Informedia @ TRECVID2008: Exploring New Frontiers

Alexander Hauptmann[1], Robert V. Baron[1]. Ming-Yu Chen[1]. Michael Christel[1], Wei-Hao Lin[1], Lily Mummert[2], Steve Schlosser[2], Xinghua Sun[1], Victor Valdes[1], and Jun Yang[1]

[1]*School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213*

[2]*Intel Research Pittsburgh, Pittsburgh, PA 15213*

Abstract.

The Informedia team participated in the tasks of Rushes summarization, high-level feature extraction and event detection in surveillance video. For the rushes summarization, our basic idea was to use subsampled video at the appropriate rate, showing almost the whole video faster, and then modify the result to remove garbage frames. Sinply subsampling the frames proved to be the best method for summarizing BBC rushes video, with other improvements not improving the basic inclusion rate, nor appreciably affecting the other subjective metrics. For the high-level feature detection, we trained exclusively on TRECVID'05 data and trying to assess and predict the reliability of the detectors. The voting scheme for combining multiple classifiers performed best, marginally better than trying to predict the best classifier based on a robustness calculation from within dataset cross-domain performance. For event detection, we found that the overall approach was effective at characterizing a presegmented event in the training data, but lack of event segmentation (information about the duration of an event and the existence of a known event resulted in a dramatically lower score in the official evaluation.

1. BBC Rushes Summarization

- Our **CMU.Base 50x** run used subsampled video at the appropriate rate, showing almost the whole video faster.
- **CMU.1** modified the base result by removing garbage frames, introducing summary audio and filling removed frames with frames in pans and zooms.
- **CMU.2** was a more aggressive version of CMU.1, removing more garbage.

.

Table 1. Results on 5-point scale subjective metrics.

	JU	RE	TE
50x baseline	2.66	2.02	1.44
CMU.1	3.02	2.28	1.76
CMU.2	2.96	2.25	1.64

As a control to help gauge success for the TRECVID summarization task across all participants, our Informedia group at Carnegie Mellon University produced a very simple baseline approach to 2% summary generation: sample every 50th frame. No consideration is given to the audio track at all, and the produced baseline has no audio component. No junk shot filtering is applied, so the percentage of junk frames in the baseline will be roughly equivalent to the percentage of junk frames in the source BBC rush video. The visual persistence of the junk video in 50x may not register with the viewer as much if all the junk shots in the source are brief. Consider a clapper shot of just under two seconds that produces exactly one frame for viewing in the 50x summary: a viewer may not notice this junk frame that appears for 1/25 of a second.

Alternatively, a 50-second sequence of color bars will be represented in the 50x summary as a one second color bar steady shot that will be noticeable. Hence, the simplicity of the baseline algorithm was expected to result in lower subjective ratings, but we purposefully kept the 50x baseline simple so as to measure the contributions of folding in junk frame removal and other automated processing techniques. Based on the reports and demonstrations from the TRECVID 2007 Video Summarization Workshop [7], most participants in this task did attempt junk frame removal in 2007, eliminating irrelevant shots as shown in Figure 1.

Figure 1. Examples of 4 types of irrelevant (junk) shots seen in BBC rushes video: white frame, black frame, clapper to mark scene takes, and color bar.

Such noise reduction attempts did not often result in video summaries that were markedly better than the baselines in 2007. Perhaps the source rushes material did not contain the volume of junk frames needed for such reduction to matter for the inclusion (IN) metric, or for the subjective measures used in 2007. For 2008, in order to more directly measure effects of junk frame removal perhaps, the published guidelines (http://wwwnlpir. nist.gov/projects/tv2008/tv2008.html) stated that human evaluators would directly rate whether the summary "contains color bars, clapboards, and/or totally black or totally white frames" – the JU metric. Given this emphasis on junk frame removal, we obviously chose to include such noise reduction attempts in both of our submitted runs CMU.1 and CMU.2. We did not mute the summaries since we felt that understanding the acoustic context would help to more quickly understand the visual events. Hence, for both CMU.1 and CMU.2 we included audio, in fact the same audio. The audio edit list was not a 50x sampling, which would have been incomprehensible, but rather a collection of 1x (normal playback) audio phrases, of course no longer time-aligned to every point of the video summary visuals.

Our approach favors playing coherent, recognizable audio segments, related to the visual segments, but loses full audio/video synchronization. Keeping some audio representation in a multimodal video summary was a recommendation from an earlier empirical study [8], which also advised that tight audiovisual synchronization may not be necessary in a video summary. For our CMU NIST-judged runs, we decided to focus on a few specific summarization features. First, how would our own junk frame removal improve on the baseline, removing clearly irrelevant material as illustrated in Figure 1? Second, does the audio component improve satisfaction with the summary or time on task? Finally, if we aggressively remove junk frames to have more space available in the 2% target size for other frames, and then backfill those frames by emphasizing sequences of importance – pan/zoom sequences – and fold in variable rate playback, is there a difference on any of the collected metrics?

