# Video Retrieval within a Browsing Framework using Key Frames

Daniel Heesch, Marcus J Pickering, Stefan Rüger and Alexei Yavlinsky

Department of Computing, South Kensington Campus,
Imperial College London, SW7 2AZ, UK
{daniel.heesch,m.pickering,srueger,agy02}@doc.ic.ac.uk

**Abstract**

We describe our system and experiments for the shot boundary detection, high level feature extraction and search tasks for TRECVID 2003. In the shot boundary detection task, we employ our tried and tested method based on the calculation of a distance measure between colour histograms of frames over a range of timescales. In the feature detection task we attempted the vegetation feature, using a colour-based region classifier. For the search task we introduce a novel browsing structure that provides an interesting complement to the traditional retrieval method of search-by-example. We compare performance of four interactive runs, including one with only browsing permitted.

## 1 Introduction

The retrieval system we used for the search tasks is a significant extension of the system employed for TREC 2002 [4]. Three points are particularly noteworthy: (i) The suite of features has been broadened to include a set of simple, semi-global texture features as well as standard colour histogram features. (ii) For each feature, we normalise the dissimilarities between any two images (for the purpose of linear combination) such that their median lies at 1. (iii) The interaction model has been augmented by a dual browsing functionality. One allows browsing backwards and forewards along the sequence of images belonging to any one shot. The other allows movement through a richly connected, pre-computed image network with images being the nodes and connections being established between any two nodes that satisfy a particular similiarity condition.

Four interactive runs are being compared to assess the value of the browsing functionality and of relevance feedback. In addition, two manual runs were performed to investigate whether visual features can significantly enhance performance over the baseline (text only) when no interaction is allowed. The paper is structured as follows. Sections 2-4 describe, respectively, the experiments and results for the shot boundary detection task, the feature detection task, and the search task.

## 2 Shot boundary detection task

### 2.1 System

The video shot boundary detection algorithm is broadly based on the colour histogram method, where the colour histograms for consecutive frames are compared and, if their difference is greater than a given threshold, a shot boundary is declared. This method is extended, based on the algorithm of Pye et al [7] for detection of gradual transitions that take place over a number of frames, and for rejection of transients, such as the effect of a flash-bulb. In order to determine the start and end points for gradual

| | All | | Cuts | | Gradual | | | |
|---|---|---|---|---|---|---|---|---|
| | Rec | Prec | Rec | Prec | Rec | Prec | F-Rec | F-Prec |
| Imperial-01 | 0.788 | 0.897 | 0.880 | 0.895 | 0.564 | 0.907 | 0.799 | 0.239 |
| Imperial-02 | 0.800 | 0.890 | 0.879 | 0.892 | 0.608 | 0.885 | 0.811 | 0.245 |
| Imperial-03 | 0.807 | 0.874 | 0.874 | 0.886 | 0.643 | 0.834 | 0.819 | 0.256 |
| Imperial-04 | 0.808 | 0.847 | 0.871 | 0.874 | 0.657 | 0.768 | 0.823 | 0.258 |
| Imperial-05 | 0.839 | 0.871 | 0.910 | 0.892 | 0.664 | 0.808 | 0.704 | 0.627 |
| Imperial-06 | 0.855 | 0.854 | 0.911 | 0.889 | 0.717 | 0.762 | 0.704 | 0.626 |
| Imperial-07 | 0.863 | 0.826 | 0.911 | 0.880 | 0.749 | 0.699 | 0.698 | 0.628 |
| Imperial-08 | 0.866 | 0.801 | 0.910 | 0.868 | 0.760 | 0.654 | 0.697 | 0.597 |
| Imperial-09 | 0.845 | 0.868 | 0.905 | 0.890 | 0.698 | 0.805 | 0.757 | 0.477 |
| Imperial-10 | 0.855 | 0.839 | 0.905 | 0.882 | 0.734 | 0.733 | 0.747 | 0.484 |
| Median | 0.795 | 0.849 | 0.910 | 0.881 | 0.487 | 0.806 | 0.750 | 0.707 |

Table 1: Shot boundary detection task – results for the ten system variants submitted by Imperial College London, followed by the median for all systems submitted by the various groups to TRECVID

transitions, we employ a method similar to that described by Zhang [12], in which a lower threshold is used to test for the start and end of a gradual transition. Our system is largely unchanged from last year, and more details can be found in our TREC 2002 proceedings paper [4].

