, **125**], video can be thought as a group of 3D volumes and decomposed into 3D subregions. 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.

Once we can represent and match individual volumes, we can use them to recognize spatio-temporal events in video. Detecting events is reduced to matching 3D shapes, as shown in Figure 1.8. Given a single template of an action, we can match other similar instances of the same action in a video database. A problem with template matching is its low generalizability, in particular to different people performing the action or to different camera views. We will propose techniques

TABLE 1.2. Datasets used in our experiments.

Dataset	# actions	# actors	Length	Description
KTH [**117**]	6	25	$\tilde{2}$ hours	Periodic motion on static background. Standard dataset.
Weizmann [**16**]	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 [**5**]	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.

that can cope with these issues and we will measure quantitatively the system's robustness.

5. 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 [**117**] and the Weizmann dataset [**16**] 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 1.12. We also apply our method to standard action classification datasets for comparison purposes even though our algorithm is designed for event detection. Table 1.2 lists

FIGURE 1.9. Example actions from the KTH [**117**] dataset.

FIGURE 1.10. Example actions from the Weizmann [**16**] dataset.

all of the datasets we used and Figures 1.9, 1.10, 1.11 illustrates some of the actions in the various datasets. Additional figures of the other datasets are included with the results.

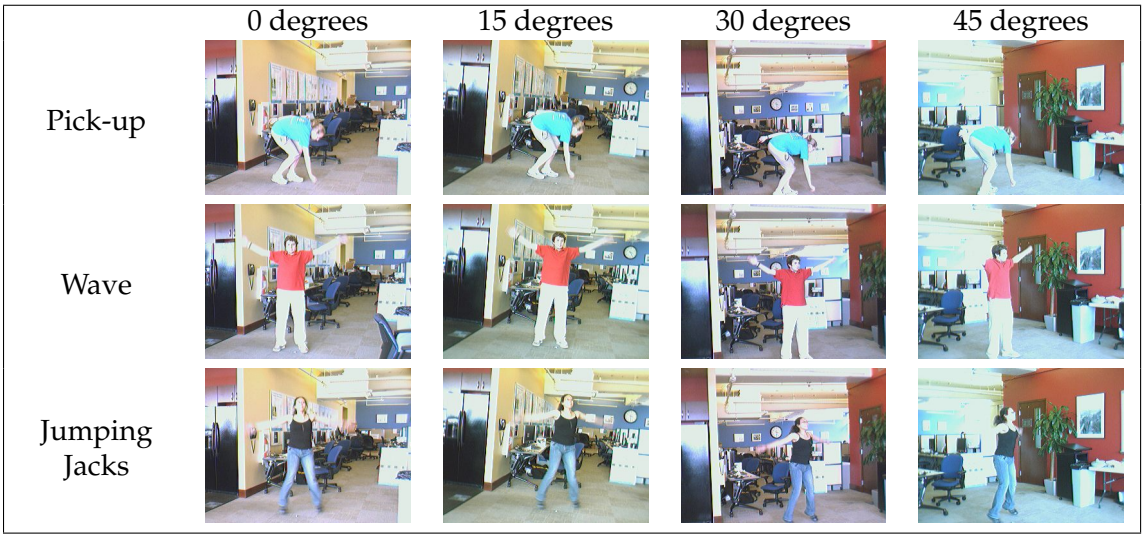


FIGURE 1.11. Multiview Dataset. Camera viewpoint change of up to 45 degrees.

FIGURE 1.12. 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.

CHAPTER 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 spatio-temporal shapes [**16**, **17**, **148**, **152**], flow [**49**, **79**, **119**], interest points [**46**, **103**, **117**], and tracking [**109**, **123**]. A more comprehensive review of historical work is presented by Aggarwal and Cai [**6**]. More recent work is surveyed by Wang *et al.* [**145**]. Our work is based on volumetric flow and shape matching and thus is most related to works by Shechtman, Blank, and Irani [**16**, **119**]. We first review the related work in each of these areas and then we describe in detail two baseline techniques that we use for comparison.

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 [**16**, **152**]. For computational efficiency and greater robustness to action variations, Bobick and Davis [**17**] project the spatio-temporal volume down to motion-history images, which Weinland *et al.* extend to motion-history volumes [**148**]. These techniques work best when the action of interest is performed in a setting that enables reliable segmentation [**139**, **147**]. 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 [**66**, **90**] and 3D [**8**, **8**, **41**, **59**, **78**]) 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. 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 [**49**]. 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 [**119**]. More recently, Zhu *et al.* uses an SVM classifier trained on histograms of optical flow to recognize tennis actions [**153**, **154**]. Jhuang *et al.* uses biologically-inspired features, which includes optical flow, for action recognition [**75**]. 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 [**140**]. Our work has recently been extended by Laptev and Perez [**85**].

3. Space-time Interest Points

Recently, space-time interest points [**84**] have become popular in the action recognition community [**46**, **103**, **117**], with many parallels to how traditional interest points [**91**] 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

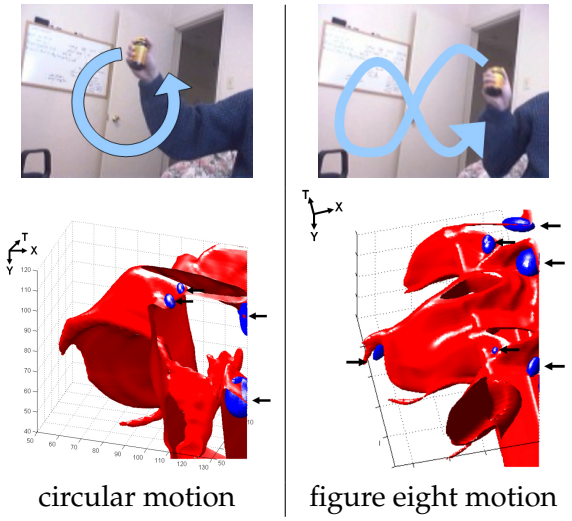


FIGURE 2.1. Two examples of smooth motions where no stable space-time interest points are detected. The 3D plots of motion through time were generated using software from [**84**]. The highlighted ellipsoids show the detected interest points. All of these detections are non-informative, caused by boundary 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.

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 2.1. Space-time interest points are also frequently triggered by the appearance of shadows and highlights in the video sequence, as shown in Figure 2.2. These unstable "events" are sensitive to lighting conditions and can reduce recognition accuracy for the action of interest.

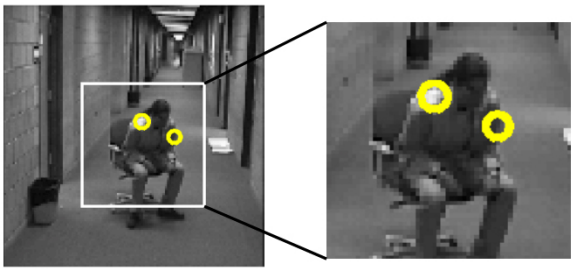


FIGURE 2.2. Space-time interest points are often found on highlights and shadows. These points are sensitive to lighting conditions and reduce recognition accuracy. This observation motivates our decision to apply volumetric features to the motion vectors rather than to the raw pixels.

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 [**110**], 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 [**9**, **68**, **72**, **86**, **137**]. 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.

5. Baseline Comparisons

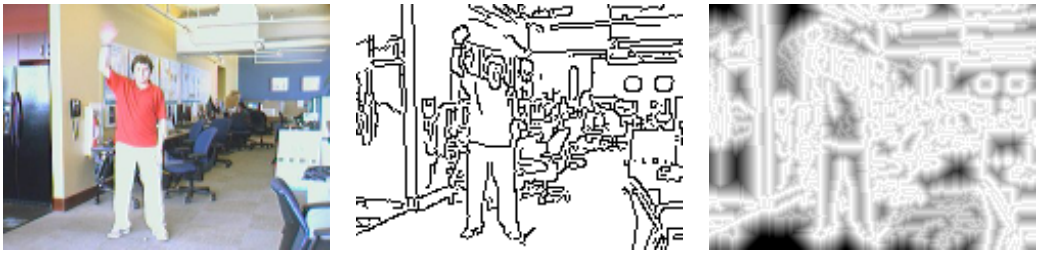
Our goal is to detect visual events in videos. While there are many proposed techniques for action classification in videos, many of them assume that the actor is well segmented from the background [**16**, **152**]. Therefore, they can not be used as a baseline for comparison in real world videos. The task is further constrained in that the algorithm must work with as few as one training example. Therefore, we selected one shape-based and one flow-based algorithm for comparison. We describe their basic matching approach and how we implemented them. The results of these baseline algorithms are discussed in Chapter 5.

5.1. 3D Chamfer Matching

Chamfer distance matching is a classic technique for recognizing shapes in images. It was first proposed for 2D shape matching by Barrow *et al.* [**12**]. It was later improved and generalized by Borgefors [**20**], Olson and Huttenlocher [**106**], and Gavrila [**61**], amongst others. It is still being used extensively for pedestrian detection [**62**, **87**, **118**] and general shape recognition [**131**]. Closely related to Chamfer distance transform is the Hausdorff distance metric [**116**]. In addition to 2D shape matching, it could also be used to match 3D shapes [**112**, **113**]. Pedestrian detection can be seen as a special case of human event detection, where people are walking in an upright position. Therefore, we compare our proposed event detection method to 3D Chamfer distance matching.

The distance transform is calculated by first running the Canny edge detector [**23**] on each frame in the video sequence.[1] We then calculate a 3D distance transform with time being the third dimension. Figure 2.3 illustrates the distance transform calculated on a video clip with a hand-wave event. In the transform image, dark areas represent larger distance to the original edge image. Notice the gradual fall-off of the distance transform. The soft boundaries give it more robustness to small deformations between the template and the target video.

[1] It would be interesting to experiment with other edge detectors as well [**47**, **93**, **99**, **129**].



Video Edges Distance Transform

FIGURE 2.3. Example of Chamfer distance transform on a hand-wave event.

5.2. Flow Consistency Matching

Shechtman and Irani proposed a flow-based method for matching actions in videos [**119**]. 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 S-I [**121**] for further details):

$$M = \begin{bmatrix} \sum P_x^2 & \sum P_x P_y & \sum P_x P_t \\ \sum P_y P_x & \sum P_y^2 & \sum P_y P_t \\ \sum P_t P_x & \sum P_t P_y & \sum P_t^2 \end{bmatrix}. \tag{2.1}$$

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

$$M^{\diamondsuit} = \begin{bmatrix} \sum P_x^2 & \sum P_x P_y \\ \sum P_y P_x & \sum P_y^2 \end{bmatrix}. \tag{2.2}$$

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 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

$$\Delta r = \text{rank}(M) - \text{rank}(M^{\diamondsuit}), \tag{2.3}$$

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

$$\Delta\tilde{r} = \frac{\lambda_2 \cdot \lambda_3}{\lambda_1^{\diamondsuit} \cdot \lambda_2^{\diamondsuit}} = \frac{\det(M)}{\det(M^{\diamondsuit}) \cdot \lambda_1} \tag{2.4}$$

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

$$\Delta\hat{r} = \frac{\det(M)}{\det(M^{\diamondsuit}) \cdot \|M\|_F}, \tag{2.5}$$

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

$$m_{12} = \frac{\Delta r_{12}}{\min(\Delta r_1, \Delta r_2) + \epsilon}. \tag{2.6}$$

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

$$d_F(T, V; l) = \frac{\sum_{i \in T, j \in (T \cap V)} m_{ij}}{|T|}. \tag{2.7}$$

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. The only optimization we use is that 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.

CHAPTER 3

Efficient Event Detection using Volumetric Features

1. Introduction

A fundamental limitation of video analysis is the large amount of data that must be processed. Many sophisticated algorithms have been developed for analyzing images and while spending a second or two per image is perfectly acceptable, the same algorithm applied to video analysis would be prohibitively slow.[1] For videos captured at thirty frames a second, this would be two orders of magnitude slower than real time. Recently, Shechtman and Irani proposed an action recognition technique based on flow correlation (described in Chapter 2 Section 5.2) [**119**]. It demonstrated that correlating flow-based volumetric features is viable, but their algorithm is computationally expensive. We propose a discriminative framework for efficient visual detection using volumetric features that runs in real time. We will propose an extended shape-based framework in Chapter 4 and discuss robustness issues in detail in Chapters 5 and 6.

Our work is motivated by Viola and Jones' highly successful face detection framework [**140**]. It is very efficient and it could process images at frame rate. We borrow two main techniques they used for efficiency – simple features which are fast to compute and a decision hierarchy for early discard of negative samples. We propose a novel appearance-based framework that employs volumetric features

[1] Some algorithms, such as mean shift used for image segmentation, could take tens of seconds per image [**30**].

FIGURE 3.1. An example of our detector recognizing the *grab-cup* action. Note that the detection volume (shown highlighted) is localized both in space and time.

for efficiently analyzing video. Applying this framework to the video's optical flow, we learn activity detectors that perform event detection in real time. The framework extends the rectangle features used by Viola and Jones [**140**] into the spatio-temporal domain for video analysis. Unlike their follow-up work in pedestrian detection in video [**141**], which uses only two adjacent frames, our *volumetric features* span longer time frames. To maintain computational efficiency and build a real-time detector, we generalize the notion of integral images [**140**] to an efficient space-time representation that we term *integral videos*. These allow us to perform event detection on video sequences in real time, as shown in Figure 3.1.

Although our system is primarily designed to detect motion events, we have also extended it to the action classification task, where each video sequence contains several repetitions of a single action. Schuldt *et al.* [**117**] propose a technique that uses local space-time features to classify six human actions (walk, jog, run, wave, clap, and box) in challenging real-world video sequences. They argue that global image measurements based on optical flow are unstable when there are multiple moving objects in the scene, or when the camera is not stationary. We show on the same dataset used in their work that our technique achieves comparable performance in the presence of camera motion, scale variation, and viewpoint changes. We argue that the same problems that hinder the use of 2D local descriptors for

object detection in static images also impact spatio-temporal local descriptors. For example, changes in object or background appearance will affect the location of the peaks found in spatio-temporal gradients. The trade-off between the stability of the interest points and the number of points found also exists in this domain. Because our technique uses dense optical flow measurements, our classifier is not limited to the sparse information found at peaks in spatio-temporal gradients nor affected by the instability of those peaks.

2. Volumetric Features

Given a video sequence, our goal is to classify whether an action event occurs in any of the space-time volumes in the video sequence. This is similar to running a classifier on all of the sub-windows in a 2D image for object detection. We now describe our framework for efficient computation of volumetric features. We will then run a discriminative classifier on these features, as we will describe in detail in Section 4.

Our framework for extracting volumetric features is general enough that it can be computed over many types of low level features, for example raw pixel intensities, spatio-temporal gradients, or optical flow. This framework has been successfully employed and extended by Laptev and Perez for action recognition using their own flow- and gradient-based volumetric features [**85**]. For our task of event detection, we believe that most of the salient information can be captured from the optical flow, and that the appearance of the object is less relevant. Initial experiments using pixel intensities performed poorly, mainly due to changes in appearance of the actor, the background, and lighting conditions.

This motivates our decision to compute our volumetric features on the video's optical flow. First, for each frame we compute its dense optical flow using OpenCV's implementation of Lucas-Kanade [**3**, **92**]. We separate the flow into its horizontal and vertical components and compute volumetric features on each component.

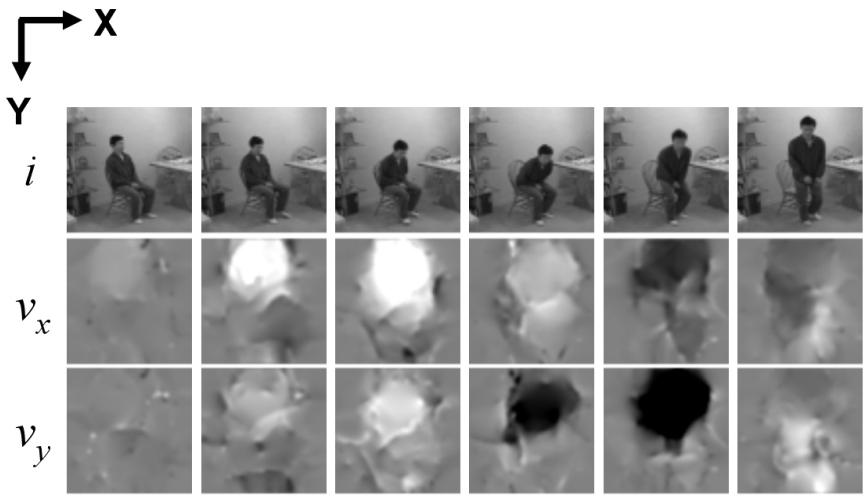


FIGURE 3.2. An example of the *stand-up* action and its optical flow. We separate the optical flow into the horizontal (v_x) and vertical (v_y) components. Lighter areas represent flow in the positive direction, while darker areas represent flow in the other direction.

Let $v_x(x, y, t)$ and $v_y(x, y, t)$ denote the horizontal and vertical optical flow components, respectively, at pixel location (x, y) and time t. Figure 3.2 shows an example *stand-up* action and its corresponding horizontal and vertical flow components. Even though the flow is very noisy and blurry (with imprecise boundaries), it has a distinctive pattern that is repeatable across many *stand-up* actions. Our discriminative classifier, discussed in Section 4, can learn to recognize these patterns and separate them from other motions.

As shown in the first row of Figure 3.3, our volumetric features consist of one-box or two-box volumes. The value of the one-box feature is simply the sum of the pixels within the volume (v_x or v_y). Correspondingly, the value of a two-box feature is the difference of their individual sums. Just as in 2D object detection [**140**], where one computes multiple rectangle features in a sub-window of the image, we compute multiple one-box and two-box features in a detection sub-volume, as illustrated in the second row of Figure 3.3. The horizontal and vertical size of the volume must be varied to match the size of the object being detected. The temporal span of the detection volume is both application and motion dependent. It is not necessary for the time duration of the volume to exactly match that of the motion to be recognized. Provided that the detection volume encompasses the motion, we

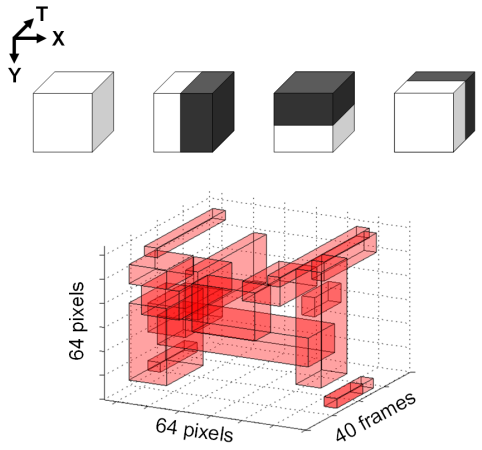


FIGURE 3.3. The top row illustrates the 3D volumetric features used in our classifiers. The first feature calculates the volume. The other three features calculate volumetric differences in X, Y, and time. The bottom row shows multiple features learned by the classifier to recognize the hand-wave action in a detection volume.

can achieve satisfactory recognition even if the action is performed at moderately different speeds (see Figure 3.8). If the training sequences are all aligned correctly at the start of the motion, the training algorithm described in Section 3 will learn that the tail ends of the sequences are noisy and thus less discriminative.

In our experiments, the features are computed over a volume of 64×64 pixels by 40 frames in time. A time-span of 40 frames is sufficient because we are interested in detecting short-term events. An exhaustive enumeration of all possible one-box and two-box features over this volume would number in the billions. Therefore one must sub-sample the feature space to reduce the set to a reasonable size for learning. We discretize the space to sample approximately one million features in our experiments, which is a large, but manageable set. The smallest feature occupies a 4×4 pixel by 4 frame spatio-temporal volume.

