TemporalBoost for Event Recognition

Paul Smith, Niels da Vitoria Lobo, and Mubarak Shah
Computer Vision Lab, School of Computer Science, University of Central Florida
{rsmith,niels,shah}@cs.ucf.edu

Abstract

This paper contributes a new boosting paradigm to achieve detection of events in video. Previous boosting paradigms in vision focus on single frame detection and do not scale to video events. Thus new concepts need to be introduced to address questions such as determining if an event has occurred, localizing the event, handling same action performed at different speeds, incorporating previous classifier responses into current decision, using temporal consistency of data to aid detection and recognition. The proposed method has the capability to improve weak classifiers by allowing them to use previous history in evaluating the current frame. A learning mechanism built into the boosting paradigm is also given which allows event level decisions to be made. This is contrasted with previous work in boosting which uses limited higher level temporal reasoning and essentially makes object detection decisions at the frame level. Our approach makes extensive use of temporal continuity of video at the classifier and detector levels. We also introduce a relevant set of activity features. Features are evaluated at multiple zoom levels to improve detection. We show results for a system that is able to recognize 11 actions.

1. Introduction

Activity recognition is an active research area in computer vision, and there has been an increasing amount of research done in this field in recent years [2] [11] [24]. The task of detecting and localizing events in video is a challenging problem. There are many different contexts where activity recognition would play a major role. Applications ranging from surveillance to Homeland Security to multimedia retrieval rely on a robust method to detect actions in video. We focus our attention on the problem of action recognition in an office environment as this gives us a rich set of actions to recognize. Though boosting paradigms have been gaining popularity they are not well studied for recognition of video events.

A number of difficulties arise when using a boosted learning framework to recognize video events. How do we determine that an event has occurred? How do we localize the event in time? Is there a way to deal with temporal variation of the same action performed multiple times. How should the “jump” from single frame object recognition to video data event detection occur. Is there any way to use the temporal continuity properties of video, so as to be able to use previous feature responses in evaluating the current frame? Another problem stems from the fact that in object detection the objects can be sized and normalized so that they are essentially aligned with one another; this makes feature design easier as the features all operate on data at the same scale. In the context of activities, it is not clear how one would normalize the activity to facilitate comparison. Would this normalization occur in time, space, illumination or all of the above? More fundamentally, in an approach such as AdaBoost, during training and testing each image is independent of the others. However in video data, if a face was viewed in one frame, it is likely it would be in the next if it was a true positive. AdaBoost should be able to decide which weak classifiers can increase their detection rate when allowed to use their own individual histories. Though a few boosting methods do operate on video data to conduct tasks such as classification, they do not use the temporal continuity of video at the weak and strong classifier levels.

Our contributions are in extending the boosting paradigm of machine learning to address the above limitations with respect to detection of video events. First we present a variation of AdaBoost, named TemporalBoost, that allows features to rely on previous frames to make a decision in the current frame. Further TemporalBoost automatically learns the optimal number of frames needed to recognize each event while detecting as few false positives as possible. Second, to detect and localize events in video one must either build specific classifiers that detect beginning and endings of events or group frame wise decisions after individual classifiers have been built. We use the latter approach which results in an additional layer of learning once the strong classifiers are built. Our extension allows both for detection and localization of actions in video.

The third contribution is in designing a set of features which is useful for activity recognition in an office environment. In the context of object detection, comparison of feature responses was simple after image normalization. It is unclear how to normalize events, thus we have chosen the

ID	Name
a_1	Talking on phone
a_2	Checking voicemail on phone
a_3	Bringing cup to face
a_4	Scratching/Rubbing face
a_5	Resting hand on face
a_6	Taking medication
a_7	Yawning with hand at mouth
a_8	Yawning with no hand at mouth
a_9	Putting on eyeglasses
a_{10}	Putting on earphones
a_{11}	Rubbing eyes

Table 1: Actions recognized by system

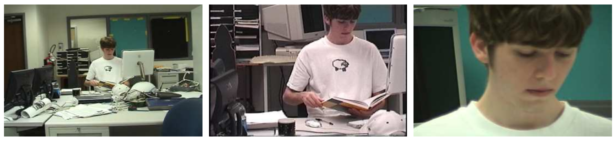

Figure 1: Example of scene showing zooms 1, 2, and 3.

alternate path of more complex feature design. The features are evaluated simultaneously at multiple zooms taken from more or less the same viewpoint. Interestingly, many events rely on features evaluated at multiple zooms. An example of a single image from each zoom is shown in Figure 1.