### 2.2 Experiments

We performed ten shot boundary detection runs. The first four runs, Imperial-01 – Imperial-04 were carried out keeping the low threshold, $T_4$, constant at 0.05, and reducing the high threshold, $T_{16}$, from 0.6 to 0.3 in decrements of 0.1. In runs Imperial-05 – Imperial-08, the low threshold was increased to 0.15 and the same four values for the high threshold were used again. The last two runs, Imperial-09 and Imperial-10 used a value for the low threshold falling between the first set of runs (0.1) and the second set, and a high (0.5) and low (0.4) value respectively for the high threshold. The thresholds for these last two runs were determined empirically from last year's TREC results.

### 2.3 Results

We show the results for our ten shot-boundary detection runs in Table 1. There was a predictable trade-off between recall and precision as the thresholds were altered. Runs 5, 6 and 9 gave particularly good points at the top right of the precision-recall graphs. Comparison with other systems submitted to TRECVID clearly reveals our systems as the third highest performing system among all groups.

## 3 Feature detection task

### 3.1 System

We attempted only one feature in the high-level feature extraction task – feature 16, the "vegetation" feature. For this we used a classifier trained as a grass detector – we did not have the time to re-train for vegetation using the TRECVID development data, and as we had not participated in the common feature annotation forum we did not have access to the common training data.

The classifier that we used works on the basis of colour. Images are segmented into regions [9] and the colour distribution of each region is encapsulated by clustering pixels in RGB space into a few points. A training set was created by identifying a few hundred positively labelled grass regions and a few hundred

| System Variant | Avg prec | Hits |
|---|---|---|
| Imperial-01 (distance-weighted) | 0.082 | 342 |
| Imperial-02 (voting) | 0.087 | 360 |
| TRECVID Median | 0.150 | 367 |

Table 2: High level feature extraction task – average precision results and hits at depth 2000 for feature 16 (vegetation)

negative examples. A test region was then classified by finding its 25 nearest neighbours in the training set. The term "nearest" was defined using the *earth mover's distance* [8].

The relevance score for a shot was taken as the square of the highest region score in the shot's key frame. The results were sorted by score and the top 2000 submitted as positive shots.

### 3.2 Experiments

Two runs were carried out, one using voting-based $k$-nn and one using distance-weighted $k$-nn. In voting-based $k$-nn, a region's relevance score is based on the proportion of the nearest neighbours that are positive. In distance-weighted $k$-nn the distance of the nearest neighbours from the test region are also factored in.

### 3.3 Results

Our results for the two runs on feature 16 are shown in Table 2. Although voting-based $k$-nn had been shown empirically to give better results on the development data, it suffered in this case from the limit of 2000 submitted results. Over 2000 shots were judged by 100% of the nearest neighbours to contain vegetation and there was no possibility for ranking within this results set, so the only option was to truncate the results list at 2000. Although distance-weighted $k$-nn had given poorer results in testing, the distances gave a greater spread of scores and hence a less arbitrary ranking. The weakness of our ranking is reflected in the fact that our number of hits at depth 2000 was very close to the median achieved in all TRECVID runs, but our average precision fell well below the TRECVID median.

## 4 Search task

### 4.1 Features

For the search task we used eleven low-level texture and colour features as well as the text from the LIMSI transcripts. As was evident from the training collection, an appreciable proportion of images contained a news line near the bottom whose colour and texture characteristics were unrelated to the actual image content. We therefore decided to disregard the bottom 52 pixel lines in our feature computation. For three simple texture features, we decided to partition the images into tiles and obtain features from each tile individually with the aim of better capturing local information. The final feature vector for these three features consisted of a concatenation of the feature vector of the individual tiles. Details of each feature are given below.

