Detecting Human Actions in Surveillance Videos

Ming Yang, Shuiwang Ji, Wei Xu, Jinjun Wang, Fengjun Lv, Kai Yu, Yihong Gong
NEC Laboratories America, Inc.
10080 N Wolfe Road, SW-350, Cupertino, CA 95014
{myang, sji, xw, jjwang, flv, kyu, ygong}@sv.nec-labs.com

Mert Dikmen, Dennis J. Lin, Thomas S. Huang
Dept. of ECE, University of Illinois Urbana
405 N Mathews Ave, Urbana, IL 61801
{mdikmen, djlin, huang}@ifp.uiuc.edu

Abstract

This notebook paper summarizes Team NEC-UIUC's approaches for TRECVid 2009 Evaluation of Surveillance Event Detection. Our submissions include two types of systems. One system employs the brute force search method to test each space-time location in the video by a binary classifier on whether a specific event occurs. The other system takes advantage of human detection and tracking to avoid the costly brute force search and evaluates the candidate space-time cubes by combining 3D convolutional neural networks (CNN) and SVM classifiers based on bag-of-words local features to detect the presence of events of interests. Via thorough cross-validation on the development set, we select proper combining weights and thresholds to minimize the detection cost rates (DCR). Our systems achieve good performance on event categories which involve actions of a single person, e.g. CellToEar*,* ObjectPut*, and* Pointing.

1. Introduction

Event detection based on human action recognition in uncontrolled environments shows great potentials for many emerging video-content analysis applications, thus it attracts more and more research interests and experiences rapid advances in recent years. Nevertheless, most of the existing approaches [3, 5, 9, 17, 4, 18, 10, 7] first address human action detection with some simplified assumptions such as known spatial locations and temporal segmentations of actions, no (or very little) scale and viewpoint changes, as well as static and clean background. Therefore human figures can be reliably extracted and aligned. However, these assumptions seldom hold in real-world surveillance videos. Even the same type of actions may exhibit enormous variations due to cluttered background, different viewpoints and many other factors (*e.g.* human-body occlusions and low-resolution videos) in unconstrained real-world environment.

This line of research suffers from a lack of standard benchmark video dataset which provides sufficient clearly defined video events together with ground truth annotations in unconstrained real-world environment. To our best knowledge, TREC Video Retrieval Evaluation (TRECVid) [14] has made the largest effort to bridge the research efforts and the challenges in real-world conditions by providing an extensive 144-hour surveillance video dataset recorded in London Gatwick Airport. There are 10 required events in TRECVid 2009 Evaluation, we concentrate on the events that involve the actions of a single person, such as *CellToEar*, *ObjectPut*, *Pointing*, and *PersonRuns*.The approaches developed by the Video Analysis Group at NEC Laboratories America based on 3D Convolutional Neural Networks and bag-of-words of local features are first elaborated in Sec. 2– 5. Then, we present the approach using brute force search in videos developed by the Image Formation and Processing (IFP) group at UIUC in Sec. 6. The experiments as well as implementation issues about computational complexity and parameter selection are discussed in Sec. 7. Concluding remarks are given in Sec. 8.

2. System Overview

For the tasks in TRECVid 2009 Event Detection Evaluation, we focus on 3 events that require understanding of articulated body motions of a single person, *i.e. CellToEar*, *ObjectPut*, and *Pointing*. We mainly follow the framework we employed in TRECVid 2008 Evaluation, which incorporates hypothesis generation, feature extraction, and classification modules. The candidate regions are generated based on human detection and tracking which not only significantly reduces the searching space but also eases the tough

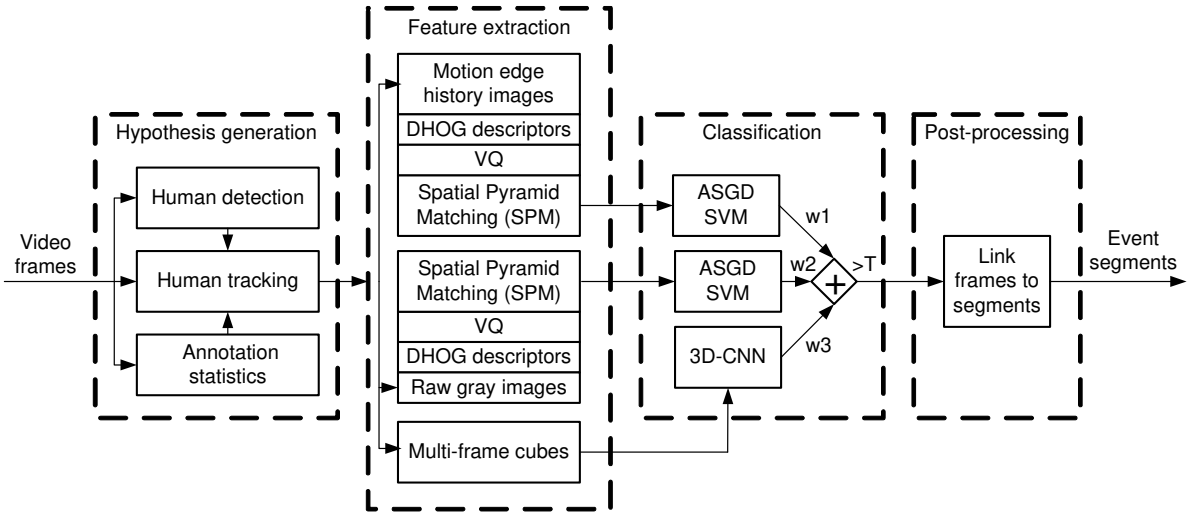

Figure 1. The system diagram of NEC's approach on human action detection.

requirements of collecting sufficient negative samples. Otherwise, the classifier in the action detector has to fulfill the job of discerning human from non-human regions and determining whether they are performing the actions of interests simultaneously. Afterwards, for each candidate region we calculate dense DHOG descriptors and build bag-of-words features based on raw gray-level images and motion edge histogram images (MEHI) to train two binary SVM classifiers for each action category. At the same time, we learn convolutional neural networks (CNN) classifiers based on 3D cubes. The classification scores of these 3 classifiers are combined to make the final decision. In the post-processing stage, the frame-based classification results are linked to event segments by heuristics. The system diagram is illustrated in Fig. 1.

