Oxford TRECVID 2008 – Notebook paper

James Philbin, Manuel Marin-Jimenez, Siddharth Srinivasan and Andrew Zisserman
Visual Geometry Group, Department of Engineering Science, University of Oxford, United Kingdom
Mihir Jain, Sreekanth Vempati, Pramod Sankar and C. V. Jawahar
Center for Visual Information Technology, International Institute of Information Technology, Gachibowli, Hyderabad, India

Abstract

The Oxford/IIIT team participated in the high-level feature extraction and interactive search tasks. A vision only approach was used for both tasks, with no use of the text or audio information.

For the high-level feature extraction task, we used two different approaches, both based on a combination of visual features. One used a SVM classifier using a linear combination of kernels, the other used a random forest classifier. For both methods, we trained all high-level features using publicly available annotations [24]. The advantage of the random forest classifier is the speed of training and testing.

In addition, for the people feature, we took a more targeted approach. We used a real-time face detector and an upper body detector, in both cases running on every frame.

One run C_OXVGG_4_4 was submitted using only Random forest approach for all concepts except two people. It performed best on Dog and hand, and came over the median for all the classes except kitchen and airplane.

In the interactive search task, our team came third overall with an mAP of 0.158. The system used was identical to last year with the only change being a source of accurate upper body detections.

1 High-level Feature Extraction

For the high-level feature task, we used two generic methods which were run for all topics, and also used more specialized methods for the people features. These results were then fused to create the final submission.

1.1 Generic Approaches

For the following approaches, we used a reduced subset of MPEG i-frames from each shot, found by clustering i-frames within a shot. Our approach here was to train a classifier for the concept in question, then score all frames in the test set using their distance from the discriminating hyper-plane. We then subsequently ranked the test shots by the maximum score over the reduced i-frames. We have developed two different methods for this task, each differing only in their classifier, but using the same features. The first uses a SVM classifier with a linear combination of kernels, with the weights of the linear combination learnt for each class. The second uses a multi-way classifier based on a random forest.

1.2 Feature descriptors

We have used different features to capture appearance, shape and color. They are described in the following subsections.

1.2.1 Appearance

These descriptors consist of visual words which are computed on a dense grid. Here visual words are vector quantized SIFT descriptors [15] which capture the local spatial distribution of gradients.

Local appearance is captured by the visual words distribution. SIFT descriptors are computed at points on a regular grid with spacing M pixels. We have used gray level representations for each image. At each grid point the descriptors are computed over circular support patches with radii r. So, each point is represented by four SIFT descriptors. These dense features are vector quantized into *visual words* using K-means clustering. Here, we have used a vocabulary of 300 words. Each images is now represented by a histogram of these visual word occurrences.

Here we have used $M = 5$, $K = 300$ and radii $r = 10, 15, 20, 25$. In order to cope with empty patches, we zero all SIFT descriptors with L2 norm below a threshold (200).

For the spatial layout representation which is inspired by the pyramid representation of Lazebnik $et.al.$ [13] , an image is tiled into regions at multiple resolutions. A histogram of visual words is computed for each image sub-region at each resolution level.

Finally, the representation of an appearance descriptor consists of a concatenation of these histograms into a single vector which are referred to as Pyramid Histogram of Visual Words (PHOW). Here, we have used 3 levels for the pyramid representation. The distance between the two PHOW descriptors reflects the extent to which the images contain similar appearance and the extent to which the appearances correspond in their spatial layout.

1.2.2 Shape

Local shape is represented by a histogram of edge orientations computed for each image sub-region, quantized into K bins. Each bin in the histogram represents the number of edges that have orientations within a certain angular range. This kind of representation is similar to the bag of (visual) words, where here each visual word is a quantization on edge orientation.

Initially, edge contours are extracted using the Canny edge detector. The oriented gradients are then computed using a 3×3 Sobel mask without Gaussian smoothing. We have used $K = 8$ bins for an angular range of $[0, 180]$. The vote from each contour point depends on its gradient magnitude, and is distributed across neighboring oriented bins according to the difference between the measured and actual bin orientation.

Finally the representation of Shape descriptor consists of concatenation of these histograms in a single vector. This descriptor is referred to as Pyramid Histogram of Oriented Gradients (PHOG). Four pyramid levels were used for this feature. Each level of PHOG is normalized to sum to unity taking into account all the pyramid levels.

1.2.3 Color histogram

Another feature used is a colour histogram combined with a spatial pyramid over the image to jointly encode global and local information. This is similar to the descriptor proposed in [5], except that we quantize colors more coarsely for all the levels. We used 10, 8, 4 and 2 bins for a cell in levels from 0 to 3 respectively. These are appended to create the final feature vector. We also used feature vector created without decreasing the number of bins with levels.