#### 4.1.1 HSV Global Colour Histograms

We use the HSV [11] colour space which has proven to work better than other 3D colour models for image retrieval in our experiments [5, 6]. HSV Global Colour histograms are quantised distributions in the 3-dimensional colour space of all pixels of one image. The corresponding feature vector is a list of the proportions of pixels which fall into the respective 3-dimensional colour bins; its length depends on

the granularity of the colour bins. Here, we do *not* use 1-dimensional component-wise histograms since (as with all marginalisations) information about the underlying colours would be lost.

HSV is a cylindrical colour space with hue H being the angular, saturation S the radial and brightness value V the height component. This brings about the mathematical disadvantage that hue is discontinuous wrt RGB coordinates and that hue is singular at the achromatic axis $r = g = b$ or $s = 0$. As a consequence we merge, for each brightness subdivision separately, all pie-shaped 3D HSV bins which contain or border $s = 0$. The merged cylindrical bins around the achromatic axis describe the grey values which appear in a colour image and take care of the hue singularity at $s = 0$. Saturation is essentially singular at the black point in the HSV model. Hence, a small RGB ball around black should be mapped into the bin corresponding to $hsv = (0, 0, 0)$ to avoid jumps in the saturation from 0 to its maximum of 1 when varying the singular RGB point infinitesimally. There are several possibilities for a natural subdivision of the hue, saturation and brightness axes; they can be i) subdivided linearly, ii) so that the geometric volumes are constant in the cylinder and iii) so that the volumes of the nonlinear transformed RGB colour space are nearly constant. The latter refers to the property that few RGB pixels map onto a small dark V band but many more to a bright V interval of the same size.

We use the HSV model with a linear subdivision into 10 hues, 5 saturation values and 5 $V$ values yielding a 205-dimensional feature vector (less than $10 \cdot 5 \cdot 5$ bins owing to the merging of all pie-shaped h-0-v bins over the hue component). We normalise the HSV colour histogram so that the components add up to 1.

#### 4.1.2 HSV Focus Colour Histograms

This feature is essentially a truncated version of the aforementioned feature. Only pixels from the central 25 % of the image are taken into account for feature computation. The feature leads to close similarities between any images that differ primarily with respect to their background.

#### 4.1.3 Colour Structure Descriptor

This feature is based on the recently introduced HMMD (Hue, Min, Max, Diff) colour space, which is used in the MPEG-7 standard. The HMMD space is derived from the HSV and RGB spaces. The hue component is the same as in the HSV space, and max and min denote the maximum and minimum among the $R$, $G$, and $B$ values, respectively. The diff component is defined as the difference between max and min. Three components are sufficient to uniquely locate a point in the colour space and thus the space is effectively three-dimensional. Following the MPEG-7 standard, we quantise the HMMD non-uniformly into 184 bins with the three dimensions being Hue, Sum and Diff (sum being defined as $(max + min)/2$) and use a global histogram. See Manjunath and Ohm [2] for details about quantisation.

In order to capture local image structure, we slide a $8 \times 8$ structuring window over the image. Each of the 184 bins of the colour structure histogram contains the number of *window positions* for which there is at least one pixel falling into the corresponding HMMD bin. This descriptor is capable of discriminating between images that have the same global colour distribution but different local colour structures. Although the number of samples in the $8 \times 8$ structuring window is kept constant (64), the spatial extent of the window differs depending on the size of the image. Thus, for larger images appropriate sub-sampling is employed to keep the total number of samples per image roughly constant. The 184 bin values are each normalised by dividing by the number of locations of the structuring window; each of the bin values falls thus in the range $[0, 1]$, but the sum of the bin values can take any value up to 64 (see Manjunath and Ohm [2] for details).