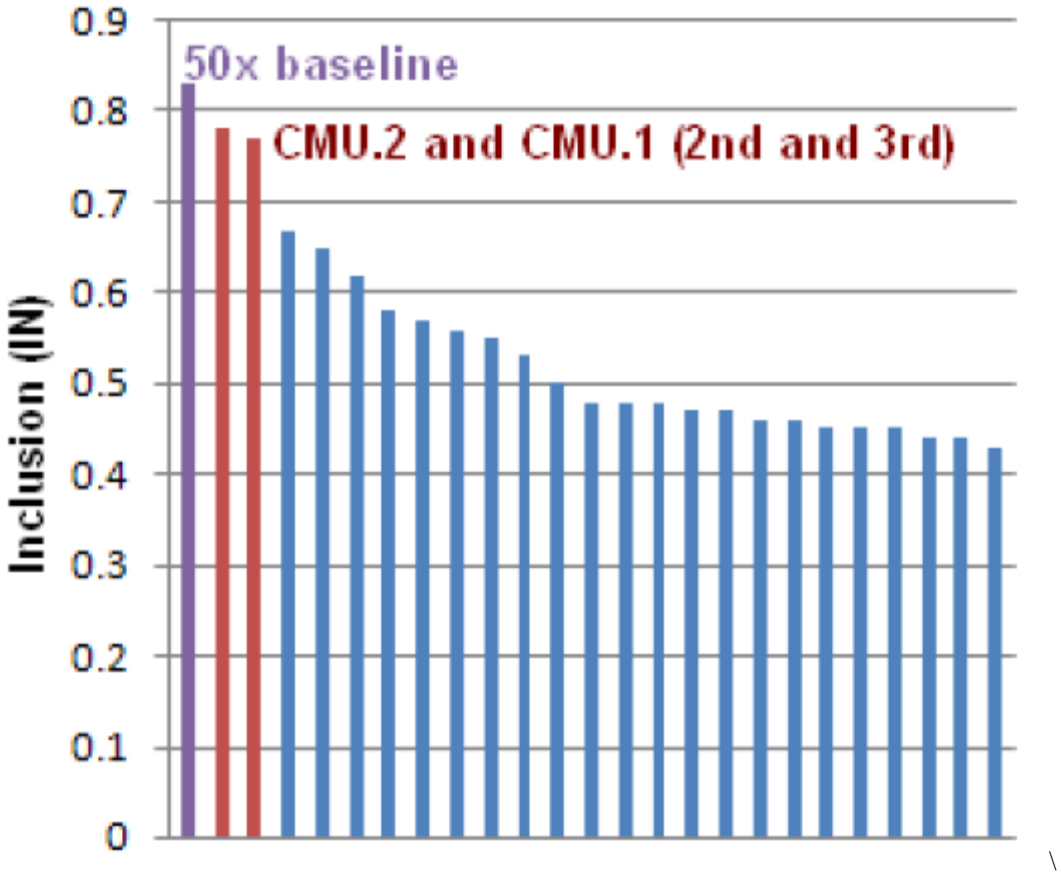

Figure 2. Top 25 IN scores for TRECVID 2008 BBC rush video summarization runs.

1.1 Visual Processing

Our NIST runs labeled CMU.1 and CMU.2 in the evaluation utilized automatic junk frame detectors. There are four different kinds of junk frames we want to remove from BBC summarizations, as shown in Figure 1. We extract three different

features to construct our junk frame detectors: HSV color histogram features, SIFT features and speech recognition features. Color histogram features provide solid performance to detect black frames, white frames, and color bar frames, as witnessed in the TRECVID 2007 summarization runs [2, 5]. However, the main challenge is to detect clapper frames. SIFT features and speech recognition features are extracted to provide different information other than global color appearance. Scale-invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. The detection and description of local image features can help in object recognition. The SIFT features are based on the appearance of the object at particular interest points, are invariant to image scale and rotation, and are also robust to changes in illumination, noise, occlusion and minor changes in viewpoint. In clapper detection, a key difficulty is that clappers vary in appearance. Hence, global features by themselves may not detect clappers accurately. Our SIFT clapper detection has three major steps: (1) SIFT feature detection and description based on [5]; (2) bag-of-word quantification; and (3) classification. Each interest point is described by a 128-dimension vector. For each key frame, the number of extracted SIFT features is different. Therefore, we try to use bag-of-word approach to quantify feature distribution of each key frame. The idea of the bag-of-word approach is to quantify SIFT features into a fixed number of types. We use k-means clustering to find the conceptual meaningful types and each cluster (or type) is treated as a visual word in bag-of-word approach. In the end, each frame is presented by a visual word histogram feature and this is the bag-of-word feature of the image. We train our clapper classification based on bag-of-word features. We also found speech recognition provides distinguishing features to detect clappers. Prior work