3. Human Detection and Tracking

We apply a human detector based on Convolutional Neural Networks (CNN) [12] and a detection-driven multiple hypotheses based tracker [2, 6, 19] integrating color, shape and texture cues to locate human heads. For the development set of TRECVid 2008 dataset, we annotated all human heads every 750 frames. The first 70% of heads are used to train the CNN human detector and the remaining 30% are used to test the performance. Denote the labeled rectangle of a head as R_L and the detected or tracked rectangles as R_D and R_T, respectively. We regard a detection as a correct one if the overlapped area is larger than one half of both the labeled and the detected head, *i.e.* $Area(R_L \cap R_D) > Area(R_L)/2$ and $Area(R_L \cap R_D) > Area(R_D)/2$. This criterion examines both location and scale of a detection, which is quite strict for head detection in fact. The reason to train a human head detector rather than a pedestrian detector based on the whole body is due to the crowded scenes in the TRECVid videos where partial occlusions of human bodies happen very frequently.

For 4 different camera views, the number of frames labeled (*# of frames*), the average number of tagged heads per frame (*avg. # of labels*), the average number of detected heads (*avg. # of detected heads*), the average number of tracked heads (*avg. # of tracked heads*), the recall rates and precision rates of the detector and tracker are summarized in Tab. 1. The performance is quite good given the extremely complex and crowded scenes in the TRECVid test videos, *e.g.*, on average there are over 24 persons in the videos of CAM2. Certainly, wrong human detections may degrade the action detection performance later on. However, in practice, applications have to cope with such imperfect detection and tracking results and do not expect accurate annotations of human figures. Some typical human detection and tracking results from different camera views are shown in Fig. 2. The detection and tracking results are stored into hard drives which are loaded when extracting training features or performing action detection on the evaluation set. The human detection and tracking runs at 0.5-2 fps depending on the number of persons in the scene, so it may take up to 48 hours to process a 2-hour video.

4. BoW Features based SVM

Given the human detection and tracking results, we crop an enlarged candidate image region or an cube around every tracked head. Then, we train binary one-against-all SVM classifiers for each action category based on bag-of-words (BoW) of dense local features extracted from the candidate region or cube. The flow chart is shown in Fig. 3 and the technical details are elaborated as follows.

We extract dense DHOG features, where DHOG is essentially a fast implementation of the SIFT descriptor [13],

Table 1. Performance of detection and tracking (per head)

Per frame	CAM1	CAM2	CAM3	CAM5	Overall
# of frames	3775	3774	3774	3772	15095
avg. # of labels	5.505	24.315	11.486	7.330	12.159
avg. # of detected heads	3.349	16.122	7.236	5.459	8.042
avg. # of tracked heads	4.120	21.545	8.940	8.070	10.668
recall of the detector	43.53%	46.25%	42.58%	45.25%	**44.81%**
precision of the detector	74.40%	67.37%	66.09%	62.21%	**66.99%**
recall of the tracker	51.68%	56.76%	48.66%	54.11%	**53.65%**
precision of the tracker	70.80%	62.42%	61.19%	51.03%	**60.80%**

Figure 2. Sample human detection and tracking results for camera view 1,2,3,5.

from both raw gray images and motion edge history images (MEHI). Local features on raw gray images preserve the appearance information. On the other hand, MEHI proposed in [20] only concerns with the shape and motion pattern. Therefore, the bag of local features extracted from these two different kinds of images are complementary to each other. The procedure to calculate MEHI is illustrated in Fig. 4. For consecutive frames, we first calculate the frame difference images which only retain the motion information, and then we perform Canny edge detection to make the observations cleaner. The motion edges are accumulated to a single image with a forgetting factor.

The spatial pyramid matching (SPM) [11] of a bag of interest point descriptors demonstrates superb performance in object and scene categorization due to its power to delineate the spatial layout of shape patterns. Given the location of a head output by the human tracker, we crop a candidate region with 4 times of the head width by 6 times of the head height, as shown in Fig. 5. Afterwards, we calculate DHOG features on a dense grid within the candidate regions, *i.e.* every 6 pixels with two patch sizes 7×7 and 16×16. Each 128D DHOG feature is softly quantized using a codebook with 512 words , then we construct the BoW features from both 2×2 and 3×4 cells in the candidate cube. The dimensionality of the final feature vector is $512 \times (2 \times 2 + 3 \times 4) = 8192$. We test such BoW features extracted from a single frame or from a cube. The cube based BoW features are constructed from 7 frames with frame interval 2 (*e.g.* frame -6, -4, -2, 0, 2, 4, 6 if the current frame is frame 0). Note we do not align human figures and the cube is composed of regions at the same location in successive frames as illustrated in Fig. 3.

