# Informedia @ TRECVID 2011

**Lei Bao**[1,2,3], **Shoou-I Yu**[1], **Zhen-zhong Lan**[1], **Arnold Overwijk**[1], **Qin Jin**[1],
**Brian Langner**[1], **Michael Garbus**[1], **Susanne Burger**[1], **Florian Metze**[1], **Alexander Hauptmann**[1]
[1]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]Laboratory for Advanced Computing Technology Research, ICT, CAS, Beijing 100190, China
[3]Graduate University of Chinese Academy of Sciences, Beijing 100049, China

## Abstract

The Informedia group participated in three tasks this year, including: Multimedia Event Detection (MED), Semantic Indexing (SIN) and Surveillance Event Detection. Generally, all of these tasks consist of three main steps: extracting feature, training detector and fusing. In the feature extraction part, we extracted a lot of low-level features, high-level features and text features. Especially, we used the Spatial-Pyramid Matching technique to represent the low-level visual local features, such as SIFT and MoSIFT, which describe the location information of feature points. In the detector training part, besides the traditional SVM, we proposed a Sequential Boosting SVM classifier to deal with the large-scale unbalance classification problem. In the fusion part, to take the advantages from different features, we tried three different fusion methods: early fusion, late fusion and double fusion. Double fusion is a combination of early fusion and late fusion. The experimental results demonstrated that double fusion is consistently better, or at least comparable than early fusion and late fusion.

## 1 Multimedia Event Detection (MED)

### 1.1 Feature Extraction

In order to encompass all aspects of a video, we extracted a wide variety of visual and audio features as shown in figure 1.

Table 1: Features used for the MED task.

| | Visual Features | Audio Features |
|---|---|---|
| Low-level Features | • SIFT [19]<br>• Color SIFT [19]<br>• Transformed Color Histogram [19]<br>• Motion SIFT [3]<br>• STIP [9] | Mel-Frequency Cepstral Coefficients |
| High-level Features | • PittPatt Face Detection [12]<br>• Semantic Indexing Concepts [15] | Acoustic Scene Analysis |
| Text Features | Optical Character Recognition | Automatic Speech Recognition |

### 1.1.1 SIFT, Color SIFT (CSIFT), Transformed Color Histogram (TCH)

These three features describe the gradient and color information of a static image. We used the Harris-Laplace detector for corner detection. For more details, please see [19]. Instead of extracting features from all frames for all videos, we first run shot-break detection and only extract features from the keyframe of a corresponding shot. The shot-break detection algorithm detects large color histogram differences between adjacent frames and a shot-boundary is detected when the histogram difference is larger than a threshold. For the 16507 training videos, we extracted 572,881 keyframes. For the 32061 testing videos, we extracted 1,035,412 keyframes.

Once we have the keyframes, we extract the three features by the executable provided by [19]. Given the raw feature files, a 4096 word codebook is acquired using the K-Means clustering algorithm. According to the codebook and given a region in an image, we can create a 4096 dimensional vector representing that region. Using the Spatial-Pyramid Matching [10] technique, we extract 8 regions from an keyframe image and calculate a bag-of-words vector for each region. At the end, we get a $8 \times 4096 = 32768$ dimensional bag-of-words vector. The 8 regions are calculated as follows.

- The whole image as one region.
- Split the image into 4 quadrants and each quadrant is a region.
- Split the image horizontally into 3 equally sized rectangles and each rectangle is a region.

Since we only have feature vectors describing a keyframe, and a video is described by many keyframes, we compute a vector representing a whole video by averaging over the feature vectors from each keyframe. The features are then provided to a classifier for classification.

### 1.1.2 Motion SIFT (MoSIFT)

Motion SIFT [3] is a motion-based feature that combines information from SIFT and optical flow. The algorithm first extract SIFT points, and for each SIFT point, it checks whether there is a large enough optical flow near the point. If the optical flow value is larger than a threshold, a 256 dimensional feature is computed for that point. The first 128 dimensions of the feature vector is the SIFT descriptor, and the latter 128 dimensions describes the optical flow near the point. We extracted Motion SIFT by calculating the optical flow between neighboring frames, but due to speed issues, we only extract Motion SIFT for the every third frame. Once we have the raw features, a 4096 dimensional codebook is computed, and using the same process as SIFT, a 32768 dimensional vector is created for classification.