1.3 Multiple Kernel SVM

To predict whether a key-frame from the test data belongs to a category, we have used SVM classifiers which use a linear combination of multiple kernels for merging different levels of PHOW descriptor. Multiple kernels have been previously used in [**?**, 25], and we follow the approach of [3]. The kernels of different levels for two image regions i and j are combined using the following formulation

$$K(i,j) = \sum_{l \in L} \gamma^l e^{-\mu^l \chi^2(H^l(i), H^l(j))} \tag{1}$$

where $H^l(i)$ is the histogram at level l, γ^l is the weight for level l, and is constrained to be positive($\gamma^l \geq 0$).

The parameter μ^l is set to be $1/(2 * mean)$ value of the χ^2 distances between the level l histograms over all the training images.

Here the χ^2 distance used is defined as follows:

$$\chi^2(p,q) = \sum_{i=1}^{N} \frac{(p_i - q_i)^2}{p_i + q_i} \tag{2}$$

The weights for descriptors at each level γ^l are learnt from a training set for each high level feature (e.g. bridge, airplane) by using the method presented in [25]. So, this leads to giving more weight to the more discriminative features, and also includes not selecting the level of the feature if $\gamma^l = 0$. We have used libSVM package for classification [6].

We have also used bootstrapped version of the above for some of the runs using the following procedure:

Algorithm 1 Bootstrapped Multiple Kernel SVM

```
1) Prepare a initial classifier $C_0$ by training on a initial training set $Train_0$ consisting of all the positive key-frames, and a subset of the negative key-frames.
2) Divide the set of positive key-frames in the training set into subsets $P_i$'s where $i = 1$ to $m$, the set of negative key-frames in the training set into subsets $N_i$'s where $i = 1$ to $n$.
for $i$ = 1 to $max(m,n)$ do
    Classify the subsets $P_i$ and $N_i$ with the classifier $C_{i-1}$.
    Add the incorrect samples to $Train_{i-1}$ to get the updated training set, $Train_i$.
    Compute the kernel matrix for the updated training set, $Train_i$.
    Train a classifier $C_i$ using an SVM with Multiple Kernels.
end for
Here $P_i$ is empty when $i > m$ and $N_i$ is empty when $i > n$.
```

Example results are shown in figures 2 and 3.

1.4 Random Forests

Random Forests were introduced in the machine learning community by [1, 4]. They became popular in the computer vision community from the work of Lepetit*et al.* [14, 19]. Many papers have applied them to various classification and segmentation tasks [3, 17, 18, 21, 22, 28].

A random forest multi-way classifier consists of a number of trees, with each tree grown using some form of randomization. The leaf nodes of each tree are labeled by estimates of the posterior distribution over the image classes. Each internal node contains a test that splits the space of data to be classified. An image is classified by sending it down every tree and aggregating the reached leaf distributions. Figure 1 shows a Random Forest which consists of T trees.

Decision tree classifiers have shown problems related to over-fitting and lack of generalization. Random Forests are trained to alleviate such problems by: (i) injecting randomness into the training of the trees, and (ii) combining the output of multiple randomized trees into a single classifier. Randomness can be injected at two points during training: in subsampling the training data so that each tree is grown using a different subset; and in selecting the node tests. Two other appealing features of Random forests which we require here are: (i) probabilistic output and (ii) the seamless handling of a large variety of visual features (e.g. color, texture, shape, depth etc.). We use a combination of features for example, PHOW + PHOG, PHOW + Color etc to train the classifier.

1.4.1 Training

One versus rest classifiers are trained for all the 20 concepts. The trees we train here are binary and are constructed in a top-down manner. For node test we have a node function (difference of two components or single component of the feature vector) and a threshold. During training of the tree each node has available only a randomly chosen subset of the entire pool of possible node functions (100 such functions were chosen). Training is achieved by finding for each non-terminal node a node function and a threshold which yields maximum information gain within such restricted, randomized search space [3, 28]. We train 100 trees with maximum allowed depth 10 and choose an optimal threshold from 10 randomly selected thresholds for each chosen node function. To further inject randomness each tree is trained using 15000 randomly selected samples from the training data. Empirical posterior distribution for the class is stored in the leaf nodes as done in [3].

To combine the features, for each node test one type of feature is randomly selected from the combination of features. Similarly a pyramid level is selected randomly and the node function is then chosen from the indices corresponding to selected level of the selected feature.

Random Forest binary classifiers were trained for all the classes using different combinations of features (PHOW or PHOW + PHOG or PHOW + Color), pyramid levels and node tests. The best combinations were obtained for each concept by testing on TRECVID 2007 data.

1.4.2 Classification

The test image is passed down each random tree until it reaches a leaf node. All the posterior probabilities are then averaged and the arg max is taken as the classification of the input image. Example results are shown in figures 4 and 5.