#### 4.1.4 Marginal RGB colour moments

For this desciptor, we formed individual histograms for each of the three colour channels and computed the mean and second, third and fourth central moment of each marginal colour distribution. In order to obtain values in a comparable range, the features we use are the corrsponding $n$-th roots of the $n$-th

colour moments

$$m_c^n := \sqrt[n]{\sum_{i_c} p(i_c)(i_c - \overline{i_c})^n},$$

where $n \in \{2,3,4\}$, $i_c$ is the colour intensity of colour $c \in \{r,g,b\}$, $p(i_c)$ is the frequency of this colour level $i_c$, and $m_c^1 := \overline{i_c}$ is the average colour level. This results in a feature vector of size 12, namely 4 numbers $m_c^1, \ldots, m_c^4$ for each colour channel.

#### 4.1.5 Thumbnail feature

This feature is obtained by scaling down the original image to $44 \times 27$ pixels and then recording the gray value of each of the pixels leaving us with a feature vector of size 1,188. It is suited to identify near-identical copies of images, e.g., keyframes of repeated shots such as adverts.

#### 4.1.6 Convolution filters

For this feature we use Tieu and Viola's method [10], which depends on the definition of highly selective features that are determined by the structure of the image, as well as capturing information about colour, texture and edges. By defining a vast set of features, each feature is such that it will only have a high value for a small proportion of images, and by discovering a number of features which distinguish the example set in question we are able to perform an effective search.

The feature generation process is based on a set of 25 primitive filters, which are applied to the gray level image to generate 25 different feature maps. Each of these feature maps is rectified and downsampled before being fed again to each of the 25 filters to give 625 feature maps. The process is repeated a third time, and then each feature map is summed to give 15,625 feature values. The idea behind the three stage process is that each level "discovers" arrangements of features in the previous level. The feature generation process is computationally quite costly, but only needs to be done once and then the feature values can be stored with the image in the database.

#### 4.1.7 Variance Feature

The variance feature is a 20 bin histogram of gray value standard deviations within a a sliding window of size $5 \times 5$ determined for each window position. The histogram is computed for each of 9 non-overlapping image tiles and the bin frequencies concatenated to give a feature vector of size 180.

#### 4.1.8 Smoothness Feature

The smoothness measure $R$ is based on the variance $\sigma^2(z)$ and defined as

$$R := 1 - \frac{1}{1 + \sigma^2(z)/L^2},$$

where $z \in \{0, \ldots, L-1\}$ is a variate denoting the gray level of a pixel. Smoothness was determined for each of $8 \times 8$ image tiles resulting in a feature vector of size 64.

#### 4.1.9 Uniformity Feature

Uniformity is another statistical texture feature defined as

$$U := \sum_{z=0}^{L-1} p^2(z)$$

where $L = 100$ is the number of gray levels and $p(z)$ the frequency of pixels of gray level $z$. For each of $8 \times 8$ image tiles, we obtain one uniformity value resulting in a feature vector of size 64.

#### 4.1.10 Text

Our text feature is derived from the speech recognition transcripts supplied by LIMSI. A full text index was built using the Managing Gigabytes search engine [1] and queries formed from the XML data supplied with each query. Managing Gigabytes supplies a numerical relevance value which is used when weighing features.

#### 4.1.11 Bag of words

Using the textual annotation obtained from the LIMSI transcripts, we computed a bag-of-words feature consisting for each image of the set of accompanying stemmed words (Porter's algorithm) and their weight. This weight was determined using the standard tf-idf formula and normalised so that the sum of all weights is 1. As this is a sparse vector of considerable size (the number of different words) we store this feature in the form of (weight, word-id) pairs, sorted by word-id.

### 4.2 Distances and Normalisation

In order to compare two images in the database we use distances of their corresponding descriptors. For these we use the $L_1$-norm throughout[1]. Some of the distances already exhibit a natural normalisation, for example when the underlying features are normalised (e.g., HSV colour histograms), others do not (e.g., the colour structure descriptor). As the distances under different descriptors are going to be combined, we normalise the occuring distances empirically, so that the median of the occuring pairwise distances in the databse for a particular feature would be around 1. $\mathrm{dist}_d(p, q)$ denotes this normalised distance between images $p$ and $q$ under descriptor $d$.

