

# Sort-Merge Feature Selection and Fusion Methods for Classification of Unstructured Video

Mitchell J. Morris and John R. Kender

Columbia University Department of Computer Science

## Abstract

We explore the problem of rapid automatic semantic tagging of video frames of unstructured (unedited) videos. We apply the Sort-Merge algorithm for feature selection on a large (>1000) heterogeneous feature set for videos showing lectures, to quickly locate low-level image features most predictive for concepts such as “key frame with text” or “key frame with computer source code”. For evaluation, we introduce a “keeper” heuristic for feature retention, which provides a baseline comparison. We then compare early fusion and late fusion of diverse feature types; based on experiments on 12,395 frames, we find that in general late fusion offers higher Average Precision accuracy at lower computation cost, compared to early fusion. However, mergers of redundant feature types do not necessarily improve performance over single feature types; exploration of both merged and unmerged performance is necessary.

**Index Terms**— *unstructured video analysis, feature selection, semantic tags, SVM, Sort-Merge*

## 1. INTRODUCTION

Classification and browsing of unstructured videos is an important problem. Many universities record course lectures and make them available to students over the internet. For example, each year the Columbia Video Network (CVN) manages a database of more than 2000 lectures distributed over 67 courses. Together they comprise about 3000 hours of unstructured video. However, production values are poor, and there is no post-production editing. Better access to such information can be supplied by enhanced video browser techniques. We hypothesize that tagging frame content with automatically learned semantic information will yield similar improvements.

### 1.1. Video Browsers

Although much work has been done on semantic browsers for edited videos, there is comparatively little work on unstructured videos, such as home videos or the casual capture of lectures, and even less on the automatic tagging of sections of unstructured videos [1]. VASTMM is a video browser designed to work with unstructured video [2].

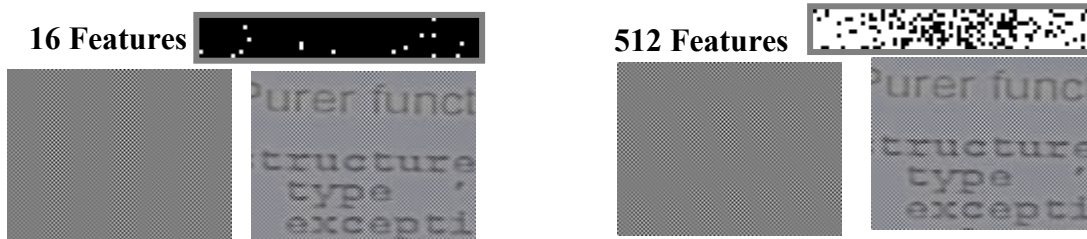
Virtual shots determined by foreground motion and by the similarity of spoken words are displayed along a timeline to assist in video browsing. Thumbnails of these key frames and a display of the words extracted from ASR are used to jump to cross-linked sections of interest. However, higher semantic concepts cannot be detected if the words are missing or missed in detection, yet anecdotal information suggests that users recall segments of interest by these tags, such as “text”, “code”, “graph”, “instructor’s underlining”, etc. Augmenting the timeline with automatically learned tags is expected to increase accuracy and reduce time of search, just as the ASR word index has.

### 1.2. Machine Learning Techniques

To label key frames with semantic tags we use an SVM implementation, LibSVM [3], to learn high-level concepts based on a very large library of extracted image features. SVMs are quadratic in the number of features, so we use Sort-Merge feature selection [4] to prune away almost all features, and extend that method by defining a definite stopping condition. Further, we use training example subsampling to speed up the costly wrapper selection method that comprises the Sort-Merge inner loop [5]. We use Average Precision at rank 100 as an empirical evaluation metric [6]. We find that even Sort-Merge can still run very slowly on such a large feature and large training set (about 48 hours at 2.8GHz on 700 features; but other feature selection methods, such as those based on forward selection, backward elimination, or genetic algorithms do not even finish). We therefore investigate hierarchical methods for combining features derived from partial analyses based on different types of low-level image detectors. The first method, early fusion, considers all detectors to be in a single homogeneous set, and Sort-Merge selects from their combined input. Late fusion does a full feature selection in each class of detectors first, then selects from the smaller set of “keepers” that result in each, before seeking a final set.