### 1.1.3 Space-Time Interest Points (STIP)

Space-Time Interest Points are computed using code from [9]. Given the raw features, a 4096 dimensional code is computed, and using the same process as SIFT, a 32768 dimensional vector is created for classification.

### 1.1.4 Semantic Indexing (SIN)

We predicted the 346 semantic concepts from Semantic Indexing 11 onto the MED keyframes. For details on how we created the models for the 346 concepts, please refer to section 2. Once we have the prediction scores of each concept on each keyframe, we compute a 346 dimensional feature that represents a video. The value of each dimension is the mean value of the concept prediction scores on all keyframes in a given video. We tried out different kinds of score merging techniques, including mean and max, and mean had the best performance. These features are then provided to a classifier for classification.

### 1.1.5 Face

We ran face detection over all videos using the PittPatt Face Detection software [12], and extracted information on the location of the face, the size of the face and whether the face is frontal or profile. In order to speed up the process, we sample 10 frames per second from each video and only perform face detection on the sampled frames. From the extracted face information, we create a 9 dimensional vector where the meaning of each dimension is as follows.

1. Number of faces in the video divided by the total number of frames

2. Maximum number of faces in a frame in the whole video.
3. Number of frames with more than (including) one face divided by the total number of frames.
4. Number of frames with more than (including) two faces divided by the total number of frames.
5. Number of frontal faces divided by the total number of faces.
6. The median of the ratio $\frac{\text{face width}}{\text{frame width}}$ for all faces in the video.
7. The median of the ratio $\frac{\text{face height}}{\text{frame height}}$ for all faces in the video.
8. Number of frames in the center of the frame divided by total number of faces. If $w$ and $h$ is the width and height of the video respectively, and let $(x, y)$ be the location of the center of a face, then the face is in the center of the frame if $\frac{w}{4} \leq x \leq \frac{3\times w}{4}$ and $\frac{h}{4} \leq y \leq \frac{3\times h}{4}$.
9. Median of the confidences of all faces in the video.

We did not perform face tracking.

#### 1.1.6 Optical Character Recognition (OCR)

We used the Informedia system [5] to extract the OCR. We extracted OCR at a sample rate of 10 frames per second. For details of the OCR process, please refer to [11]. Once we have the OCR output, we create TF-IDF [13] bag-of-words features for each video. Since OCR rarely gets a word completely correct, the vocabulary we use here is a trigram of characters. For example the word "rarely" will be split into "rar", "are", "rel" and "ely". In this way, if one of the characters was miss recognized, there are still some trigrams that are correct.

#### 1.1.7 Automatic Speech Recognition (ASR)

We run automatic speech recognition using the Janus [17] and the Microsoft ASR system. Once completed, for each video, we combine the output of each system into one file and view it as a document. We then perform stemming using the Porter Stemmer, and calculate the TF-IDF [13] bag-of-words vectors for each video.

#### 1.1.8 Mel-Frequency Cepstral Coefficients (MFCC)

We extracted Mel-frequency cepstral coefficients (MFCC) features using the Janus system. Given the raw features, we treat the raw features as a computer vision feature (e.g. SIFT) and run the MFCC features through the same computer vision pipeline. Therefore, we compute a 4096 word codebook and aggregate all MFCC features in one video to create a 4096 dimensional bag-of-words vector. Spatial Pyramid Matching is not reasonable here, so it is not applied.

#### 1.1.9 Acoustic Scene Analysis (ASA)

An expert manually annotated about 3 hours ($\approx$ 120 files) of videos with 42 semantic concepts, which can be derived from the audio: a small ontology links the annotated "small engine" sound concept to video concepts, and the words mentioned in the event kits. Using these labels, we trained 42 Gaussian Mixture Models, which we connected as an ergodic Hidden Markov Model, used to decode the test data with Viterbi. The symbol sequence generated by this step is treated as a bag-of-word, and fed into an SVM classifier.

#### 1.1.10 Performance of features

Table 2 and 3 show the performances of the above features when we use non-linear support vector machine as classifier. The mean minNDC score on 10 Events is used to measure their performances. The smaller mean minNDC score means the better performance. From Table 2, we can find that:

- Generally, comparing low-level visual features, high-level visual features and text visual features, low-level visual features work best.
- Comparing three kinds of image-based low-level features: SIFT, CSIFT and TCH, SIFT describes the gradient information, TCH describes the color information and CSIFT describes both gradient and color information. The performance of TCH is much worse than

SIFT. It means the gradient information is more discriminant than color information in MED task and also explains why the performance of CSIFT is slightly worse than SIFT.