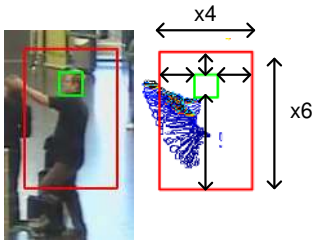

Figure 5. Extraction of the candidate regions.

To train the SVM classifier for each action category, we label the positive samples as many as possible and collect a vast number of negative samples. The total number of training samples is about 520K where each sample is an 8192D feature vector. Thus, only the storage for one set of training data requires 17G bytes ($520K \times 8192 \times 4 \simeq 17G$). The huge memory requirements and the enormous computations make learning of SVM classifiers extremely challenging for this task. As far as we know, no off-the-shelf SVM package can fulfill this task. Thus, we develop a new averaged stochastic gradient descent (ASGD) method to train linear SVM classifiers to deal with the huge amount of data. The first order stochastic gradient descent (SGD) [15] is as good as any second order SGD with optimal matrix valued step size. Our ASGD based SVM learning is very efficient since we only need to access each sample once in an iteration. The training of 5 classifiers for 5-fold cross-validation and

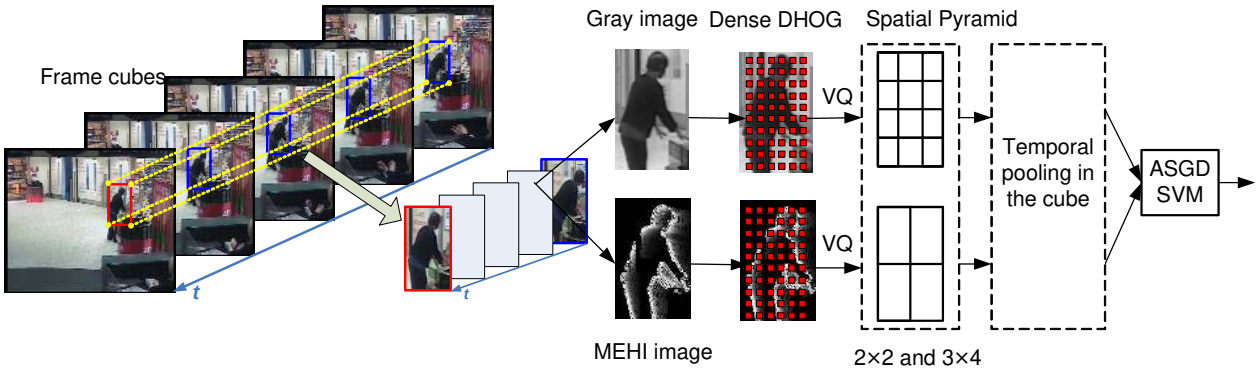

Figure 3. Flow chart of NEC's approaches based on bag of local features.

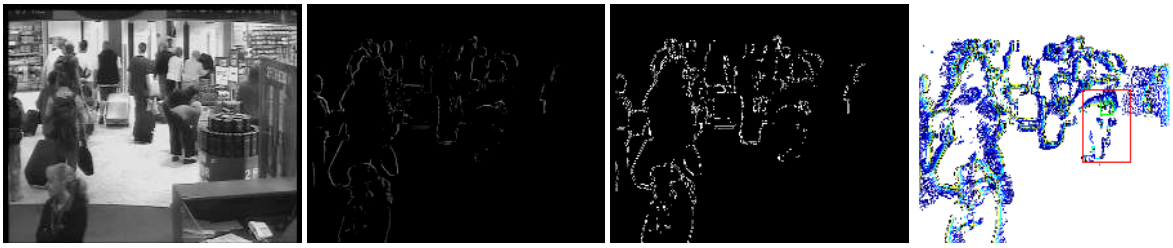

Figure 4. Illustration of MEHI extraction. From left to right: (a) the original frame, (b) the frame difference image, (c) Canny edge detection, (d) the accumulated motion edge history image

the final classifier for 3 events using the 520K samples takes about 12.5 minutes in total on a 64bit blade server with CPU Intel Xeon 2.5GHz (8 cores) and 16GB RAM.

5. Cube based Convolutional Neural Networks

Conventional paradigm of pattern recognition usually consists of two steps in which the first step computes handcrafted features from raw inputs and the second step learns classifiers based on the obtained features. The overall performance of the system is largely determined by the first step, which is, however, highly problem dependent and requires extensive human intervention. Convolutional neural networks (CNN) are a class of deep networks in which multiple stages of learned feature extractors are applied directly on the raw input images and the entire system can be trained end-to-end in a supervised or unsupervised manner [12, 16]. It has been shown recently that, when trained with appropriate regularization, CNN can achieve superior performance on image classification tasks [1, 21]. We consider the application of CNN to video action recognition in TRECVid. A simple approach for CNN in video processing is to treat the video frames as still images. However, such approach does not take advantage of the motion information carried by multiple contiguous frames. We propose a 3D CNN architecture, in which the motion information of video data is captured by performing convolution in both space and time.

In traditional CNN, convolution and subsampling are applied on the 2D feature maps in the previous layer to compute the feature maps in current layer. When applied to video processing problems, it is desirable to capture the motion information conveyed by multiple contiguous frames. We propose the 3D CNN architecture in which multiple contiguous frames are fed into CNN and the convolution is performed in both space (in a single frame) and in time (among multiple contiguous frames). In particular, the feature maps in the 3D convolution layer is connected to multiple contiguous frames in the previous layer, and contiguous feature maps are connected to contiguous feature maps in the previous layer in a overlapping manner similar to the convolution in space. To perform convolution operations along the time axis, we also require that the same set of weights are applied repeatedly with a specified temporal window size. Fig. 6 shows the 3D convolution with a temporal window size of 3.