### 1.3. Sort-Merge

Sort-Merge is a heuristic approach to feature selection. Like forward selection and backward elimination [7], it is a wrapper method, but not an exhaustive one. Initially, it forms feature groups that consist of only a single feature. At each step of the selection process each group of features is



**Figure 1: An example of feature sets at two of the levels in the Sort-Merge tree while using the FFT feature to classify code given text. Each grouping chosen was the best performing feature set at that level. Each feature set is displayed in a truncated frequency domain (scaled up 2x for clarity, to the right of the title), as a spatial representation of the filter itself (left image), and as that representation superimposed over a typical example image from the dataset (right image).**

evaluated for classification performance. In the Sort step, the groups are then ranked in order of their performance. In the Merge step, adjacent pairs of ranked groups are merged into new feature groupings. Successive rounds of Sorting and Merging occur until a feature group is found that meets a predetermined condition. Sort-Merge is designed to work in very high dimensional feature spaces and outperforms more classical algorithms by several decimal orders of magnitude in speed, while retaining accuracy [4]. It appears to derive its power from the observation that if features are heavily abundant (and therefore redundant), extreme care in selection is not only unnecessary, it is counterproductive.

#### 1.4. Features

In the domain of unedited lecture videos, much content can be discriminated on the basis of its texture (for example, text, code segments, charts, images, etc.) Such textural elements of key frames have previously been recognized by classification systems using low-level texture features [8,9]. Wavelet features can be used to measure color, texture, or a combination of both. Therefore, as in [8], we exploit the observation that wavelet features are effective at discriminating text [10]. Fast Fourier Transforms can also be used to measure gross properties of texture and of shape [9], particularly if the appearance of regular features has been distorted through affine or perspective transformations, as is common in these videos of blackboards, slides, and screens.

## 2. APPROACH

### 2.1. Sort-Merge and Keeper Heuristic

The number of possible low-level image features is large, but few are relevant to a particular semantic tag. For example, our wavelet texture measure generates 567 features per image, of which only about 6% appear to be useful for detecting “code” key frames. The Sort-Merge feature selection algorithm works well on sets of such magnitude [4], but does not define a definite stopping condition, leaving that to the experimenter. We improve upon this method with the Keeper Heuristic.

This is a heuristic method of selecting good feature sets

based on their comparison to a baseline of the classification performance of the entire set of features. Since the entire set is usually redundant, its absolute performance is actually suboptimal. At each level of the Sort-Merge tree, any feature set of equal or greater performance than this baseline is saved as a “keeper”. When the Sort-Merge tree is completed, only these keepers (which usually vary in feature set cardinality) are tested against a validation set of images. The best performing one, regardless of feature set cardinality, is considered the best keeper.

### 2.2. Feature Fusion

Feature fusion is the method of combining multiple different features, for example, color features and texture features, in the feature selection process. However, Sort-Merge was designed to only use homogeneous feature sets. We enhance the Sort-Merge algorithm to deal with heterogeneous feature sets, but first define how relative performances among the different features are to be weighted. We explore both early fusion, which combines all the features from the different feature sets equally from the beginning and then Sort-Merges this combined set, and late fusion, which first selects the keepers from each feature type separately (where performance measures are fairer) and then Sort-Merges these smaller sets.

## 3. DETAILS OF EXPERIMENTS

### 3.1. Data Set

We use a data set taken from the Columbia Video Network. These videos consist of course lectures and student midterm and final exam presentations. We used 169 videos consisting of 172.8 hours. A segmentation algorithm developed for unstructured video as defined in [1], set to maximum granularity, generated 66130 virtual key frames. We employed random subsampling to speed the experiments, as in [5], and three-fold cross-validation for the SVM learning. Each fold therefore used 441 positive and 2864 negative examples in training the classifier; 6610 images in testing (to determine the ordering in the sort step of the Sort-Merge); and 2480 images in validation (to determine keepers at the end of the Sort-Merge algorithm).