### 4.3 Retrieval using $k$-nearest neighbours

Retrieval is performed using the $k$-nearest neighbours approach, which is based on the intuitive notion that if we have seen and already identified something, then anything we later see that is the same, or similar, (based on some defined characteristics) is probably the same kind of thing. In practice, we provide positively and negatively classified examples, and classify all test images according to their proximity to these examples.

We use a variant of the distance-weighted $k$-nearest neighbour approach [3]. Positive examples are supplied by the user, and a number of negative examples are randomly selected from the database. The distances, for descriptor $d$, from the test image $T_i$ to each of the $k$ nearest positive or negative examples (where 'nearest' is defined by the Euclidean distance in feature space) are determined, and a distance measure $d$ calculated as follows:

$$\mathrm{d}_f(Q, T_i) = \frac{\sum_{n \in N} \left(\mathrm{dist}_f(T_i, n) + \varepsilon\right)^{-1}}{\sum_{q \in Q} \left(\mathrm{dist}_f(T_i, q) + \varepsilon\right)^{-1} + \varepsilon},$$

where $Q$ and $N$ are the sets of positive and negative examples respectively amongst the $k$ nearest neighbours, such that $|Q| + |N| = k$. $\varepsilon$ is a small positive number to avoid division by zero. Images are ranked according to the convex combination

$$D_s(Q, T_i) = \sum_f w_f \cdot \mathrm{d}_f(Q, T_i), \tag{1}$$

where $w_f$ is the weight of descriptor $f$ subject to $\sum w_f = 1$ and $w_f \geq 0$.

[1] except for the bag-of-stemmed-words descriptor which is the $L_1$ norm raised by the power of 3; otherwise the typical distance of images with a large overlap in vocabulary would still be much larger than 0/5

## 4.4 Relevance feedback

Retrieved images $T_1, T_2, \ldots$ are displayed as thumbnails such that their respective distance from the centre of the screen is proportional to the dissimilarity $D_s(Q, T_i)$ (given by Equation 1) of thumbnail $T_i$ to the query set $Q$. Using this semantics of thumbnail location on the screen, the user can provide relevance feedback by moving thumbnails closer to the centre (meaning they are more relevant than the system predicted) or further away (indicating less relevance). The user effectively supplies the system with a real vector of distances $D_u(Q, T_i)$, which, in general, differ from the distances $D_s(Q, T_i)$ which the system computes using the set of weights $w_f$. The sum of squared errors

$$SSE(w) = \sum_{i=1}^{N} [D_s(Q, T_i) - D_u(Q, T_i)]^2 = \sum_{i=1}^{N} \Big[ \sum_f w_f \cdot d_f(Q, T_i) - D_u(Q, T_i) \Big]^2 \qquad (2)$$

gives rise to an optimisation problem for the weights $w_f$ such that (2) is minimised under the constraint of convexity. Using one Lagrangian multiplier we arrive at an analytical solution $w'$ for the weight set which changes the distance function. We get a different ranking of images in the database and, with equation (1), a layout for the new set of top-retrieved images on the screen.

## 4.5 Browsing

We allowed for two types of browsing. The first type allowed the user to move along the sequence of consecutive keyframes of a particular shot, and will henceforth be referred to as temporal browsing. The second type involved moving along nodes of a pre-computed image network, a process which we will refer to as lateral browsing. We will describe each method in turn.

### 4.5.1 Temporal Browsing

From the keyframe identifier, it was possible to determine for each keyframe of the test collection its left and right neighbours. This information was gathered offline, stored in an index file, and used at runtime to show the neighbours of any image over which the user moved the mouse pointer. Buttons associated with each image allowed the user to shift the tape back or forwards (see Figure 1).