by FX Palo Alto signaled the possible benefits of audio for clapper detection [1], but we looked for spoken words rather than the clapper sound as we also wanted to detect clapper sequences where there was no audible "snap" of a board clapping against another board. Just as the visual cues varied for clappers, so did the audio, but based on inspection of the development set we felt that there were some distinguishing key phrases with clappers, like "Take One!", "Take Two!", and "Action!" We set up heuristic rules to retrieve speech recognition to utilize this information to detect clappers by audio appearance. We combined the detection results from global appearance (HSV color histogram), local appearance (SIFT features) and audio appearance (speech recognition). Table 2 shows our cross-validation detection performance on the BBC Rush training set.

Table 2. Performance of different automated clapper detectors against the training set.

	Color	SIFT	ASR	Combination
Precision	0.65	0.7	0.17	0.31
Recall	0.22	0.37	0.2	0.58

Color in the table means HSV color histogram, SIFT is the result from SIFT features, ASR is the result from speech recognition and combination is the result by combining all three features. From the result, we can see ASR has fairly poor precision but overall it helps recall. Since all three features are from three different views, they complement each other to detect more clappers. This results in overall recall that is much higher; however, the precision drops because of the poor ASR result. For CMU.1, we make use of the SIFT clapper detector, finding fewer clappers but with higher precision. For CMU.2, we attempt to score better on the junk frame existence subjective scale (JU) by more aggressively removing clapper frames according to the Combination clapper detector. We acknowledge that some non-clapper content is thrown out, but we have better recall of clapper frames as well. CMU.2 has more space available after junk frame removal for backfilling. As was done in our 2007 runs [2, 5], we emphasize automatically detected camera pans and zooms, using those frames to replace the junk frames removed by the two methods. The replacement frames come from the longest pans/zooms found for the original video with reasonable confidence. On average, CMU.2 eliminated 6.8% more frames as junk frames based on the more aggressive Combination clapper filtering, producing a video summary with more pan/zoom frames than CMU.1.

The visual processing begins with the frames that are in the 50x sampling, which we will term CMU keyframes. Black, white, and color bar junk frames are eliminated from this set based on the HSV color histogram classifier that CMU also used last year. Clapper shots are detected via SIFT for CMU.1, and more aggressively folding in ASR cues for CMU.2. Temporal cues are used to more aggressively eliminate junk frames: CMU keyframes adjacent to detected junk frames are also considered junk and removed. Also, the temporal neighborhood is broadened for clappers: CMU keyframes adjacent to frames adjacent to clapper frames are also removed. Detected pans/zooms are added back in as space allows at a playback rate of 2x (if space allows) or 4x so that the pan or zoom can be easily interpretable. As first noted in [5], we are leveraging from cinematic principles that pans and zooms are used to emphasize important visuals, but unfortunately for rush videos this is not always the case. A pan or zoom may also be occurring at the setup of a take to get the camera focused appropriately. We opted for slower playback of pans/zooms so that the pan/zoom event could be more easily recognized by the viewer of the video summary.

1.2 Aural Processing

We first ran the SAIL LABS Technology (http://www.sailtechnology.com) speech recognizer over the video. The source footage is not ideal for automatic speech recognition (ASR), as in rushes materials the speaker is often not properly microphoned and environmental aural noise not controlled because the sound track is anticipated to be cleaned up later during the production process. Since the ASR error rate was quite high, we did not rely heavily on the content of the spoken text, but exploited other characteristics. The output of the recognizer was split into phrases, based on the duration of silences in the speech and whether a speaker change was detected. We wished to collect audio snippets bounded by silences based on earlier research on skims [3] showing that choppy audio is very distracting, and in that research we had successfully used the SNR segmentation to obtain reasonable acoustic phrases in news skims.

To achieve a balance in coverage of the video, we divided the video into equal segments, where the number of segments was determined by the target length of the summary video (1/50th of the original full video) and the average duration of the SAIL-recognized phrases, i.e.

Number of Segments = TargetLength / AvgPhraseDuration

Now we divided the full video into the same number of segments, and the algorithm then selected one phrase of near average duration which occurred in each of the segments in the full video. This allowed us to insert some audio from every portion of the video. Near the end of the target time of the summary video, shorter phrases became acceptable if longer phrases no longer fit into the overall time budget. At the end, if the resulting audio segment was shorter than the target time, the remaining time was padded with silence.

We also placed a number of constraints on the phrases that were added. A phrase had to contain at least two new words to be included in the summary. This avoided adding repetitions of identical or nearly identical utterances from repeated scenes. A certain set of phrases was always ignored or eliminated. This included the very first phrase in video since this phrase frequently contained the director's instructions or startup talk. We also excluded all phrases containing the words "Action", "Quiet", "Take", "Scene", as well as any phrases containing numbers (as in "take two", "scene five"). Deleting any phrase with one of these keywords allowed us to remove phrases containing director's instructions despite a high speech recognition error rate for anything not spoken directly near the microphone. One word phrases were also eliminated from the process.