1.5 People

For the people feature two approaches that used every frame, rather than only I-frames or key-frames, were employed. Both are targeted on detecting people, one by their frontal faces, the other by their upper bodies. In both cases the detected person is "tracked" throughout a shot so that all the detected instances are associated with a common identity. The output is a count of the number of people in each shot, and shots are ranked by their confidence for single people, two people, etc.

1.5.1 Upper body detection

We describe here the approach used to detect upper bodies and track them over time.

We start by detecting frontal upper bodies in every frame separately, by using the method of Ferrari *et al.* [**?**] which is based on the detector of Dalal and Triggs [7]. This approach slides a window over the image at various locations and scales and classifies each as either an upper body or not. The classifier is based on the spatial distributions of oriented gradients.

At this point, the system is not aware how many people are in the shot, and doesn't know yet how they evolve over time. For this purpose, we associate detections over time using a graph clustering algorithm [11] to maximize the temporal continuity of the detected bounding-boxes. Each of the resulting tracks links detections of a different person, as shown in figure 6. In this manner the number of (near frontal) people in each can be determined quite reliably.

1.5.2 Face Detection

This section describes our face detection approach (for details see [2] which presents a real-time version of the method described below). The aim here is to find video footage of people where their face is visible with a low false positive rate. The same processing pipeline is applied to all frames of the training data and test data. In the training data, a very high precision (99%) was achieved for low recalls (first 2000 shots).

Face detection and tracking. The first stage of processing is frontal face detection which is done using the Viola-Jones cascaded face detector [26]. When a new individual has been detected, a kernel-based regressor is trained to track that individual such that the tracking performance is both fast and more robust to non-frontal faces in comparison to cascaded face detection [27]. Face detection is used to collect several exemplars of an individual's face which may vary in pose and expression. A training set consisting of image patches that are offset from the face center and at a slightly different scale, and the respective transformations back to the original face location and scale, are artificially generated from the face detections. This dataset is used to train a kernel-based regressor to estimate the position (x, y) and scale (w) of a face.

Feature localization. The output of the face tracker gives an approximate location and scale of the face, but does not

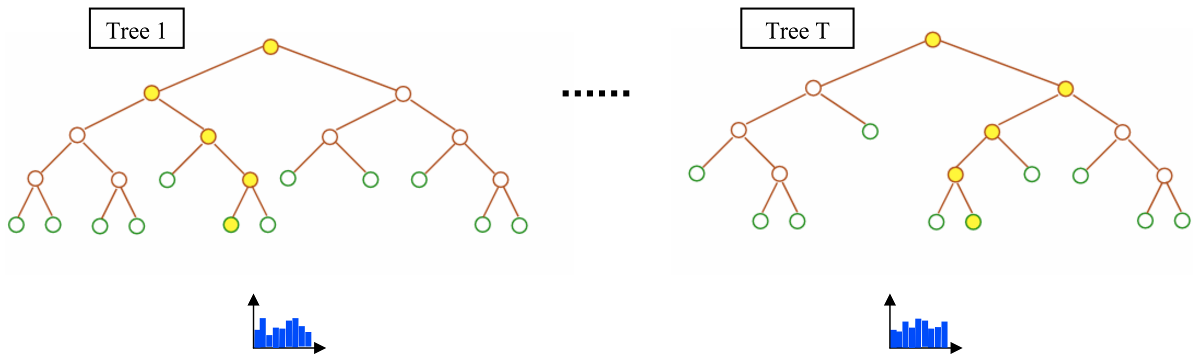

Figure 1: Random Forest with T trees, leaf nodes are shown in green. Training samples are traversed from root to leaf nodes and posterior distributions (blue) are computed. An Image is classified by descending each tree and then aggregating the distributions at each reached leaf. The paths formed while descending are shown in yellow.

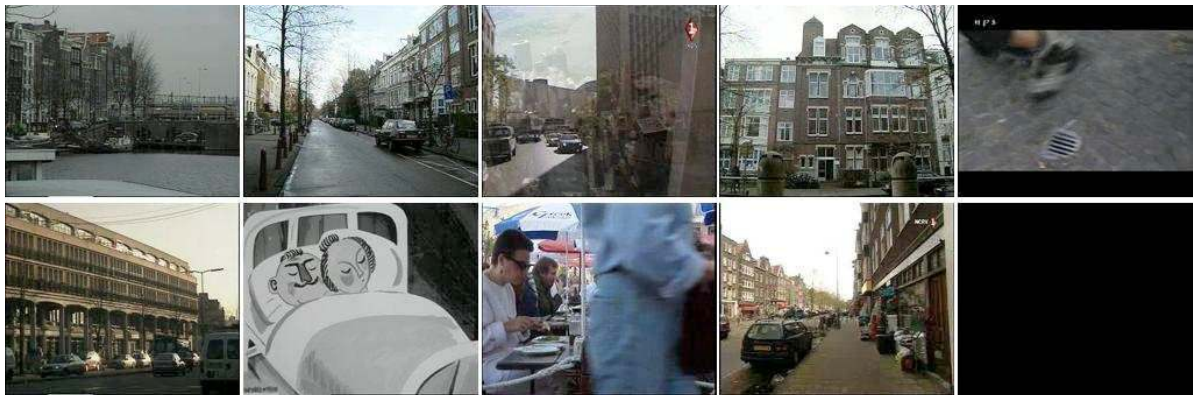

Figure 2: Top 10 results (distinct scenes) of Cityscape using Multiple Kernel SVM