Summing every pixel for every learned feature would be expensive to compute. Viola and Jones used an integral image to reduce these sums to a few table-lookups [**140**]. Similarly, we use an "integral video" data structure to efficiently calculate the box features described above. This is a direct spatio-temporal generalization of the integral image. An integral video at pixel location (x, y) and time t is defined as the sum of all pixels at locations less than or equal to (x, y, t). More

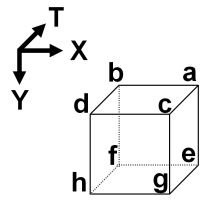


FIGURE 3.4. The volume of this box can be computed from the integral video with eight array references: $e - a - f - g + b + c + h - d$.

formally, the integral video iv, is defined as

$$iv(x, y, t) = \sum_{x' \leq x} \sum_{y' \leq y} \sum_{t' \leq t} i(x', y', t'),$$

where $i(x, y, t)$ is the pixel value at the original image. In our framework, we compute two integral volumes for the two optical flow components, and use $v_x(x, y, t)$ and $v_y(x, y, t)$ in place of $i(x, y, t)$. iv can be easily computed using the following recurrences:

$$\begin{aligned} s_1(x, y, t) &= s_1(x, y - 1, t) + i(x, y, t) \\ s_2(x, y, t) &= s_2(x - 1, y, t) + s_1(x, y, t) \\ iv(x, y, t) &= iv(x, y, t - 1) + s_2(x, y, t), \end{aligned}$$

where $s_1(x, -1, t) = s_2(-1, y, t) = iv(x, y, -1) \equiv 0$. Clearly, computing the integral video structure is only marginally more expensive than computing a series of integral images in a video sequence. We only need to examine the last frame in an integral video to add a newly captured frame. To compute the one-box feature shown in Figure 3.4 at any scale, location, or time, only 8 array references to the integral video structure are needed. Long video sequences could accumulate large sums that could overflow. Therefore, one must reset the sum periodically to maintain numerical precision.

3. Learning the Classifier

Since our volumetric features can be computed efficiently, we employ a sliding-window approach over the video to detect actions. This is an example of a rare-event detection problem, and fits well in the framework of cascaded classifiers. We use the direct forward feature selection method of Wu *et al.* [**149**] to select a small subset and arrange them in a cascade for efficient detection. The algorithm

selects features and their associated thresholds (termed filters), to classify whether an event occurs in a particular detection volume. The filters are binary classifiers that vote on the classification of the volume. While we could have used Adaboost as did Viola and Jones, Wu *et al.*'s method enables us to train the classifier in a reasonable amount of time despite the much larger set of initial candidate features. We summarize our implementation below.

Given a set of positive examples P, negative examples N, and features F, we construct a cascaded classifier that achieves high detection rate on the positive examples and low detection rate on the negative examples. For each node in the cascade (to be learned), we randomly choose a set of negative examples $N' \subseteq N$, that have been misclassified by the previous stages, where $|N'| = |P|$. Based on P and N', we select the optimal threshold θ_i for each feature $f_i \in F$ that minimizes the classification error rate independent of the other features. The optimal threshold for each feature can be found in $O(|P|\log(|P|))$ time by sorting the feature value of the positive and negative examples. We then iteratively add filters to the ensemble to maximize its detection rate or minimize its false positive rate. The decision output of the ensemble is simply the majority vote of the individual filters. The stopping criteria for a node in the cascade is when its target detection rate (100%) and false positive rate (less than 20%) are reached, or when it is not possible to improve its performance by adding more filters. Once the stopping criteria is reached, we eliminate the negative examples that were correctly classified and train the next node in the cascade using the remaining examples.

While training the cascade, it is critical to have a representative sampling of the space of negative samples. First, it is very difficult to pre-compute all of the negative examples. Because the cascade must have a very low false positive rate, we quickly exhaust all of the pre-computed negative examples after training only a few nodes. Second, this problem is compounded by our use of three dimensional features, where the feature space is much larger than the ones used for object detection on images. Sung and Poggio [**130**] propose a bootstrapping method which we use to generate more negative examples after training each node. As we train the new nodes in the cascade, we first run the already-trained cascade on video

sequences and extract all of the false detections. We use these detections as negative examples to train the new node, guaranteeing a sufficient supply of negative examples.

4. Detection

To detect events in a video sequence, we first compute the integral video of the sequence. We then scan a detection volume over all locations in space and time. To make the detector work at different scales, we vary the width and height of the detection volume and the associated filters. This is analogous to varying the size of the detection window in 2D object detection on images. Our smallest window size is 64×64 pixels. We slide the position of the window in increments of $1/16$ of the window size, and we increase the window size in increments of 1.25 in scale. A complication is that different people can perform the same actions at different speeds. In principle, this would indicate that we should also vary the time scale of the detection window. However, we chose instead to train the detector on actions performed at varying speeds and to detect actions using a fixed time scale window. Section 5.2 confirms that this approach works well in practice.

As with all similar scanning-window type detection systems, there will be multiple detections at adjacent locations, scales, and time around each "true" event. One might argue that this is a drawback of such systems, but we use it to our advantage to increase detection precision. Since we employ a purely discriminative technique (binary classifier), there is no explicit notion of detection "strength". However, the number of detections found in a small area in space-time *is* indicative of the quality of the detections. In other words, a dense cluster of detections is likely to indicate a true detection, while isolated detections are more likely to be false positives. We model this formally as follows, similar to Rosenberg's work in object detection [**115**].

Suppose that our goal is to detect the event E at space-time location L. Let D_i indicate that the detector also detects this event at some nearby space-time location L_i relative to L. Intuitively, the more D_i's that are true, the more likely that the

event E occurred at L. Given the set of D_i's, we wish to find the likelihood ratio between the event occurring and the event not occurring at L, and threshold at some value:

$$\frac{P(E_L|\{D_i \ldots D_n\})}{P(\neg E_L|\{D_i \ldots D_n\})} > T_1.$$

Using the naive Bayes assumption, this is simply

$$\frac{P(D_1|E_L) \ldots P(D_n|E_L)}{P(D_1|\neg E_L) \ldots P(D_n|\neg E_L)} > T_2.$$

Assuming that $P(D_n|\neg E_L)$ has identical uniform distribution, this becomes

$$P(D_1|E_L) \ldots P(D_n|E_L) > T_3$$

$$\sum_{i=1 \ldots n} \log P(D_i|E_L) > T_4,$$

which is equivalent to convolving the detections with a kernel that is dependent on the distribution of D_i's, and thresholding the output. We model this distribution as a 3D Gaussian kernel with diagonal covariance. We find the peaks of all clusters that are greater than the threshold and report them as events.

5. Evaluation

The first set of experiments examine our system's ability to detect and recognize non-periodic events in a long video sequence. Our detector is trained and tested on real videos with the *sit-down*, *stand-up*, *close-laptop*, and *grab-cup* actions. We discuss our system's robustness to different camera views, scale variations, and changing speeds at which actions are performed. We also modify our action detector to do action classification to evaluate our system on existing public data sets. We compute optical flow vectors using two off-the-shelf algorithms (Gautama *et al.* [**60**] and standard Lucas-Kanade [**92**]) and achieve similar results.

5.1. Action Detection

We train our system to detect the following four non-periodic actions: *sit-down*, *stand-up*, *close-laptop*, and *grab-cup*. The training set consists of four people performing a total of 60 repetitions of these actions. The test set consists of a different group

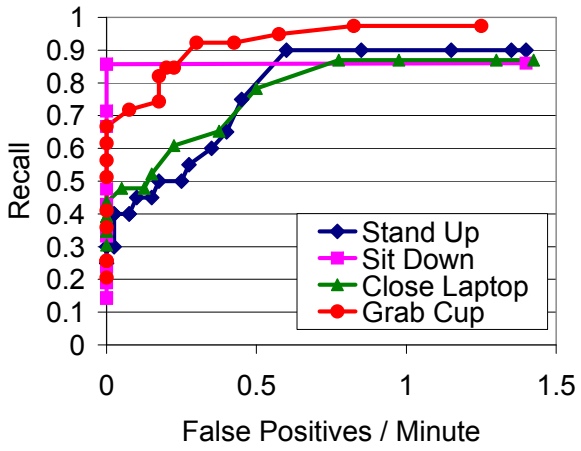


FIGURE 3.5. ROC curves for our event detectors. Note that the sit detector achieved 86% detection rate with no false positives.

of five people performing 20 repetitions. The ROC curve (see Figure 3.5) is generated by varying the threshold in the likelihood ratio test. For each action detector, we measure its false positive rate by running it on 40 minutes of video with moderate amounts of movement where none of these actions occur. The *sit-down* detector performs extremely well; it detects 86% of the events with no false positives. The *stand-up* detector achieves 90% recall with 0.6 false positives per minute. The *close-laptop* detector achieves 78% recall at 0.5 false positives per minute. Finally, the *grab-cup* detector achieves 92% recall at only 0.3 false positives per minute. Using Lucas-Kanade for computing the optical flow, our system runs at thirty frames a second on 160×120 pixel size videos.

5.2. Analysis

Our detector is trained on the optical flow of the video, and as with all methods that operate on low level features (as opposed to recovered 3D pose), its performance is dependent on factors such as variations in camera view, the objects' scale, and the speed at which the actions are performed. First, we investigate how changes in camera view affect our system. The effects are motion dependent, where horizontal moving actions are more likely to be distorted than vertical motions as we pan the camera around the room. We give anecdotal evidence that our system is robust to modest changes in camera view, roughly within 45 degrees. Our data

Chair sequence

Laptop sequence

FIGURE 3.6. Our system is trained and tested in different environments and camera views.

is gathered in uncontrolled office environments with no precise placement of cameras. Figure 3.6 shows two typical frames in the *sit* and *stand* sequences, where the two people face slightly different directions and sit on different chairs.

As described in Section 4, we make our detector scale invariant by adjusting the size of the detection volume. Figure 3.7 shows the amount of scale variations in the data, where the variations between training and testing objects are 1-2$\times$ for our data and 1-3$\times$ for the KTH dataset. Part of this scale variation is caused by the fact that we train the detector on 64$\times$64 pixel size videos and test on 160$\times$120 pixel size videos, and part of the variation is caused by the differences in distance of the object to the camera as well as camera zoom. The results show that our system is robust to the scale changes.

Because people perform actions at different speeds, our detector must also be robust to such variations. Figure 3.8 shows how three people perform the hand-clapping action at different speeds over 40 frame sequences. The sequences are shown to be aligned at the start of the motion, which we define to be when the hands are closed. Note that the actions diverge at the end of the sequences, where one person's arms are closed while another's arms are open. Our classifier automatically learns to ignore the noisy tail ends of the sequences. This is shown at the

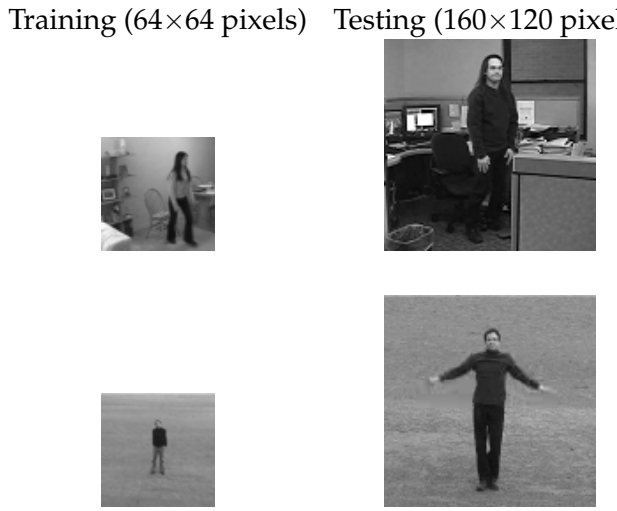


FIGURE 3.7. Examples of the amount of scale changes in our data (2×) and the KTH dataset (3×). We trained the system using 64×64 pixel size videos (left) and tested on 160×120 pixel size videos (right).

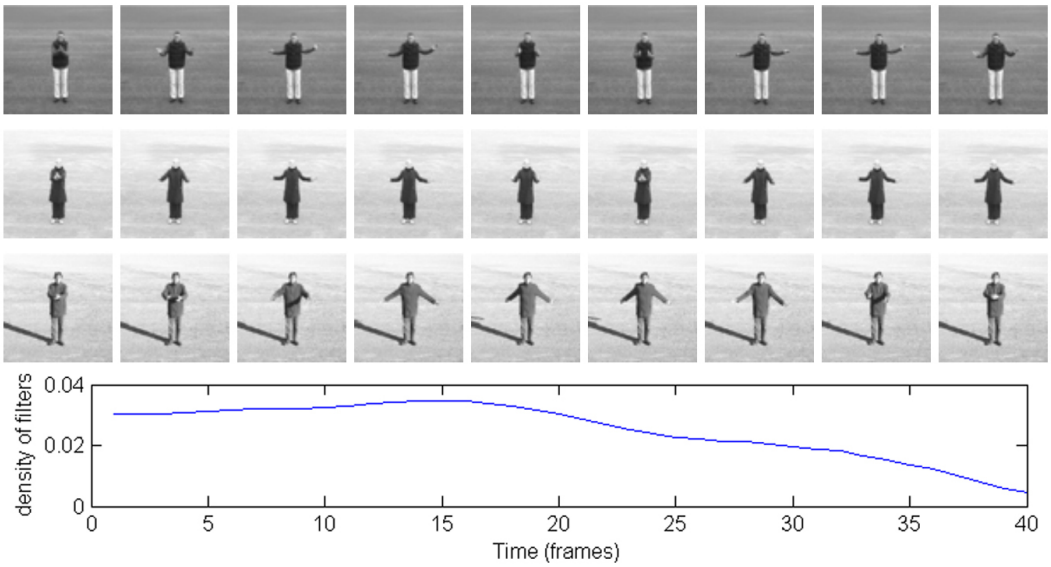


FIGURE 3.8. This figure shows three people performing the hand-clapping action at different speeds. Since the beginning of the sequences are aligned, our classifier learns that this region is more discriminative. The classifier selects more filters to classify the start of the sequences and fewer filters to classify the end of the sequences, where they diverge.

bottom of Figure 3.8, where the density of filters is greater near the beginning of the sequence.

FIGURE 3.9. One cycle of the hand-wave motion, from the KTH dataset.

5.3. Action Classification

The second set of experiments compares our method against Schuldt *et al.*'s method [**117**] in classifying periodic actions performed in short video sequences. We use the same training and test sequences as in their paper, which contains eight people in the training set and nine in the testing. Each person repeats six actions (walk, jog, run, box, clap, and wave) in each of four scenarios (outdoor, outdoor + camera zoom, outdoor + change of clothes, indoor). Each video clip, which contains one person performing one action in one scenario, is about twenty seconds long. Since we designed our detector to recognize aperiodic motion, we manually segment several instances of each action for training, all starting at the same phase to be positive examples. For example, we define a simple hand wave action to be the one shown in Figure 3.9. For each action, we choose random parts of the other action sequences in addition to the pre-recorded videos to generate the negative examples. Figure 3.10 shows examples of the first few filters chosen to classify the hand-wave and boxing motions. It is easy to see that the hand-wave classifier is quite symmetric, capturing motion on both sides of the body. On the other hand, the boxing filters are on one side and high, representing the punch thrown by the boxer. We classify the entire sequence by summing likelihood ratios that pass the threshold and select the action whose detector reported the highest sum.

Figure 3.11 shows the confusion matrix for these six actions for each technique. We have an average accuracy of 63% while Schuldt *et al.*'s is 72%. Our results are only slightly worse, which is encouraging since our system was trained to detect a single instance of each action within arbitrary sequences while Schuldt *et al.*'s system has the easier task of classifying each complete sequence (containing several repetitions of the same action) into one of six classes.

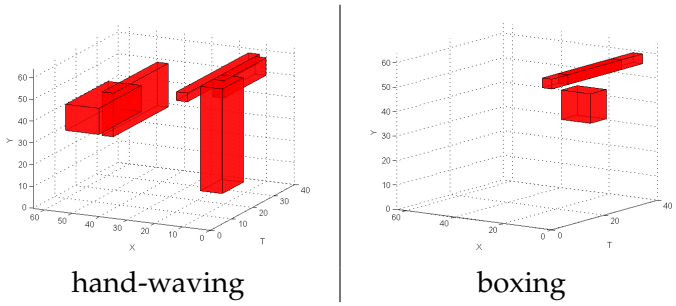


FIGURE 3.10. Examples of first few filters learned for the hand-waving and boxing actions.

.81	.11	.08	.00	.00	.00
.31	.36	.33	.00	.00	.00
.03	.25	.44	.00	.28	.00
.00	.03	.11	.69	.11	.06
.00	.00	.06	.36	.56	.03
.00	.06	.00	.03	.00	.92

(a) Our method

.84	.16	.00	.00	.00	.00
.23	.60	.17	.00	.00	.00
.06	.39	.55	.00	.00	.00
.01	.00	.00	.98	.01	.01
.01	.00	.00	.35	.60	.04
.01	.00	.00	.21	.05	.74

(b) Schuldt *et al.*

FIGURE 3.11. Confusion matrices for (a) our method and (b) Schuldt's LF + SVM method. We have an accuracy of 63% and they have an accuracy of 72%.

6. Limitations

While the proposed volumetric features are very efficient to compute, the framework also has some limitations. Because we use a discriminative classifier, we require many training examples. Typical algorithms for learning a face detector require thousands of training examples for good performance **[69]**. It would be very difficult to find this many examples to train an event detector. It is much more labor-intensive to scan and label events in video than in images. A person can quickly determine whether an image has a face and draw a rectangle around it to label it. To find a *sit-down* event, on the other hand, one might need to sift through hours of video to find a few dozen examples.

FIGURE 3.12. An example of an incorrect detection: the "clapping" motion shown in top row triggers the "boxing" detector. This is because the left-to-right motion in the shaded area (caused by the clap) is very similar to the motion generated by the extension of the arm in the boxing action shown in the bottom row. This illustrates the limitation of relying on motion features alone, as a model of appearance would easily be able to distinguish between the two actions.

In addition to the manual labor, it is also difficult to train an event detector due to computation time and memory constraints. We needed to sub-sample the number of candidate features in order to train the classifier. Our implementation of Wu *et al.*'s feature selection algorithm required one gigabyte of memory to train the classifier [**149**]. It would be difficult to learn from thousands of training examples even if we could label them manually.

There is a fundamental limitation of relying exclusively on motion information for event detection. It does not explicitly take into account the shape of the person. For example, Figure 3.12 shows an example where a "clapping" motion incorrectly triggers the "boxing" detector because the left-to-right motion in the shaded area is the same for both motions, even though they vary considerably in appearance. An appearance-based detector should be able to easily determine that the grassy area is not a person boxing. The challenge is to use the right kind of appearance information. We believe that appearance in the form of silhouette outlines are useful, while texture patches are not.

Another limitation of using only optical flow is that a cluttered and moving background could dominate the motion field. Figure 3.13 shows the an example hand-wave event and its corresponding optical flow. We see that the strongest flow occurs in the background due to the moving crowd. A detector that relies on optical flow alone is likely to fail in this scenario because the only motion caused by the person is in their arms, which occupy a relatively small area.

(a) Hand-waving event (b) Corresponding optical flow

FIGURE 3.13. The moving background dominates the optical flow in this video. Any detector that relies on flow information alone is likely to fail. The motion induced by arms moving occupies a relatively small area.

These limitations motivate us to explore alternative frameworks for event detection. In the rest of the thesis, we propose a framework for event detection using as few as one training example. The proposed algorithm incorporates both flow and shape information in a principled framework. We show good detection results in cluttered and dynamic environments.