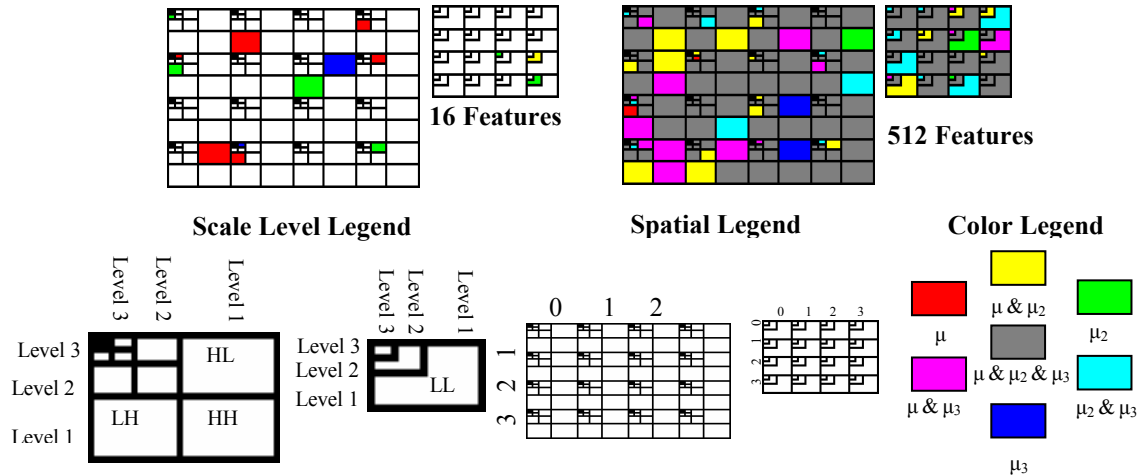


Figure 2: Best feature set at each level for the wavelet feature while classifying code, given images with text. Each grouping chosen was the best performing feature set of that level. The features chosen are as indicated by the legends at the bottom of the figure, where  $\mu$  represents the mean and  $\mu_n$  represents the  $n^{\text{th}}$  central moment of the pixel intensities of the resultant transform.

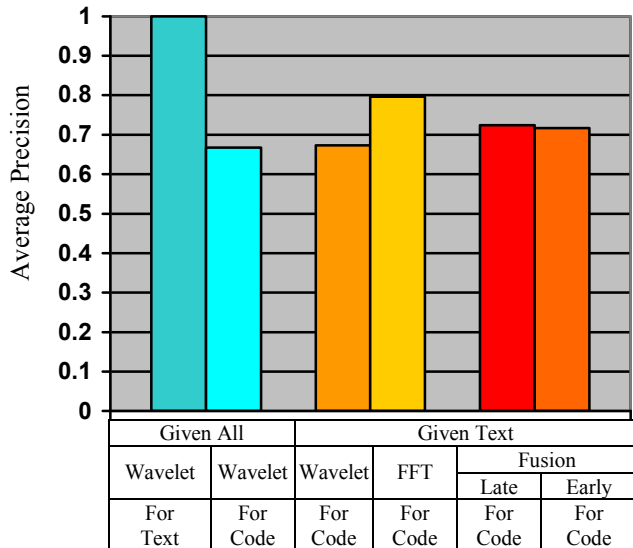


Figure 3: Average Precision of Fusion Experiments.

### 3.2. Fourier Features

We use Fourier features as one method for finding fixed-width rendered text (“code”). All images are regularized to 320 pixels wide through cropping. As we are only interested in images of text that are inclined within a 16 degree angle of vertical, we represent and examine only that portion of the Fourier domain that corresponds to this inclination band. The resulting FFT image is then decimated by a factor of 5. Each of the 700 values of this resulting representation is used as a feature [9]; see Figure 1.