1.3 RESULTS: 50x baseline, CMU.1, CMU.2

As expected based on the empirical study conducted with BBC rush video from TRECVID 2007 [2], the inclusion score (IN) metric showed that the 50x strategy, and our derivatives CMU.1 and CMU.2, provide excellent coverage of the original video. The IN score, an estimation of recall, is plotted for the submitted runs in Figure 2. The simplistic strategy of Nx sampling, with N=50 to provide a 2% summary, worked as well as the 25x 4% summary did in evaluations of TRECVID 2007 rush summaries [2, 5].

From 2007, we reported that "if the main objective of the summary is to maximize recall of text inclusions, i.e., produce the highest IN score, then 25x is an excellent method, with its 0.87 mean (0.92 median) far outstripping …all other NIST submitted runs" [2]. We echo that same conclusion: if the main objective of the summary is to provide excellent coverage of the original video to maximize recall of text inclusions, then 50x is an excellent method. There is a limit on N for Nx speed-ups, and we have gathered some empirical evidence from the TRECVID 2007 rush video test set that 100x does not work as well as 50x [2], but at 50x, we see confirmation in the results of Figure 2 that users can capably perform on the inclusion task armed with 50x summaries or the derivatives CMU.1 and CMU.2 that were based on 50x.

Such excellent performance comes at a cost: the time on task TT metric for 50x, CMU.1, and CMU.2 averaged 59.59, 56.66, and 56.43 seconds, respectively, the slowest, third slowest, and fourth slowest of all of the NIST graded runs. The lengths of the summaries were backfilled to be 2% summaries, on purpose to focus attention on differences in summary make-up rather than summary length. The average length was 31.31 seconds, so that the task time was about double the video summary duration. To accomplish greater than 75% inclusion score, the viewer had to invest 4% of the time that it would take to view the full video, so while these CMU.1, CMU.2, and 50x baseline summaries did slow down time on task, there is still tremendous savings over watching the original full video: 4% vs. 100%.

We anticipated a separation of our two CMU runs and the 50x baseline on the subjective metrics JU (summary contained lots of junk, 1 strongly agree – 5 (best) strongly disagree), RE (summary contained lots of duplicate video: 1 strongly agree – 5 (best) strongly disagree), and TE (summary had a pleasant tempo/rhythm: 1 strongly disagree – 5 (best) strongly agree). Table 1 shows the results. The baseline scored the absolute lowest for all NIST graded runs on these measures, with the CMU runs likewise scoring at the bottom of the scale for RE and TE. The one metric where we attempted to improve based on some cleverness in better clapper recall was in the JU metric, and with JU the CMU.1 and CMU.2 separate themselves from the baseline, but not from each other. Hence, the increased aggressiveness in removing potential clapper shots with CMU.2 over CMU.1 was not distinguishing enough to produce significant differences in any of the collected metrics IN, TT, JU, RE, and TE.

2. Event Detection

- **CMU** Our single event detection submission found interest points, clustered them into visual keywords and use a classifier to detect activities based on trained SVM models. Segmentation was done in a multi-resolution framework, where all activity durations found in training were tried