- Comparing two kinds of motion-based feature: MoSIFT and STIP. MoSIFT works around 8% better than STIP, which indicates MoSIFT is a better motion-based feature for MED task.
- Comparing high-level based feature SIN with low-level features, the performance of SIN is comparable to CSIFT and MoSIFT features, better than TCH and STIP, and around 6% worse than SIFT. Generally, with only SIN feature, the system also can get a reasonable performance.

Table 2: The performances of visual features

| features | SIFT | CSIFT | TCH | MoSIFT | STIP | SIN | Face | OCR |
|---|---|---|---|---|---|---|---|---|
| mean MinNDC | 0.689 | 0.717 | 0.778 | 0.724 | 0.782 | 0.730 | 0.985 | 0.90 |

From Table 3, we can find that:

- Generally, audio features work worse than visual based features. However, these two kind of features are very complementary. When we just simply combined audio and visual features by average late fusion, the mean minNDC can be improved around 12%. It decreased from 0.600 to 0.528.
- Comparing low-level audio feature(MFCC), high-level audio feature (ASA) and text feature (ASR), low-level audio feature works best.
- Comparing high-level audio feature ASA with high-level visual feature SIN, ASA is much worse than SIN. It is only slightly better than random. The reason could be that we only have 42 audio concepts and there are not enough to describe the 10 events, however, the SIN provides 346 visual concepts which are reasonable large enough to describe the 10 events.

Table 3: The performances of audio features

| features | MFCC | ASA | ASR |
|---|---|---|---|
| mean MinNDC | 0.805 | 0.981 | 0.897 |

### 1.2 Classifier Training and Fusion

A large variety of classifiers exist for mapping the feature space into score space. In our final submission, three classifiers are adopted, i.e. non-linear support vector machine (SVM) [2] and kernel regression (KR), and Sequential Boosting SVM in Section 2.2. SVM is one of the most commonly used classifier due to its simple implementation, low computational cost, relatively mature theory and high performance. In TRECVID MED 2010, most of the teams [8] [7] use SVM as their classifiers. Compared to SVM, KR is a simpler but less used algorithm. However, our experiment shows that the performance of KR is consistently better than the performance of SVM.

For combining features from multiple modalities and the outputs of different classifiers, we use three fusion methods, which are early fusion, late fusion and double fusion.

Early Fusion [4] is a combination scheme that runs before classification. Both feature fusion and kernel space fusion are example of early fusion. The main advantage of early fusion is that only one learning phase is required. Two early fusion strategies, i.e., rule-based combination and multiple kernel learning [16], have been tried to combine kernels from different features. For rule-based combination, we use the average of the kernel matrix. Multiple kernel learning [16] is a natural extension of average combination. It aims to automatically learn the weights for different kernel matrix. However,our experimental results show that the performance of multiple kernel learning is only slightly better than average combination. Considering that average combination is much less time consuming than multiple kernel learning, average combination is used as our early fusion method for final submission.

In contrast to early fusion, late fusion [4] happens after classification. While late fusion is easier to perform, in general, it needs more computational effort and has potential to lose the correlation in mixed feature space. Normally, another learning procedure is needed to combine these outputs, but in general, because of the overfitting problem, simply averaging the output scores together yields better or at least comparable results than training another classifier for fusion. Compared to early fusion, late fusion is more robust to features that have negative influence. In our final submission, we use both average combination and logistic regression to combine the outputs of different classifiers.

In our system, we also use a fusion method called double fusion, which combines early fusion and late fusion together. Specifically, for early fusion, we fuse multiple subsets of single features by using standard early fusion technologies; for late fusion, we combine output of classifiers trained from single and combined features. By using this scheme, we can freely combine different early fusion and late fusion techniques, and get benefits of both. Our results show that double fusion is consistently better, or at least comparable than early fusion and late fusion.