For the TRECVid video data, bounding boxes for the humans that perform the actions have been obtained by human detection, tracking, and manual labeling. To apply the 3D CNN, we also apply the same bounding box to frames before and after the current frame with certain step size. In this application, the step size is set to 2. So, suppose the current frame is 0, we extract a bounding box at the same position from frames -6, -4, -2, 0, 2, 4, and 6. From the multiple

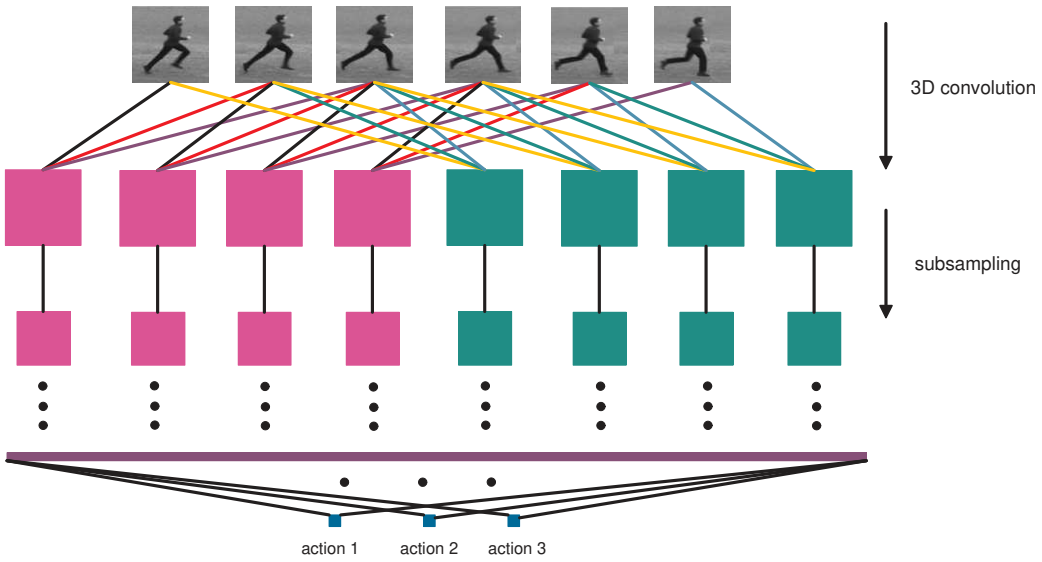

Figure 6. The 3D CNN architecture in which the convolution is performed both in space and in time. In the 3D convolution layer, the connection weights with the same color share the same set of weights. The 3D convolution and sampling can be applied alternately multiple times.

input frames, we first compute the gradients on each input frame and the optical flow from contiguous frames along x and y directions. Along with the gray image of each input we obtain a total of 5 channels in which three of them contain 7 feature maps and the other two that derived from optical flow contain 6 feature maps. In this architecture, the 3D convolution is performed separately on the the different channels (RGB, gradient, and optical flow) with a temporal window of size 3. The proposed architecture is shown in Fig. 7. The bounding box on each frame is scaled to 60×40 and the sizes of the filters and the subsampling ratios are shown in Fig. 8. The classification scores of this method is denoted by *3D-CNN*.

6. Alternative System

We also demonstrate an alternative system developed at the University of Illinois. We take a general approach for detecting a variety of short duration actions (average duration of one second) with characteristic motion or appearance patterns. The system is applied on 4 events: *Pointing*, *ObjectPut*, *PersonRuns*, *CellToEar*. We follow the sliding window detection paradigm, in which all possible locations in every frame of the video are evaluated to see whether it contains the event or not. The candidate windows for event detection are 30 frames in time and of varying spatial size depending on the camera and horizontal location of the bounding box. However, the horizontal to vertical aspect ratio of the bounding box is always fixed at 4 to 6. That means our candidate windows are space-time rectangles with the vertical edge longer than the horizontal one. This is consistent with the shape of the most subjects in the video corpus (unless they are seated). We infer the size-location relation of subjects in the video corpus by computing the linear relation between the locations and head sizes of manually annotated people in each of the five cameras.

To accomplish the goal of event detection, we take advantage of two types of features describing appearance and motion respectively. The framework uses a bag of features approach for motion features, while the appearance feature is extracted from a dense grid on the entire candidate window. Finally the two sets of descriptors in each candidate window are concatenated together and classified with a linear kernel SVM as shown in Fig. 9. To accelerate the sliding window search, K-nearest neighbor (KNN), feature extraction and SVM kernel evaluation are all completely or partially CUDA, which gives us a 15x to 30x speed boost in most cases [1].