We show an application of detecting events in an office environment recognizing 11 events with good accuracy. The target actions to be recognized are listed in Table 1. A visual sample of each action is given in Figure 2. Many of the events we recognize are similar which makes the problem quite challenging. Medication, drinking, and yawning with hand are all very similar. Our training process was the following: label all training videos; compute all individual feature responses (Section 3); expand the features in scale, position, and parameter space (Section 3); find best threshold for each weak classifier; use TemporalBoost to find strong classifier for each action and its optimal temporal window (Section 2).

1.1. Related Work

This research touches on many aspects of activity recognition, so we review previous work in the following areas: AdaBoost Learning, Activity recognition and event representation. AdaBoost was developed first in [10]. Work in [25] generated much interest in the computer vision community, and there have been many improvements to AdaBoost, such as FloatBoost [14]. Recently many interesting applications have also emerged, among those [18]. These systems all make a decision in an object detection context. There has been some recent work in using AdaBoost for speech recognition: In [6], a unique training approach using AdaBoost and HMMs to sequence learning is proposed; Research in [13] also develops an AdaBoost framework to improve recognition of sequence data. In [27] AdaBoost is used for automatic visual feature formation to boost HMMs for speech recognition. However in most of these speech based methods the features are taken to be averages over some N frame window. This is not good for localization. In [4] a method is presented for facial expression analysis using Adaboost. However the method specifically trains on only two frames for each facial expression (a neutral expression and a frame during the facial expression). That work essentially does not use video data for training as our approach does. Temporal features were introduced in [26]. The features were designed to operate on two frames, though temporal information is not used after feature design. The above methods do not discover inherent temporal dependencies (if they exist) both between the classifier responses and between the feature level responses.

In order to recognize activities much previous work has utilized point trajectories or contours of the objects in question [20]. Work in [12] and [17] focuses on large-scale activities and makes use of both object and trajectory properties of objects in question. In [8] a motion descriptor based on optical flow measurements is used to classify activities at low resolution. HMM's have also been widely used to recognize activities [5] [16], though the large number of features in our context might not be suitable to such an approach. Related work can also be seen in the context of video summarization [28]. Other work in activity recognition focuses on detailed views of persons and faces[1]. A variety of facial expression analysis methods are explored in [7]. We seek to employ features at both the coarse and fine level to recognize a broader class of activities.

The work in [21] relies on a representational framework for actions using various logical and temporal constraints. Results are shown on detailed views of hands to analyze actions. Other approaches in representational models for activities include [19], [12], [15], and [3]. We do not focus on event representations in our work in this paper.

[22] gave a method to automatically track the head and hands across cameras with different zoom. We employ [22] to acquire hands and head tracks in all views, which we require in the development of the activity recognition features.

The rest of the paper is organized as follows: The machine learning algorithm is presented in Section 2; Details on the features used are presented in Section 3; Results are presented in Section 4; and finally we conclude.

2. TemporalBoost Learning

We first present an overview and some intuition to TemporalBoost. In the TemporalBoost procedure, when choos-

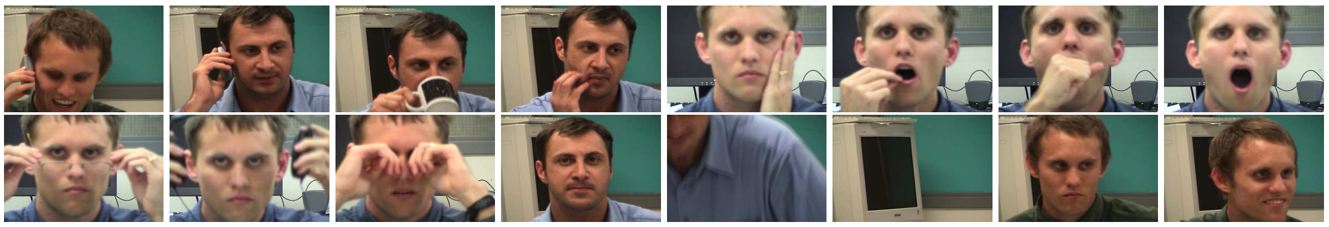

Figure 2: Visual Sample of Target Actions. They are shown in the same order as they are listed in Table 1. From left to right and top to bottom. The five figures in the lower row right corner show some “non-action” events.

ing the best weak classifier for a given stage we modify the boosting process to allow the weak classifier to use its previous classifier responses (previous frames in the temporal sense) if it helps decrease the overall error for this classifier. There are many ways to use previous weak classifier response data. We consider perhaps the two simplest. These allow a weak classifier to respond positively for the current frame if 1) any of the previous t frames were classified as being of the positive class (OR operation) or 2) if all of the previous t frames were classified as being of the positive class (AND operation). Case 1 will allow more true positives at the expense of possibly allowing false positives. Case 2 will reduce false positives at the expense of possibly reducing true positives. Other functions could be considered. For intuition, suppose a feature classifies an action correctly in the previous n frames, but misclassifies the action in the current frame. TemporalBoost allows this feature to classify the current frame correctly based on the fact that the previous n frames classified the action correctly if the overall error was decreased. n is learned automatically; if $n = 0$ it means previous temporal information did not help this feature.