Table 4: Comparison of classifiers, and Fusion Methods, MinNDC is used as evaluation criteria.

| Classifier | Early Fusion | Late Fusion | Double Fusion |
|---|---|---|---|
| SVM | 0.632 | 0.528 | 0.519 |
| Sequential Boosting SVM | 0.651 | 0.556 | 0.554 |
| KR | 0.585 | 0.516 | 0.506 |

### 1.3 Submissions

A submission for a MED 2011 event consists of a video list with scores and a threshold. The score for each video is computed by a classifier that is trained on a number of features in Section 1.1. We experimented with three different types of classifiers: SVM, Sequential Boosting and Kernel Regression in Section 1.2. For each classifier we explored early fusion, late fusion and a technique that we call double fusion in Section 1.2. Given the scores for each video per event, we have two methods to compute the actual threshold. The first method is a simple cutoff at 1800. This number guarantees us that the false alarm rate will be lower than 6%, because we return less than 6% of all videos. Moreover the number of videos that our system did not detect will be as low as possible within the 6% false alarm criteria. Notice that this method is very conservative and will likely not be close to the best possible threshold. The second method is to use the best threshold of our cross validation experiments in the training data. This method is obviously less conservative and turned out to be very unstable in weighted fusion techniques, but as we will see later it performs well for average fusion techniques.

#### 1.3.1 primary run

Since the primary run is the most important, we prefer it to be our best run, however we don't want to risk the chance that we are over-fitting on the training data and therefore the results have to be stable across different splits in our training data. Moreover the actual threshold also plays an important role in the evaluation and should therefore be stable as well. We believed that both our early and late fusion approach would have a decent performance, but from our experiments none of them was consistently better even though there sometimes was a significant difference between the two approaches. Double fusion on the other hand showed very promising results, better than or comparable to both early and late fusion. Moreover the actual thresholds seemed to be stable, based on the number of retrieved videos for each event. Therefore we decided to submit a double fusion run, using SVM and average fusion in the late fusion stage of double fusion, as our primary run. We chose SVM, because the average performance of our Sequential Boosting SVM and Kernel Regression experiments were very similar. Furthermore we preferred average fusion over a weighting scheme learned by logistic regression, because we believed that the weighting scheme was likely over-fitting to the training data.

#### 1.3.2 DoubleKernelLG

In our experiments performed on the training data, we got better performance using a weighting scheme learned by logistic regression for the late fusion part of double fusion. Also kernel regression gave slightly better results than the other classifiers. However we believe that this behavior might

be explained by over-fitting and therefore not transform to the testing data. Nonetheless it is worth being a non-primary run in case that it actually would be better.

#### 1.3.3 Double3ClassifierLG

This is slightly more conservative run compared to the DoubleKernelLG run, because we use a weighted combination learned by logistic regression of the three different classifiers: SVM, Sequential Boosting SVM and Kernel Regression. For the average performance across events, this does not make a significant difference on average. However it does reduce the variance within events, because not all three classifiers perform equally well for all events. Therefore it is a more stable run when we take individual events into account.

#### 1.3.4 Late3ClassifierAverage

The previous three submissions are all dependent on the early fusion performance, while this run omits that part completely. This gives us more diversity in our runs in addition to our believe that a late fusion approach is likely to perform within the 6% false alarm and 75% miss detection limits. By setting the actual threshold to 1800 videos, we have a high chance of satisfying this criteria. Similarly to the Double3ClassifierLG, we again performed an average fusion on the results from the three classifiers to reduce the variance within events.

### 1.4 Results

The results on the MED 2011 evaluation data are shown in Figure 1. We can see that all our runs have similar performance, because we use the same features across all runs. The slight differences in performance are therefore mainly due to overfitting, learning weights for different features in a logistic regression setting did harm our performance in the final evaluation data. On the other hand additional experiments showed that using kernel regression does perform better than SVM.