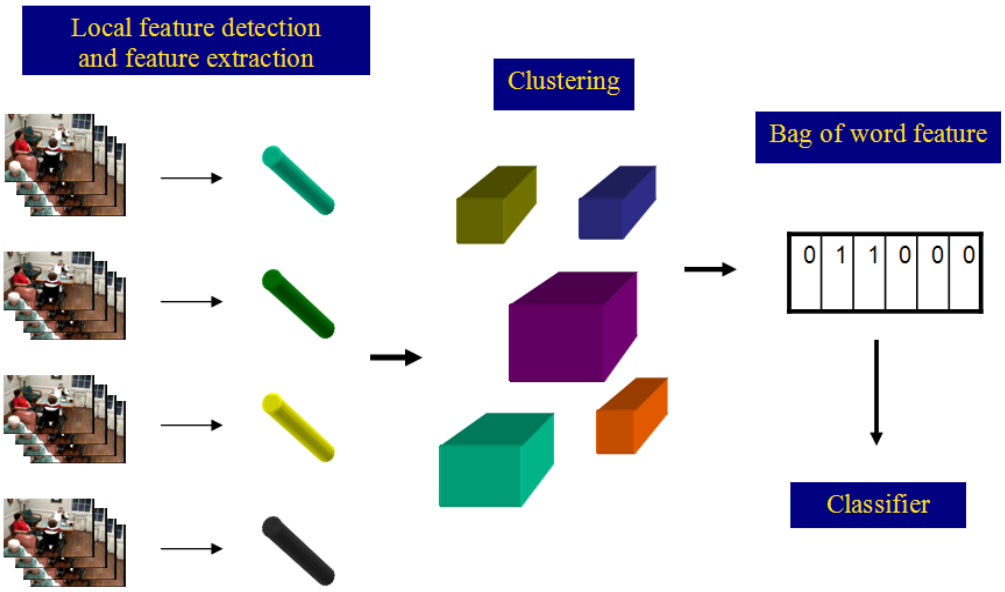

Figure 1: Framework of the proposed human behavior recognition. It includes 3 major stages: (1) feature point detection and video cube extraction, (2) clustering and bag-of-words representation based on video codebooks, and (3) classification to achieve behavior recognition.

2.1 Human Behavior Detection/Recognition

We characterize human behaviors in surveillance video through the use of spatio-temporal video cubes. A spatio-temporal video cube is a small, short and localized video sequence extracted around an interest point to capture and represent a small but informative motion in the video. These small motions can be movements such as raising a finger, bending a knee or lips moving. We assume that a behavior can be described though a combination of these different types of small movements. Since the extracted cubes are small, we believe they can capture local appearance and are less affected by global appearance, posture, illumination and occlusion. Therefore, the main problem of comparing the similarity of two behaviors transforms to a search for similar, conceptually meaningful components exhibited in the video. Figure 1 illustrates our framework.

2.2 Interest point detection and video cube extraction

In object recognition, there are a variety of methods to detect interest points. Typically, a response function is calculated at every pixel in the image and an interest point is a local maxima pixel corresponded to the response function. A popular response function is to detect corner points. In object detection in 2D images, this is a spatial corner. In video, a spatio-temporal corner can be defined as a spatial corner which also contains non-constant movements in a temporal sequence. The extended Harris corner detector [11] embodies a very simple and elegant algorithm. The basic idea is to extend gradients not only along x and y, but also along t (x and y are coordinates in space and t is the coordinate in time). The spatio-temporal corners are defined as regions with strong local gradients in orthogonal directions along x, y and t. Therefore, a spatio-temporal corner is a spatial corner in video images whose velocity vector is changing. Although the extended Harris corner detection is simple and elegant. In practice, spatio-temporal corners are quite rare. This has been proved troubling in detection and recognition tasks observed by Lowe [12]. Therefore, another local sptaio-temporal interest point detector was proposed to detect periodic movements

[13]. It applies 1D Gabor filters temporally and tries to capture periodic motions. This provides a richer set of features but many complicated actions cannot be represented merely by periodic motions only.

Our proposed interest point detection is also inspired by the Harris corner detector. Instead of corner points in spatial positions, we extract points along edges with velocity vectors by simply replacing the 2nd moment gradient matrix with gradient magnitudes of x, y and t. The goal is to find high contrast points both in space and time. This identifies the points which are along edges in a video image and contain velocity. This approach will provide very dense features rather than sparse features resulting from the extended Harris detector. It will also contain points with all different kinds of motions rather than only periodic motions.

The proposed formula for interest point calculation is as follows:

$$L(x, y, t, \sigma_x, \sigma_y, \sigma_t) = I(x, y, t) * g(x, y, t, \sigma_x, \sigma_y, \sigma_t)$$

$$G_x = \left(\frac{\partial L(x, y, t, \sigma_x, \sigma_y, \sigma_t)}{\partial x} \right)$$

$$G_y = \left(\frac{\partial L(x, y, t, \sigma_x, \sigma_y, \sigma_t)}{\partial y} \right)$$

$$G_t = \left(\frac{\partial L(x, y, t, \sigma_x, \sigma_y, \sigma_t)}{\partial t} \right)$$

$$R(x, y, t) = \sqrt{G_x^2 + G_y^2 + G_t^2}$$

L() denotes a smoothed video. I() is the original video and g() is the Gaussian smoothing kernel. $\sigma_x, \sigma_y, \sigma_t$ control smoothing scales in space and temporal sequence. Gx, Gy, and Gt are the gradients among x, y and t. The response function, R(), combines these 3 magnitudes. We calculate the response function value for each pixel and extract interest points as local maxima pixels. The gradient over time fulfills a similar function as background subtraction to remove still background pixels and preserve moving objects.

At each interest point, a cube is extracted which contains the spatially and temporally windowed pixel values. The window size is normally set to contain most of the volume of data that contributed to the response function. In other words, the window size corresponds to σ_s and σ_t. Gray scaling and normalization are applied to pixel values in cubes to make the features invariable to small translations, slight variations in appearance or motion, changes in lighting and so on. We extracted brightness gradients in x, y and t directions to present video cubes.