Figure 1: Moving over a displayed image immediately causes its neighbouring shots to pop up as well. Clicking the << and >> buttons allows the user to move along the tape.

### 4.5.2 Lateral Browsing

In addition to temporal browsing, a novel browsing structure was implemented with the aim of enabling the user to efficiently move through the collection. Given a query image $Q$ and a vector $F$ of feature-specific similarities $F_i$, we compute the overall similarity between two images as the weighted sum over the feature specific similarities, i.e.

$$S(Q, I) = \sum_f w_f F_f$$

with the convexity constraint $\sum_f w_f = 1$, $w_f \geq 0$. According to our construction principle, an image $Q$ is connected to an image $I$ by a directed edge $I \rightarrow Q$ if and only if $I$ is the nearest neighbour of $Q$ for at least one combination of features, i.e. if there is at least one instantiation of the weight vector $w$ such that it causes the image $I$ to have the highest similarity $S(Q, I)$ among all images of the collection (excluding itself). The number of possible weight combinations is infinite since the weight vector is defined in a continuous vector space. However, because the overall similarity is a linear sum, small changes in any of the weights will induce correspondingly small changes in the similarity value. Points that are close in weight space should therefore produce a similar ranking and in particular, the same nearest neighbour. We can think of the weight space as being partitioned into a set of regions such that all weights from the same region result in the same nearest neighbour. For each image we store the set of its nearest neighbours. For each nearest neighbour we also store the proportion of feature combinations for which that image was ranked top. This number constitutes our measure of similarity between two images in the network.

To access the network, we initially display the top $k$ images from the collection with the highest number of neighbours thus allowing the user to branch out into as many regions of the network as possible. The user may select any of the images as an entry point into the browsing structure. Clicking on an image causes all of its neighbours in the network to be displayed on the screen. The position of each image in the collection is determined at random once and for all at the beginning. The initial display is shown in Figure 2.

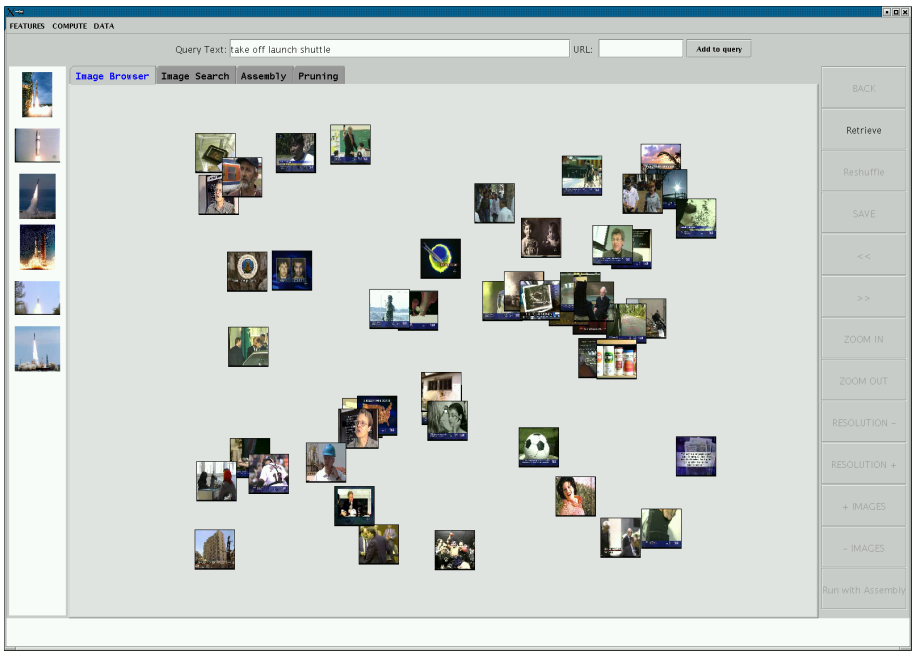


Figure 2: Initial access to the browsing structure is provided through high-connectivity nodes.