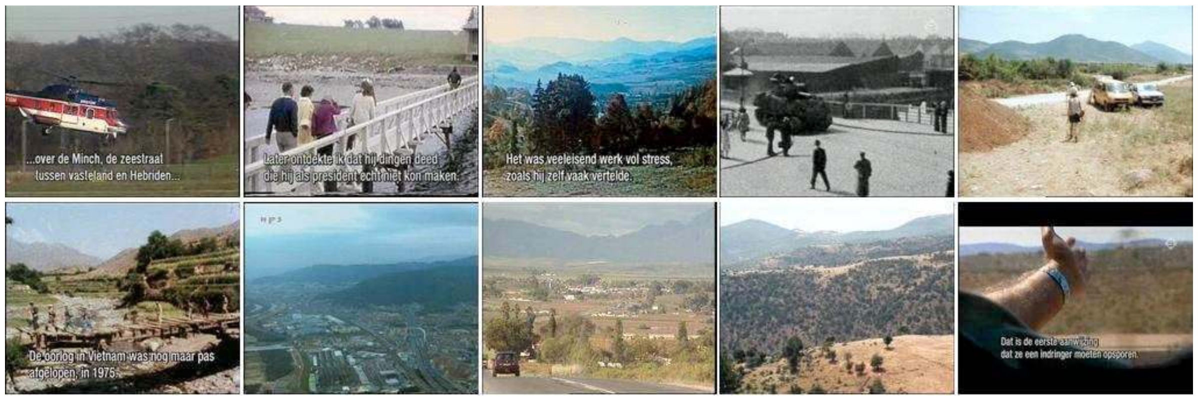
...over de Minch, de zeestraat tussen vasteland en Hebriden...
Later ontdekte ik dat hij dingen deed die hij als president echt niet kon maken.
Het was veeleisend werk vol stress, zoals hij zelf vaak vertelde.
De oorlog in Vietnam was nog maar pas afgelopen, in 1975.
Dat is de eerste aanwijzing dat ze een indringer moeten opsporen.