2.3 Clustering and video codebook creation

Our approach is based on the assumption that although two instances of the same behavior may vary significantly in their overall appearance, many of the video cubes extracted from them will be similar. According to this assumption, even though the number of cubes extracted from each instance is different, they can be mapped into a set of similar "types". We have all observed how easily children can use a small number of types of LEGO blocks to build many different objects. Based on the same idea, we decompose a behavior into different components represented by types of video cubes. Analogously, in our behavior recognition task, the individual video cube becomes unimportant, but its conceptual type matters. This solves an important problem. Since two instances of the same behavior may vary a lot, the number of cubes extracted from each instance can be quite different. If we want to measure the similarity of two instances, having different numbers of cubes extracted from each instance makes any comparison extremely hard. We may then need to model the distribution of the video cubes in each instance and compare the two distributions. However, if we transform video cubes into a fixed number of types, we can easily construct a feature vector for each behavior instance. A simple similarity function can then be applied to these two vectors to easily obtain a similarity comparison.

There are two essential questions: How many types are enough to describe behaviors? How can these types be discovered? We try to discover these cube prototypes with an unsupervised learning algorithm. We apply a clustering algorithm on video cubes extracted from video data to search for cube prototypes. We assume every cluster corresponds to one cube type. However, determining the number of types/clusters is

still an unsolved problem. Currently, we use a large number of clusters which previous experimental results show won't hurt your recognition performance [14], but may be inefficient. In practice, cross validation is widely used to determine the optimal number of clusters. Using cluster prototypes is a very simple and powerful method to reduce variability of the data while maintaining its richness. After we finish our clustering, each video cube is assigned to one of the known prototypes. We refer to these cluster prototypes as our "video codebook". Every cluster/prototype is a video code word in the codebook. Using this video codebook, we can furthermore extract video cubes from new data by the same surveillance source and encode every new incoming video a collections of prototypes. Our hope is that each code servers to describe a specific part of some behaviors and has its own conceptual meaning. In the case of facial expression detection, some clusters denote lips movement and eyebrow raising [15].

2.4 Behavior descriptor and classification

After we extract video cubes from the video, we can discard the original video and use the video cubes to characterize behaviors. Our video codebook maps video cubes into their closest prototypes. A behavior descriptor represents the distribution of types for the current event. In a straightforward manner, we could use a histogram to describe this type distribution. However, we decided to borrow an idea from document classification. We treat each code in the video codebook as if it were a word in a document. In text classification, we typically have different length documents, using a limited number of words. For our behavior descriptors, we now have the same characterization. Different behaviors can vary in global appearance but they are represented by a fixed number of cube types. We choose also to use TF or TF/IDF to represent our type distribution for an event in our descriptors. A behavior descriptor is thus a vector that represents the type distribution for the current event. To measure the similarity of two instances, a distance function such as the Euclidean or X^2 distance can be simply applied. In our framework, we use Support Vector Machines (SVM) to achieve high recognition accuracy when given training data where only the presence absence of a behavior is labeled in time. However, this behavior descriptor is a key building block in the video retrieval system that uses behaviors as examples to retrieve similar events.

In our system, we need labeled training data to build our SVM models as behavior classification models. The labeled data isn't provided in the form of bounding boxes to locally circumscribe the events. We only need the information whether a certain action occurred in current segment or not. After we construct our behavior models, we can simply extract video cubes from new incoming video and represent them with behavior descriptors. SVM behavior models will then be applied to recognize behaviors from the descriptors.

We have a key assumption that independent video cubes and cube prototypes provide enough information to characterize human behaviors in our work. Therefore, we are discarding the relative positions between video cubes. Some research has used position information integrated with the cubes [18]. Because we focus on large data archives, we avoid modeling relative positions since this adds significant computation to the process. However, we are still looking for efficient ways to achieve this in our future work.

2.5 System setting

In our system, we extract 5x5x10 video cubes from interest points. $\sigma_x, \sigma_y, \sigma_t$ are 5, 5 and 10 respectively for the Gaussian smoothing kernel. The raw pixel feature we generated from video cubes thus contains 250 dimensions. These raw pixels are gray-scaled and normalized to achieve moderate invariance to lighting and environment. We apply brightness gradients to video cubes and extract 64 dimensional (4 different orientations in x, y and z coordinates respectively) features to present video cubes. We built our video codebook by 500 video words. For the video code word descriptor, a term frequency-inverse document frequency (tf-idf) representation is used to generate a bag of words feature. We apply radian basis kernel support vector machine (RBF-SVM) to train recognition models. Since we have multiple actions in both dataset, we adopt a one-against-all strategy in the SVM classifiers.

We apply sliding window with different scale to predict evaluation set. We analyze mean and variety of each activity and set up different scale level to scan though evaluation videos. The detection result is in the table 3.