We refer to this step as the DiscoverTemporalDependence step. It has a twofold effect. First it automatically allows for different features to respond in different ways to the input, allowing each feature to use as much temporal information as it can while minimizing its error for the current boosting iteration. Second, it allows temporal smoothing to be embedded in the boosting process. Our search is influenced by the current weights and is different than giving all features temporal scale.

Video events happen over multiple frames, but the standard boosting framework makes a decision as to whether an event is happening in every frame. Our second algorithmic extension to AdaBoost, provides the necessary extensions to learn the allowable variation in action length while keeping the false positive rate as low as possible. Suppose that the yawning event is occurring but for the first x frames of the event, both the yawning and medication classifiers respond that their event is occurring (or only the medication classifier in the multiclass case). Suppose after $x + \delta$ frames only the yawning classifier responds. Suppose that when the medication event does happen the medication classifier responds for $y \gg x$ frames. We encountered this phenomenon frequently in training. If we require that the medication classifier responds for at least y frames, it would prevent the same action from being recognized if it were performed faster in the testing data. Further, if we allow the medication classifier to respond after x frames we would get many false positives. Instead the minimum number of frames needed to achieve high true positive and high true negative rates should be used. This learning process occurs after the classifiers are built. We refer to this step as the LearnEventVariation step. The full algorithm is in Table 2. The algorithm in Table 2 is the one-against-all algorithm. Given a target action (class), a, the algorithm will learn a strong classifier that can recognize the action. Aseparate classifier for each target action is built.

By using multiple one-against-all classifiers two events occurring simultaneously can be recognized, which is not possible in a single multiclass classifier. Nonetheless, the method has been implemented and tested on both the one-against-all and the multiclass training approach. Similar results were obtained in both cases. Due to space limitations we leave out the adaptations to **AdaBoost.M1**.

For clarity, let us point out that in steps 4.b and 4.d the term $[z_i = z_{i-k}]$ is a 1/0 binary predicate that avoids temporal coherence being exploited across boundaries of training videos. Once the optimal k is found for action a in step 6, detection and localization of this action in video is straightforward. We emphasize that each action a can have a different optimal temporal extent k. The action starts when the action is present for k frames and ends when this contiguous detection ends. By keeping the final decision rule simple it allows us to focus on the lower level task of activity detection rather than on how to combine frame responses, which is a higher level task needed at the semantic level. Of course in individual frames multiple classifiers can respond positively, but in order for our method to declare an event is occurring, the classifier for action a has to fire for k frames. Rarely do two classifiers fire in the same time frame. It is handled by taking the action with the maximum classifier response.

Table 2: TemporalBoost Learning Algorithm

For a given class a:

1. Given labeled video data $(x_1^a, y_1^a, z_1^a), \dots, (x_n^a, y_n^a, z_n^a)$: x_i^a represents one video frame. Its label is $y_i^a = 0, 1$. z_i^a is the unique video sequence index. The z_i^a label allows multiple training videos to be concatenated and trained together. All frames from the same video have the same sequence index.
    - (a) Build the set Ω_a for the a^{th} event category by collecting the contiguously labeled frames as events to obtain the start and end of each event in the video
    - (b) Each $\omega \in \Omega_a$ has ω_s and ω_e indicating the starting and ending frames for this event.
2. Set $w_{i,1}^a = \frac{1}{2m^a}, \frac{1}{2l^a}$ for y_i^a=0,1 where m^a, l^a are the number of negatives and positives respectively.
3. $\forall i$ Set $h_i^{a,t} = 0$. In this notation h_i is the weak classifier that corresponds to f_i, h_i^a indicates this classifier is for action a, and $h_i^{a,t}$ indicates the temporal extent of the feature, which is initialized to zero.
4. For $s = 1, \dots, S$:
    - (a) Normalize weights $$w_{i,s}^a \leftarrow \frac{w_{i,s}^a}{\sum_{j=1}^n w_{j,s}^a}$$

DiscoverTemporalDependence (Steps 4b-4e)

- (b) For each feature, j, train a classifier h_j^a which is restricted to using a single feature.