7. Conclusion

We introduced a novel volumetric feature framework for analyzing video by extending Viola and Jones' work for static-scene object detection to the spatio-temporal domain. Applying this framework on the video's optical flow, we learn activity detectors that can recognize events in video. We have shown that computing integral videos and box features is only a small constant factor slower than that of a series of integral images, and therefore we achieve real time recognition performance. We have demonstrated good performance on detecting non-periodic motions with low false positive rates on long sequences of video. Despite camera movements and scale changes, our results on classifying the six actions are comparable to the system by Schuldt *et al.*

We have successfully shown that object detection in static scenes and activity recognition in video can be integrated into a common framework. One possible future direction is to add an appearance model to increase accuracy. At the very least, this will enable us to differentiate between the clapping and boxing actions

shown in Figure 3.12 by discriminating between pixels that correspond to a person rather than the background.

CHAPTER 4

Volumetric Shape Matching

1. Introduction

We now propose a method for event detection using shape-based features. This work is motivated by silhouette-based approaches to action recognition by characterizing the shape of the actor's silhouette through space-time [**16**,**17**,**95**,**152**]. Because shape-based techniques ignore texture and other appearance inside the silhouette, they are robust to variations in clothing and lighting. They show that the deformation of a person's silhouette over time is an important and distinctive feature for action recognition.

There are two major limitations with silhouette-based approaches. First, they assume that the silhouettes can be accurately delineated from the background. Second, they assume that the entire person is represented as one region. Therefore, such techniques typically require static cameras and a good background model.

Bend Jumping jacks Jump Two-handed wave

FIGURE 4.1. Example noisy silhouettes extracted using background subtraction from the Weizmann dataset.

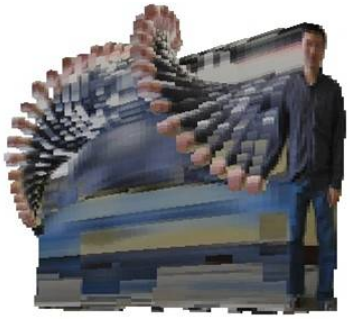


FIGURE 4.2. 3D shape of an example hand-wave action. Event detection is reduced to 3D shape matching of the volumetric shapes extracted from a video.

Unfortunately, even state-of-the-art background subtraction techniques generate holes when parts of the actor blend in with the background, or create protrusions on the silhouette when strong shadows are present. Examples of noisy silhouettes extracted from background subtraction are shown in Figure 4.1. These artifacts consequently reduce the accuracy of shape-based action recognition techniques.

We propose a simple yet effective shape-based representation for matching videos that does not require background subtraction, nor explicit background models. Event detection is reduced to 3D shape matching of the volumetric regions found in the video. Figure 4.2 illustrates the 3D shape of an example hand-wave action. Our shape-based matching consists of spatio-temporal region extraction and region matching. For region extraction, we employ an unsupervised clustering technique to segment the video into three-dimensional volumes that are internally consistent in appearance; we term these "supervoxels" since they are conceptually analogous to superpixels [**114**]. We observe that real object boundaries in spatio-temporal volumes typically fall on supervoxel borders, just as superpixel borders correspond to useful segmentation boundaries [**102**]. As with all bottom-up segmentation techniques, we do not expect the region extractor to segment the entire object as a single region, and thus we err on the side of over-segmentation. We propose a shape matching technique that works on over-segmented videos. This is similar in spirit to recent work in shape-guided figure-ground segmentation [**19**, **38**].

FIGURE 4.3. Example video and its resulting segmentation. The segmentation is volumetric (3D); we can only show its 2D projection.

To achieve the best performance, we should use flow information in addition to shape. Therefore, we combine our shape-based method with recent flow-based techniques and demonstrate improved recognition performance. We discuss the limitations of shape- and flow-based techniques for event detection and argue that their complementary nature allows them to mitigate each other's limitations. To show the benefits of the combined features, we incorporate Shechtman and Irani's flow-based features **[119]** into our classifier and demonstrate improved performance on a challenging event detection task and a standard video classification task.

The baseline technique proposed in this chapter can recognize events with only one training example. This is useful in scenarios where we can not quickly acquire many training examples. For example, if we see a suspicious event in a surveillance tape and we want to recognize other instances of that event, we could train using the instance that we saw. However, we recognize that having more training examples will improve performance and the algorithm's ability to generalize. We will explore these issues in Chapter 6. We now describe the details of our baseline shape matching algorithm.

2. Spatio-Temporal Region Extraction

We first automatically segment the video into 3D spatio-temporal volumes. An ideal region extractor would not only automatically segment individual objects in space, but it would also track their motion through time. Stable object segmentation is currently difficult for images [**94**] and video [**143**]. Because we want the region extraction to be general for many types of applications, we use mean shift [**29**, **34**] to cluster the video into regions. Instead of individually segmenting video frames and then linking the regions temporally (which causes unstable regions), we segment the three-dimensional spatio-temporal volume of pixels created by stacking a sequence of frames. The smallest processing unit is a voxel, taken from a 1×1 pixel from one 1 frame. The voxel location and color are used as features for mean shift. Our method works well despite having used simple features because it is not dependent on the precise segmentation of the object from the background. Figure 4.3 shows an example video frame and its segmentation. We show only a two dimensional projection although the original segmentation is in 3D. In general, this approach works with any segmentation algorithm. Future work could explore whether there is benefit from using more sophisticated algorithms.

2.1. Mean Shift

Standard mean shift is an iterative procedure that finds local modes in the data [**58**]. Researchers have successfully used it to segment images [**34**] and video [**45**, **143**]. Wang *et al.* uses anisotropic kernels to segment video, which produces excellent results for anisotropic shapes. However, their work only focuses on segmentation and the regions were not used for recognition. DeMenthon and Doermann's work is similar to ours in that they both segment the video and build a representation for the regions for recognition. However, the representation consists of only the cluster modes and does not take into account the shape of the cluster region. Therefore, it is optimized for full frame descriptions, and not for specific objects or event descriptions. There is no notion of an "object" in that work.

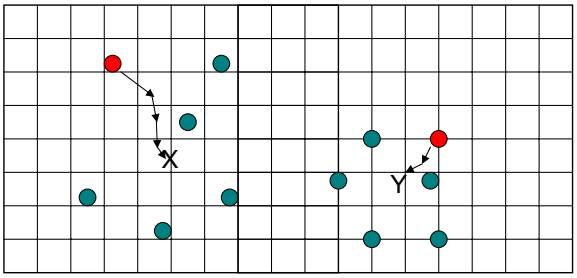


FIGURE 4.4. Mean shift moves the data points to the modes of the data density. This figure shows two points moving iteratively to their cluster centers (X and Y).

Let us describe mean shift in detail. Let f_i be a d-dimensional point in a set of n feature points, $f_1 \ldots f_n$. To cluster the points, we find the mode near f_i using mean shift. We calculate

$$y_{(i,1)} = f_i \tag{4.1}$$

$$y_{(i,j+1)} = \frac{\sum_{k=1}^{n} f_i g(||\frac{y_{(i,j)}-f_k}{h}||^2)}{\sum_{k=1}^{n} g(||\frac{y_{(i,j)}-f_k}{h}||^2)}, \tag{4.2}$$

where $j = 1, 2, \ldots$, with kernel g, typically a Gaussian kernel, and bandwidth h. The mode y_i is the limit of the series, where $y_{(i,j)}$ converges to a fixed point, as shown in Figure 4.4. As a post-processing step, all data points within some distance (typically h) of each other are grouped and labeled as one cluster. Note that this segmentation process only needs to be performed once on the video. The resulting segmentation can be used to match *any* template event. Therefore, the segmentation can be thought as a preprocessing step and it is not a limiting factor when searching for many events.

2.2. Efficient Region Clustering

One of the limitations with using mean shift to cluster data is that it is computationally intensive. At every iteration, it needs to find the nearest neighbors for every data point. Near neighbor search, especially in high dimensions, can be quite slow. Previous work have focused on reducing the cost of near neighbor search [**34, 63, 73**], or other methods [**151**]. Despite all of these efforts, a 640×480 image could take up to a minute to segment [**63**] and much longer to segment an

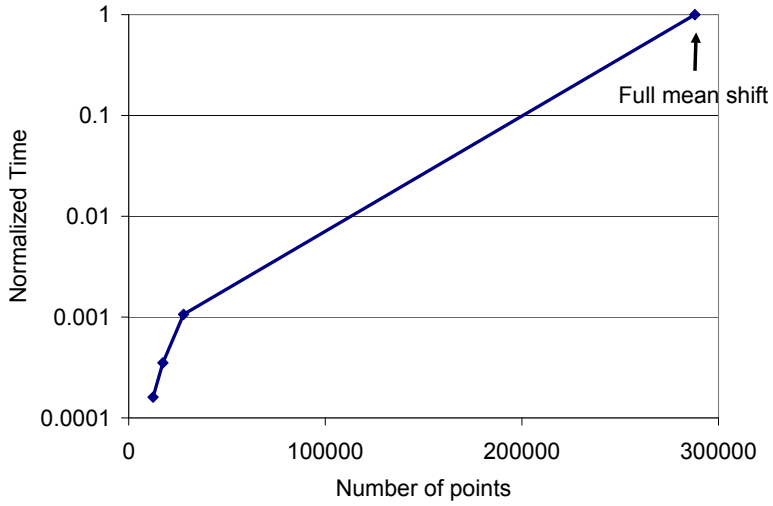


FIGURE 4.5. Effect of using an initial octree clustering prior to mean shift. Using larger (and thus fewer) octree clusters dramatically decreases the amount of time needed to for mean shift to segment the video without significantly decreasing the clustering quality. The amount of time that mean shift uses without the octree clustering is normalized to 1 (upper right). The experiments were done on a $160 \times 120 \times 15$ block of video.

entire video clip. Reducing the number of data points that needs to be clustered is another method that can reduce the cost of mean shift. A typical 320x240 by 15 frame video clip has over a million data points. It takes about 20 iterations for each point to converge. Even with an approximately $O(\log n)$ algorithm to search for nearest neighbors, it is still very expensive to do mean shift clustering. Therefore, we use octrees to perform an initial coarse clustering of voxels that are similar in color. Each group of voxels is then clustered again using mean shift to find more accurate object boundaries. This enables a factor of 100-1000 speedup in computation time without significant loss of quality. Figure 4.5 shows the amount of time it takes to cluster a block of frames, where the timing for the full mean shift computation is normalized to 1. An example octree clustering output is shown in Figure 4.6. It would be interesting to explore other optimization techniques to further decrease the running time [**24**] or try methods other than mean shift to do the volumetric clustering.

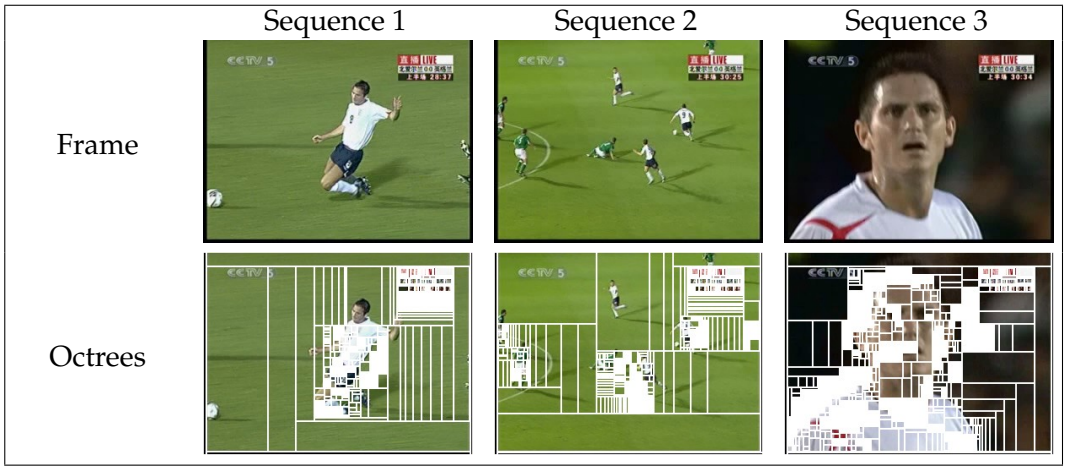


FIGURE 4.6. A 2D illustration of the octree clustering on the video. The actual clustering is done spatio-temporally. Notice that the octree divides regions with high detail, which also have high variance into smaller blocks. This enables us to adaptively adjust the block size to match the amount of detail in the scene. The size of the segments used in practice are much smaller than the ones shown in this figure.

2.3. Hierarchical Clustering

A critical parameter that must be chosen for mean shift, and nearly all clustering algorithms, is the kernel bandwidth size. Intuitively, the bandwidth size encodes the prior on the size of the objects that should be segmented. A small bandwidth will correctly segment small objects, but will over-segment large objects into multiple parts. Conversely, a large bandwidth will correctly segment large objects, but will incorrectly group small objects together. While there are proposed methods for adapting the kernel bandwidth **[33]** or automatically choosing a stable bandwidth based on scale-space theory **[45, 89]**, it is inherently impossible to choose the correct bandwidth for segmentation without higher-level semantic knowledge. Therefore, we perform hierarchical clustering using mean shift that segments the image into a pyramid of region sizes **[44]**. Because of the small scaling factor of the region sizes, this only increases the number of regions by a small constant factor, while enabling us to deal with arbitrarily-sized objects. Figure 4.7 shows hierarchical mean shift applied to a video sequence. As expected, larger regions are extracted with larger bandwidths. At run-time, we search over the hierarchy using the matching algorithm described below.

FIGURE 4.7. Hierarchical mean shift automatically finds differently-sized regions at various levels of the hierarchy. The bottom level, with the smallest bandwidth, finds the smallest regions. Larger regions are found at higher levels of the hierarchy.

3. Volumetric Shape Matching

We present a novel method for matching an action template to an over-segmented video which 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.

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 set difference between the union and the intersection of the regions, *i.e.*, $|A \cup B \setminus A \cap B|$.[1] We adapt this distance metric to work with over-segmented regions as

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

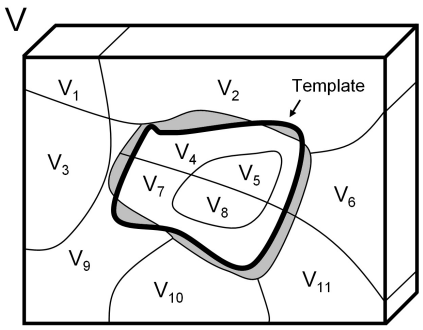


FIGURE 4.8. Example showing how a template is matched to an over-segmented volume using the Region Intersection method. The template is drawn in bold, and the distance (mismatch) is the area of the shaded region.

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.

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

$$\hat{S} = \bigcup_{i \in S} V_i, \tag{4.3}$$

and S is defined by

$$\underset{S \subset \{1,\ldots,k\}}{\operatorname{argmin}} \; |((\cup_{i \in S} V_i) \cup T) \setminus ((\cup_{i \in S} V_i) \cap T)|. \tag{4.4}$$

In Figure 4.8, we have drawn the template T in bold and overlaid it 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

$$d(T, V; l) = \sum_{i=1}^{k} d(T, V_i; l), \tag{4.5}$$

where

$$d(T, V_i; l) = \begin{cases} |T \cap V_i| & \text{if } |T(l) \cap V_i| < |V_i|/2 \\ |V_i - T(l) \cap V_i| & \text{otherwise,} \end{cases} \tag{4.6}$$

where $T(l)$ 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 $V_{\{1\ldots k\}\setminus i}$. 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.

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

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.

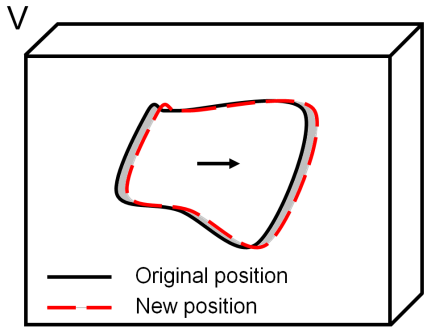


FIGURE 4.9. When we shift the template to a nearby location, only the shaded area changes. Therefore, we only need to update an area that is proportional to the surface area of the template, rather than the volume of the template. This dramatically decreases the running time during retrieval.

the distance function at each location. Figure 4.9 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. We define the shaded region as follows. Let T' be the set of points for the template at location L'. Let T be the set of points for the template at a new location L. The shaded region that needs to be updated is $T \cup T' \setminus T \cap T'$. We further divide

Input: C: 3D array of region labels for each voxel.
S: Array containing the size of each region.
N: Number of regions.
L: New location to calculate matching distance.
overlap: Array of amount of overlap between T and region V_i at location L'.
T_P: Positive update region.
T_N: Negative update region.

Output: Matching distance

foreach p **in** T_P **do**
    increment(overlap[C[$\vec{L} + \vec{p}$]]) ;
end

foreach p **in** T_N **do**
    decrement(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 2: Updating the shape matching distance as we move the template to a new location nearby.

the update region into two subregions, the positive update region

$$T_P = (T \cup T') \setminus T', \tag{4.7}$$

and the negative update region

$$T_N = (T \cup T') \setminus T. \tag{4.8}$$

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. The algorithm is summarized in Algorithm 2.

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 4.10 shows example false positives

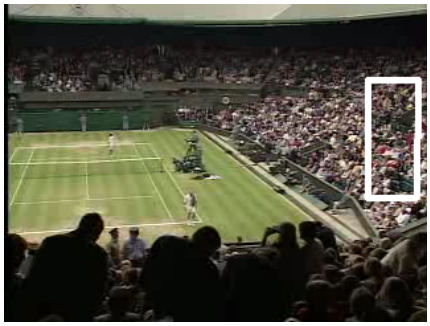


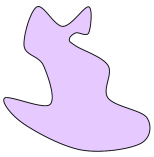
FIGURE 4.10. False positive in cluttered video regions. Because of the extreme over-segmentation of these regions, the tiny regions can be arbitrarily grouped to match any template.

of a tennis serve in a crowded region of a video segment. 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 $|V|$ 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 regularization term that balances the template match by the target volume's inherent flexibility. This is illustrated in Figure 4.11, 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). We propose a normalization model as follows.