Figure 3: Top 10 results (distinct scenes) of Mountain using Multiple Kernel SVM

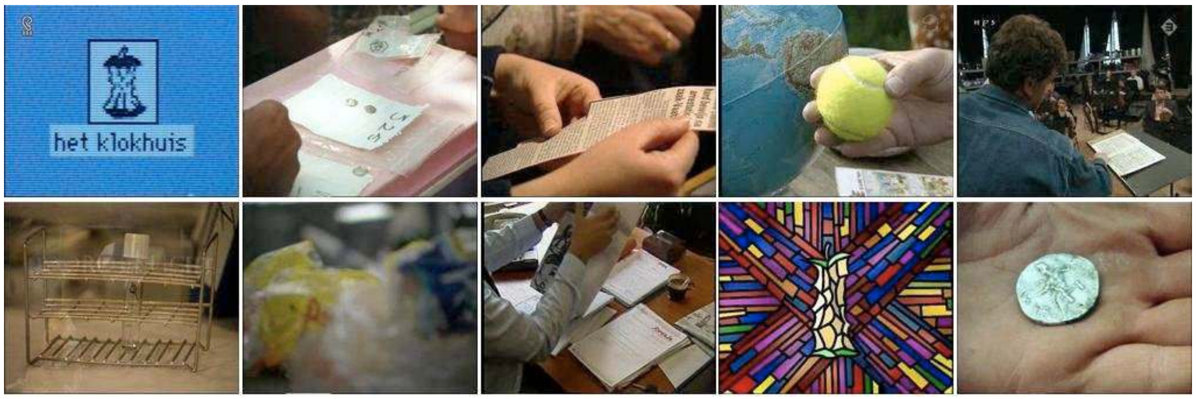

Figure 4: Top 10 results (distinct scenes) of Hand using Random Forests

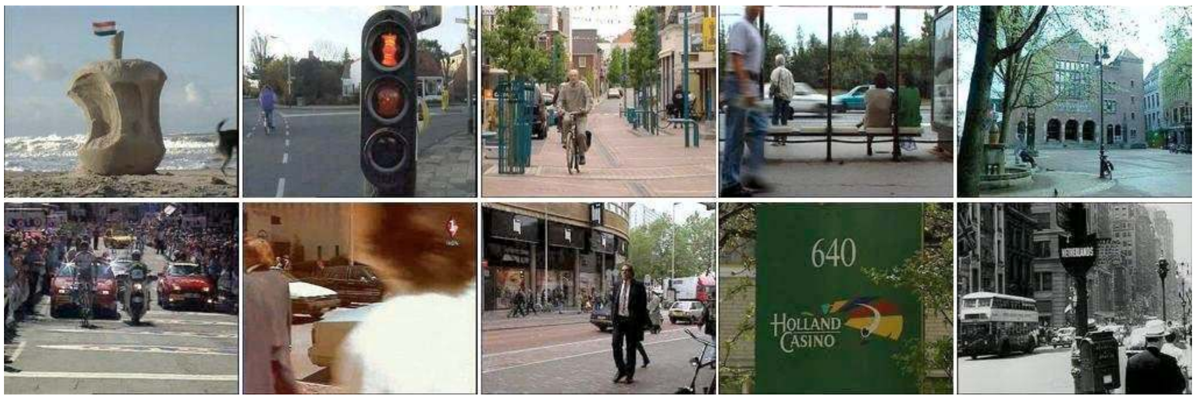

Figure 5: Top 10 results (distinct scenes) of Street using Random Forests

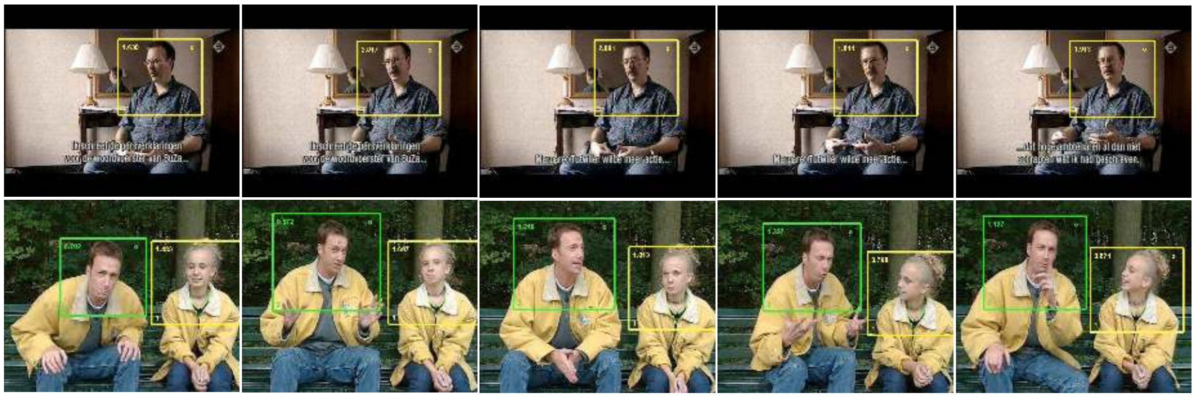

Figure 6: Upper body detections linked throughout a shot – the superimposed rectangle on each frame indicates the region detected. The top row is an example containing a single person, and the lower row an example of two people. Using the temporal continuity of the detections allows false positive detections to be rejected.

provide a confidence in this measure. To achieve a low false positive rate, features at the corners of the eyes, nose and mouth are located to verify the existence of a face. Where multiple successive frames achieve a poor localization confidence, the track is terminated.

To locate the features, a model combining a generative model of the feature positions with a discriminative model of the feature appearance is applied. The probability distribution over the joint position of the features is modelled using a mixture of Gaussian trees, a Gaussian mixture model in which the covariance of each component is restricted to form a tree structure with each variable dependent on a single "parent" variable. This model is an extension of the single tree proposed in [10], and further details can be found in [8, 9].

Ranking face shots. The output of the face tracking system is a set of face tracks for each shot that include the location (x, y), scale (w) and confidence (c) of the face at each frame in the track. This information is combined to get a score (s) for each shot

$$s_i = \frac{1}{N_{\mathcal{T}}} \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}_t} c_f w_f. \qquad (3)$$

where $\mathcal{T}$ is the set of tracks in shot i which have had all faces with a low confidence c_f removed and are at least 15 frames long, $\mathcal{F}_t$ is the set of faces in track t, w_f is the width of the face f and $N_{\mathcal{T}}$ is the number of tracks in shot i. The final confidence depends on the facial feature localization confidence, and the size and position of the detected face. The weights in the confidence measure are learnt from training data.

2 Interactive Search

Our system for the interactive search was essentially the same as last year. Unfortunately, due to a change in the rules that was missed by the authors we took 15mins to answer each query instead of 10mins (which is the new maximum). Our results should be taken with this caveat in mind.

For interactive search, we use the ranked results obtained from the high-level feature detection task coupled with some external images, such as those supplied by NIST for each query or images found from Google Image Search. These external images are indexed in real-time and this allows us to use a number of different expansion-search algorithms to harvest new shots similar to any i-frame in the database or an external image. Taken together with an extremely high-speed, efficient interface, this allows us to answer queries quickly and with high precision.