	Act. RFA	Act. PMiss	Act. DCR	Min RFA	Min PMiss	Min DCR	Det Score
CellToEar	488.56	0.92	3.36	9.66	0.99	1.04	0.448
ElevatorNoEntry	0.97	--	--	--	--	--	--
Embrace	389.68	0.79	2.74	23.14	0.99	1.11	0.179
ObjectPut	1068.64	0.83	6.18	76.21	0.99	1.38	1.000
OpposingFlow	2.93	1.00	1.01	2.93	1.00	1.01	0.102
PeopleMeet	1360.74	0.69	7.49	3.42	0.99	1.01	0.426
PeopleSplitUp	826.38	0.72	4.85	0.00	0.99	0.99	0.967
PersonRuns	390.25	0.87	2.83	1.78	0.99	1.00	0.434
Pointing	1555.74	0.84	8.62	142.51	0.99	1.70	1.000
TakePicture	0.97	1.00	1.00	0.97	1.00	1.00	0.018

Table 3: RFA denotes Rates of False Alarms. PMiss denotes probability of missed event. DCR denotes Detection Cost Rate. DetScore indicates the detection score from TRECVID evaluation.

3. High-Level Feature Extraction

Our 6 submissions to the high-level semantic features, all trained on TRECVID'05 data are:

- **a_CMU.mostrel_gen_linear**: model selected using a universal generalizability model trained with SVR with linear kernel.
- **a_CMU.mostrel_gen_rbf**: model selected using a universal generalizability model trained with SVR with RBF kernel.
- **a_CMU.mostrel_fea_linear**: model selected using feature-specific generalizability model trained with SVR with linear kernel.
- **a_CMU.mostrel_fea_rbf**: model selected using feature-specific generalizability model trained with SVR with RBF kernel.
- **a_CMU.voting** which aggregates the results of the other 5 submissions using weighted voting.
- **a_CMU.Best_CV** constituted the baseline run based on cross-validation.

In high-level feature (HLF) extraction task, our approach is to build models from "out-of-domain" data and then to choose from them the ones that generalize (relatively) well to the test data in TRECVID 2008 corpus. Instead of being trained on the TRECVID 2008 development set , the models used in all of our 6 runs are trained exclusively from the TRECVID 2005 development set and labels provided by LSCOM. Therefore, all of our runs belong to category (a). Their performance are considerably lower than that of runs of type (A), (B), and (C), because their training data are news video while the test data consist of footage of news magazines, documentaries, and educational programming.

For each high-level feature, we build a set of SVM models with RBF kernel using different features and model parameters. There are 7 types of low-level features, and 4 of them are provided by Columbia University, which are grid-based color comments (GCM), Gabor texture (GBR), edged direction histogram (EDH), and their combination (Columbia374). The other 3 features are grid-based HVC color histogram (3hvc) and its compact version (3hvc.hstat), and SIFT-based bag-of-words feature (vw2000). Moreover, we use 3 values of cost factor (1, 3, and 10) and 6 values of gamma parameter (0.01, 0.05, 0.1, 0.2, 0.4, and 1) of the RBF kernel function. This results in a total of 126 models for each high-level feature. For each model configuration, we compute its performance in TRECVID 2005 development data using 5-fold cross-

validation, denoted as AP_{TV08}. Our goal, however, is to choose the model with the highest performance in TRECVID 2008 test data, denoted as AP_{TV08}. Obviously, we cannot compute AP_{TV08}, and we cannot even approximate it using the performance on TRECVID 2008 development data, since to be type (a) submissions no labels in TRECVID 2008 data are allowed to be used. One thing we are sure is that AP_{TV08} is most likely lower than AP_{TV08} due to the difference on data types.

Meta-feature	**Description**	**Correlation with decline**
pos_ratio	the ratio of positive data in training data	-0.60
SV_ratio	the ratio of support vectors in training data	-0.70
pos_SV_ratio	the ratio of support vectors in positive training data	0.21
score_range	the gap b/w the largest and smallest model output	-0.66
max_score	the largest model output on test data	-0.62

Our baseline run (**CMU_Best_CV**) selects the model with the highest AP_{TV08}, in the hope that this model also has the highest AP_{TV08}. However, our preliminary study suggests that this is not always the case. A model with high AP_{TV08} may not generalize well to TRECVID 2008 data and end up with low AP_{TV08}. The key here is to know the generalizability of each model, which is measured by the *relative decline* between the two performance, i.e., $\frac{(AP_{TV08} - AP_{TV08})}{AP_{TV08}}$. We find that this decline has strong correlations with the meta-features of a SVM model, including the ratio of positive data, the ratio of support vectors in the model, and the distribution of the model's output in the test data. We have summarized these meta-features and their correlation coefficient with decline in Table ?.