## 4.6 Assembly

To provide the user with a way of keeping track of the relevant images encountered in the course of a search session, we allow images to be added to a separate assembly panel by clicking the $A$ button associated with each image (see Figure 1). We use a hash table to ensure that an image can only be added to the assembly once. Towards the end of the search session, the user may specify the order in which the images collected on the assembly panel should appear in the final ranked list. This is done simply by clicking on them in the desired order. A "Retrieve with Assembly" button will perform a

search using the current weight set and the current query with the set of images from the assembly panel being prepended in the specified order to the ranked list thus obtained. Finally, duplicates are removed from the list (the one of lower rank) and the first 1000 images recorded for submission.

## 4.7 Experiments

### 4.7.1 Manual runs

Two manual runs were carried out. One was the compulsory run with only the text feature permitted. For the other run, we used topic-specific weight vectors to weigh each of the eleven features described above. The weight vectors were our crude guesses of the optimal weight sets and were based on inspection of the query images only.

### 4.7.2 Interactive runs

Four interactive runs were carried out which differed with regard to the type of user interaction allowed. Temporal browsing formed a core functionality that was enabled in all runs.

I Search + Browsing + Relevance Feedback: The users had available the full functionality of the system. They could freely formulate a query by selecting features, adding or removing query images and by modifying the text query. Users could attempt to optimise the weight set by performing relevance feedback on retrieved objects. In addition the users had full access to the browising structure.

II Search + Browsing: same functionality as in run I but without relevance feedback.

III Browsing: The users were not allowed to formulate any query but were allowed to see the query images and the text annotation that came with each topic. Images were selected by employing the browsing structure only.

IV Search + Relevance Feedback: Browsing was disallowed, but users could search and optimise weights using relevance feedback.

### 4.7.3 Block design

With a total of four users, 24 topics and four runs to be compared, the natural design choice for the interactive runs was a $4 \times 4$ latin square with the topics being divided into groups of six.

| | T100-105 | T106-111 | T112-117 | T118-124 |
|---|---|---|---|---|
| U1 | I | II | III | IV |
| U2 | IV | I | II | III |
| U3 | III | IV | I | II |
| U4 | II | III | IV | I |

where the rows and columns represent, respectively, the set of users and the set of topic blocks. For each user we randomised the precise temporal sequence of topics within each of the four blocks of topics. This should have helped to somewhat reduce the effect of learning on search performance.

## 4.8 Results

The results for the two manual runs are shown in Table 3. Although the inclusion of visual features does result in a higher mean average precision value, the difference is not significant at a 0.05 significance level. The smaller standard deviation for the second run with visual features included may be accounted for by the fact that multiple features have a moderating effect on performance.

The results for the four interactive runs are shown in Table 4. Three of our four runs were among the top eight overall. The best performance was achieved when only search with relevance feedback

| | mean average precision | rank out of 38 |
|---|---|---|
| TRECVID Best | 0.218 ± 0.168 | |
| TRECVID Median | 0.0720 | |
| TRECVID Mean | 0.0851 ± 0.0665 | |
| Text only | 0.074 ± 0.1125 | 19 |
| Text + Visual Features | 0.076 ± 0.0937 | 18 |

Table 3: Search task results for the manual runs averaged over all 25 topics (ranks based on mean average precision).

| | mean average precision | rank out of 36 |
|---|---|---|
| TRECVID Best | 0.4573 ± 0.276 | |
| TRECVID Median | 0.1939 | |
| TRECVID Mean | 0.182 ± 0.088 | |
| S + RF + B | 0.257 ± 0.219 | 5 |
| S + RF | 0.259 ± 0.210 | 4 |
| S + B | 0.234 ± 0.240 | 8 |
| B | 0.132 ± 0.187 | 27 |