This year we came third with an mAP score of *0.158*. This was behind two entries from the University of Amsterdam. We won/drew in 5 categories which were: "Find shots of a person opening a door", "Find shots of one or more vehicles passing the camera", "Find shots of waves breaking onto rocks", "Find shots of food and/or drinks on a table" and "Find shots of one or more people, each in the process of sitting down in a chair".

Our expansion-search algorithms allow us to search for images with similar textural layout (bag-of-words, image-SIFT), similar colour layout (spatial colour histograms) or which are nearly identical (near-duplicates). We also used the output from the upper-body detector previously described. The results of these detections were combined to give "N-people" rankings, where a shot contained exactly N people. This proved very useful for some queries.

2.1 (Spatial) texture and colour search

Texture-like search expansion was performed by pre-computing the 20 most similar i-frames to each reduced i-frame in the corpus using the bag-of-words representation from the concept task with a χ^2 distance measure. The user also had access to a spatial texture expansion method, which used a spatial pyramid based bag-of-words representation to return images with similar structure [12].

We also implemented a global gradient orientation descriptor similar to SIFT [16] to give more varied texture results for images.

For our colour expansion method we used a very fast spatial colour histogram, used with success in our entry from last year [20], with an L_2 distance for measuring similarity.

2.2 Near duplicate detection

We also allowed the user to find near-duplicate scenes to any i-frame in the corpus using a method described in [5]. This used a bag-of-words representation coupled with a min-hash search algorithm to quickly compute an approximate set overlap score.

2.3 User Interface

In designing a successful user-interface for TRECVid it is important to specify which goals such an interface should meet. The system must make it easy for a user to combine the many different streams of data in an efficient and intuitive manner. In our case, the main data sources were:

1. Bag-of-words concept rankings.
2. Face detections.
3. Upper body detections.
4. "N-people" detections. E.g. "exactly 2 people", "exactly 3 people", etc.

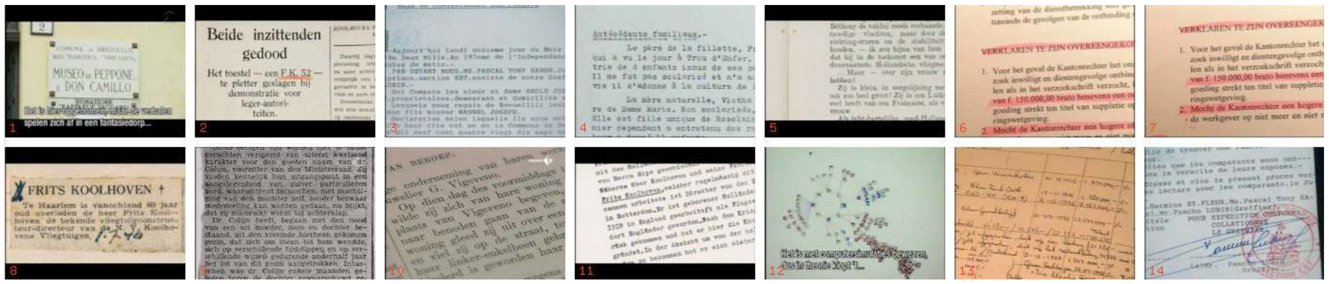
Figure 7: BOW expansion on “text”. The query image is labelled 1, with the top 14 results shown.

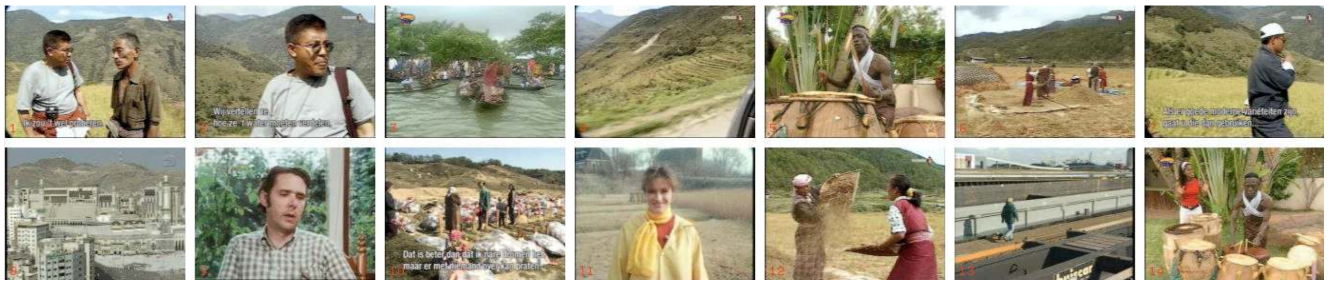
Figure 8: Colour expansion on “mountain”.