$$\epsilon_j^a = \min\left(\sum_i w_i^a \left|\left\lceil \frac{1}{h_j^{a,t}+1}\sum_{k=0}^{h_j^{a,t}} h_j^a(x^a_{i-k\cdot[z_i=z_{i-k}]})\right\rceil - y_i^a\right|, \sum_i w_i^a \left|\left\lfloor \frac{1}{h_j^{a,t}+1}\sum_{k=0}^{h_j^{a,t}} h_j^a(x^a_{i-k\cdot[z_i=z_{i-k}]})\right\rfloor - y_i^a\right|\right)$$

- (c) set $\epsilon_{prev}^a \leftarrow \epsilon_j^a$
- (d) while $\epsilon_{prev}^a \geq \epsilon_j^a$
    - i. $\epsilon_{prev}^a \leftarrow \epsilon_j^a$, $h_j^{a,t} \leftarrow h_j^{a,t} + 1$
    - ii. $$\epsilon_j^a = \min\left(\sum_i w_i^a \left|\left\lceil \frac{1}{h_j^{a,t}+1}\sum_{k=0}^{h_j^{a,t}} h_j^a(x^a_{i-k\cdot[z_i=z_{i-k}]})\right\rceil - y_i^a\right|, \sum_i w_i^a \left|\left\lfloor \frac{1}{h_j^{a,t}+1}\sum_{k=0}^{h_j^{a,t}} h_j^a(x^a_{i-k\cdot[z_i=z_{i-k}]})\right\rfloor - y_i^a\right|\right)$$
    - iii. if $\epsilon_{prev}^a \geq \epsilon_j^a$ goto step 4.d
    - iv. else $\epsilon_j^a \leftarrow \epsilon_{prev}^a$, $h_j^{a,t} \leftarrow h_j^{a,t} - 1$ goto step 4.e
- (e) Choose the classifier, h_s^a with the lowest error ϵ_s^a
- (f) Update weights

$$w_{i,t+1}^a = w_{i,t}^a \beta_s^{(1-e_i^a),a} \text{ where } e_i^a = 0 \text{ if example } x_i^a \text{ is classified correctly, } e_i^a = 1 \text{ otherwise, and } \beta_s^a = \frac{\epsilon_s^a}{1-\epsilon_s^a}$$

5. The final strong classifier is $h^a(x) = \begin{cases} 1 & \sum_{s=1}^S \alpha_s^a \Phi_s^a\left(\frac{1}{h_s^{a,t}+1}\sum_{k=0}^{h_s^{a,t}} h_s^a(x_{i-k}^a)\right) \geq \frac{1}{2}\sum_{s=1}^S \alpha_s^a \\ 0 & \text{otherwise} \end{cases}$

   where $\alpha_s^a = \log\frac{1}{\beta_s^a}$ and $\Phi_s^a(x)$ is evaluated either as $\lceil x \rceil$ or $\lfloor x \rfloor$ depending on whichever resulted in the lower error term ϵ_s^a for the correspondingly selected weak classifier during stage s.

LearnEventVariation (Steps 6a-6b):

6. After the strong classifier has been built, for action a.
    - (a) Run the strong classifier over the training data to obtain the candidate labeling. We refer to this set of candidate activities as $\overline{\Omega}_a$. By grouping contiguously labeled frames then each $\overline{\omega} \in \overline{\Omega}_a$ has $\overline{\omega}_s$ and $\overline{\omega}_e$ indicating the starting and ending frames for this event.
    - (b) compute $\operatorname*{argmin}_k\ error_a(k)$ where $error_a(k) = \sum_{\overline{\omega} \in \overline{\Omega}_a} [\overline{\omega}_e - \overline{\omega}_s > k] \cdot \Psi(\omega, \overline{\omega})$

      $[x]$ is a true/false predicate that evaluates to 1 or 0, respectively and

$$\Psi(\omega, \overline{\omega}) = \begin{cases} -1 & \omega_s \leq \overline{\omega}_s \leq \omega_e || \omega_s \leq \overline{\omega}_e \leq \omega_e || \overline{\omega}_s \leq \omega_s \leq \overline{\omega}_e \\ 1 & \text{otherwise} \end{cases}$$

3. Guide to Building Features

Our features were selected by first intuitively determining what constituted each event. For instance, event a_3 requires a hand to come to the face, with an object, and bring the object to the mouth region. Further the mouth might open as the event is taking place. From this high level analysis for each action we then built features that specialized in responding positively for each of these sub tasks. These features compute higher level semantics. Generally, all the features compute one of the following: 1) Determining if an object is in the hand, 2) Determining where on the face an action is occurring (i.e., a phone would go to the ear region, whereas a cup would go to the mouth region), 3) Determining how many hands are involved in the action, 4)Determining if the face is moving, 5) Determining if the mouth is opening. These 5 higher level semantics are referred to in Table 3 under the heading Purpose.