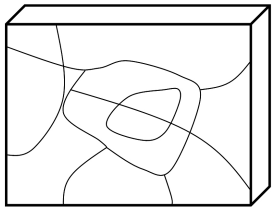
We divide the distance metric $d(T, V)$ by a normalization term so that it becomes

$$d_N(T, V; l) = \frac{d(T, V; l)}{E_{\mathcal{T}}[d(\cdot, V; l)]}, \tag{4.9}$$

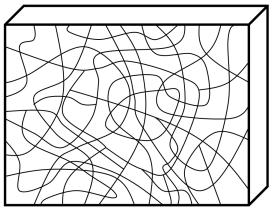
where the denominator is the expected distance of a template to volume V, averaged over $\mathcal{T}$, the set of all possible templates that fit within V. Essentially, this is an estimate of the match confidence. Enumerating through all possible templates to compute the expected value may seem intractable at first, but we show that it is



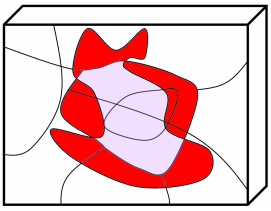
(a) Template



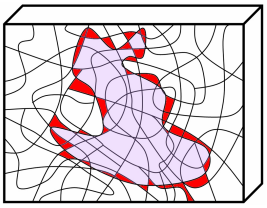
(b) Normal segmentation



(c) Extreme over-segmentation



(d) Distance on normal segmentation



(e) Distance on extreme over-segmentation

FIGURE 4.11. 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 regularization term in the distance metric.

possible to compute this efficiently. Writing out the definition of the expectation, we have

$$E_{\mathcal{T}}[d(\cdot, V)] = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} d(\tau, V) \tag{4.10}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{i=1}^{k} d(\tau, V_i), \text{ by Equation. 4.5} \tag{4.11}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{i=1}^{k} \sum_{\tau \in \mathcal{T}} d(\tau, V_i), \text{ by independence.} \tag{4.12}$$

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

$$E_{\mathcal{T}}[d(\cdot,V)] = \frac{1}{2^{|V|}}\sum_{i=1}^{k}\sum_{j=1}^{|V_i|-1} 2^{|V|-|V_i|}\binom{|V_i|}{j}\min(j,|V_i|-j) \tag{4.13}$$

$$= \sum_{i=1}^{k}\frac{1}{2^{|V_i|}}\sum_{j=1}^{|V_i|-1}\binom{|V_i|}{j}\min(j,|V_i|-j). \tag{4.14}$$

This can be simplified to (see Appendix B):

$$E_{\mathcal{T}}[d(\cdot,V)] = \sum_{i=1}^{k} f(|V_i|), \text{ where} \tag{4.15}$$

$$f(n) = \begin{cases} \frac{n}{2} - \frac{1}{2^n}\binom{n}{n/2}(n/2), & n \text{ even}, \\ \frac{n}{2} - \frac{1}{2^n}\binom{n-1}{(n-1)/2}n, & n \text{ odd}. \end{cases} \tag{4.16}$$

There exists a simple recurrence for computing $f(n)$ (Eqn. 4.16) exactly. Let $f(n) = n/2 - T(n)$. If n is even, then $T(n)$ is defined as

$$T(n) = \frac{1}{2^n}\binom{n}{n/2}(n/2). \tag{4.17}$$

The recurrence can be calculated as

$$T(n+2) = \frac{n+1}{n}T(n) \tag{4.18}$$

$$T(2) = 1/2. \tag{4.19}$$

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

$$T(n) = \frac{1}{2^n}\binom{n-1}{(n-1)/2}n. \tag{4.20}$$

The recurrence can be calculated as

$$T(n+2) = \frac{n+2}{n+1}T(n) \tag{4.21}$$

$$T(1) = 1/2. \tag{4.22}$$

To get an intuition of how this normalization function behaves, $f(n)$ is illustrated in Figure 4.12 as a log-log plot. As n increases, $f(n)$ approaches $n/2$ as expected. Note that Equation 4.15 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.

Not only does this algorithm automatically filter out cluttered backgrounds, it also chooses the best level in the segmentation hierarchy against which to match.

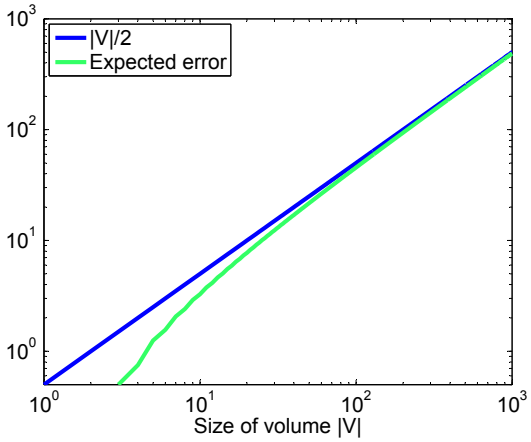


FIGURE 4.12. Illustration of Equation 4.16 on the size of the volume $|V|$ in a log-log plot. The expected error contribution, $f(n)$, approaches $n/2$ as n increases.

Objects that are under-segmented will not match the template at all. And although over-segmented objects will match the template, they will be penalized for having too many regions. The "correct" segmentation level, the one that least over-segments without under-segmenting, will get the highest score.

4. Complementary Nature of Shape and Flow

We 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. While 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 4.13 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 in cases where an action cannot be distinguished from its silhouette alone.

FIGURE 4.13. 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.

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 observed in Chapter 3 Section 6, in the KTH action recognition database, the flow of the boxing action looks very similar to that of the hand-clap (see Figure 3.12). This is because the horizontal trajectories of the arms are 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.

Despite the normalization, our shape-based correlation algorithm can sometimes generate false positives on highly-textured regions, which are finely segmented (Figure 4.14a). However, we can obtain accurate flow measurements on these regions and a flow-based algorithm such as Shechtman and Irani's flow consistency [**119**] 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,

(a) shape fp (b) flow fp

FIGURE 4.14. False positives on 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.

we observe that the majority of false-positives occur in low-textured regions (Figure 4.14b). 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 6.

5. Detection

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 Chapter 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 Chapter 2 Section 5.2) to measure the flow matching distance [**121**]. 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:

(4.23) $d(T, V; l) = d_N(T, V; l) + \alpha d_F(T, V; l)$ from Eqns. 4.9 and 2.7,

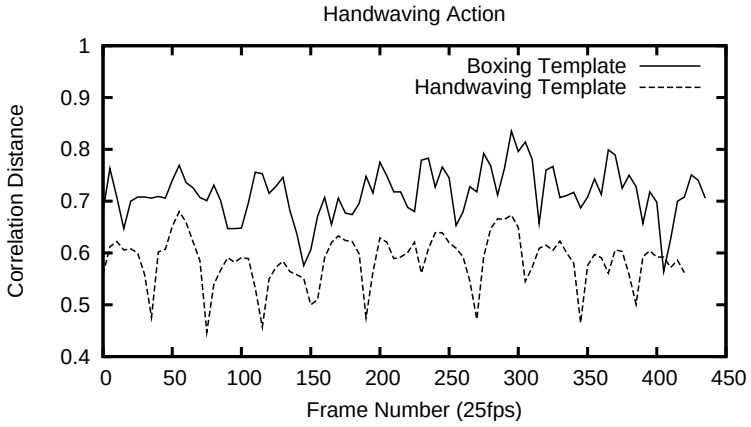


FIGURE 4.15. The minimum correlation distance of two templates on a hand-waving action. Notice the cycles in the action, where the distance is minimized when the phase of the template matches that of the action.

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 4.15 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 Chapter 6 on how to generalize to multiple templates.

6. Evaluation on Tennis Dataset

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

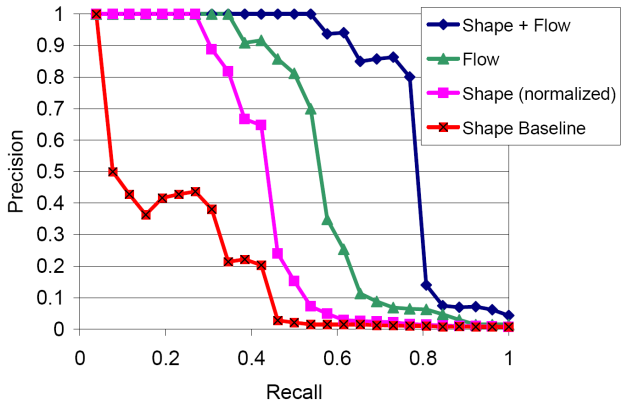
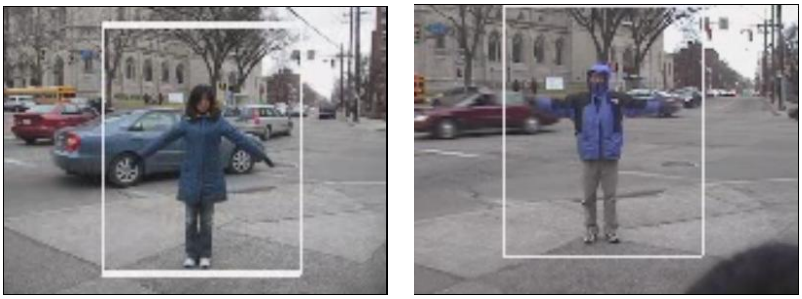


FIGURE 4.16. Comparison of various features on 30 minutes of tennis video in an event detection task.

(*e.g.*, Figure 4.14a) and only a few instances of the actions are present in the video. We manually selected an example of Rafter serving (Figure 4.18a) 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 4.16 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. Qualitative examples of other actions we can detect are illustrated in Figures 4.17 and 4.18.



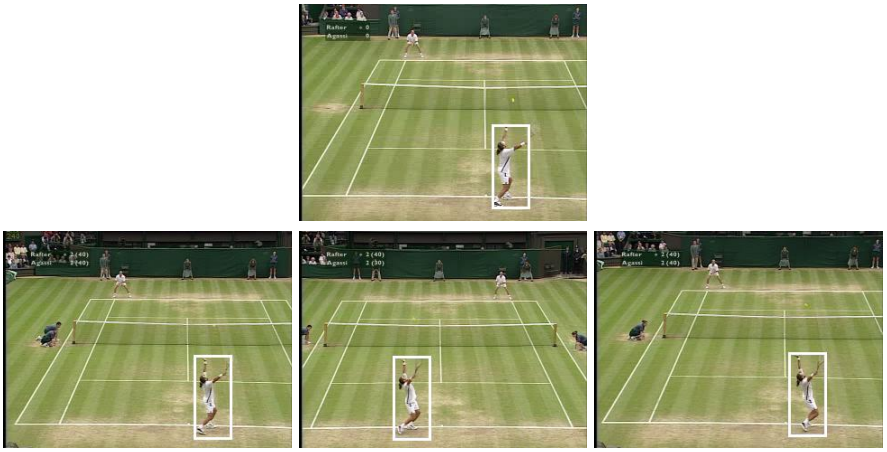
(a) (b)

FIGURE 4.17. Hand-wave detections in a cluttered scene and with a moving background.

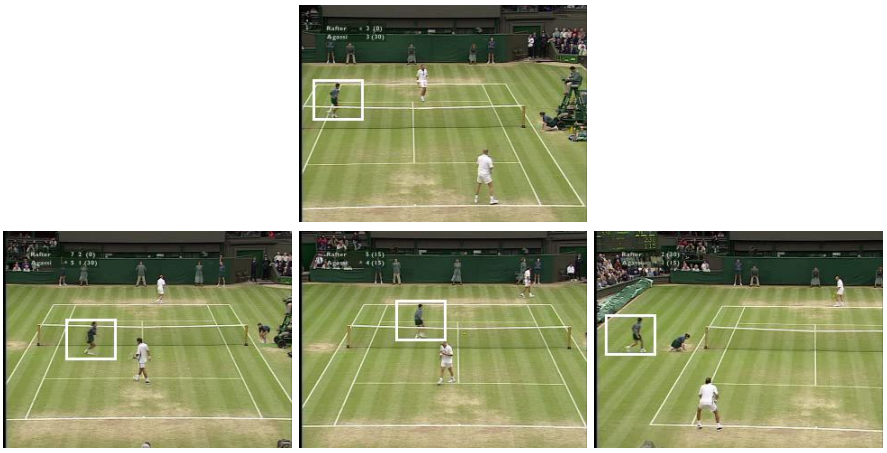
7. Comparison to Standard Datasets

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 [**117**]. Each video clip contains one person performing an action multiple times. This dataset is difficult because it contains drastic lighting, clothing, and scale changes (Figure 4.19). Different people also perform the actions at different speeds and orientations. The videos were recorded using a handheld camera, which prevents 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.

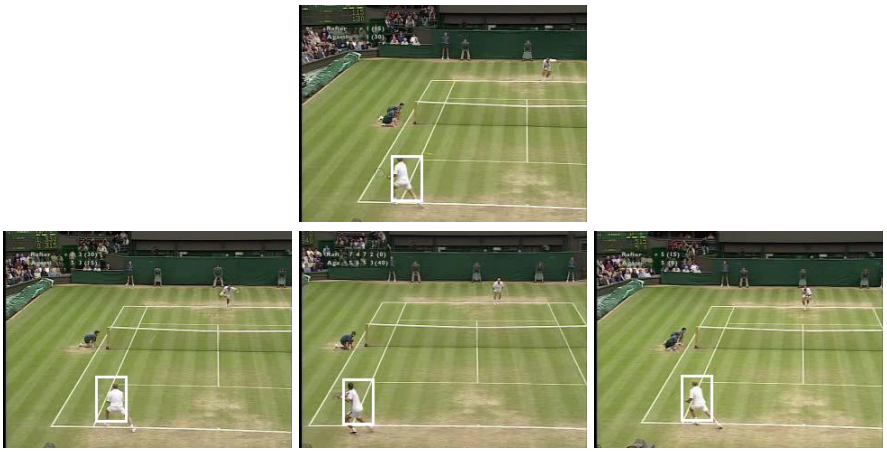
To classify the video sequences, we first classify the individual frames and then we assign the class label with the highest frame count to each video clip. While there are a number of classifiers that one could use to classify the frames, we chose to use SVM because of its reported success in a wide range of applications. We train the SVM (using LIBSVM [**25**] 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



a) serve action



b) run right action



c) return serve action

FIGURE 4.18. Illustration of actions detected in a tennis sequence. Top row: templates; Bottom row: example detections.

(a) (b)

FIGURE 4.19. Notice the difference in scale between some videos in the KTH dataset. The contrast is also low making segmentation difficult.

feature vector corresponds to a distance between the candidate video location and a template action.

Following the methodology of Niebles *et al.* [**103**], we use leave-one-out cross-validation grouped by person to measure the classification accuracy. To generate the training 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 4.21 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 4.1). Unfortunately, we can only loosely compare the results in Table 4.1 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 4.21 shows the effect of using different features and training on different number of

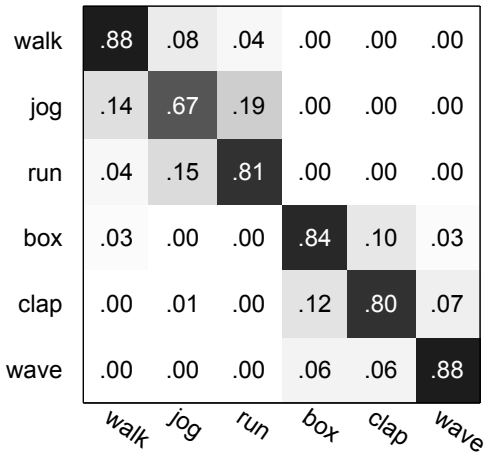


FIGURE 4.20. KTH Actions confusion matrix. Walk, jog, and run actions are most easily confused. Box, clap, and wave actions are sometimes confused.

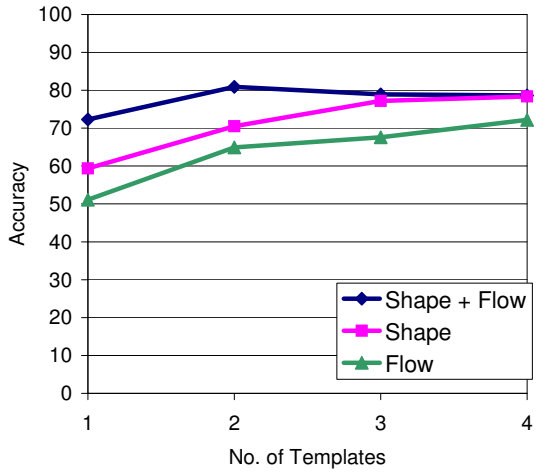


FIGURE 4.21. Results on the KTH actions dataset. Training on more templates improves results with diminishing returns. The combined shape and flow features perform better than either alone, especially with few training examples.

templates. We 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.

TABLE 4.1. Although our method is not specifically designed for whole-video classification, our results on the KTH actions dataset [**117**] are competitive with recent studies.

Related work	Accuracy
Our Method (shape + flow)	**80.9%**
Flow only from Chapter 3 [**79**]	63.0%
Schuldt *et al.* [**117**]	71.7%
Dollar *et al.* [**46**]	81.2%
Niebles *et al.* [**103**]	81.5%
Jiang *et al.* [**76**]	84.4%

8. Limitations

Although our baseline technique works well, it is still limited because it is trained from one template. Due to the variations in how people perform an action, we would like to train on many examples of an action. Similarly, our baseline algorithm is not scale nor view invariant. We would need to scan over many (spatio-temporal) scales and train from multiple viewpoints. Instead of scanning over many templates, a possible solution to improve the generalizability of a single template is to make it deformable. Our baseline technique only scans rigid templates. These and additional issues are further explored in Chapter 5.

A limitation of searching over many templates is the speed of the algorithm. We initially chose mean shift because of its high accuracy and success in image segmentation. However, since our algorithm does not require an exact segmentation and since objects could be over-segmented, we could replace mean shift with other faster segmentation algorithms [**10**, **15**, **24**, **122**]. Another way to improve the matching aspect is to do a coarse-to-fine approach in matching. This is essentially an early discard technique that attempts to avoid computation on in which we are confident that it could not match. On areas where there are possible matches, we perform more higher precision calculations to determine the match. These ideas are explored in related work for shape [**38**] and flow descriptors [**119**].

9. Conclusion

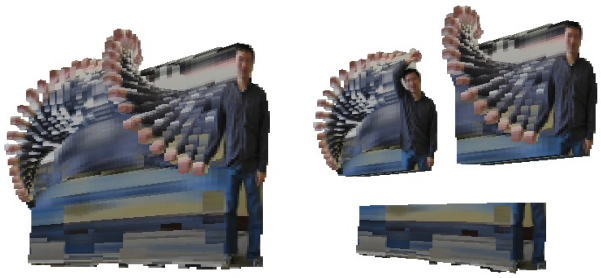
We described our baseline method for event detection using volumetric shape matching. The main contribution of this part is that we are able to do shape matching on videos without having to rely on a foreground mask. Unlike previous work that extracted the foreground mask by background subtraction, and thus requiring videos with static backgrounds, we can match volumetric shapes on automatically segmented videos. First, we segment the video volumetrically using mean shift. The result is an over-segmentation of the video that tries to preserve object boundaries, by may cut an object into many segments. We then proposed a shape matching distance based on region intersection that works on over-segmented videos. To avoid false positives in finely segmented regions, we introduced a normalization term that accounts for the segmentation granularity of the video. We further combined our shape descriptor with Shechtman and Irani's volumetric flow matching technique for improved performance. The baseline algorithm shows good results in the Tennis dataset and is comparable to other algorithms on the standard KTH dataset.

CHAPTER 5

Parts Based Recognition

1. Introduction

We propose a parts-based shape matching technique that improves the generalization power of our baseline technique described in the previous chapter. The main strength of the baseline algorithm is that it can perform shape matching without precise object masks in the input video [**16**, **17**, **152**]. 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 [**17**,**119**], 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 [**146**] and video [**18**, **120**]. Therefore,



(a) Whole (b) Parts

FIGURE 5.1. 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.

we extend our baseline matching algorithm by introducing a parts-based volumetric shape-matching model. Specifically, we extend the pictorial structures framework [**53**, **56**] to video volumes to model the geometric configuration of the parts and to find the optimal match in both appearance and configuration in the video.

2. Parts Based Shape Descriptor

A key feature of our baseline algorithm is that it can perform shape matching on 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 hand-wave template into parts, as shown in Figure 5.1. 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 hand-wave 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.

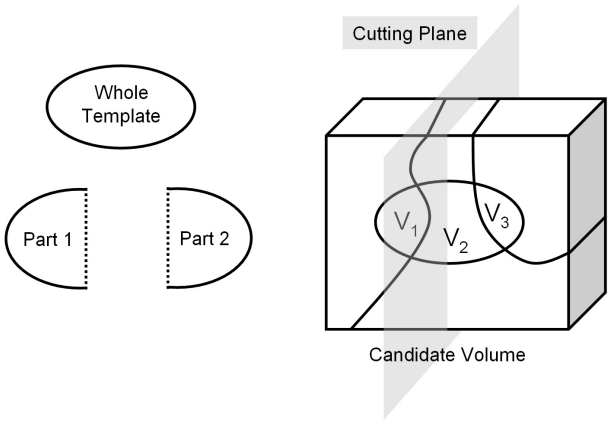


FIGURE 5.2. Illustration of how we artificially cut the candidate volume to match how the whole template is split into its constituent parts. The candidate volume is dynamically cut as we slide the template parts along it. The cutting process is very efficient.

We now generalize our baseline algorithm (in Chapter 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 of Figure 5.2. 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 by 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.