### 3.3. Wavelet Features

For our wavelet features we use Haar wavelets; this feature is known to be useful in finding frames which

contain readable text (“text”) [8]. The image is divided into a 4x4 grid, and in each region the wavelet is calculated to 3 levels. For each of the sub-bands (HH, LH, HL, LL) of each level, the first 3 moments of the transform values are used as features. This gives 576 features per image; see Figure 2.

Best Feature Set Per Level Building Classifier for Code Given Text Using Late Fusion

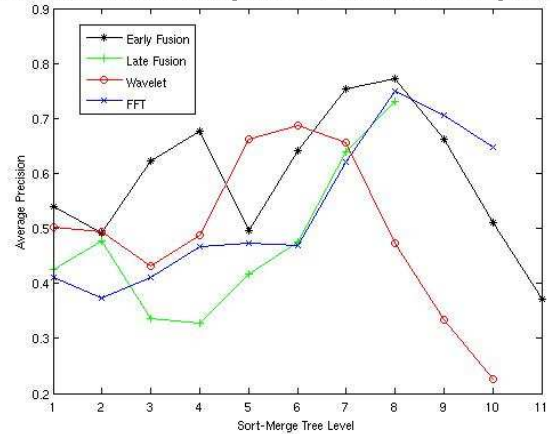
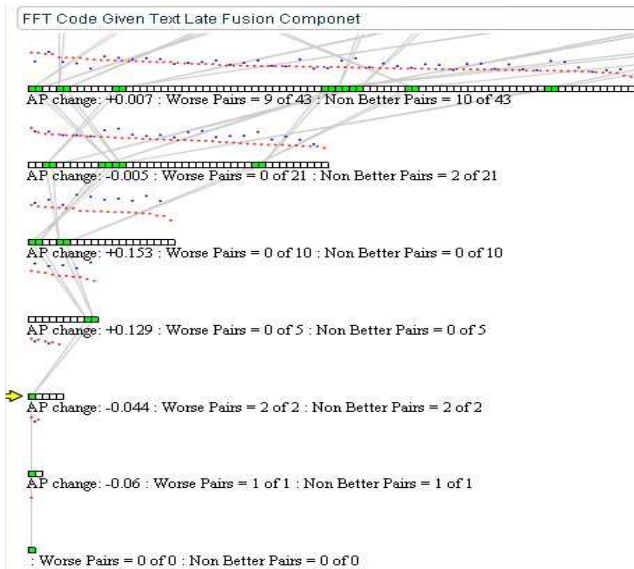


Figure 4: Average precision at each Sort-Merge level.

## 4. RESULTS OF EXPERIMENTS

### 4.1. Single Feature Experiments

The performance of the best keepers for each feature class can be found in Figure 3 where Average Precision (AP) is the metric for feature set performance. Identifying frames with readable text (“text”) using an SVM trained with features selected from wavelet transforms proved to be easy, resulting in a perfect AP of 1 for the best keeper (of size 32). Our initial attempt to identify frames that contained some form of computer code (“code”: assembly, Java, Matlab, C, others) using selected wavelet features resulted in an AP of 0.667 for the best keeper (of size 32).



**Figure 5: Genealogy Tool screenshot. Squares represent individual feature sets; each row shows all sets at a Sort-Merge level. Sets are sorted by AP, graphed by red dots. The AP of the merger of two adjacent feature sets is graphed by blue dots. Yellow arrow is best level.**

Identifying code was more accurate if the SVMs were cascaded and “code” was sought only within the context of “text” frames, discriminating non-code text from code text. See Figure 3, where “for code given text” refers to this cascaded method; performance increased to an AP of 0.79.