The whole idea behind boosting is that we can feed it numerous weak classifiers and AdaBoost will select those that are best. Some of our features might seem inappropriate, but AdaBoost itself will discover which features are relevant and which are not. There is no harm in giving AdaBoost an overabundance of weak classifiers. It is with this philosophy that we designed our features. This is compatible with the original AdaBoost proposal [10].

Most boosting approaches normalize size of the training images so that the whole training image contains only the object to be detected (for a positive example). This allows features to operate directly on the pixel data. In videos it is unclear how to perform such normalization since different parts of the image are needed simultaneously. This has prevented us from using such Haar-like features.

We now explain how to compute the features. We started out with more features than these, but our boosting algorithm selected a subset of the initial weak classifiers. Features h_1-h_3 were not selected in training. We describe them to show the kinds of feature that were not selected. Features h_1-h_3 analyze various properties of background subtraction artifacts. We use 1)difference pictures 2)[23], and 3) [9] to acquire foreground images D_1, D_2, and D_3. For simplicity let us consider one particular zoom (or camera) C_a. Connected components are $(B_{1,1}, B_{2,1}, \ldots, B_{N_1,1})$, $(B_{1,2}, B_{2,2}, \ldots, B_{N_2,2})$, and $(B_{1,3}, B_{2,3}, \ldots, B_{N_3,3})$ for D_1, D_2, and D_3. For each triple $B_{i,1}, B_{j,2}, B_{k,3}$ compute the corresponding centroids $[x_1 \quad y_1]^T, [x_2 \quad y_2]^T, [x_3 \quad y_3]^T$. Size of each is $S(B_{i,j})$.

$\mathbf{h_1}$: The spatial agreement is computed as

$$e = (|[x_1 \quad y_1]^T - [x_2 \quad y_2]^T| + |[x_1 \quad y_1]^T - [x_3 \quad y_3]^T| + |[x_2 \quad y_2]^T - [x_3 \quad y_3]^T|)/3 \quad (1)$$

This gives an idea on how to compute the features. In order to present them in a more concise manner we present the remaining features in Table 3. We give the feature ID in Column 1. Column 2 gives the purpose of this feature. Column 3 contains a high level description of the feature. Column 4 gives the steps to compute each feature.

α and β are size constraints. These values are not hard coded because we will have multiple features with varying values for α and β. In this way AdaBoost itself can select which values of α and β work best.

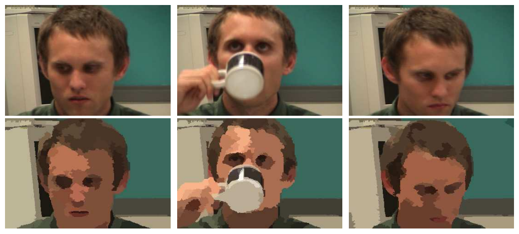

Figure 3: Top row: Sequence showing person drinking from a mug. Bottom row: Mean shift segmentation performed on the above three frames. The idea of this feature is to count the number of segments in each given region. For this example computation the region is the whole image. Frames 2854, 2897, and 2990 are shown.

In Figure 3 three input frames and the corresponding mean shift segmentations are shown. The plot over time of the mean shift segments in this sequence is shown in Figure 4. One can see how when the object comes into view (near frame 2890) the number of mean shift segments spikes up. We do not claim to have the optimal feature set, but they are a reasonable effort at designing activity features.

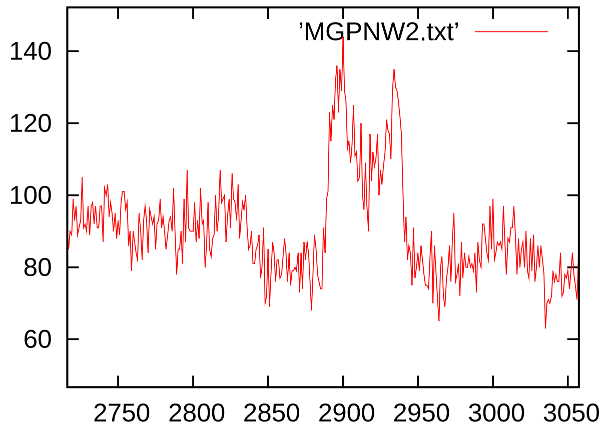

Figure 4: Plot of the number of mean shift segments in the images of Figure 3. The jump in mean shift segments occurs around frame 2897, when the object is near the face.