## 2 Semantic Indexing (SIN)

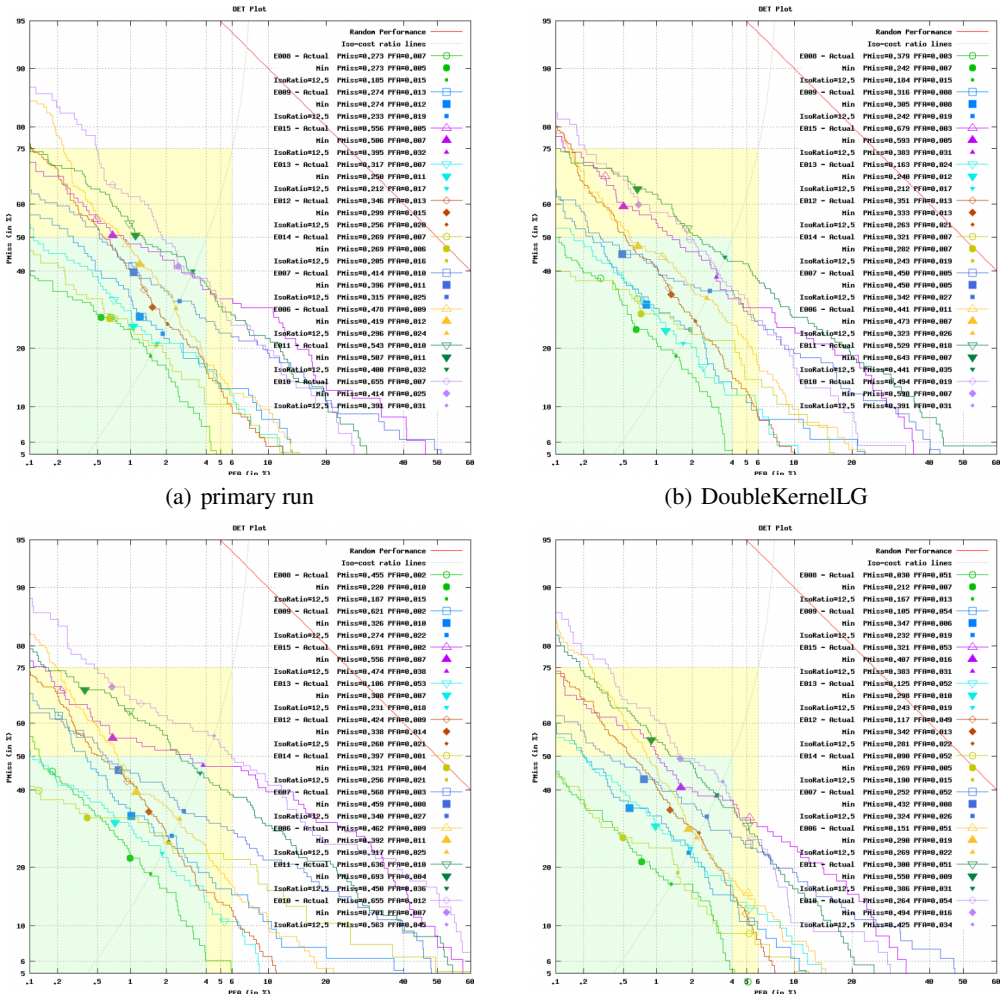
### 2.1 Feature Extraction

As we know, MED task focus on the multimedia content analysis on video level. SIN task focus on the video clips (shot) level. We can expect that, most of the useful low-level features in MED task can also be useful for SIN task. However, considering the time-consuming problem, we only used three most representative features for SIN task: SIFT, Color SIFT (CSIFT) and Motion SIFT (MoSIFT). SIFT and CSIFT describe the gradient and color information of images. MoSIFT describes both the optical flow and gradient information of video clips. Since the Harris-Laplace detectors only can detect a few feature points for some simple scenes such as sky, we also used dense-sampling detector to sample feature points besides Harris-Laplace detector. The more details about these features please refer to Section 1.1. Generally, these three features provide most of the useful information for SIN task.

### 2.2 Sequential Boosting SVM

#### 2.2.1 Problem Analysis

In feature extraction, we got the spatial bag-of-words feature representation for every shot. With these low-level feature representations, the most popular solution is to train a two-class non-linear kernel SVM classifier for every concept. However, as the increasing of training samples, this solution has some problems. For this year's SIN task, the development set includes around 11,000 videos and 26,000 shots. Obviously, we are facing a large-scale classification problem. As shown in the Figure 2(a), among 346 concepts, there are 152 concepts have over 50,000 labeled samples. Only 56 concepts have less than 10,000 labeled samples. The time cost becomes a big issue if we want to train a non-linear kernel SVM classifier on the over 50,000 training samples. Furthermore, the labeled samples for each concept are extreme unbalanced. In Figure 2(b), we analyzed the ratio between negative samples and positive samples. There are 65 concepts whose number of negative samples is over 1000 times than the number of positive samples. There are 189 concepts which the ratio between negative and positive samples is over 100. Only 46 concepts have reasonable balanced