We build a regression model to predict the decline from AP_{TV08} to AP_{TV08} based on the meta-features in Table ?. We use support vector regression (SVR) as the learning algorithm. To avoid overfitting, this generalizability model is trained using another set of high-level features that do not overlap with those used in TRECVID 2008. Using the predicted decline, we can easily compute the estimated performance on TRECVID 2008 data as $\widehat{AP}_{TV08}$, and select the model with the highest $\widehat{AP}_{TV08}$ to generate the results. We generate 4 submissions using this generalizability-based model selection strategy. They differ in two implementation decisions regarding the generalizability model: 1) whether to train a universal model for all the features, or feature-specific models for each feature; and 2) whether to use linear or RBF kernel in the regression model. These 4 submissions are:

- **a_CMU.modsel_gen_linear**: model selected using a universal generalizability model trained with SVR with linear kernel.
- **a_CMU.modsel_gen_rbf**: model selected using a universal generalizability model trained with SVR with RBF kernel.
- **a_CMU.modsel_fea_linear**: model selected using feature-specific generalizability model trained with SVR with linear kernel.
- **a_CMU.modsel_fea_rbf**: model selected using feature-specific generalizability model trained with SVR with RBF kernel.

We also include a submission (**a_CMU.voting**) which aggregates the results of the other 5 submissions using weighted voting. As the evaluation results show, this voting run performs the best, followed by the model selection run based on universal generalizability model trained with RBF kernel (**a_CMU.modsel_gen_rbf**), and then the baseline run based on cross-validation. This shows that one can do better model selection based on careful analysis of generalizability than based on cross-validation performance, although the difference is relatively small.

4. Acknowledgments

This work was supported by the National Science Foundation under Grant No. IIS-0205219 and Grant No. IIS-0705491.

5. REFERENCES

[1] Chen, F., Cooper, M., and Adcock, J. Video Summarization Preserving Dynamic Content. In Proc. ACM Workshop on TRECVID Video Summarization (Augsburg, Germany, Sept. 2007), 40-44.

[2] Christel, M.G., Lin, W.-H., and Maher, B. Evaluating Audio Skimming and Frame Rate Acceleration for Summarizing BBC Rushes. In Proc. CIVR (Niagara Falls, July 2008).

[3] Christel, M.G., Smith, M.A., Taylor, C.R, & Winkler, D.B. Evolving Video Skims into Useful Multimedia Abstractions. In Proc. ACM CHI '98 (Los Angeles, April 1998), 171-178.

[4] Hauptmann, A.G., Christel, M.G., Lin, W.-H., Maher, B., Yang, J., Baron, R.V., and Xiang, G. Clever Clustering vs. Simple Speed-Up for Summarizing BBC Rushes. In Proc. ACM Workshop on TRECVID Video Summarization (Augsburg, Germany, Sept. 2007), 20-24.

[5] Lowe, D. G. Object Recognition from Local Scale-Invariant Features. In Proc. International Conf. on Computer Vision (Kerkyra, Greece, Sept. 1999), 1150-1157.

[6] Over, P., Smeaton, A.F., and Awad, G. The TRECVID 2008 BBC Rushes Summarization Evaluation. In TVS '08: Proc. ACM Workshop on TRECVID Video Summarization (Vancouver, BC, Canada, Oct. 2008), 1-20.

[7] Over, P., Smeaton, A.F., and Kelly, P. The TRECVID 2007 BBC Rushes Summarization Evaluation Pilot. In Proc. ACM Workshop on TRECVID Video Summarization (Augsburg, Germany, Sept. 2007), 1-15.

[8] Song, Y., and Marchionini, G. Effects of Audio and Visual Surrogates for Making Sense of Digital Video. In Proc. ACM CHI '07 (San Jose, CA, April-May 2007), 867-876.

[9] Taskiran, C.M., Pizlo, Z., Amir, A., Ponceleon, D., and Delp, E. J. Automated Video Program Summarization Using Speech Transcripts. IEEE Transactions on Multimedia 8(4), 2006, 775-791.

[10] Wildemuth, B.M., Marchionini, G., Yang, M., Geisler, G., Wilkens, T., Hughes, A., and Gruss, R. How Fast Is Too Fast? Evaluating Fast Forward Surrogates for Digital Video. In Proc. Joint Conf. Digital Libraries (Houston, TX, May 2003), 221-230.

[11] Laptev, I. and Lindeberg, T. 2003, Space-time interest points, In ICCV, p. 432-439, 2003

[12] Lowe, D.G., 2004, Distinctive image features from scale invariant key points, IJCV, November 2004

[13] Dollar, P., Rabaud, V, Gottrell, G. and Belongie, S. 2005. Behavior Recognition via Sparse Spatio-Temporal Features, In VS-PETS 2005, page 65-72

[14] Yang, J., Jiang Y.G., Hauptmann, A. and Ngo, C.W. 2007, Evaluating bag-of-visual-word representation in scene classification, MIR'07 ACMMM, September 2007

[15] Fergus, R., Perona, P., and Zisserman, A. 2003, Object class recognition by unsupervised scale-invariant learning, *In CVPR,* 2003