3. 3D Pictorial Structures

We now describe how the framework of pictorial structures [**53**, **56**] 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 [**53**], 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:

$$L^* = \operatorname*{argmin}_L \left(\sum_{i=1}^{n} a_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \tag{5.1}$$

The correlation distance $a()$ is a linear combination of our normalized distance metric (Equation 4.9) and Shechtman and Irani's flow-based correlation distance (Equation 2.7):

$$a_i(l_i) = d_N(T_i, V; l_i) + \alpha d_F(T_i, V; l_i), \tag{5.2}$$

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,

$$d_{ij}(l_i, l_j) = \beta \mathcal{N}(l_i - l_j, s_{ij}, \Sigma_{ij}), \tag{5.3}$$

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 F&H [**53**], 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.

3.1. 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 5.3). 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[1]. 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 160x120 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.

For each event, we create the model from a single instance by interactively segmenting the spatio-temporal volume using an interactive video cutout tool similar to work by Wang *et al.* [**142**]. 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 5.3. 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,

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

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 and to reduce the number of potential false positives, we project the map to a one-dimensional vector of scores, keeping only the best detection for each frame, as shown in Figure 5.4(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.

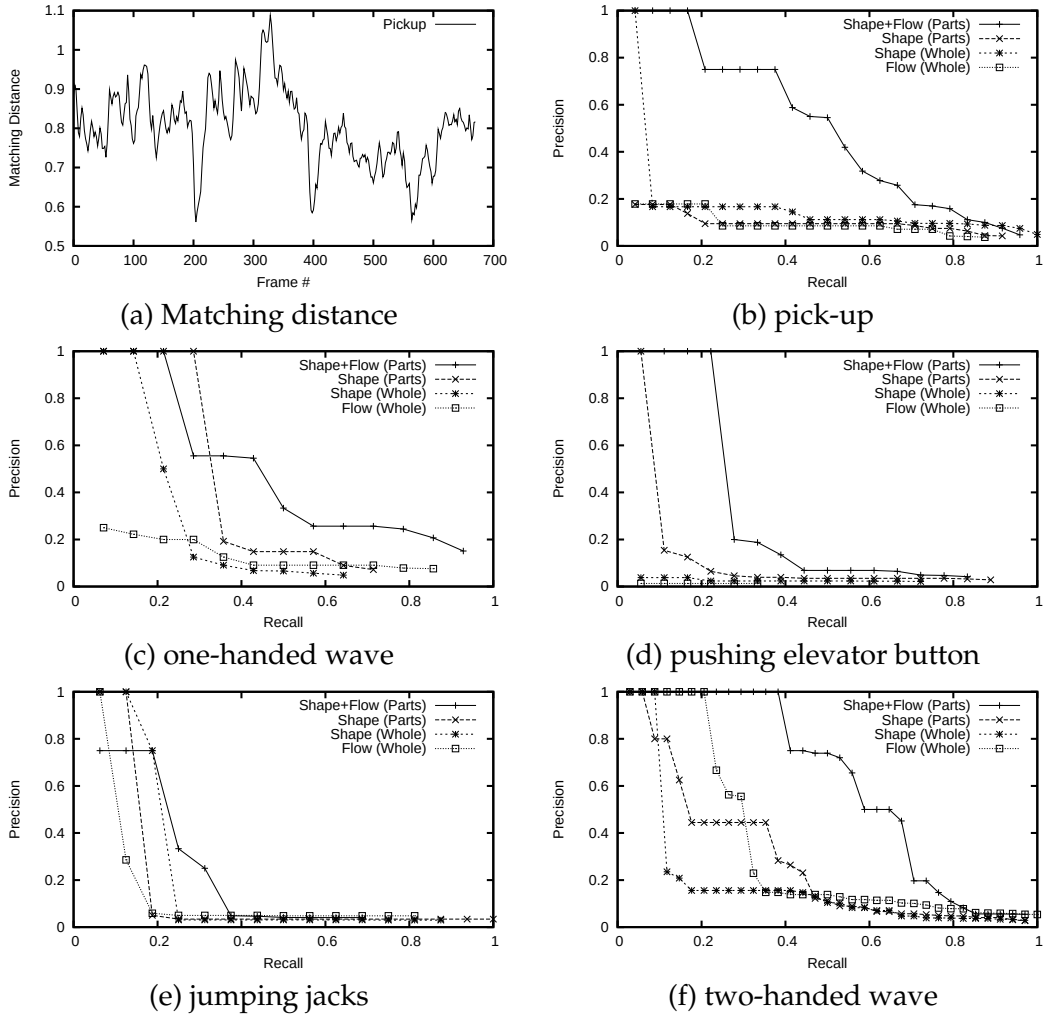
We now analyze the performance of our algorithm and compare it to the baseline methods. Figure 5.3 shows example detections using our algorithm with the parts-based shape and flow descriptor in crowded scenes. Note the amount of clutter and movement from other people near the event. The precision-recall graphs for all of the actions are shown in Figures 5.4(b)–(f). We compare our results to Shechtman and Irani's flow consistency method **[119]** 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, which confirms our previous findings **[81]**. 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 illustrate 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 5.5 shows the difference between the whole and parts-based templates as they are overlaid (in pink)

FIGURE 5.3. 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 overlaid on the test sequence, and a bounding box drawn around each part.

onto the video. There was a difference in hand-waving 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.

3.2. Comparison to Chamfer Distance Matching

It is important to compare our parts-based event detector to other baseline techniques, in particular the classic technique of Chamfer distance matching. We use the 3D spatio-temporal extension of the traditional 2D Chamfer distance matching, as described in Chapter 2. Similar to the 2D case, the spatio-temporal extension can also be computed in $O(n)$ time, where n is the number of pixels. At run-time,



(a) Matching distance (b) pick-up

(c) one-handed wave (d) pushing elevator button

(e) jumping jacks (f) two-handed wave

FIGURE 5.4. (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 [**119**], labeled as "Flow (Whole)", achieves low precision and recall in most actions, demonstrating the difficulty of our dataset.

we slide the template along the video and at each iteration, we can calculate the distance in $O(m)$ time, where m is the number of pixels that lie on the surface of the template, and is typically much smaller than the volume. A typical action template captured from 320x240 pixel by 15 frame video might occupy approximately 70,000 voxels, but the surface is only 8,800 voxels. Therefore, Chamfer distance matching in the 3D case is still efficient to compute. Figure 5.6 shows an example jumping-jacks event, its edges, and the Chamfer distance transform. The

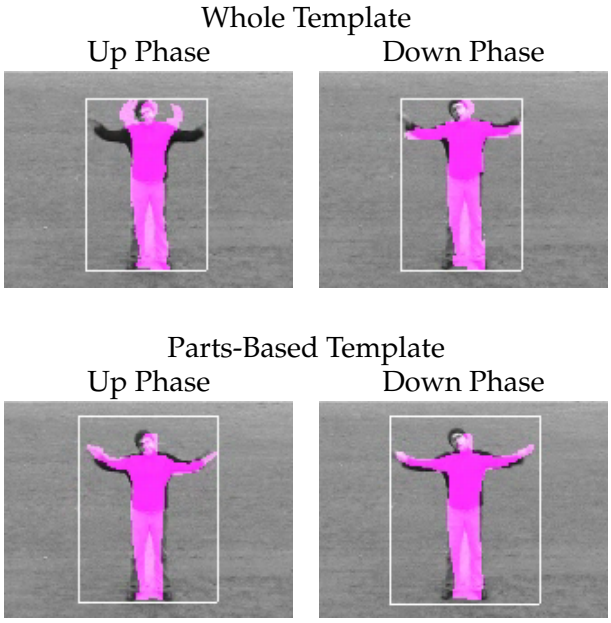


FIGURE 5.5. Illustration of how our parts-based model can stretch to accommodate the mismatch in speed between the template and the detected action. While the down-phase of the hand-wave is matched to the whole template, the up-phase is clearly misaligned. The parts-based template matches both phases well and thus enables better detection accuracy.

FIGURE 5.6. Example of Chamfer distance transform on a jumping-jacks event.

edges are extracted using OpenCV's Canny edge detector with threshold parameters $(20, 200)$ and aperture size of 3 **[3]**. Figure 5.7 shows the results of comparing our parts-based event detector with the baseline 3D Chamfer distance matching. Chamfer distance fails in almost all cases except the *push-elevator-button* action. That category has the cleanest background, which is probably why Chamfer distance matching performed well.

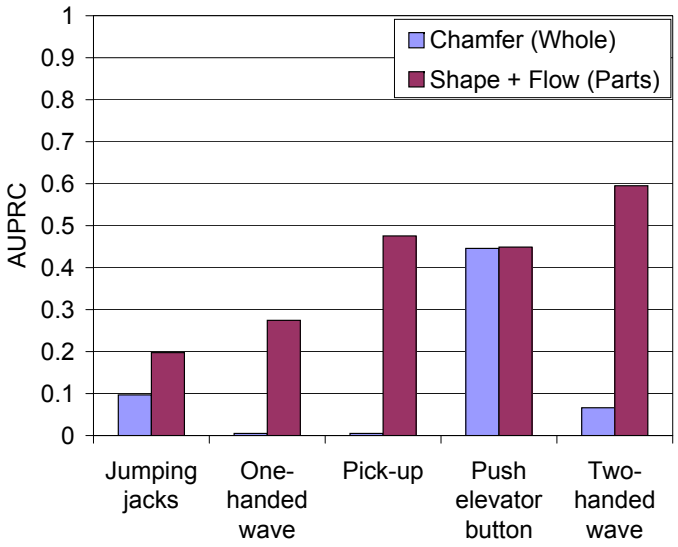


FIGURE 5.7. Comparison to Chamfer distance matching. Chamfer distance on Canny edges.

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 5.1a to look like the parts in Figure 5.1b. Given many training examples, we can automatically learn the optimal place to cut the templates. Intuitively, the cuts divide the templates into parts 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 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 5.8 shows the

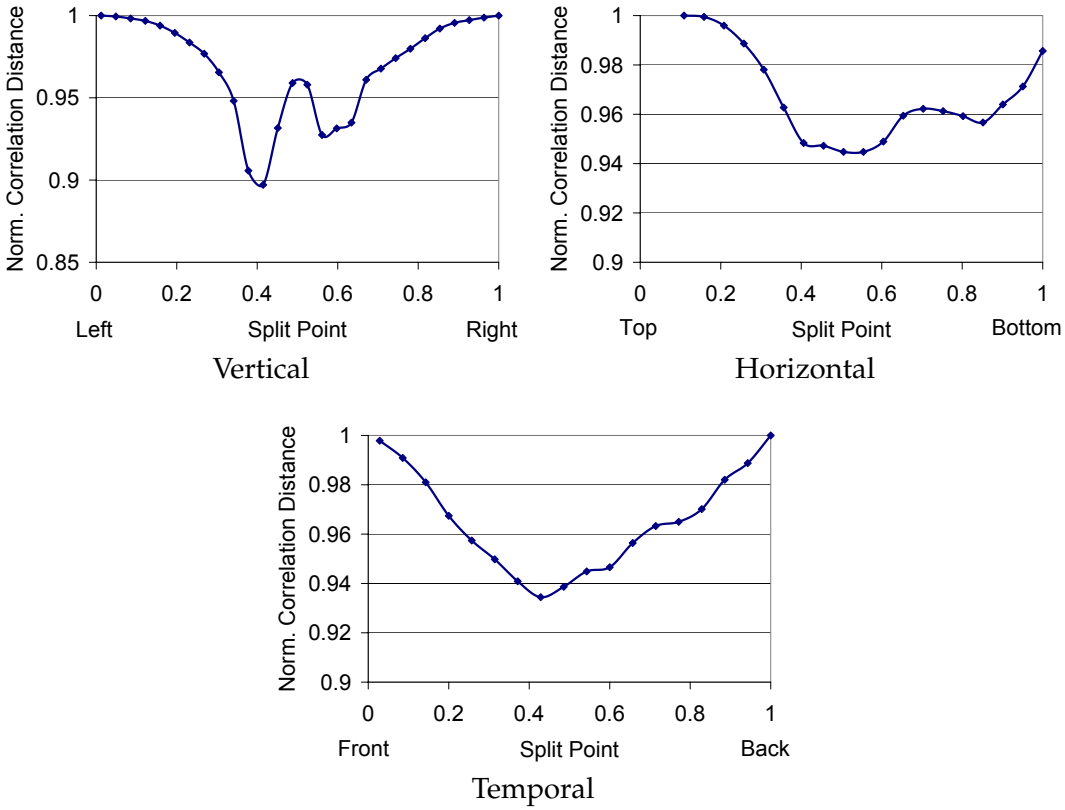


FIGURE 5.8. Matching distance at different cut locations along each axis.

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 5.9 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 5.9. Further divisions can be made by recursively dividing the individual parts. Once we have the parts, we can learn the part configuration parameters, which we describe next.

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

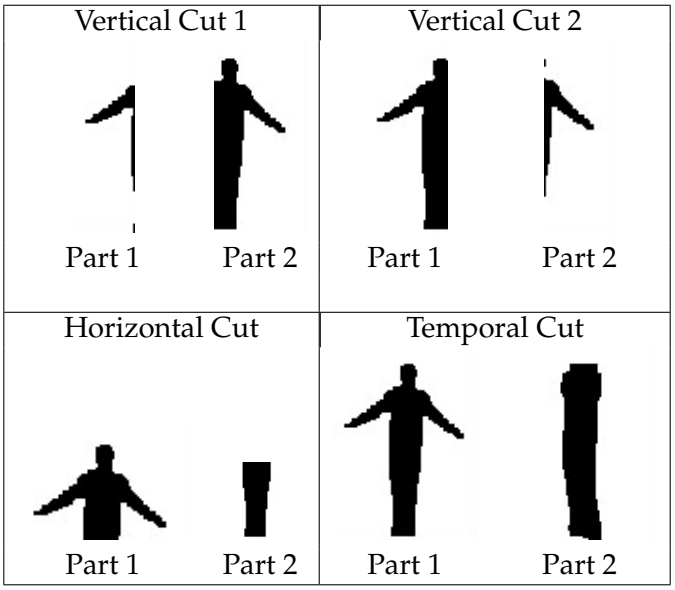


FIGURE 5.9. Automatically cutting whole template into parts.

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, $T = \{T_1 \dots T_m\}$. 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.

$$s_{ij} = \frac{1}{n} \sum_{k=1}^{n} (l_i^k - l_j^k). \tag{5.4}$$

$$\Sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} (l_i^k - l_j^k - s_{ij})^2. \tag{5.5}$$

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.

FIGURE 5.10. Illustration of how the part offset parameters are initialized.

The overall learning is an EM-like iterative process [97]. 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 3. 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 5.10 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 Equation 5.1). Given the set of part locations, we can now re-estimate s_{ij} and Σ_{ij} using Equations 5.4 and 5.5. 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 3.

6. Conclusion

We extended our baseline shape matching algorithm to detect event parts (sliced in both space or time), and we generalized the model to recognize actions with higher actor variability. The parts are combined using pictorial structures to find

Input: $E = \{E_1 \dots E_n\}$: Location of training events.
s', Σ': Initial estimates of s and Σ.
$T = \{T_1 \dots T_m\}$: 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 Σ'.
  end
  Estimate s, Σ using part locations for all events (Equations 5.4 and 5.5).
  $s' = s$;
  $\Sigma' = \Sigma$;
until s, Σ *converges* ;

return s, Σ ;

Algorithm 3: Estimating part configurations.

the optimal configuration. Our approach detects events in difficult situations containing highly-cluttered dynamic backgrounds, and significantly out-performs the baseline method [**119**]. The biggest limitation of the current work is that the model is derived from a single exemplar of the event, thus limiting our ability to generalize across observed event variations. While we are able to generate the parts and learn the part parameters from several training examples, we are still limited to matching against one template. In the next chapter, we discuss ways to use multiple templates and explore other robustness issues.

CHAPTER 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. For example, Figure 6.1 shows the large amount of camera movement there exists in a 2.5 second video clip. We analyze how our system performs in various settings and how robust it is to these kind of variations.

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

FIGURE 6.1. Example video sequence illustrating the a large amount of camera movement.

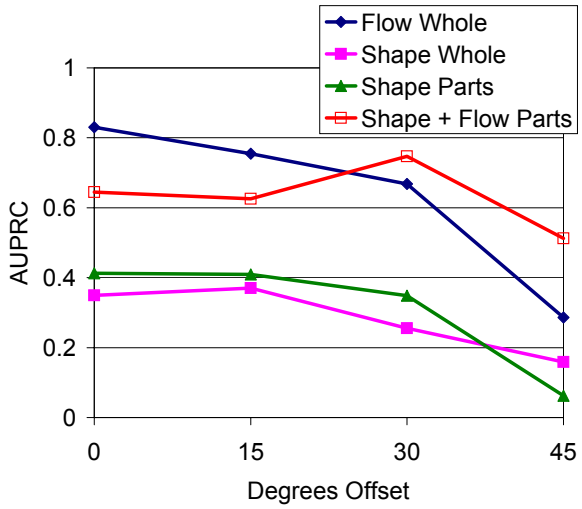


FIGURE 6.2. Robustness to camera view change. The performance is good for a skew of up to 30 degrees.

frames are illustrated in Figure 1.11. We averaged the area under the precision-recall curve (AUPRC) overall actions and plotted them against the camera viewpoint, shown in Figure 6.2.[1] 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.

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 6.3. Column (a) shows the detection results using a single template per

[1] See Appendix A for a more in-depth discussion of AUPRC.

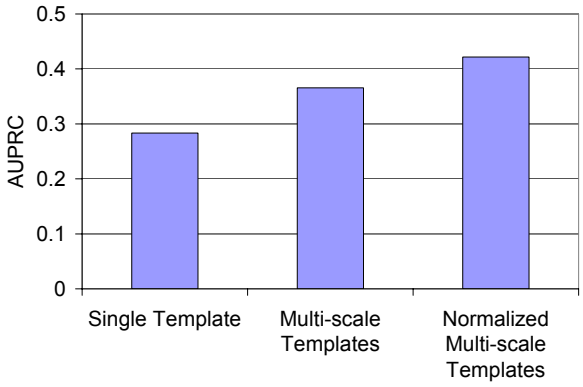


FIGURE 6.3. Effectiveness of scanning multiple scales in space-time (on the Multiview dataset). Left: Single template. Middle: in 9 scales (3 in space and 3 in time). Right: Multi-scale normalized templates to account for differences in template size.

action. Column (b) shows the results with the multi-scale templates and it clearly improves over the single template.

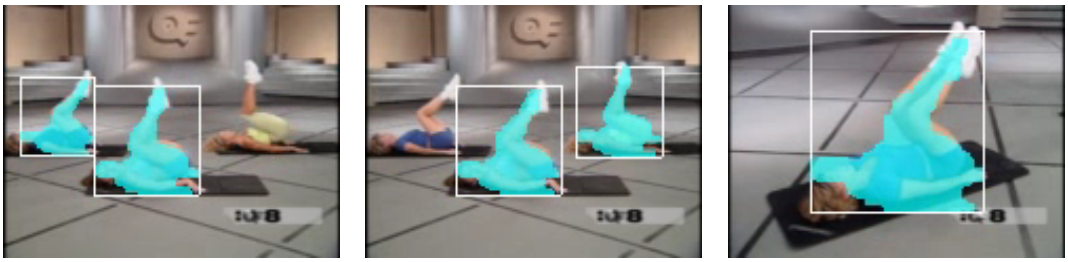
A potential problem with using multi-scale templates is that since the templates are different sizes, they cannot be directly compared to each other using the same detection threshold. This is evident by looking at our distance metric, shown below.

$$d(T, V_i; l) = \begin{cases} |T \cap V_i| & \text{if } |T(l) \cap V_i| < |V_i|/2 \\ |V_i - T(l) \cap V_i| & \text{otherwise.} \end{cases} \tag{6.1}$$

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 as follows.

$$d_N(T, V; l) = \frac{d(T, V; l)}{E_{\mathcal{V}}[d(T, \cdot)]} \tag{6.2}$$

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 template T and many segmented videos. Using this normalized distance metric, we achieve the best detection results, as shown in column (c). Figure 6.4 shows anecdotal results on the Aerobics dataset. The spatial scale difference between the two views is 2.3x and we successfully detect all of the events. The bounding box and template overlay show that we correctly detected the scale of the events.