(a) primary run

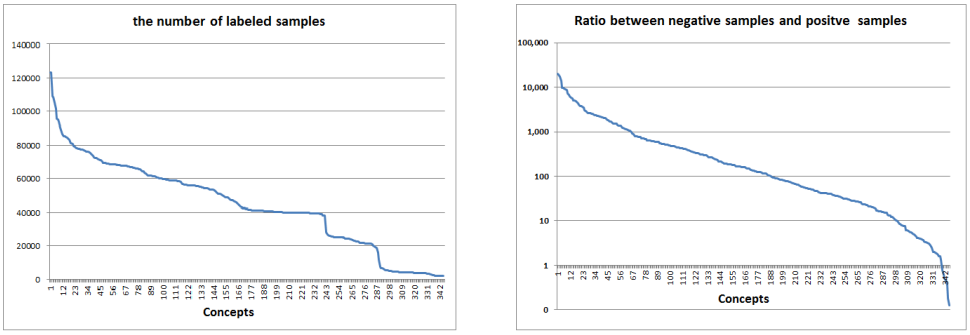
(b) DoubleKernelLG

(c) Double3ClassifierLG

(d) Late3ClassifierAverage

Figure 1: Preliminary results.

training data. Their ratios between negative and positive samples are between 10 and 0.1. As a result, the SVM's optimal hyperplane will be biased toward the negative samples due to the unbalance of training samples. Therefore, we proposed the Sequential Boosting SVM to deal with large-scale unbalance classification problem.




(a) the number of labeled samples

(b) ratio between positive and negative samples

Figure 2: SIN Label Analysis

### 2.2.2 Bagging and AdaBoost

The proposed Sequential Boosting SVM comes from the idea of Bagging and AdaBoost. The main ideas of Bagging and AdaBoost are:

- **Bagging** [1]: The basic idea of Bagging is to train multiple classifiers. The training samples for each classifier are generated by uniformly sampling with replacement. The final prediction is the combination by average the multiple classifiers.
- **AdaBoost** [6] [14]: The basic idea of AdaBoost is to train a sequence of weak classifiers by maintaining set of weights over training samples and adaptively updating these weights after each Boosting iteration: the samples that are misclassified gain weight while the samples that are classified correctly lose weight. Therefore, the future weak classifier will be forced to focus on the hard samples. Finally, the combination of these weak classifiers will be a strong classifier.

### 2.2.3 Sequential Boosting SVM

Intuitively, Bagging strategy can help us to solve the large-scale unbalance classification problem. Firstly, Bagging strategy divides the large-scale training problem to several smaller training problems. Each of them only contains a reasonable number of training examples. The training time cost will not a big issue. Meanwhile, to overcome the unbalanced problem, we can keep all of the positive examples and only execute the random sampling on negative examples. The number of sampled negative examples of each set is the same with the number of positive samples. Therefore, each classifier will be trained on a balanced number of positive and negative samples. This is the Asymmetric Bagging strategy proposed in [18]. However, since the training data for SIN task is extreme unbalanced and its size is large, in most of cases, the sampled examples for each bagging classifier cannot cover the whole training examples. This will hurt the final performance.

To improve the performance of bagging classifiers, an intuitive solution is to choose the most importance examples for each bagging classifier. Therefore, even the bagging classifiers only use a limited number of training examples, the sampled most importance examples already contain the most information of the whole training set. Inspired by the main idea of AdaBoost weighting [14], we proposed the Sequential Boosted Sampling strategy. The adaptively updating weights of training examples are used as a metric to measure the importance of training examples. Examples that can be easily misclassified get high possibility to be sampled. Examples that can be easily classified get low possibility. Therefore, the small classifier will focus on the hard examples, which will boost the performance even only a small part of training examples are used.

The algorithm of Sequential Boosting SVM is described in Algorithm 1.