#### 4.2. Fusion Experiments

Given the heuristic nature of the Sort-Merge method, we needed to verify that no significant interactions of the two different feature types were overlooked. We conducted two fusion experiments: early and late. Figure 3 shows the results. Late fusion is not only considerably faster (96 hrs. vs. 116 hrs.) than early fusion but yields a higher AP. However, FFT alone performs the best; attempting to fuse a weaker feature with a strong one appears only to have weakened the better of the two.

Figure 4 shows the AP performance, on test data, of the best feature set at each level of the Sort-Merge tree. The graph shows that at a certain point adding more features is detrimental to classification, probably due to overfitting. (Although the peak of the early fusion curve here is greater than that of the FFT curve--the opposite of what is shown in Figure 3--this is because Figure 4 displays test set rather than validation set results.)

#### 4.3. Feature Set Genealogy

In order to better understand the evolution of keepers, we designed a tool to trace the genealogy of a feature set; see

Figure 5. This tool shows which two feature sets were merged to produce a given set, and how well the combination performed. The tool is interactive, allowing the trace of individual features backwards and forwards through the Sort-Merge levels.

This tool visualizes the effect that merging two feature sets has on performance. Red dots graph by their vertical height the AP of the feature sets, and blue dots graph the AP of the merger of each pair of adjacent sets. Because Sort-Merge exploits the heuristic that large volumes of features are redundant, the AP of the merged pair cannot be predicted; the resulting AP can be greater than, lesser than, or in between the separate APs.

However, we have noticed that after a certain level, every pairing of features has a lower AP than both of its parents. Additionally, the best keeper has always come from this level or a previous level. This criterion can be used as a much cheaper stopping point for the Sort-Merge algorithm. This obviates the costly need to train an SVM on the full feature set to establish a performance baseline, and allows rapid exit from the algorithm without the costly training of SVMs at levels where feature sets are largest.

## 5. CONCLUSIONS

Based on these experiments we conclude that selecting from a single class may be the best method. It appears that using both early and late fusion results in a performance that is, already bounded by the performance of the stronger set.

## 6. REFERENCES

- [1] F. C. Li, A. Gupta, E. Sanocki, L. He, Y. Rui, “Browsing Digital Video”, CHI, pp. 169-176, 2000.
- [2] A. Haubold, J. R. Kender, “Analysis, User Interface, and their Evaluation for Student Presentation Videos”, ICME, pp. 863-866, 2007.
- [3] C.-C. Chang and C.-J. Lin, “LIBSVM : A Library for Support Vector Machines”, [www.csie.ntu.edu.tw/~cjlin/libsvm](http://www.csie.ntu.edu.tw/~cjlin/libsvm), 2001.
- [4] Y. Liu, J. R. Kender, “Fast Video Segmentation Retrieval by Sort-Merge Feature Selection, Boundary Refinement and Lazy Evaluation”, CVIU, 95:2, pp. 147-175, 2003.
- [5] N. V. Chawla, K. W. Bowyer, “Random Subspaces and Sub-sampling for 2-D Face Recognition”, CVPR, pp. 582-589, 2005.
- [6] S. Marchand-Maillet, M. Worring., “Benchmarking Image and Video Retrieval: an Overview”, MIR, pp. 297-300, 2006.
- [7] I. Guyon, A. Elisseeff, “An Introduction to Variable and Feature Selection”, JMLR 3, pp. 1157-1182, 2003.
- [8] H. Li, D. Doermann, O. Kia, “Automatic Text Detection and Tracking in Digital Videos”, TIP, 9:1, pp. 147-156, 2000.
- [9] J. Laaksonen, E. Oja, M. Koskela, S. Brandt, “Analyzing Low-Level Visual Features Using Content-Based Image Retrieval”, ICONIP, pp. 1333-1338, 2000.
- [10] D.K. Iakovidis, D.E. Maroulis, S.A. Karkanis, I.N. Flaounas, “Color Texture Recognition in Video Sequences using Wavelet Covariance Features and Support Vector Machines,” Euromicro, pp. 119-205, 2003.