(a) (b) (c)

FIGURE 6.4. 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.

TABLE 6.1. 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

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 [**49**]. For 640×480 resolution videos, this translates to a maximum of 16x in scale variation. Temporal variations, on the other hand, are much smaller. Table 6.1 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.

FIGURE 6.5. Video with three people performing an action at the same time. Using single-instance detection only detects one of the three people.

3. Multi-instance Detection

Our baseline detection algorithm projects the three dimensional distance map onto the temporal axis by preserving only the best detection for each frame. This has two main advantages. First, false positives are minimized to a maximum of one per frame. This is especially useful because of the high number of locations that must be checked for the events. The other advantage is that it is simple to check for local minimums. We only need to check on the horizontal axis. If no projection was were, we would need to do local non-maximal suppression in 3D to transform the distance map into a set of discrete detections. However, the drawback is that if two events occur very close in time (less than a couple of frames), one of them is likely to be missed. Events from different actions can still be detected simultaneously, however, since they the distance maps are projected independently of each other. Many types of videos do not have simultaneous events, and this type of detection scheme will work well for them because it minimizes the number of false positives. For example, in a tennis match, only one person serves at a time and the serves are spaced far apart. However, in the Aerobics dataset shown in Figure 6.5, there are multiple people performing the action simultaneously. The single-instance detector only found one instance of the events. In order to detect all of the instances, we would need to be able to detect multiple events that start within a single frame.

Fortunately, the three-dimensional distance map contains all of the required information for multiple instance detection. The two main challenges are that we

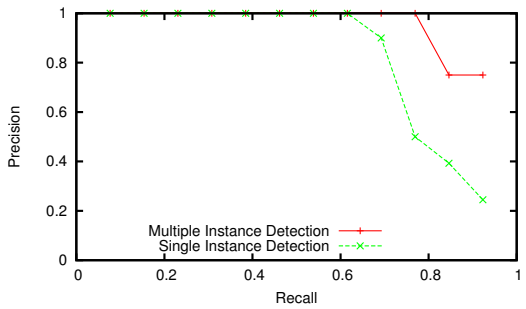
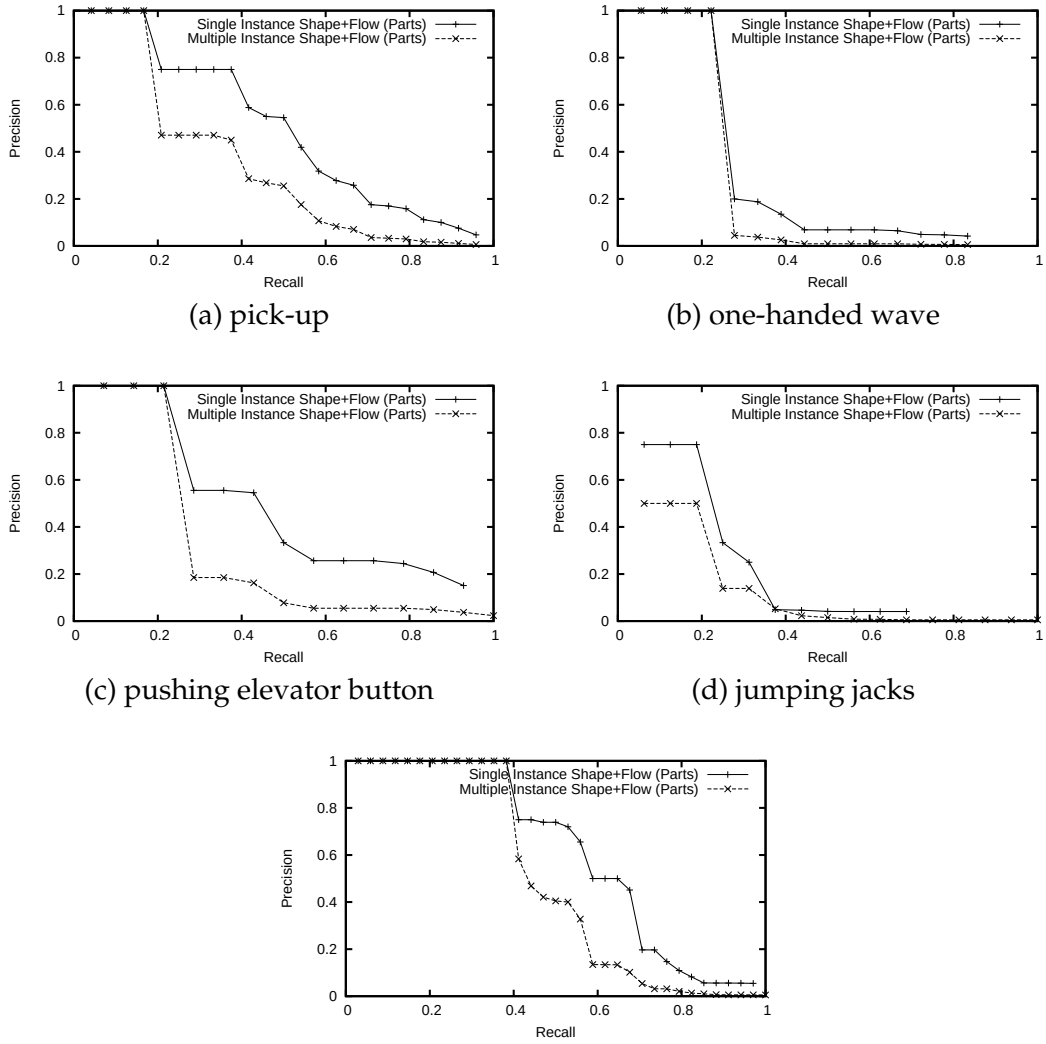


FIGURE 6.6. Multiple instance detection on Aerobics dataset

need to detect all local minima and we need to perform local non-minimal suppression in three dimensions. Both of these can be done simultaneously using mean shift, similar to mean shift tracking **[32]**. First, we threshold the distance map at some value. We then perform mean shift to form a set of clusters, where each center is a candidate event detection. Small clusters (those with low weight) are considered as noise and are discarded. The remaining clusters are then the final detections. Using this method, we are able to detect instances of the same event occurring simultaneously. Figure 6.6 shows the precision-recall of detecting the raise-knee event in the Aerobics dataset. Using multiple-instance detection increases the performance because the single-instance detection missed some events that occurred simultaneously. Figure 6.4 shows example detections obtained using the multi-instance detector. The three events that occurred in this small time period were all detected. A potential problem with multiple-instance detection is that the precision might drop dramatically because we could have many false positives in a single frame, as opposed to a maximum of one for single-instance detection. We ran the algorithm on the Cluttered Videos dataset and compared the result to single-instance detection, as shown in Figure 6.7. Since there are no simultaneous events in this dataset, there is no benefit in using multiple-instance detection and thus we expect the performance to drop due to the increase in the number of false positives. The performance only slightly drops in this dataset, showing that the increased number of false positives is manageable.



(a) pick-up (b) one-handed wave

(c) pushing elevator button (d) jumping jacks

(e) two-handed wave

FIGURE 6.7. Multiple instance detection on cluttered videos.

4. Training with Multiple Templates

Given multiple examples of an event, we would like to use them to train a set of optimal templates for recognition. We performed experiments on the Weizmann dataset to show how the system can utilize multiple exemplars for improved performance. The Weizmann dataset is well suited for this task because all of the events have foreground masks and therefore we can automate the template extraction during training. The foreground masks are never used during testing; only the spatio-temporal segmentations are used for shape matching. Since there are nine

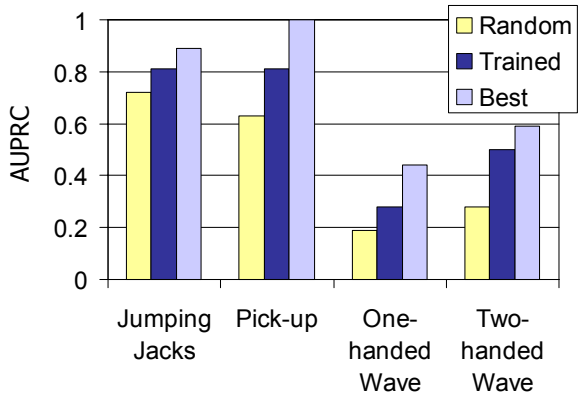


FIGURE 6.8. Given multiple exemplars for training, we see that training for the best template using cross-validation (Trained) gives better performance than using a random template (Random). "Best" denotes the upper-bound in performance if we knew which template would give the best results. Experiments performed on the Weizmann dataset.

actors, every action class has at least 9 exemplars; many have up to 36 if the actors perform multiple instances of an action in one video clip. Therefore, we can divide the data into training, validation, and testing sets and perform cross-validation performance evaluations. Only the whole shape feature is used for the following experiments.

Our first experiment is to show whether we can train a (single) good template for detection given multiple training exemplars. For each action, we randomly set aside the videos from 4 people for testing. Using videos from the remaining 5 people, we find the single best template using cross-validation. This template is then run on the testing set to measure its recognition performance. In Figure 6.8, the label "Trained" shows the results from this experiment. We compare it to "Random", where we randomly sample a training template and run it on the testing set. We also compare it to "Best", where given a testing set, we evaluate the performance from the best possible training template. This is the upper-bound on performance given the data. The graphs were generated by repeating the experiments 50 times and averaging the AUPRC. The results show that training using multiple exemplars performs better than choosing a random template, as expected.

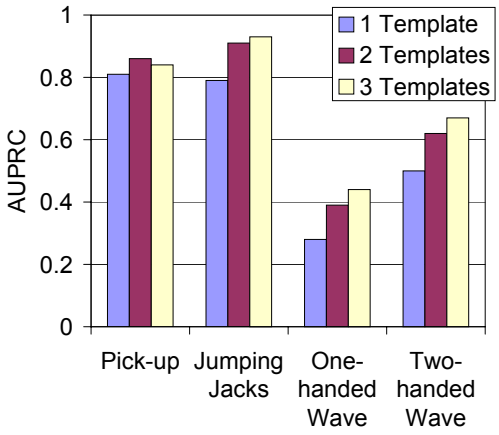


FIGURE 6.9. Training on and using multiple templates simultaneously for detection improves performance than using a single template. Experiments on the Weizmann dataset.

Of course, we need not limit ourselves to a single template during training. We can use a set of templates simultaneously for detection. The templates are correlated with the testing data and the events are considered to be detected if they fall below a threshold to any of the templates. For this experiment, we again use the videos from 4 randomly selected people for testing and use the remaining 5 for training and validation. If we train for n templates, n people's videos are used for training and $5 - n$ people's videos are used for validation. The results are shown in Figure 6.9. We see that in general, using more templates for detection improves performance.

5. Volumetric Segmentation

Using volumetric segmentation is better than segmenting the video on a frame-by-frame basis. Figure 1.6 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 6.10 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 the average number of octree

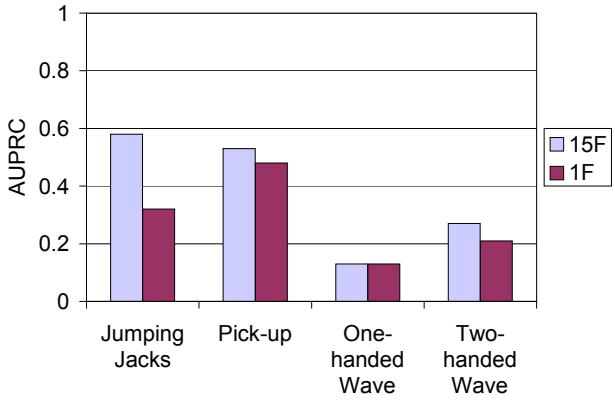
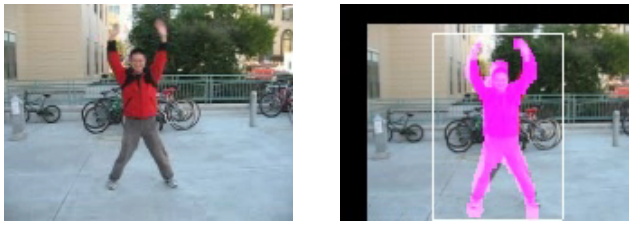


FIGURE 6.10. Recognition results comparing volumetric segmentation (15F) versus single-frame (1F) segmentation on the Weizmann dataset. We see that segmentation using blocks of 15 frames gives better results than single-frame segmentation.

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 to capture the object boundaries more consistently through time. Given the same number of regions, volumetric segmentation can more accurately represent the boundaries.

6. Robustness to Camera Movement

While our features are robust to small amounts of camera motion such as those from a hand-held camera, they are 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 [**105**], 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 [**9**, **13**, **98**, **132**]. 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 6.1 shows an example of camera shake and Figure 6.12 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



(a) Video with Movement (b) Stabilized Video

FIGURE 6.11. 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 [**105**], we were able to easily detected events in the stabilized sequence (b).

FIGURE 6.12. 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.

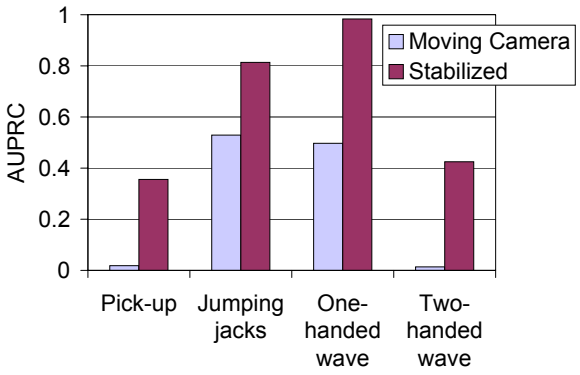


FIGURE 6.13. Detection results on the Moving Camera dataset. Using camera stabilization dramatically increases the recognition rate.

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 6.11 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 6.13. The results for all actions improve when we add camera stabilization.

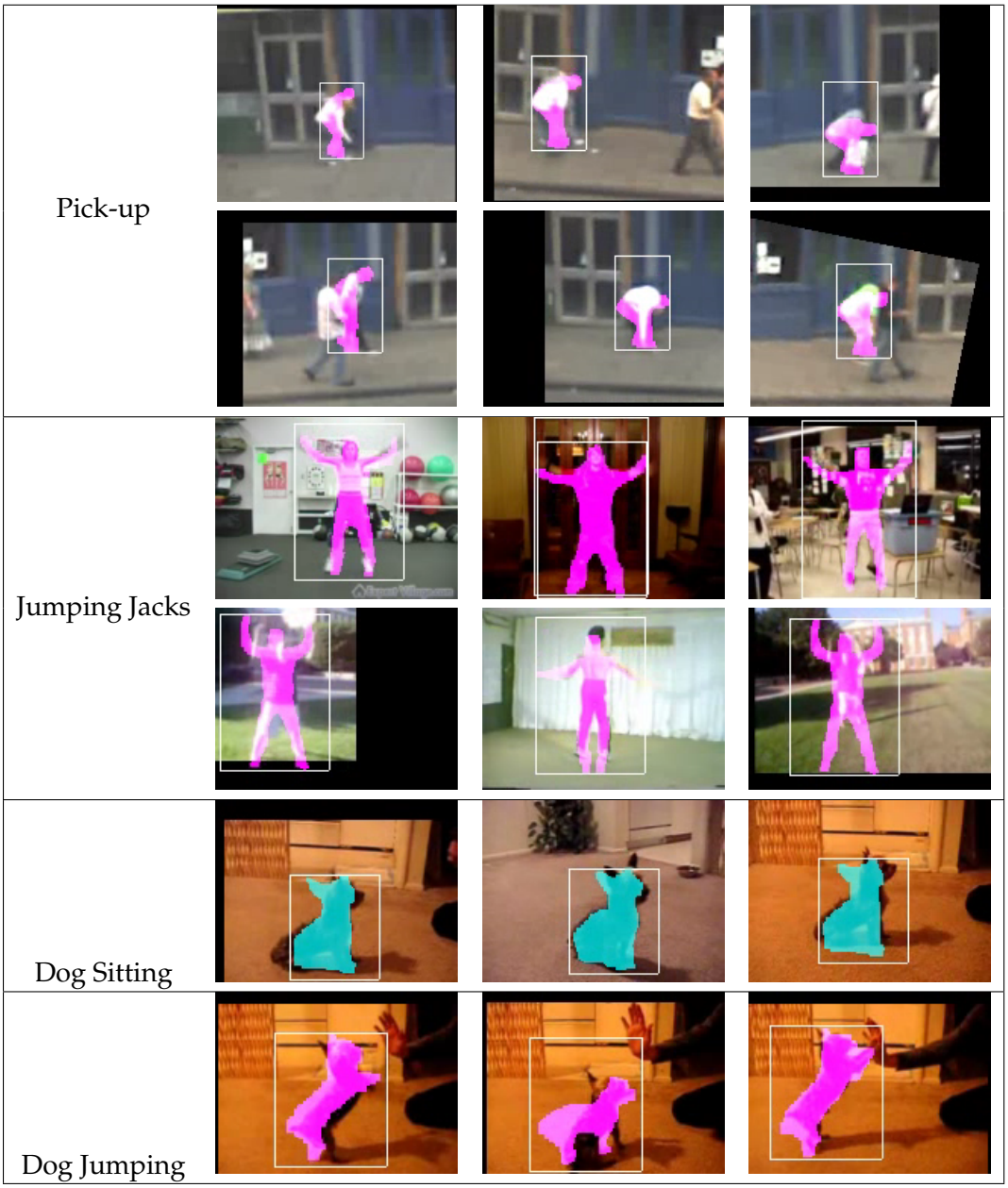


FIGURE 6.14. Example detections of events on unscripted YouTube videos.

7. 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 6.14 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

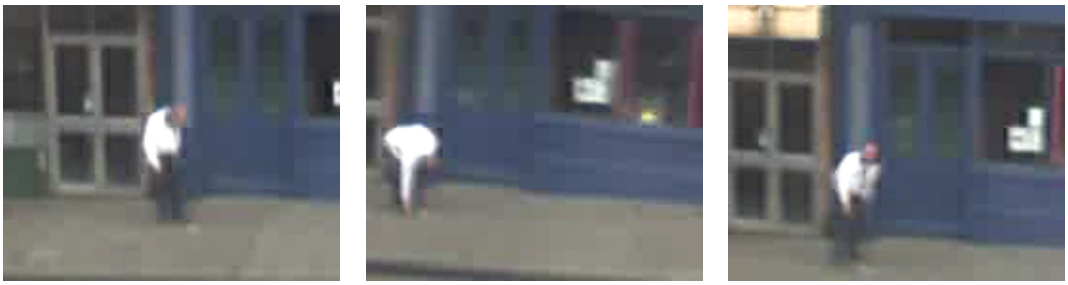


FIGURE 6.15. These frames show how much camera movement there is in a typical video sequence. The frames were extracted in a two second window.

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 6.15 shows how much camera movement there is in a typical video.

8. Extracting Training Templates

Automatic template extraction would be useful during the training process. From a video clip of the event, we need to find the person's silhouette boundaries to build the volumetric shape, as shown in Figure 6.16. Given controlled environments, the easiest way extract the silhouette is to have the actor perform the action against a static background and use background subtraction. This is done for the Weizmann dataset, for example [**16**]. Even on static backgrounds, background-subtraction is not perfect and there may be cases where we need to refine the extracted silhouette.