## 2.3 Fusion

Generally, there are two kinds of fusion methods. One is early fusion, which combines different features before training classifier. The other is late fusion, which fuses the prediction scores of different features' classifiers. Considering the time cost to train classifier, we only took early fusion, which only need train a classifier. In order to explore multi-modal features, we also design a multi-modal Sequential Boosting SVM. In each layer, not only the training samples will be re-sampled by their weights, but also the using feature will be change Sequentially. Since we extracted five kinds of features for SIN task: MoSIFT spatial bag-of-word (MoSIFT), SIFT spatial bag-of-word by Harris-Laplace detector (SIFT-HL), SIFT spatial bag-of-word by dense sampling (SIFT-DS), Color SIFT spatial bag-of-word by Harris-Laplace detector (CSIFT-HL) and Color SIFT spatial bag-of-word by dense sampling (CSIFT-DS). We pre-computed the distance matrix between training data for all of these five features. In early fusion part, we just weighted fused their distance matrix. We tried several different kinds of fusion combination and got some combined features.

- **SIFT-HL-DS**: averagely fuse SIFT-HL and SIFT-DS;
- **CSIFT-HL-DS**: averagely fuse CSIFT-HL and CSIFT-DS;
- **MoSIFT-SIFT-CSIFT**: averagely fuse MoSIFT, SIFT-HL and CSIFT-HL;
- **MoSIFT-SIFT2-CSIFT2**: averagely fuse MoSIFT, SIFT-HL-DS and CSIFT-HL-DS.

**Algorithm 1:** Algorithm of Sequential Boosting SVM.

**Input**: positive example set $\mathbf{S}^+ = (x_1^+, y_1^+), \ldots, (x_{N^+}^+, y_{N^+}^+)$, where $y_i^+ = 1$; negative example set $\mathbf{S}^- = (x_1^-, y_1^-), \ldots, (x_{N^-}^-, y_{N^-}^-)$, where $y_i^- = 0$; SVM classifier $\mathbf{I}$; number of generated classifiers: $\mathbf{T}$; sample $\mathbf{K}^+$ positive examples and $\mathbf{K}^-$ negative examples in each iteration.

**begin**

$D_1^+(i) = 1/N^+$;

$D_1^-(i) = 1/N^-$;

**for** $t \leftarrow 1$ **to** ***T*** **do**

**Sample**:

- Sample positive example set $\mathbf{S}_t^+$ from $\mathbf{S}^+$ via distribution $D_t^+$, $|\mathbf{S}_t^+| = \mathbf{K}^+$;
- Sample negative example set $\mathbf{S}_t^-$ from $\mathbf{S}^-$ via distribution $D_t^-$, $|\mathbf{S}_t^-| = \mathbf{K}^-$;

**Train** SVM classifier: $C_t = \mathbf{I}(\mathbf{S}_t^+, \mathbf{S}_t^-)$;

**Predict**: $C^*(x_i) = \frac{1}{t}\sum_{p=1}^{t} C_p(x_i)$;

**Update**:

- $D_{t+1}^+(i) = \frac{D_t^+(i)}{Z_t^+} \times (1 - C^*(x_i^+))$, where $Z_t^+$ is a normalization factor (chosen so that $D_{t+1}^+$ will be a distribution);
- $D_{t+1}^-(i) = \frac{D_t^-(i)}{Z_t^-} \times (1 - C^*(x_i^-))$, where $Z_t^-$ is a normalization factor (chosen so that $D_{t+1}^-$ will be a distribution);

**Output**: classifier $C^*(x_i) = \frac{1}{\mathbf{T}}\sum_{p=1}^{\mathbf{T}} C_p(x_i)$

### 2.4 Submission

This year, we trained 4 different kinds of models and submitted 4 runs.

- **MoSIFT_model**: we used MoSIFT spatial bag-of-word feature and trained 10-layers Sequential Boosting SVM classifier. We submitted this as **CMU_1** run.
- **MoSIFT-SIFT-CSIFT_model**: we used MoSIFT-SIFT-CSIFT feature and trained 10-layers Sequential Boosting SVM classifier. We submitted this as **CMU_2** run.
- **MoSIFT-SIFT2-CSIFT2_model**: we used MoSIFT-SIFT2-CSIFT2 feature and trained 10-layers Sequential Boosting SVM classifier. We didn't submit this run.
- **MoSIFT-SIFT2-CSIFT2_multimodal**: we used MoSIFT, SIFT-HL-DS and CSIFT-HL-DS to train 20-layers multi-modal Sequential Boosting SVM. The order of the features is MoSIFT, SIFT-HL-DS and CSIFT-HL-DS. We submitted this run as **CMU_3** run.
- **MoSIFT-SIFT2-CSIFT2_latefusion**: we averagely fused the prediction scores from MoSIFT-SIFT2-CSIFT2_model and MoSIFT-SIFT2-CSIFT2_multimodal and submitted this as **CMU_4** run.