6.1. Appearance features

We construct histograms of oriented gradients (HoG) [8] to describe the appearance of the subject in the candidate window. We have empirically found that the 10th frame of each annotated event contains the most characteristic appearance of the subjects for each event (*e.g.* for Pointing subject usually have their arms fully extended). HoG feature is a static image feature, which we extract from the 10th frame of each candidate window. The bins of the histograms are over the orientations. We use eight orientation

[1]The CUDA source codes are available to download at http://libvivid.sourceforge.net

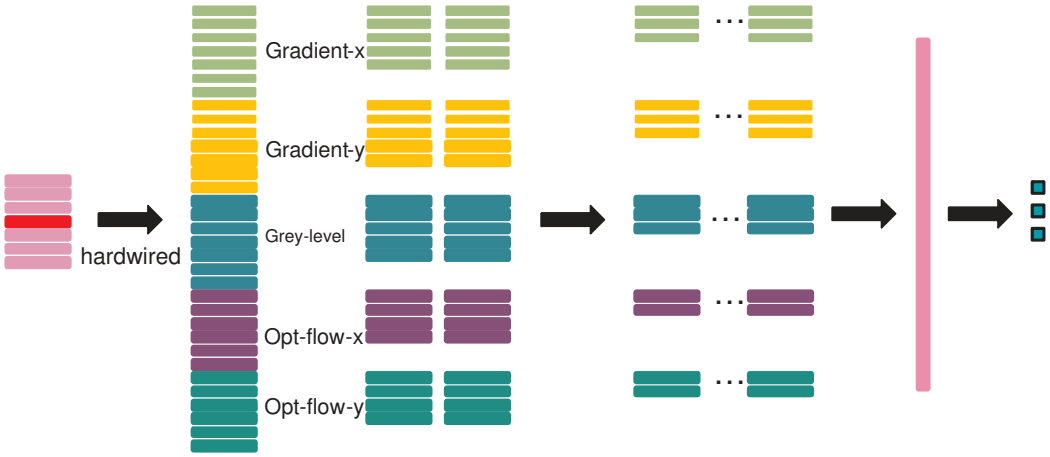

Figure 7. The 3D CNN architecture employed for human action recognition. The first layer is hard-wired in which the gradient on each input frame and the optical flow computed from contiguous frames along x and y directions are computed along with the gray image of each input. In this architecture, the 3D convolution is performed separately on the different channels (RGB, gradients, and optical flow) with a temporal window of size 3. Subsampling layers are applied after each convolution layer and they are not shown in this figure.

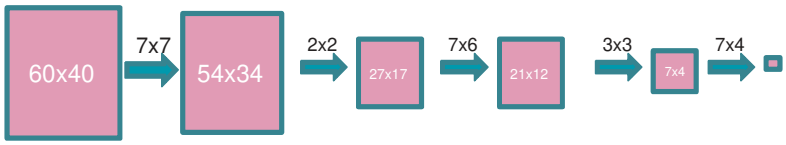

Figure 8. The sizes of kernels and feature maps employed in the 3D CNN architecture.

bins equally spaced from 0 to 180 degrees of image gradient direction.

We define a $(n_x \times n_y)$ fixed grid within the candidate window. Each grid point corresponds to a 15×15 square region from which an 8 dimensional histogram of gradient orientations will be computed. The direction of the gradient at the pixel determines which bin of the histogram the pixel belongs to, and the magnitude of the gradient at the pixel determines the weight of the vote cast. Finally the histograms of all blocks are concatenated to form the holistic appearance feature vector of the candidate window. The spatial dimensions of the candidate windows are re-scaled to 90×60 pixels, and we set n_x and n_y to 15 and 9 producing a $15 \times 9 \times 8 = 1080$ dimensional feature vector for appearances.

6.2. Motion Features

As opposed to shape and appearance, it is hard to localize motion patterns. Therefore for describing motion, we pursue a bag of features approach, where the location of the motion within the candidate window has no effect on the feature descriptor. We first find points of motion in the candidate window and we randomly sample a fraction of them. Then we use motion history images [3] to describe the motion around each sampled point. In the next step, we find the nearest codeword from a dictionary of motion history descriptors which was learned offline by clustering a large number of motion history descriptors randomly collected from the video corpus. Finally we count the number of nearest neighbors to each dictionary codeword from the extracted motion history image descriptors in the candidate window. This gives us a histogram equal to the size of our codeword dictionary. We use this histogram as our motion feature. The size of the trained dictionary is 1000 codewords.

6.2.1 Random sampling on the motion boundary (RSMB)

To find the points of motion in the candidate windows, we randomly sample from the motion boundaries obtained by thresholding the pixel-wise difference between two consecutive frames. This sampling method is rather naive, yet it is still more structured than uniformly sampling from all pixels in the video. It is logical in this case since motion information relevant to action detection is mostly contained around the pixels of the motion boundary. The random sampling process is conducted by finding all the motion boundary pixels in a candidate window and then ran-

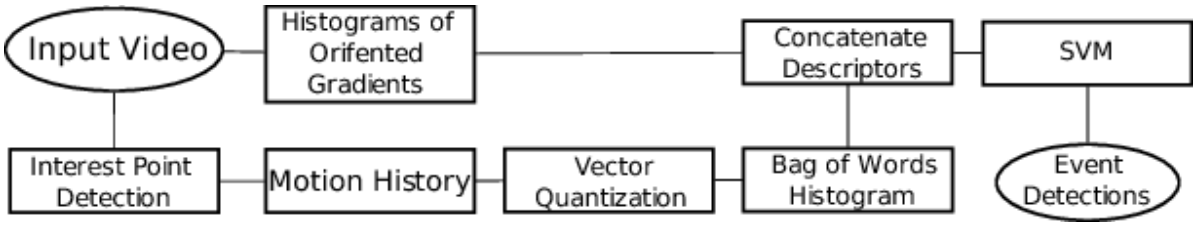

Figure 9. Diagram of the UIUC System. Two types of features are concatenated at the classification stage to improve the detection accuracy

domly selecting 50% of them. This approach has the advantage that more samples are collected from frames with large amounts of motion and few get selected from frames with little motion. Thus we can quickly discard candidate windows with little to no motion. Our studies indicate that densely sampling points from the motion boundary, even randomly, yields superior results to the recognition performance obtained with state of the art space-time interest point detectors.