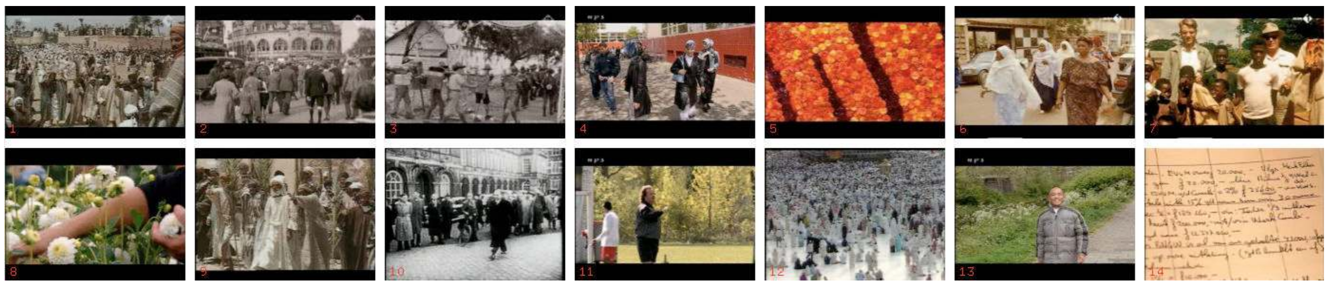
Figure 9: Texture expansion on “crowd”.

5. Texture expansions.
6. Colour expansions.
7. Near-duplicate detections.

Inspired by the *CrossBrowser* approach [23], the main interface view contains two axes (see figure 10). The x-axis represents the temporal ordering of the shots in the corpus and enables the user to move backwards and forwards in time. The y-axis displays the rank ordering for the currently loaded list. There is often a high level of temporal coherency between subsequent shots and exploiting this is crucial to good interactive performance. Frequently, in searching forwards and backwards from one relevant shot, the user would find more shots relevant to the query.

As with last year we used a "temporal zoom" function facility which allowed the user to specify the temporal granularity. Set to the finest level, the interface allowed the user to do video "scrubbing" with the mouse, which is important for answering the action queries in this year's competition. Additionally, every shot on the screen could be played at high speed simultaneously. Surprisingly, it seems quite easy to spot particular actions whilst viewing multiple videos.

The interface allows for rapid access to the data sources mentioned in the following ways. Any of the pre-generated results can be loaded into one of ten "live" lists in the system. Lists can then be appended, trimmed or fused at will to give the user a list which can be labelled (as correct or not for the topic). Once some good examples have been found, the user can then use any of the expansion methods to "grow" the positive examples. New external images could be dragged and dropped onto the interface and then run through the various expansion methods to generate extra results. We found that using the external images provided by NIST for each query gave some good initial results.

References

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees, 1997. Neural Computation, 9(7).

[2] N. E. Apostoloff and A. Zisserman. Who are you? real-time person identification. In *Proceedings of the British Machine Vision Conference*, 2007.

[3] A. Bosch, A. Zisserman, and X. Munoz. Image classi.cation using random forests and ferns, 2007. ICCV.

[4] L. Breiman. Random forests, 2001. ML Journal, 45(1).

[5] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, Jul 2007.

[6] Chih chung Chang and Chih jen Lin. Libsvm: a library for support vector machines, 2001.

[7] N. Dalal and B Triggs. Histogram of oriented gradients for human detection. In *Proc. CVPR*, 2005.

[8] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2006.

[9] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2006.

[10] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005.

[11] V. Ferrari, T. Tuytelaars, and L. Van Gool. Real-time affine region tracking and coplanar grouping. In *Proc. CVPR*, Dec 2001.

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

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognitzing natural scene categories., June, 2006. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2, pages 2169-2178, New York.

[14] V. Lepetit and P. Fua. Keypoint recognition using randomized trees, 2006. IEEE PAMI, 28(9).

[15] D. Lowe. Distinctive image features from scale invariant keypoints, 2004. International Journal of Computer Vision,60(2):91-110.

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] R. Mar'ee, P. Geurts, J. Piater, and L Wehenkel. Random subwindows for robust image classification, 2005. CVPR.

[18] F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests, 2006. NIPS.

[19] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code, 2007. CVPR.