ID	Purpose	What Feature is Computing	Computational Steps Required
h_1	1	Spatial artifact agreement	Computed above
h_2	1	Relative size agreement	$S(B_{i,1}) - S(B_{j,2}) + S(B_{i,1}) - S(B_{k,3}) + S(B_{i,2}) - S(B_{j,3})$
h_3	1	Absolute size agreement	$\sum_i (S(B_{l,i}) > \alpha) \cdot (S(B_{l,i}) < \beta$
h_4	1	Artifact distance to face	Distance from each artifact centroid to the face
h_5	1,2	Number edges in region R	Count number of edges in R
h_6	2	Percent hand head overlap	(area of intersection)/(area of head bounding box)
h_7	1,2,4,5	Number moving pixels in R	$\sum_{x \in R} [I(x) - I'(x) > \alpha_0]$
h_8	1,2	Number moving pixels in R of specified color, C	$\sum_{x \in R} [I(x) - I'(x) > \alpha_0] \cdot [RGB(x) \in C]$
h_9	3	Distance hand to head	$e^{-\frac{d}{2\sigma^2}}$, where d is the distance between the head and hand
h_{10}	3	Distance both hands to head	$e^{-\frac{d_1+d_2}{2\sigma^2}}$ where d_1, d_2 are the distance between each hand and the head
$h_{11} - h_{13}$	2	Percentage overlap sides of face	computes what percentage the hand overlaps each side of the face
h_{14}	1	Number Mean Shift Segments in R	Count the number of unique segments that occur in the given region
h_{15}	1	Number pixels with color C in R	$\sum_{x \in R} [RGB(x) \in C]$
h_{16}	5	Number dark pixels in region R	$\sum_{x \in R} (I(x) < \alpha_1)$
h_{17}	4	SSD Based Head Motion	Performing an SSD match on the found head region
h_{18}	4	Global flow head motion	Measure global flow estimate of head region.

Table 3: Description of all features and how to compute them. Col. 1 is the Feature ID. Col. 2 is the purpose of this feature. The purpose values are described in the first paragraph of Section 3. Col. 3 gives a short description of what the feature is computing. Col. 4 gives details on how to compute the feature.[x] is a 1/0 binary predicate.

4. Results & Discussion

To test the AdaBoost activity recognition framework we ran a number of experiments. We report these results and give other implementation details of the training process. We have tested the system on the actions listed in Table 1. All results were obtained using a separate one-against-all classifier for each action.

We performed a variety of experiments. Multiple people were used in the training/testing phase and the method achieved good success rates. We limited the system to allowing a maximum of seven weak classifiers for each action because we have higher level features. This also prevents over-training. We trained using the basic features presented in Section 3. Each feature was computed at all three zoom levels. For those features that were computed in a given region (i.e., $h_5, h_7, h_8, h_{14} - h_{16}$), the regions we computed these features at were the whole image and the found head and hand regions, respectively.

We now present results obtained by our algorithm. Overall we had 140 video events to train/test on totaling nearly 20,000 video frames. Using these events we performed a variety of training and testing setups. We report results both on detection of actions and localization in time of actions.

In all instances the strong classifier built in training did better than any single feature. Generating ground truth for start and end of events is somewhat difficult, because the start and end of an event are not easily defined. We had a person not involved with the project annotate the start and end frames of events and we report results against this annotation. Tables 4 and 5 give detailed results for each action on the training and testing data respectively. We first compare our method to the best individual feature for each classifier

Action	Frequency /# Frames	Best Feature: True +ve / True -ve %	Classifier: True +ve / True -ve %	TP	FP	Localize True +ve / True -ve %
a_1	2/263	68/96	91/91	2	0	90/91
a_2	1/46	84/80	100/67	1	0	99/66
a_3	6/412	82/98	99/98	5	0	94/97
a_4	8/303	80/76	94/93	6	0	91/92
a_5	11/501	70/85	93/97	10	3	86/95
a_6	9/348	94/79	95/94	9	2	92/94
a_7	7/205	95/79	98/97	7	0	96/97
a_8	6/464	89/94	95/99	6	0	93/99
a_9	9/136	86/87	97/94	8	3	92/95
a_{10}	6/136	97/80	97/96	6	0	90/86
a_{11}	8/336	97/78	99/98	7	0	94/93

Table 4: Results on training data. Col. 1 gives the action id. 2 gives the # actions and total # frames for each action. 3-4 give a head to head comparison between the best feature and the strong classifier. 5-6 (relevant only for TemporalBoost) give the true positive and false positive action detection rate. 7 gives TemporalBoost localization percentages.