6.2.2 Motion history images (MHIST)

Motion History Images [3] are one of the first features proposed for representation and recognition of simple action categories. The initial studies have assumed that the subjects are well localized and their scales are known. Unfortunately we do not have such kind of clean environment in a general surveillance setting, and despite our sliding window approach, we have to allow for flexibility of the features in both time, space and scale space. Therefore we make a slight adjustment to the original definition of the motion history image, where we use MHIST as a local region descriptor in which we extract the motion history image of an $(X \times Y \times T)$ region centered around the sampling point. The drawback of MHIST features is that the non moving pixels of the foreground objects are not registered in the descriptor. However we use this feature to specifically describe motion and augment it with the static appearance descriptor explained in the previous sub section. This approach allows us to cover a significant range of information relevant for characterizing the action in the candidate window.

The motion history image H of a $(X \times Y \times T)$ sized region R centered around a point (x_0, y_0) at time t_0 in the video volume $V(x, y, t)$ is described as follows:

$$D(x, y, t) = |V(x, y, z) - V(x, y, t-1)| \quad (1)$$

$$h(x, y, t) = \begin{cases} 0, & D(x, y, t) < k \\ 1, & D(x, y, t) \geq k \end{cases} \quad (2)$$

$$H_{x_0,y_0,t_0}(\hat{x}, \hat{y}) = \max_{0 \leq \hat{t} \leq \tau} (1 - \tfrac{\hat{t}}{\tau}) h(x_0 + \hat{x}, y_0 + \hat{y}, t_o + \hat{t}) \quad (3)$$

$$|\hat{x}| \leq \tfrac{X}{2}, |\hat{y}| \leq \tfrac{Y}{2}$$

In this system, we set $X = Y = 11$ and $T = 12$.

6.3. Training of the system

We used TRECVid 2008 Development corpus as our development data since we had added location labels to a part of the time labeled ground truth. We trained separate systems on all 4 testing events and all 5 camera views. The labeled events are very sparse compared to the rest of the corpus. Therefore while it is relatively easy to obtain negative non-event samples, training becomes massively unbalanced due to the large amounts of negative samples. To overcome this imbalance, we adopt a bootstrap classifier training strategy, where we start out with a balanced number of positive and negative training samples. A classifier trained using this set and is applied to the training corpus. The false alarms found are added to the set of negative samples from the previous round to form a new set of negative samples. We then train a new classifier and repeat this process adding "tougher" negative samples to the training set at each iteration. The classifier training is terminated when a reasonable rate for false positive production is achieved on the training corpus. The results are denoted by UIUC-1 in the experiments.

7. Experiments

TRECVid 2009 Event Detection Evaluation [14] provides 99 hours videos in the development set and about 44 hours videos in the evaluation set, where the videos were captured using 5 different cameras with image resolution 720×576 at 25 fps. From the statistics of events in the development set, we find out there are hardly any events in the videos of CAM4, so we exclude those videos from our experiments to save some computation power. Even though, to detect human actions in such a huge dataset, computational efficiency remains the topmost concern to ensure the experiments can be done within reasonable time. So we save intermediate results to hard drives to avoid repeated computations and try to utilize parallel computing with multi cores as much as possible. The system is implemented using C++ and compiled with Intel Compiler 10.1 to utilize multiple cores. The experiments are mainly performed on 64bit blade servers with Intel Xeon 2.5GHz CPU (8 cores) and 16GB RAM. It took about 20 days to complete all the experiments. Next, we elaborate the details about training sample preparation, feature extraction, parameters selection based on 5-fold cross-validation.

7.1. Training sample preparation

Collecting training samples is not trivial. The ground truths about time intervals of actions in the development set were provided by NIST. We further label the locations of the persons performing the actions of interests, *i.e. CellToEar*, *ObjectPut*, and *Pointing*, every 3 frames as the positive samples. We further generate 6 additional positive samples from each labeled positive sample by perturbing the locations and scales. The total numbers of positive samples for *CellToEar*, *ObjectPut*, and *Pointing* are about 25.2K, 39.3K, 152.2K, respectively. The negative samples include two subsets from human labeling and detection results: 1) the same persons performing the actions are labeled as negative samples in two 30-frame intervals before and after the action occurs; 2) the detected persons that are not performing the actions when the actions occur. The total number of negative samples is about 303K. The collection of negative samples need to ensure the classifiers do not learn action models only for some specific persons. Quite often the persons performing the actions are too hard to find even for our labelers, so we only manage to label about one half of the action instances in the ground truths provided by NIST.

7.2. Feature extraction and classification

We train two codebooks with 512 words on the DHOG features extracted from gray images and MEHIs in 8 hours videos on the day 2007/11/12 using K-Means. After loading detection and tracking results from the files, we extract BoW features and train SVM classifiers using the ASGD algorithm described in Sec. 4. We train 4 SVM classifiers based on a single image and cube of gray images and MEHIs, where the classification results are denoted by *Gray-Frame*, *Gray-Cube*, *MEHI-Frame*, and *MEHI-Cube*, respectively. The training features are only extracted where the labels are available, so it only takes about 8 hours to obtain one set of features. Evaluation on a 2-hour video requires feature extraction and classification for all persons in every frame, which may take about 1-2 days depending on the number of detected persons. We save the classification scores of all persons from these 4 classifiers and those of the *3D-CNN* method to hard drives.