We propose an alternative technique for segmenting the person from the background based on Wang *et al.*'s interactive video cutout technique [**142**]. The goal is to use a small amount of manual labeling to approximately delineate the foreground and background. The algorithm then computes the exact boundaries. Figure 6.17 shows an example of this process. We summarize our implementation of the video cutout.

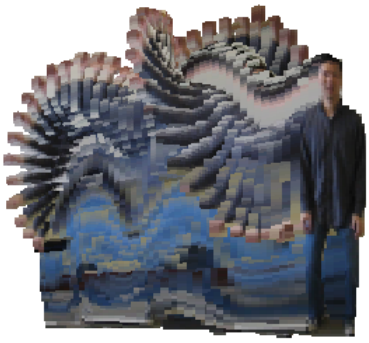


FIGURE 6.16. Jumping jacks shape template.

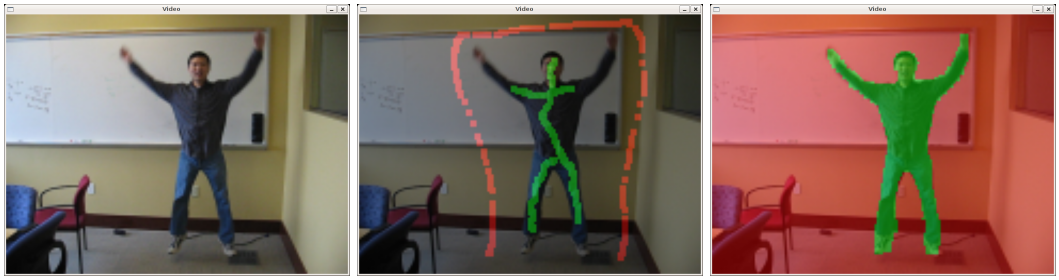


FIGURE 6.17. Illustration of how we generate model templates using an interactive video segmentation process similar to Wang *et al.* [**142**]. Given the training video (left), we draw a few strokes to label the foreground and background regions (middle), and graph-cut generates the complete segmentation mask (right).

First, the user selects a block of frames (typically 15) in the video to segment. The pixels in this video block is represented as a graph, a 3D grid where all adjacent pixels (in space and time) are connected to each other. The weight between adjacent pixels is similarity in RGB color between the pixels. Pixels with similar colors are more likely to belong to the same component, and therefore have higher weight. All labeled foreground pixels are connected to the source with infinite weight and the sink with zero weight [2]. Similarly, the background pixels are connected to the sources and sinks with the appropriate weights. Finally, the unlabeled pixels are connected to the sinks and sources with weights depending on the probability that they are foreground and background pixels. This probability is computed based on the color distribution of the labeled pixels. First, we quantize the color space into 5 bins per dimension for a total of 125 bins. All of the labeled pixels are quantized

[2] We use a large value for the weights instead of infinity to maintain numerical stability.

and thus we can compute the probability of a pixel belongs to the foreground or background based on its color.

We use the min-cut/max-flow algorithm by Boykov and Kolmogorov to compute the figure/ground segmentation boundary [**21**]. The algorithm finds the minimum cut between the nodes in the graph. This corresponds to the edges that is likely to be the person's silhouette boundary. Sometimes, label refinement is needed, so the user marks additional foreground and background pixels and reruns graph cut. After a few iterations, an accurate silhouette is extracted, as shown in Figure 6.17.

9. Timing Experiments

Our system was designed with recognition accuracy as a high priority. Run time performance is also important in production systems. Our system does not run in real time, but there are many opportunities for improvement. We give timing results on various components of the system to show where one might achieve the most performance gains during optimization. Figure 6.18 shows the amount of time each component takes relative to the length of the video processed. The experiments were done on a computer with a 1.4GHz Intel Pentium®III processor 2GB of RAM, averaged over 7 minutes of video. There are many factors that affect the precise timing of the calculations, such as template size, complexity of the video, or number of parts. Therefore, we focus on the relative speed of each component rather than the precise measurements.

We see that from the table that flow matching takes the most amount of time because it must do correlation in 3D. Region extraction also takes a significant portion of time, but it only needs to be computed once. We can run any number of event detectors on the extracted regions.

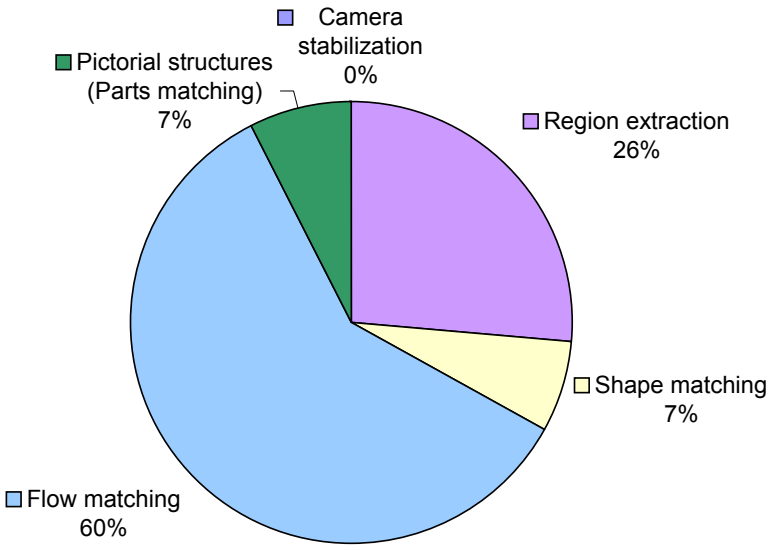
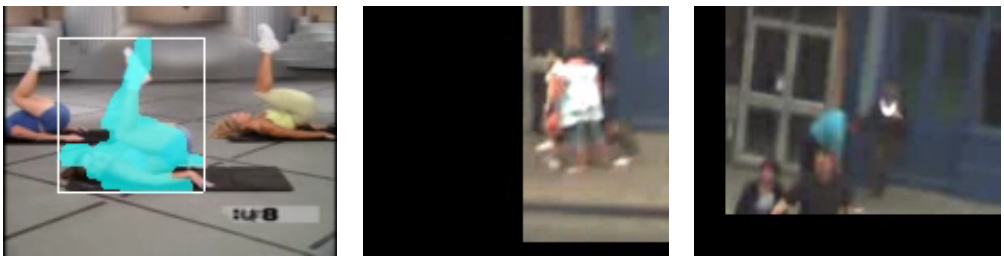


FIGURE 6.18. Timing results. Flow matching currently takes the most amount of time.



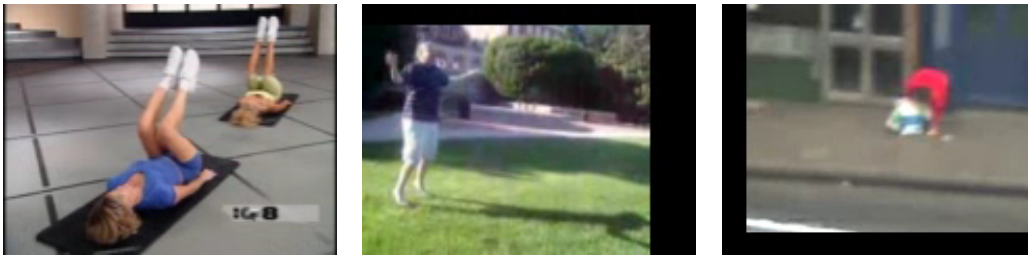
Aerobics Pick-up Pick-up

FIGURE 6.19. Failure cases due to occlusion. In the aerobics video, the left-most person's head falls outside the frame.

10. Limitations

We have discussed some of the failure modes of our system and how we address them in previous robustness experiments. However, it is useful to see examples of where the system fails. Our system is sensitive to the same kind of problems that affect other view-based approaches. Figure 6.19 shows examples of missed detections due to occlusion. Figure 6.20 shows examples where the viewpoint change is more than 45 degrees and therefore the templates fail to detect the events.

Another possible failure case is when the scale (in space or time) of the template does not match the scale of the performed action. We previously addressed



Aerobics Jumping jacks Pick-up

FIGURE 6.20. Failure cases due to viewpoint change.

this issue by scaling the template in both space and time and scanning the video at all scales. The templates were scaled linearly, but the differences between actions in the temporal axis may not be linear. These nonlinear changes raise the question of how to generalize to the human variations in how an action is performed. In another example, walking normally, walking with a limp, and walking with a cane are all technically "walking", but they look different visually. We propose that instead of training for semantic classes, we should train for visual classes, and therefore group these into three different types of events. Each one of them would need to be trained separately.

A more fundamental limitation of view-based approaches is that they work best when the actor is performing full body movements. Since our volumetric shape matching algorithm does not use a discriminative classifier, it does not know which part of the shape discriminates it against all their actions. The distance is measured by the volumetric overlap between the template and the video, and sometimes the discriminative part of the action is dominated by other irrelevant parts of the template. Consider the *pushing-elevator-button* action in the third row of Figure 5.3. The only difference between this action and *standing-still* is the small part of the arm that extends forward. The arm occupies a relatively small volume when compared to the entire template, and therefore its weight contribution to the matching score is also small. Ideally, we would want to give higher weight to the matching score of the arm, and lower weight to the score of the rest of the template. In general, events with larger, full body movements will match better than events with small movements.

CHAPTER 7

Conclusion and Future Directions

In this dissertation, we have proposed flow- and shape-based volumetric features for event detection. Our models are appearance-based and operates directly on the video without the need for background subtraction. Our first approach uses cascaded discriminative classifiers and we built a system for event detection that runs in real time [**79**]. We designed volumetric features that are spatio-temporal extensions of Viola and Jones' box features and operate on the optical flow extracted from the video. The features are efficient to compute, but required many training examples.

Next, we designed volumetric shape features that can operate on over-segmented videos [**81**]. Because we do not require background subtraction, this technique can work in cluttered scenes with dynamic backgrounds. The shape matching algorithm is based on volume intersection between event templates and segmented videos. Because it is based on shape correlation, the system works with as few as one training example.

Finally, we proposed a number of techniques to generalize our matching algorithms and demonstrate their effectiveness on real-world videos. To make the templates more deformable, we divided the templates into parts and matched them using 3D pictorial structures [**80**]. We also used pictorial structures to combine volumetric shape and flow features because they are complementary to each other. The resulting templates can better match differences in speed and scale of event. To further generalize the templates, we used multiple training templates and scaled

them in both space and time to search the videos. Finally, we demonstrated that our system works in videos with large camera movements by using off-the-shelf image stabilization techniques.

1. Future Directions

We have shown the usefulness of using volumetric features for event detection. However, it is only one piece of the puzzle in solving human activity recognition. We describe several broad areas that may be useful for further investigation.

1.1. Speed Optimizations

We have not yet optimized for the run-time performance of our system. For the system to be useful in production systems, it would have to run much faster than its current state. There is already concurrent work in making many parts of the system run in real-time. There are two general directions for speed optimization – algorithmic and parallelization. Because the system was designed for the best achievable accuracy, we did not focus on the run-time performance of the system. We can improve the performance of several components of our system as follows.

Segmentation. We originally chose mean shift because of its reported success and high accuracy in image segmentation. However, other region segmentation algorithms can be used [**10**, **15**, **24**, **40**, **82**, **122**, **124**, **144**, **150**]. If the segmentation algorithm can be significantly faster, we can incorporate additional features for segmentation. Our method only uses color as inputs. It would be useful to add texture and flow information as well. In addition, it might be useful to incorporate other cues into the segmentation process. If we knew the cameras were stationary, we could use background subtraction as an additional feature. Figure 7.1 shows an example where we combine background subtraction with automatic segmentation. Although background subtraction incorporated the shadow into the foreground, the automatic segmentation was able to separate it from the background. This is faster and more accurate than using either method alone.

(a) Video Frame (b) Foreground Mask (c) Over-segmentation

FIGURE 7.1. Combining background subtraction and automatic segmentation. Although background subtraction included the shadow into the foreground, the automatic segmentation was able to separate it from the person.

Although we have advocated using volumetric 3D segmentation for the highest accuracy, segmentation in 2D is significantly faster. One might argue that the speed increase outweighs the performance hit over 3D segmentation. These tradeoffs can be assessed in production systems.

Flow Matching. In the latter part of the thesis, we used Shechtman and Irani's flow consistency measure for flow correlation. While it fits well into our framework, it is one of the slowest parts of our system. S-I optimized their system to run close to real-time by doing a hierarchical decomposition and skipping computation. However, this inevitably reduces accuracy when compared to doing the full computation. We propose two possible methods for correlating flow efficiently.

One way to compute flow correlations uses the volumetric feature framework proposed in Chapter 3. Instead of using the features in a discriminative framework, one could find regions in the template that have similar flow. The entire region's flow can be described with its sum, which can be efficiently computed using integral videos.

We make the observation that flow consistency calculations are simply functions on video gradients. In particular, they are functions on the six-dimensional M matrices as described in Equation 2.4. Instead of computing the function at every pixel, one could quantize the input matrices and perform table look-ups on them to obtain the result directly. We have on-going work in this direction.

Coarse to Fine Search. Coarse to fine search [**11**], also sometimes referred to as hierarchical search [**54**,**88**] or branch and bound [**22**,**28**], is an optimization technique to reduce computation during search. Event detection is an instance of a rare-event detection problem, and therefore only a tiny percentage of locations in the video will contain the event. The goal of coarse to fine search is to use the least amount of computation to discard large number of locations that we are certain will not contain the event. The cascaded classifier used in Chapter 3 is an example of this technique. We can apply similar techniques used by Shechtman and Irani [**119**] or Cour and Shi [**38**] in our task. The idea is to search first search low resolution videos to quickly discard locations with no-events, and then search in progressively higher resolution at locations of possible events.

Parallelization. With the emergence of terascale computing, we are beginning to see massively parallel hardware and software architectures at affordable, commodity prices [**70**]. Parallel execution architectures such as MapReduce can easily distribute tasks to millions of computing nodes [**31**,**43**]. Event detection (and video processing in general) is relatively easy to parallelize. Because the events are isolated (by definition), different parts of the video can be processed independently of each other. The video can be divided in space or time, and each template and template variation can be scanned independently of each other. Parallelization can be used in conjunction with all of the other algorithm speed optimizations.

1.2. Accuracy and Robustness

Active Shape Models. Active shape models and their derivatives have been highly successful in matching deformable shape templates in images [**35**–**37**,**67**,**83**]. They work well because they have learned priors on the shape and they can deform to match the local contours in the image. Local shape refinement can also be used for shape-based event detection. While the prior work on active shape models are too computationally expensive to be used as general search strategies, they could be used to check whether candidate matches are correct. The shape template could be locally deformed and matched against the video to compute and the amount of deformation could be used as a distance measure.

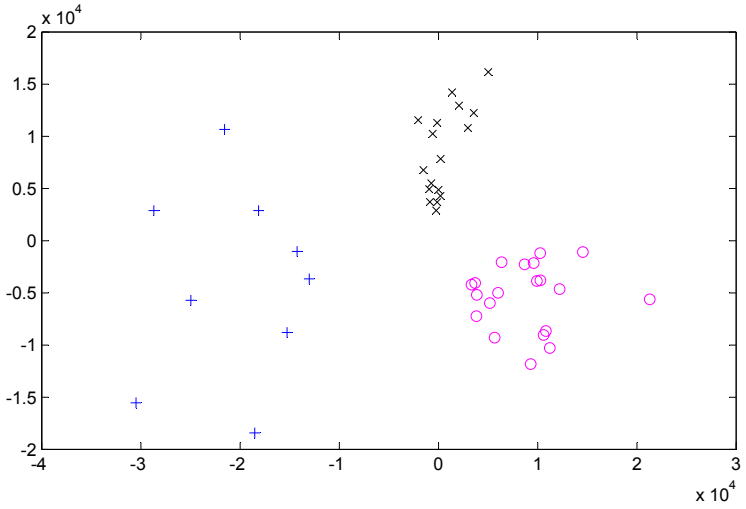


FIGURE 7.2. MDS projection of event shapes. The three event types form three distinct clusters.

Discriminability of Appearance-based Shape Models. Determining the intrinsic discriminative power of our features is an interesting area of research. Blank *et al.* demonstrated that shape invariants can classify the actions in their database very well, but they require background subtraction [**16**]. We would like to know how well the action shapes, based on the region-intersection distance, differ from one another. If they are well separated, then it confirms that this distance metric could work (without relying on background subtraction). Similar studies could be done on combined shape and flow descriptors. Figure 7.2 shows a plot of the distances between three types of events in the Weizmann dataset – *bend*, *jumping-jacks*, and *one-handed wave*. We extracted the individual events from these actions and aligned them in location and scale. We measured the region-intersection distances between all of the events and projected them using multidimensional scaling (MDS) [**39**]. The different types of markers denote different types of actions. We see that the events are well separated into three distinct clusters. As a preliminary experiment, we used medoidshift to cluster these points, shown in Figure 7.3 [**122**].

Exhaustive Grouping of Regions. While there are many approaches to matching 3D shapes using shape invariants [**78**], we can not directly use them for event detection because of the over-segmentation problem. They require the object to

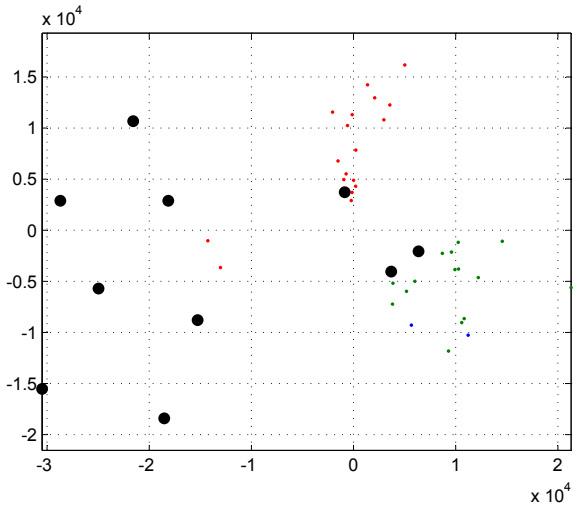


FIGURE 7.3. Medoidshift clustering of event shapes.

be composed of one 3D shape. We could try to exhaustively group the over-segmented regions to match the event template, but this scales exponentially with the number of regions. If segmentation algorithms develop to the point where objects are only slightly over-segmented, exhaustive grouping could work. This would allow us to use more powerful shape matching algorithms for event detection.

1.3. Alternative Classifiers

We have proposed several classifiers for event detection, including a discriminative cascade classifier [**149**], SVM [**136**], nearest neighbor [**64**], and pictorial structures [**53**]. The discriminative cascade is very efficient and results in a real-time event detector, but requires many training examples. The nearest neighbor and pictorial structures classifier can be used with only one training example. To improve their generalizability, we must scan over many scales and train with multiple examples. Unfortunately, these classifiers scale linearly with the number of training examples. We would like to improve both the efficiency and matching performance by sharing features among many training examples [**7**, **50**, **134**]. One aspect of pictorial structures that we did not exploit is the ability to parametrize the

appearance of each part. For example, Felzenszwalb and Huttenlocher used rectangles with parametrized aspect ratios to match the limbs of people [**53**]. Our parts are appearance-based, and therefore have no intrinsic parameters. However, we observe that practical implementations of pictorial structures do not search over continuous part parameters. Instead, they search over a discrete set of labels for efficiency. Therefore, we can potentially mix and match parts from different training examples, similar to the Mr. Potato Head toys. This can increase the generalizability of our training data exponentially while only linearly increasing the computational cost. In addition to shape parts, it would be interesting to build a set of motion primitives as parts as well [**74**, **104**, **138**]. Finding the optimal set of parts and how they can be shared is for future research.