(Columns 3-4). The features make a decision on a frame by frame basis. So we count the number of frames that the best feature correctly responded that a given action was occurring and divide by the total number of frames for this action to get the true positive rate. We use an analogous procedure to determine the true negative rate. We compare this result with the results of TemporalBoost's frame by frame decisions. It can be seen that the classifier outperforms the individual features. We would get even bigger improvements if more features were in the strong classifier. Columns 5-6 show the number of true positive and false positives and indicate how well the TemporalBoost procedure was able to correctly detect events. Localization results (Column 7) are computed in the following manner. For a given action class a_i the start and end frame of each instance is known via the ground truth. Our method also gave estimated start and end frames for each action of class a_i. We compute the total number of frames that the proposed method overlapped with the ground truth for action a_i. We divide this by the total number of frames a_i occurred to obtain the localization true positive rate. An analogous procedure is used to compute the localization true negative rate. The localization results are often times lower than the frame wise % (Column 4). This is so because it is harder to find start and end of events. If an event is missed then every frame of that event is counted as a miss, whereas a classifier can still get some of the individual frames correct. Table 6 gives a partial listing of some of the features selected for a subset of actions. It is interesting that action a_3 relied most on color information, while actions a_4 and a_{11} both made most use of contextual hand information. Figure 5 shows some example detections from the testing sequences. The detection rates for a_4 and a_5 indicate they were two of the harder events for the system. The problem comes from the fact that these two events are so similar (i.e., hand coming to face in arbitrary area). This is true because for both events the hand must move a lot initially, which looks like a_4 even if it is a_5. These results could be improved with the addition of more features. Some of our results for action recognition (about 70 actions) on the testing data set are included in the supplemental material.

It is interesting to note that in the LearnEventVariation optimization step similar activities seem to compete for the correct classification with the correct classifier eventually being able to discriminate against the incorrect classifiers. This happens for example with the events a_4 and a_5. In both cases the hand is moving as it is being brought to the face. It is only when the hand rests on the face that the event is distinguished. This would not be as readily observed in a multiclass setup. Though, we have also incorporated TemporalBoost learning into the multiclass algorithm **AdaBoost.M1**. The results obtained are similar to each action having a separate classifier. We are able to recognize 11 dif-

Action	Frequency /# Frames	Best Feature: True +ve / True -ve %	Classifier: True +ve / True -ve %	TP	FP	Localize True +ve / True -ve %
a_1	4/230	68/94	91/90	3	0	90/91
a_2	2/70	82/80	99/68	2	0	90/74
a_3	5/400	82/97	98/98	4	1	90/91
a_4	9/276	79/79	93/92	7	1	92/92
a_5	12/468	70/83	93/95	10	2	89/94
a_6	9/367	94/79	90/89	8	1	87/89
a_7	7/268	97/81	97/96	6	0	96/96
a_8	6/398	89/93	94/97	6	0	93/98
a_9	9/132	86/85	95/93	7	3	85/86
a_{10}	6/141	95/77	95/96	4	1	85/87
a_{11}	8/283	96/79	98/97	7	1	90/91

Table 5: Results on testing data. Col 1 gives the action name. 2 gives the # actions and total # frames for each action. 3-4 give a head to head comparison between the best feature and the strong classifier. 5-6 are relevant only for TemporalBoost and give the true positive, false positive action detection and rate. 7 gives localization percentages.

Action	Features Selected
a_3	$h_{15}, h_9, h_{10}, h_{13}, h_8$
a_4	$h_9, h_{15}, h_7, h_{18}, h_7$
a_{11}	$h_9, h_8, h_{16}, h_7, h_{18}$

Table 6: This table shows some of the features from the strong classifiers selected by the TemporalBoost algorithm during training. Action index is from Table 1.

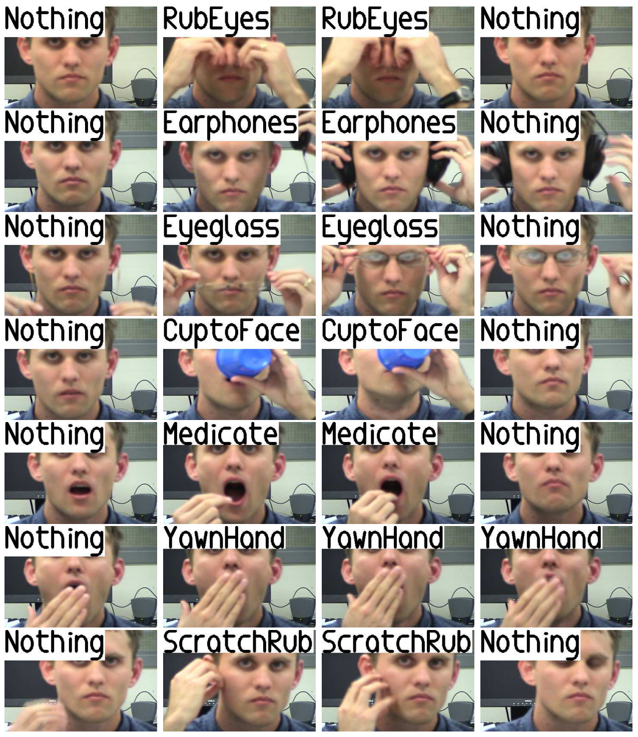

Figure 5: Example output frames from testing sequences showing images labeled automatically by TemporalBoost. They go from left to right.