7.3. 5-fold cross-validation performance

For each candidate region, the classification scores of three classifiers are combined linearly. If the combined confidence is larger than a threshold T, this frame is regarded as positive. The frame based results are linked to generate the event segments by heuristics considering the spatial and temporal smoothness and consistency. We limit the maximum number of output events for one 2-hour videos by 20 and output 2 events at most for the short video clips in the evaluation set.

We exhaustively search the combining weights (with step 0.1) and the threshold (with step 0.01) to minimize the DCR directly. Towards this end, we implement DCR calculation with C++ which is very critical to the efficiency. Moreover, due to the computational issue, we can only afford exhaustive searching of combining weights of 3 sets of classification scores. The videos in the development set of TRECVid 2009 were recorded on 10 different days, so we perform 5 fold cross-validation using the training features from 8 days to train and test on the other 2 days. We have tried different combinations and find the following two are good in terms of DCR scores in the cross-validation: Method 1 *Gray-Frame + Gray-Cube + MEHI-Cube*; and Method 2 *Gray-Frame + MEHI-Frame + 3D-CNN*.

The average DCRs per event per camera of 5-fold cross-validation are shown in Tab. 2 and Tab. 3, where the number of events, true detections, and false positives are shown in the parenthesis, *e.g.* (806/21/244). These tables about the cross-validation performance guide us what methods shall be included in the final submissions. The final combining weights and the threshold $\{\omega_1, \omega_2, \omega_3, T\}$ ($\omega_1 + \omega_2 + \omega_3 = 1$) are selected by exhaustively searching the best common parameters that yield the lowest DCR per event per camera, which are shown in Tab. 4 and Tab. 5. We expect the DCR on the evaluation set is in between of the average DCR obtained by the cross-validation and the lowest DCR obtained by searching common parameters.

We submitted event detection results of 4 systems. Submission NEC-1 and NEC-2 correspond to the 2 aforementioned methods. Submission NEC-3 selectively combines NEC-1 and NEC-2 based on the cross-validation performance per events per camera. Submission UIUC-1 includes the detection results of UIUC's system. From Tab. 6, we can see the evaluation performance is in line with our expectation. Our detection results on the events *CellToEar*, *ObjectPut*, and *Pointing* outperform all other participants.

7.4. Discussion

The false positives rates are still fairly high. A considerable portion of the false positives appear similar to the true ones in terms of the motion patterns, *e.g.* touching hair is occasionally misclassified to *CellToEar* and it is very hard to distinguish between *ObjectPut* and *ObjectGet*. The majority of the false positive are induced by cluttered background, occlusions in a crowd, and the complicated interactions among people. The combination weights of 3 classifiers vary dramatically *w.r.t* different events in different cameras, which indicates that the performance and the generalization ability are not stable.

8. Conclusions

The strengths of our system are on 3-fold: 1) the description power of the BoW features and 3D-CNN; 2) the efficient ASGD learning algorithm to utilize vast number of training samples; and 3) the thorough cross-validation

Table 2. 5-fold cross-validation performance of *Gray-Frame + Gray-Cube + MEHI-Cube*

	CellToEar	ObjectPut	Pointing
CAM1	1.0000 (40/0/0)	0.9979 (706/9/38)	0.9973 (926/5/9)
CAM2	1.0015 (265/0/6)	0.9937 (1122/7/11)	0.9990 (999/3/8)
CAM3	1.0053 (262/0/21)	1.0010 (843/1/9)	1.0023 (1056/0/9)
CAM5	0.9526 (239/21/217)	1.0000 (432/0/0)	1.0070 (1048/2/36)
Overall	**0.9896** (806/21/244)	**0.9981** (3103/17/58)	**1.0014** (4029/10/62)

Table 3. 5-fold cross-validation performance of *Gray-Frame + MEHI-Frame + 3D-CNN*

	CellToEar	ObjectPut	Pointing
CAM1	1.0000 (40/0/0)	0.9915 (706/13/33)	0.9978 (926/4/7)
CAM2	1.0000 (265/0/0)	1.0059 (1122/2/34)	1.0000 (999/2/8)
CAM3	1.0313 (262/0/125)	1.0010 (843/0/4)	1.0033 (1056/3/25)
CAM5	0.9507 (239/17/132)	1.0003 (432/0/1)	1.0088 (1048/9/71)
Overall	**0.9954** (806/17/257)	**0.9997** (3103/15/72)	**1.0025** (4029/18/111)

which finds the reliable working point in terms of DCR. TRECVid Event Detection Evaluation not only provides the chance to test the performance of the state-of-the-art approaches in realistic settings but also motivates us to investigate parallel computing and learning algorithms dealing with huge number of samples. Event detection in unconstrained surveillance videos remains an open and challenging problem for computer vision and machine learning in the near future.