The performances of the above runs are in Table 5. As we can see:

- MoSIFT spatial bag-of-word feature is a good feature for SIN task. Only MoSIFT itself can help us get reasonable performance (mean infAP: 0.1064).
- SIFT-HL and CSIFT-HL are very complemental features for MoSIFT. After we combined SIFT-HL and CSIFT-HL feature with MoSIFT feature, the mean infAP is improved from 0.1064 to 0.1337. We got about 30% improvement from SIFT-HL and CSIFT-HL feature.
- SIFT-DS and CSIFT-DS can improve the performance of SIFT-HL and CSIFT-HL in SIN task. Based on CMU_2, after we combined SIFT-DS and CSIFT-HL, we got 5% improvement from 0.1337 to 0.1407.
- Multi-modal Sequential Boosting SVM works slightly better than early fusion. The performance of MoSIFT-SIFT2-CSIFT2_multimodal is 0.1464, which is 4% better than the performance of early fusion 0.1407.

Table 5: The performances of submissions

| Run_ID | model | mean infAP of 50 concepts |
|---|---|---|
| CMU_1 | MoSIFT_model | 0.1064 |
| CMU_2 | MoSIFT-SIFT-CSIFT_model | 0.1337 |
| | MoSIFT-SIFT2-CSIFT2_model | 0.1407 |
| CMU_3 | MoSIFT-SIFT2-CSIFT2_multimodal | 0.1458 |
| CMU_4 | MoSIFT-SIFT2-CSIFT2_latefusion | 0.1464 |

### 2.5 Future work

In feature part, we only tried three most representative visual features. Obviously, for some concepts in SIN task, such as Speech, Singing and Talking, the audio feature can be very useful. In our current experiment of MED task, MFCC bag-of-word feature works well for MED task and is very complementary to visual features. We will try MFCC audio feature to improve the current SIN performance. In classification part, Sequential Boosting SVM works well for SIN task. However, there is an open issue that how to decide the number of classifier layers. That will be another future work for SIN task. In fusion part, Multi-modal Sequential Boosting SVM is a good solution to combine different modalities. Currently, we used a fixed feature order. However, it will be an interesting question that how to choose the most useful feature for next layer classifier.

## 3 Surveillance Event Detection (SED)

Please refer to our another report Informedia@TRECVID 2011: Surveillance Event Detection.

## References


[1] L. Breiman and L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[3] M.-Y. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos, 2009.

[4] C. Cortes, M. Mohri, and A. Rostamizadeh. L 2 regularization for learning kernels. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 109–116. AUAI Press, 2009.

[5] D. Das, D. Chen, and A. G. Hauptmann. Improving multimedia retrieval with a video OCR. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6820 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, January 2008.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1997.

[7] G. Iyengar. Discriminative model fusion for semantic concept detection and annotation in video. *Proceedings of the eleventh ACM international Conference on Multimedia*, pages 255–258, 2003.

[8] Y. Jiang, X. Zeng, G. Ye, S. Bhattacharya, D. Ellis, M. Shah, and S. Chang. Columbia-UCF TRECVID2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching. In *NIST TRECVID Workshop*, 2010.

[9] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR (2)*, pages 2169–2178, 2006.

[11] H. Li, L. Bao, Z. Gao, A. Overwijk, W. Liu, L. Zhang, S. Yu, M. Chen, F. Metze, and A. Hauptmann. Informedia @ TRECVID 2010. *TRECVID Video Retrieval Evaluation Workshop, NIST*, 2010.

[12] PittPatt. Pittpatt face detection. http://www.pittpatt.com/.

[13] G. Salton and e. M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[14] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning*, pages 80–91, 1999.

[15] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVID. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[16] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM International Conference on Multimedia*, pages 399–402. ACM, 2005.

[17] H. Soltau, F. Metze, C. Fügen, and A. Waibel. A One-pass Decoder based on Polymorphic Linguistic Context Assignment. In *Proc. Automatic Speech Recognition and Understanding (ASRU)*, Madonna di Campiglio, Italy, Dec. 2001. IEEE.

[18] D. Tao, X. Tang, X. Li, and X. Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:1088–1099, July 2006.

[19] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.