In Chapter 4 Section 7, we used an SVM on the distances to several action templates to classify individual frames. We can further exploit the use of discriminative classifiers as follows. Suppose that we are trying to detect n different events. Our current framework independently compares the distances between a space-time location in the video and event templates. It would be useful to use the distances to all templates in a classifier for each action. The intuition is that some events (one-handed waves) look similar to other events (two-handed waves), and therefore the distances to these events are correlated. We can learn these correlations in a classifier and we could achieve better results. There should be minimal additional computational cost since we must scan for all of the templates and the discriminative classifiers are typically very efficient.

1.4. Integration

Event detection is just one part of the overall goal of human activity recognition. A complete system would need to integrate many components including event detection. We propose that event detection should be combined with both lower-level recognition, such as pose estimation, and higher-level recognition, such as activity recognition. For example, a side effect of detecting a particular event is that we know the actor's pose at every time step [**51**]. By tracking a person's movement through the video, we can considerably reduce the number of locations that

we have to scan to detect an event. Context is an important consideration and Hoiem *et al.* recently demonstrated its usefulness in object detection [**71**]. In addition to geometric context, temporal context can greatly improve the accuracy of the system. There is a lot of prior work on incorporating domain specific knowledge to help in activity recognition [**100**, **101**, **108**, **111**, **128**, **153**, **155**]. For example, after we see a person swing a tennis racket, we are likely to see more racket swings rather than dancing moves. This motivates the need for integration with systems that understand the high level semantics of the scene.

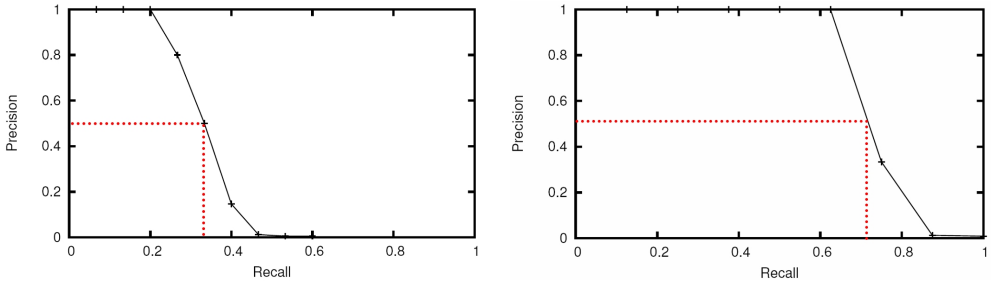
1.5. Applications

In Chapter 1, we discussed a number of applications that can utilize event detection. In this thesis, we explored some of those applications such as sports annotation and surveillance. It would be interesting to extend this and build large scale search applications, for example on the entire YouTube or movie database. This raises a number of interesting application-specific questions such as how to present the search results, how to manage large databases, and how to easily specify training examples? The issue of generating training examples is particularly interesting because while it is easy to type in a particular search term, it would be unwieldy to capture an example of someone performing an action before one can search for it in the database. One might imagine a labeled database of human actions that are available for training. All of these issues will have to be solved for a real deployment of the system.

APPENDIX A

Area Under the Precision-Recall Curve

The receiver operating characteristic (ROC) curve is commonly used for measuring classifier performance. The area under the ROC curve conveniently summarizes the classifier's discriminability with a single number. However, for skewed datasets, where the number of positives and negatives are very different, precision-recall is a better measure of classifier performance [**42**]. Our event-detection task is an example of a skewed dataset, where the number of negatives far outweighs the number of positives. Therefore, we use precision-recall as our main measure and we summarize it with the *area under the precision-recall curve* (AUPRC). The AUPRC enables us to more easily observe trends over many actions. Empirically, we observe that the AUPRC is approximately equal to the recall at 50% precision, as illustrated in Figure A.1. The reader is invited to read Davis and Goadrich's work for a more in-depth view of the relationship between precision-recall and ROC curves [**42**].



(a) AUPRC = 0.30 (b) AUPRC = 0.67

FIGURE A.1. The area under the precision-recall curve (AUPRC) is empirically observed to approximately equal the recall at 50% precision.

APPENDIX B

Approximation of Region Intersection Background Model

We would like to simplify and approximate the expected distance between the volume and a random template, as described in Chapter 4 Section 3.3. This gives us a better understanding of this term. Further, we can use the approximation when the exact computation is too expensive for large numbers.

$$E_{\mathcal{T}}[d(\cdot, V)] = \sum_{i=1}^{k} \frac{1}{2^{|V_i|}} \sum_{j=1}^{|V_i|-1} \binom{|V_i|}{j} \min(j, |V_i| - j) \tag{B.1}$$

1. Preliminaries

Stirling's Approximation:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \tag{B.2}$$

Approximation I: Approximating n choose $n/2$. n is even.

$$\binom{n}{n/2} = \frac{n!}{\left(\frac{n}{2}\right)! \left(\frac{n}{2}\right)!} \tag{B.3}$$

$$\sim \frac{\sqrt{2\pi n} n^n}{e^n} \frac{e^{n/2}}{\sqrt{\pi n}\left(\frac{n}{2}\right)^{n/2}} \frac{e^{n/2}}{\sqrt{\pi n}\left(\frac{n}{2}\right)^{n/2}} \tag{B.4}$$

$$= \sqrt{\frac{2}{\pi n}} 2^n \tag{B.5}$$

Identity I:

$$\sum_{1}^{n} \binom{n}{j} j = n2^{n-1} \tag{B.6}$$

Identity II:

$$\binom{n}{j}(n-j) = \frac{n!}{j!(n-j)!}(n-j) \tag{B.7}$$

$$= \frac{n!}{j!(n-j-1)!} \tag{B.8}$$

$$= \frac{n!}{(j+1)!(n-(j+1))!}(j+1) \tag{B.9}$$

$$= \binom{n}{j+1}(j+1) \tag{B.10}$$

Identity III: Suppose n is even.

$$\binom{n}{n/2+1}(n/2+1) = \frac{n!}{(n/2+1)!(n-(n/2+1))!}(n/2+1) \tag{B.11}$$

$$= \frac{n!}{(n/2+1)!(n/2-1))!}(n/2+1) \tag{B.12}$$

$$= \frac{n!}{(n/2)!(n/2)!}(n/2) \tag{B.13}$$

$$= \binom{n}{n/2}(n/2) \tag{B.14}$$

Identity IV: Suppose n is odd.

$$\binom{n}{\frac{n+1}{2}}\frac{n+1}{2} = \frac{n!}{\left(\frac{n+1}{2}\right)!\left(n-\frac{n+1}{2}\right)!}\left(\frac{n+1}{2}\right) \tag{B.15}$$

$$= \frac{n!}{\left(\frac{n+1}{2}\right)!\left(\frac{n-1}{2}\right)!}\left(\frac{n+1}{2}\right) \tag{B.16}$$

$$= \frac{n!}{\left(\frac{n-1}{2}\right)!\left(\frac{n-1}{2}\right)!} \tag{B.17}$$

$$= n\binom{n-1}{(n-1)/2} \tag{B.18}$$

2. Approximation for even n

For simplification of notation, we use n in place of $|V_i|$. We approximate the term inside the outer sum. Suppose n is even, so that $n/2$ is an integer.

$$\frac{1}{2^n}\sum_{j=1}^{n-1}\binom{n}{j}\min(j,n-j) \tag{B.19}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{n/2}\binom{n}{j}j+\sum_{n/2+1}^{n-1}\binom{n}{j}(n-j)\right] \tag{B.20}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{n/2}\binom{n}{j}j+\sum_{n/2+1}^{n-1}\binom{n}{j+1}(j+1)\right], \text{ by Identity II} \tag{B.21}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{n/2}\binom{n}{j}j+\sum_{n/2+2}^{n}\binom{n}{j}j\right] \tag{B.22}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{n}\binom{n}{j}j-\binom{n}{n/2+1}(n/2+1)\right] \tag{B.23}$$

$$= \frac{1}{2^n}\left[n2^{n-1}-\binom{n}{n/2+1}(n/2+1)\right], \text{ by Identity I} \tag{B.24}$$

$$= \frac{1}{2^n}\left[\frac{n}{2}2^n-\binom{n}{n/2}(n/2)\right], \text{ by Identity III} \tag{B.25}$$

$$\sim \frac{1}{2^n}\left[\frac{n}{2}2^n-\sqrt{\frac{2}{\pi n}}2^n(n/2)\right], \text{ by Approximation I} \tag{B.26}$$

$$= \frac{1}{2^n}\left[\frac{n}{2}2^n-\sqrt{\frac{n}{2\pi}}2^n\right] \tag{B.27}$$

$$= \frac{n}{2}-\sqrt{\frac{n}{2\pi}} \tag{B.28}$$

3. Approximation for odd n

Now, suppose n is odd, so that $(n-1)/2$ and $(n+1)/2$ are integers. The approximation is very similar to the even case.

$$\frac{1}{2^n}\sum_{j=1}^{n-1}\binom{n}{j}\min(j,n-j) \tag{B.29}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{(n-1)/2}\binom{n}{j}j + \sum_{(n+1)/2}^{n-1}\binom{n}{j}(n-j)\right] \tag{B.30}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{(n-1)/2}\binom{n}{j}j + \sum_{(n+1)/2}^{n-1}\binom{n}{j+1}(j+1)\right], \text{ by Identity II} \tag{B.31}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{(n-1)/2}\binom{n}{j}j + \sum_{(n+1)/2+1}^{n}\binom{n}{j}(j)\right] \tag{B.32}$$

$$= \frac{1}{2^n}\left[\sum_{1}^{n}\binom{n}{j}j - \binom{n}{(n+1)/2}(n+1)/2\right] \tag{B.33}$$

$$= \frac{1}{2^n}\left[n2^{n-1} - \binom{n}{(n+1)/2}(n+1)/2\right], \text{ by Identity I} \tag{B.34}$$

$$= \frac{1}{2^n}\left[\frac{n}{2}2^n - n\binom{n-1}{(n-1)/2}\right], \text{ by Identity IV} \tag{B.35}$$

$$\sim \frac{1}{2^n}\left[\frac{n}{2}2^n - n\frac{2^{n-1}}{\sqrt{\pi(n-1)/2}}\right], \text{ by Approximation I} \tag{B.36}$$

$$\sim \frac{1}{2^n}\left[\frac{n}{2}2^n - \sqrt{\frac{n}{2\pi}}2^n\right] \tag{B.37}$$

$$= \frac{n}{2} - \sqrt{\frac{n}{2\pi}} \tag{B.38}$$

4. Combined Approximation

Combining the two cases, we have

$$E_{\mathcal{T}}[d(\cdot,V)] \sim \sum_{i=1}^{k}\frac{|V_i|}{2} - \sqrt{\frac{|V_i|}{2\pi}} \tag{B.39}$$

Notice that for large $|V_i|$, $|V_i|/2$ dominates $\sqrt{|V_i|/(2\pi)}$, and therefore the entire sum approaches $|V|/2$.

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] Open Computer Vision Library, 2008. `http://sourceforge.net/projects/opencv/`.

[4] Wikipedia: Youtube, 2008. `http://en.wikipedia.org/w/index.php?title=YouTube&oldid=184514165`.

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

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

[7] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of International Conference on Machine Learning*, 2007.

[8] 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.

[9] 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.

[10] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *Proc. ICCV*, 2007.

[11] A. Bandapadhay and J. L. Fu. Searching parameter spaces with noisy linear constraints. In *Proc. CVPR*, 1988.

[12] H. G. Barrow, J. M. Tenenbaum, R. Bolles, and H. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. IJCAI*, 1977.

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

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

[15] S. Beucher. The watershed transformation applied to image segmentation. In *10th Pfefferkorn Conference on Signal and Image Processing in Microscopy and Microanalysis*, 1992.

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

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

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

[19] E. Borenstein and J. Malik. Shape guided object segmentation. In *Proc. CVPR*, 2006.

[20] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *PAMI*, 10(6), 1988.

[21] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9), 2004.

[22] T. M. Breuel. A comparison of search strategies for geometric branch and bound algorithms. In *Proc. ECCV*, 2002.

[23] J. Canny. A computational approach to edge detection. *PAMI*, 8(6), 1986.

[24] M. A. Carreira-Perpinan. Acceleration strategies for gaussian mean-shift image segmentation. In *Proc. CVPR*, 2006.

[25] 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`.

[26] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 1999.

[27] D. Chen and J.-M. Odebez. Video text recognition using sequential Monte Carlo and error voting methods. *Pattern Recognition Letters*, 26(9), 2005.

[28] J.-Y. Chen, C. A. Bouman, and J. P. Allebach. Multiscale branch-and-bound image database search. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 133–144, 1997.

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

[30] C. M. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. In *Proc. ICPR*, 2002.

[31] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, and A. Y. Ng. Map-Reduce for machine learning on multicore. In *NIPS*, 2006.

[32] R. Collins. Mean-shift blob tracking through scale space. In *Proc. CVPR*, June 2003.

[33] D. Comaniciu. An algorithm for data-driven bandwidth selection. *PAMI*, 25(2), 2003.

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

[35] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. ECCV*, 1998.

[36] T. F. Cootes and C. J. Taylor. Active shape models - 'smart snakes'. In *Proc. BMVC*, 1992.

[37] T. F. Cootes, C. J. Taylor, and D. H. Cooper. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1), 1995.

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

[39] M. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and hall, 2001.

[40] D. Cremers and S. Soatto. Variational space-time motion segmentation. In *Proc. ICCV*, 2003.

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

[42] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of International Conference on Machine Learning*, 2006.

[43] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of Symposium on Operating System Design and Implementation*, 2004.

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

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

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

[47] P. Dollar, T. Zhuowen, and S. Belongie. Supervised learning of edges and object boundaries. In *Proc. CVPR*, 2006.

[48] G. Dorko and C. Schmid. Selection of scale invariant neighborhoods for object class recognition. In *Proc. ICCV*, 2003.

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

[50] B. Epshtein and S. Ullman. Semantic hierarchies for recognizing objects and parts. In *Proc. CVPR*, 2007.

[51] A. Fathi and G. Mori. Human pose estimation using motion exemplars. In *Proc. ICCV*, 2007.

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

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

[54] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *Proc. CVPR*, 2007.

[55] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. CVPR*, volume II, pages 264–270, 2003.

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

[57] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. In *IEEE Computing Magazine*, 1995.

[58] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, 21(1), 1975.

[59] 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.

[60] T. Gautama and M. Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Trans. Neural Networks*, 13(5):1127–1136, 2002.

[61] D. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proc. ICPR*, 1998.

[62] D. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. ECCV*, 2000.

[63] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A texture classification example. In *Proc. ICCV*, 2003.

[64] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinovo. Neighbourhood component analysis. In *NIPS*, 2004.

[65] L. Gomes. Will all of us get our 15 minutes on a youtube video? *The Wall Street Journal*, 2006.

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

[67] L. Gu, E. P. Xing, and T. Kanade. Learning GMRF structures for spatial priors. In *Proc. CVPR*, 2007.

[68] 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.

[69] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. In *Proc. CVPR*, 2001.

[70] J. Held, J. Bautista, and S. Koehl. From a few cores to many: A tera-scale computing research overview. Technical report, Intel Corporation, 2006.

[71] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. CVPR*, 2006.

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

[73] P. Indyk and R. Motwani. Approximate nearest neighbor – towards removing the curse of dimensionality. In *Proceedings of Symposium on Theory of Computing*, 1998.

[74] O. C. Jenkins and M. J. Mataric. Deriving action and behavior primitives from human motion data. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2002.

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

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

[77] M. Jones and P. Viola. Face recognition using boosted local features. In *Proc. ICCV*, 2003.

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

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

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

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

[82] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *Proc. CVPR*, 2001.

[83] A. Koschan, S. Kang, J. Paik, B. Abidi, and M. Abidi. Color active shape models for tracking non-rigid objects. *Pattern Recognition Letters*, 24(11), 2003.

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

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

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

[87] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Proc. CVPR*, 2005.

[88] W. H. Leung and T. Chen. Hierarchical matching for retrieval of hand-drawn sketches. In *Proc. ICME*, 2003.

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

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

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

[92] 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.

[93] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5), 2004.

[94] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, 2001.

[95] O. Masoud and N. P. Papanikolopoulos. A method for human action recognition. *Image and Vision Computing*, 21(8), 2003.

[96] M. McCahill and C. Norris. *CCTV*, chapter Estimating the extent, sophistication and legality of CCTV in London. Perpetuity Press, 2003.

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

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

[99] P. Meer and B. Georgescu. Edge detection with embedded confidence. *PAMI*, 2001.

[100] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *American Association for Artificial Intelligence*, 2002.

[101] D. J. Moore, I. A. Essa, and M. H. Hayes III. Exploiting human actions and object context for recognition tasks. In *Proc. ICCV*, 1999.

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

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

[104] F. Nori and R. Frezza. Nonlinear control by a finite set of motion primitives. In *IFAC Symposium on Nonlinear Control Systems*, 2004.

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

[106] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1), 1997.

[107] M. J. Pickering, S. M. Rger, and D. Sinclair. Video retrieval by feature learning in key frames. In *Proceedings of International Conference on Image and Video Retrieval*, 2002.

[108] R. Polana and R. Nelson. Detection and recognition of periodic, nonrigid motion. *IJCV*, 1997.

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

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

[111] N. Rea, R. Dahyot, and A. Kokaram. Semantic event detection in sports through motion understanding. In *Proceedings of International Conference on Image and Video Retrieval*, 2004.

[112] E. Remy and E. Thiel. Computing 3D medial axis for chamfer distances. In *Discrete Geometry for Computer Imagery*, 2000.

[113] E. Remy and E. Thiel. Optimizing 3D chamfer masks with norm constraints. In *Int. Workshop on Combinatorial Image Analysis*, 2000.

[114] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. ICCV*, 2003.

[115] C. Rosenberg. *Semi-Supervised Training of Models for Appearance-Based Statistical Object Detection Methods*. Ph.D. Thesis, Carnegie Mellon University, 2004.

[116] W. Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. Springer, 1991.

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

[118] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele. An evaluation of local shape-based features for pedestrian detection. In *Proc. BMVC*, 2005.

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

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

[121] 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.

[122] Y. Sheikh, E. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *Proc. ICCV*, 2007.

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

[124] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proc. ICCV*, pages 1154–1160, 1998.

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

[126] J.-C. Shim, C. Dorai, and R. Bolle. Automatic text extraction from video for content-based annotation and retrieval. In *Proc. ICPR*, 1998.

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

[128] J. R. Smith, M. Campbell, M. Naphade, A. Natsev, and J. Tesic. Learning and classification of semantic concepts in broadcast video. In *Proceedings of International Conference on Intelligent Analysis*, 2005.

[129] A. N. Stein and M. Hebert. Using spatio-temporal patches for simultaneous estimation of edge strength, orientation, and motion. In *Beyond Patches*

Workshop at IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[130] K. Sung and T. Poggio. Example-based learning for view-based human face detection. *PAMI*, 20(1), 1998.

[131] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. CVPR*, 2003.

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

[133] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. CVPR*, 2004.

[134] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29(5), 2007.

[135] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[136] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

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

[138] D. D. Vecchio1, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics based primitives with application to drawing tasks. *Automatica*, pages 2085–2098, 2003.

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

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

[141] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proc. ICCV*, 2003.

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

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

[144] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *The IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625–638, September 1994.

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

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

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

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

[149] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *NIPS*, 2002.

[150] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *Proc. CVPR*, 2004.

[151] C. Yang, R. Duraiswami, D. DeMenthon, and L. Davis. Mean-shift analysis using quasi-newton methods. In *Proceedings of The International Conference on Image Processing*, 2003.

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

[153] 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.

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

[155] Z. Zivkovic, F. van der Heijden, M. Petkovic, and W. Jonker. Image processing and feature extraction for recognizing strokes in tennis game videos. In *Proceedings of the Annual Conference of the Advanced School for Computing and Imaging*, 2001.