ferent activities. A number of experiments were performed to demonstrate the effectiveness of our approach. This is an encouraging result.

5. Conclusion and Future Work

In this paper we have introduced a new boosting paradigm to handle various difficulties arising when using boosting for event detection. We have demonstrated a robust method to perform activity recognition in an office environment. The presented framework is able to combine information from cameras in multiple ways to increase overall system performance. The method selects the best features and zooms necessary to recognize a variety of actions. Activity features were developed and used in a boosting framework, which we call TemporalBoost. A more complete analysis of the features used would be useful to gain more insight into what kinds of new activity features would be useful. Recognizing more events and increasing system robustness would be good directions of future research. There is no reason why our algorithm, TemporalBoost, cannot be used in other kinds of video data. That is, though we demonstrate our method in the context of events, many things can be looked at as video events. Object detection, for instance, in the context of video events, would see each "event" to be a series of video frames in which a particular object was present. It would be interesting to use TemporalBoost to detect faces in video. Features such as the standard Haar wavelets could be used. TemporalBoost would be able to discover dependencies in the video data at the weak classifier and detector levels. We plan to explore how other classes of problems can fit into our learning framework. Designing more generic features is another direction of future research.

Acknowledgements

This material is based upon work funded in part by the U. S. Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

References

[1] *Sixth IEEE FGR , 2004*. IEEE Computer Society, 2004.

[2] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *CVIU*, 73(3), 1999.

[3] A.Kojima, T.Tamura, and K.Fukunaga. Natural language description of human activities from video images based on concept hierarchy of actions. *IJCV*, 2002.

[4] M. S. Bartlett, G. Littlewort, C. Lainscsek, I. Fasel, and J. Movellan. Machine learning methods for fully automatic recognition of facial expressions and facial actions. *CVPR*, 2004.

[5] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. *CVPR*, 1997.

[6] C. Dimitrakakis and S. Bengio. Boosting hmms with an application to speech recognition. *ICASSP*, 2004.

[7] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski. Classifying facial actions. *TPAMI*, 1999.

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

[9] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *ECCV*, 2000.

[10] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer and System Sciences*, 1997.

[11] D. M. Gavrila. The visual analysis of human movement: A survey. *CVIU*, 73(1), 1999.

[12] S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *CVIU*, 2004.

[13] O.-W. Kwon and T.-W. Lee. Optimizing speech/non-speech classifier design using adaboost. *ICASSP*, 2003.

[14] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *TPAMI*, 2004.

[15] N. Maillot, M. Thonnat, and A. Boucher. Towards ontology based cognitive vision. *Machine Vision and Applications*, 2004.

[16] M. Naphade and T. Huang. Extracting semantics from audiovisual content:the final frontier in multimedia retrieval. *IEEE T-NN*, 2002.

[17] R. Nevatia, T. Zhao, and S. Hongeng. Hierarchical language-based representation of events in video streams. *CVPR Workshop on Event Mining*, 2003.

[18] K. Okuma, A.Taleghani, N. Freitas, J. Little, and D. Lowe. A boosted particle filter:multitarget detection and tracking. *ECCV*, 2004.

[19] S. Park and J. Aggarwal. Semantic-level understanding of human actions and interactions using event hierarchy. *CVPR*, 2004.

[20] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *IJCV*, 2002.

[21] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. *CVPR*, 2004.

[22] P. Smith, M. Shah, and N. da Vitoria Lobo. Integrating and employing multiple levels of zoom for activity recognition. *CVPR*, 2004.

[23] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *PAMI*, 2000.

[24] T. Syeda-Mahmood, I. Haritaoglu, and T. H. editors. Special issue on event detection in video. *CVIU*, 2004.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[26] P. A. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *ICCV*, 2003.

[27] P. Yin, I. Essa, and J. M. Rehg. Asymmetrically boosted hmm for speech reading. *CVPR*, 2004.

[28] D. Zhong and S.-F. Chang. Real-time view recognition and event detection for sports video. *JVCIR*, 2004.