Table 4: Search task results for the interactive runs averaged over all 24 topics (ranks based on mean average precision). S = search, RF = relevance feedback, B = lateral browsing

was performed (25.9% mean average precision). The performance difference to the run with browsing included (25.7% mean average precision) is not significant (one-sided $t$-test with $\alpha = 0.05$). As expected, the exclusion of relevance feedback had an adverse effect on performance with a decrease of mean average precision to 23.4%. The standard deviations are generally appreciable and thus the performance difference turns out not to be significant. Arguably the most surprising result is the fairly respectable performance achieved by the run in which only browsing was allowed. With 13.2% mean average precision, performance is markedly lower than for any of our other three runs (the differences to our two top-ranked runs are significant at $\alpha = 0.05$) but exceeds that of 25% of all the submitted runs. A closer inspection of the performance for individual topics reveals that performance tends to be either very bad, or above the median (as evidenced by the relatively high standard deviation). Searches for particular people (Arafat, the Pope, Freeman, Souder) proved particularly hard to execute successfully with the browsing structure, but it produced excellent results for topics where colour and textural cues were important (baseball, aerial view) but also for the unknown soldier's monument, with results well above the TRECVID median. This agrees with our own experience during search execution: For some queries it was difficult to find even one relevant shot, but once a relevant image had been found, the browsing structure made it easy to gather more relevant images by inspecting the first and second degree neighbours, or moving along a relevant shot using the temporal browsing functionality.

# 5 Conclusions

Our shot boundary detection algorithm once again proved highly accurate, despite minimal training in this domain, producing some of the best results among all systems. Our feature detection approach is promising, and we would expect improved performance with more thorough training.

With regard to the search task, this year's participation has brought a few pleasant surprises. Three of our four interactive runs produced very good absolute performance results. Although our fourth run, which involved browsing only, resulted in significantly lower performance than most of our other runs, it fared well in a global comparison. An important question which we did not address in our experiments

concerns the relative merit of the two browsing modes. Because temporal browsing was allowed in all the four interactive runs, our experiments do not allow us to differentiate between the effects of temporal and lateral browsing. It would be interesting to see how much gain in performance for the browse-only run was due to temporal browsing. Intuitively, lateral browsing is ideal to find a first set of relevant images at which point temporal browsing can take over, allowing the user to find even more relevant images. As such, the effect of temporal browsing should crucially depend on the effectiveness of lateral browsing.

**Acknowledgements:** This work was partially supported by the EPSRC, UK.

# References

[1] Managing Gigabytes search engine. http://www.cs.mu.oz.au/mg/.

[2] B. S. Manjunath and J.-S. Ohm. Color and texture descriptors. *IEEE Transactions on circuits and systems for video technology*, 11:703–715, 2001.

[3] T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[4] M. J. Pickering, D. Heesch, R. O'Callaghan, S. Rüger, and D. Bull. Video retrieval using global features in keyframes. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2003.

[5] M. J. Pickering and S. Rüger. Evaluation of key frame based retrieval techniques for video. *Computer Vision and Image Understanding*, 92(2):217–235, 2003.

[6] M. J. Pickering, S. M. Rüger, and D. Sinclair. Video retrieval by feature learning in key frames. In *Proceedings of International Conference on Image and Video Retrieval (CIVR)*, July 2002.

[7] D. Pye, N. J. Hollinghurst, T. J. Mills, and K. R. Wood. Audio-visual segmentation for content-based retrieval. In *5th International Conference on Spoken Language Processing, Sydney, Australia*, Dec. 1998.

[8] Y. Rubner. The earth-mover's distance as a metric for image retrieval. Technical Report STAN-CS-TN-98-86, Stanford University, 1998.

[9] D. Sinclair. Voronoi seeded colour image segmentation. Technical Report 1999.3, AT&T Laboratories, Cambridge, 1999.

[10] K. Tieu and P. Viola. Boosting image retrieval. In *5th International Conference on Spoken Language Processing*, Dec. 2000.

[11] D. Travis. *Effective Color Display.* Academic Press, San Diego, CA, 1991.

[12] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *ACM Multimedia Systems*, 1:10–28, 1993.