References

[1] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV'08*, 2008. 4

[2] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR'98*, pages 232–237, Santa Barbara, CA, June 23-25, 1998. 2

[3] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(3):257–267, Mar. 2001. 1, 6, 7

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS'05*, pages 65–72, Beijing, Oct. 15-16, 2005. 1

[5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV'03*, volume 2, pages 726–733, Nice, France, Oct. 13-16, 2003. 1

[6] M. Han, W. Xu, H. Tao, and Y. Gong. An algorithm for multiple object trajectory trackingt. In *CVPR'04*, volume 1, pages 864–871, Washington, DC, Jun.27-Jul.2 2004. 2

[7] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV'07*, Rio de Janeiro, Brazil, Oct. 14-21, 2007. 1

[8] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. *Lecture notes in computer science*, 3667:91, 2006. 5

[9] I. Laptev and T. Linderg. Space-time interest points. In *ICCV'03*, volume 1, pages 432–439, Nice, France, Oct. 13-16, 2003. 1

[10] I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV'07*, Rio de Janeiro, Brazil, Oct. 14-21, 2005. 1

[11] S. Lazebnik, C. Achmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR'06*, volume 2, pages 2169–2178, New York City, June17 - 22 2006. 3

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based leraning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, Nov. 1998. 2, 4

[13] D. G. Lowe. Distinctive image features from scale invariant keypoints. *Int'l Journal of Computer Vision*, 60(2):91–110, 2004. 2

[14] National Institute of Standards and Technology (NIST): TRECVID 2009 Evaluation for Surveillance Event Detection. http://www.nist.gov/speech/tests/trecvid/2009/ and http://www.itl.nist.gov/iad/mig/tests/trecvid/2009/doc/eventdet09-evalplan-v03.htm, 2009. 1, 7

[15] B. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July 1992. 3

[16] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR'07*, 2007. 4

[17] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR'04*, volume 3, pages 32–36, Cambridge, UK, Aug. 23-26, 2004. 1

[18] E. Shechtman and M. Irani. Space-time behavior based correlation. In *CVPR'05*, volume 1, pages 405–412, San Diego, CA, June 20-25, 2005. 1

[19] M. Yang, F. Lv, W. Xu, and Y. Gong. Detection driven adaptive multi-cue integration for multiple human tracking. In *ICCV'09*, pages 1554 – 1561, Kyoto, Japan, Sept.29 - Oct.2, 2009. 2

[20] M. Yang, F. Lv, W. Xu, K. Yu, and Y. Gong. Human action detection by boosting efficient motion features. In *IEEE Workshop on Video-oriented Object and Event Classification in Conjunction with ICCV*, Kyoto, Japan, 28, 2009. 3

[21] K. Yu, W. Xu, and Y. Gong. Deep learning with kernel regularization for visual recognition. In *NIPS'08*, 2008. 4

Table 4. Parameter selection of *Gray-Frame + Gray-Cube + MEHI-Cube*

	CellToEar	ObjectPut	Pointing
CAM1	1.0003 (40/0/0) (0.3,0.3,0.4,1)	0.9871 (706/17/33) (0.3,0,0.7,0.67)	0.9971 (926/6/10) (0,0.6,0.4,0.77)
CAM2	1.0003 (265/0/0) (0.3,0.3,0.4,1)	0.9944 (1122/7/12) (0.7,0,0.3,0.72)	0.9968 (999/6/10) (0.2,0.2,0.6,0.65)
CAM3	1.0003 (262/0/0) (0.3,0.3,0.4,1)	1.0002 (843/1/3) (0,0.3,0.7,0.74)	1.0001 (1056/0/1) (0,0.5,0.5,0.92)
CAM5	0.9591 (239/20/218) (0,0.3,0.7,0.15)	0.9991 (432/1/0) (0.2,0.3,0.5,0.86)	0.9963 (1048/8/11) (0.2,0,0.8,0.81)
Overall	**0.9892** (806/20/218)	**0.9946** (3103/26/48)	**0.9970** (4029/20/32)

Table 5. Parameter selection of *Gray-Frame + MEHI-Frame + 3D-CNN*

	CellToEar	ObjectPut	Pointing
CAM1	1.0003 (40 0 0) (0.3,0.3,0.4,1)	0.9866 (706/16/30) (0.5,0.3,0.2,0.54)	0.9961 (926/6/6) (0.6,0.3,0.1,0.74)
CAM2	1.0003 (265/0/0) (0.3,0.3,0.4,1)	0.9974 (1122/9/32) (0.1,0.5,0.4,0.55)	0.9982 (999/4/6) (0.3,0.7,0,0.73)
CAM3	1.0000 (262/0/0) (0.3,.3,0.4,1)	1.0000 (843/1/2) (0.5,0.5,0,0.67)	0.9994 (1056/3/7) (0.5,0.1,0.4,0.59)
CAM5	0.9529 (239/17/152) (0,0.6,0.4,0.41)	0.9994 (432/1/1) (0.4,0.4,0.2,0.67)	0.9968 (1048/18/59to) (0,0.6,0.4,0.46)
Overall	**0.9877** (806/17/152)	**0.9953** (3103/27/66)	**0.9970** (4029/31/78)

Table 6. Evaluation performance of our submissions.

CellToEar	#Ref	#Sys	#CorDet	#FA	#Miss	Act.DCR	Min.DCR
NEC-1	194	35	3	32	191	**0.995**	**0.991**
NEC-2	194	20	1	19	193	1.001	0.998
NEC-3	194	20	1	19	193	1.001	0.998
UIUC-1	194	183	0	58	194	1.019	1.060
ObjectPut	#Ref	#Sys	#CorDet	#FA	#Miss	Act.DCR	Min.DCR
NEC-1	621	10	2	8	619	0.999	0.997
NEC-2	621	11	3	8	618	0.998	0.998
NEC-3	621	5	2	3	619	**0.998**	**0.997**
UIUC-1	621	555	1	190	620	1.061	1.020
Pointing	#Ref	#Sys	#CorDet	#FA	#Miss	Act.DCR	Min.DCR
NEC-1	1063	6	2	4	1061	0.999	0.999
NEC-2	1063	5	2	3	1061	**0.999**	**0.998**
NEC-3	1063	6	2	4	1061	0.999	0.999
UIUC-1	1063	774	13	225	1050	1.062	1.006