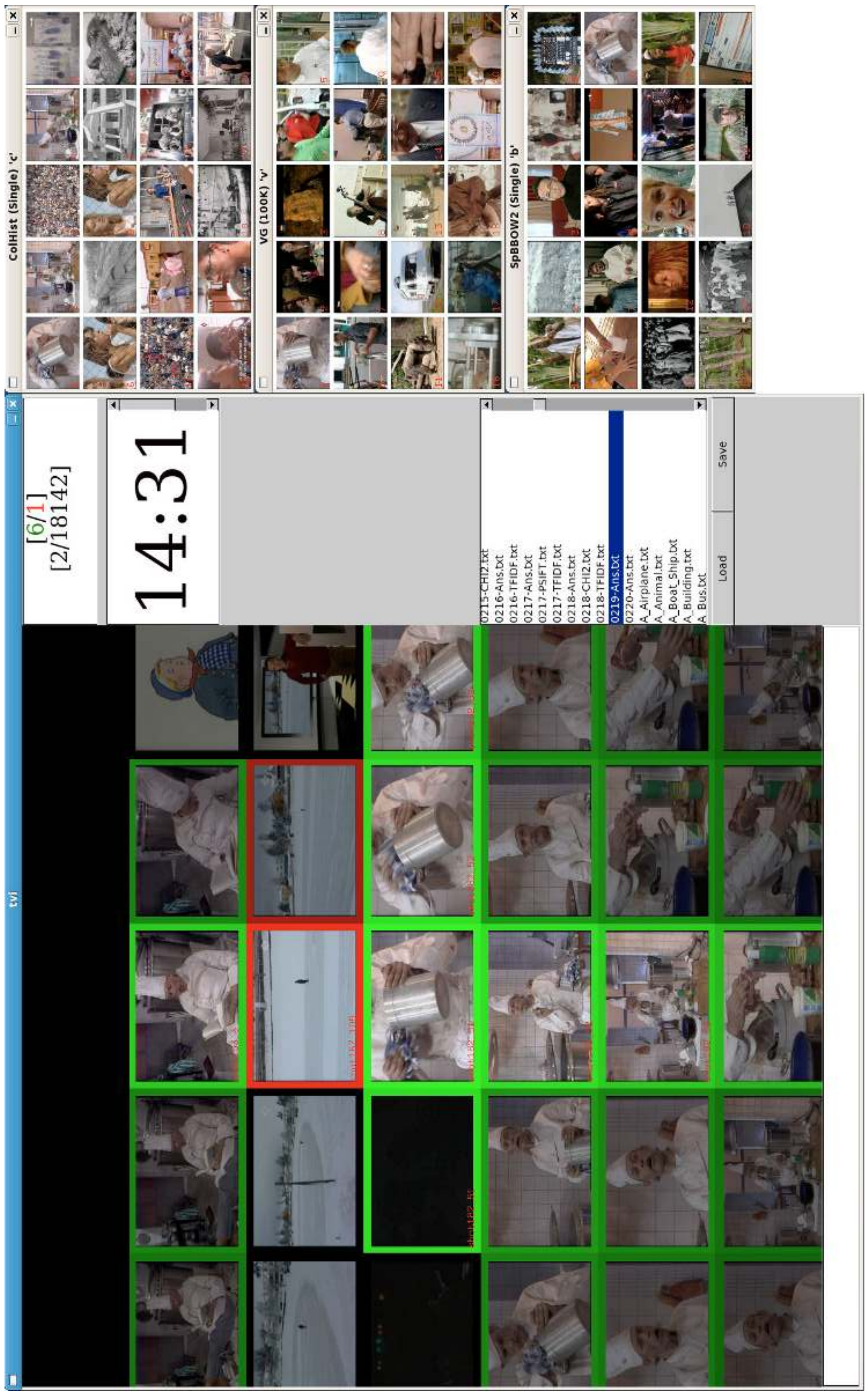

Figure 10: Finding the chef character from the Clockhuis program. The currently found positive examples are labelled green, the negative examples are labelled red. The horizontal axis shows shots in temporal order, the vertical axis shows shots in current rank order. Ranked lists from the various different sources can be loaded or fused from the pane to the right. All of the expansion methods are invoked using a single key press, and are shown in separate panes.

[20] J. Philbin, A. Bosch, O. Chum, J. Geusebroek, J. Sivic, and A. Zisserman. Oxford trecvid 2006 - notebook paper. In *Proceedings of the TRECVID 2006 Workshop*, Nov 2006.

[21] F. Scroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests, 2008. BMVC.

[22] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation, 2008. CVPR.

[23] C. Snoek, J. van Gemert, J. Geusebroek, B. Huurnink, D. Koelma, G. Nguyen, O. de Rooij, F. Seinstra, A. Smeulders, C. Veenman, and M. Worring. The mediamill TRECVID 2005 semantic video search engine. In *Proceedings of the 3rd TRECVID Workshop*, November 2005.

[24] Georges Qunot Stphane Ayache. Video corpus annotation using active learning. *30th European Conference on Information Retrieval*.

[25] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off., October, 2007. In Proceedings of the IEEE Internation Conference on Computer Vision, Rio de Janerio, Brazil.

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages 511–518, 2001.

[27] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proc. ICCV*, 2003.

[28] J. Winn and A. Criminisi. Object class recognition at a glance, 2006. CVPR.