Oxford TRECVid 2007 – Notebook paper

James Philbin, Ondřej Chum, Josef Sivic, Vittorio Ferrari,
Manuel Marin, Anna Bosch, Nicholas Apostolof
and Andrew Zisserman
Department of Engineering Science
University of Oxford
United Kingdom

Abstract

The Oxford 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 sparse visual features. One used a standard bag-of-words representation, while the other additionally used a lower-dimensional "topic"-based representation generated by Latent Dirichlet Allocation (LDA). For both methods, we trained χ^2-based SVM classifiers for all high-level features using publicly available annotations [3].

In addition, for certain features, we took a more targeted approach. Features based on human actions, such as "Walking/Running" and "People Marching", were answered by using a robust pedestrian detector on every frame, coupled with an action classifier targeted to each feature to give high-precision results. For "Face" and "Person", we used a real-time face detector and pedestrian detector, and for "Car" and "Truck", we used a classifier which localized the vehicle in each image, trained on an external set of images of side and front views.

We submitted 6 different runs. `OXVGG_1`(0.073 mAP) was our best run, which used a fusion of our LDA and bag-of-words results for most features, but favored our specific methods for features where these were available. `OXVGG_2`(0.062 mAP) and `OXVGG_3`(0.060 mAP) were variations on this first run, using different parameter settings. `OXVGG_4`(0.060 mAP) used LDA for all features and `OXVGG_5`(0.059 mAP) used bag-of-words for all features. `OXVGG_6`(0.066 mAP) was a variation of our first run. We came first in "Mountain" and were in the top five for "Studio", "Car", "Truck" and "Explosion/Fire". Our main observation this year is that we can boost retrieval performance by using tailored approaches for specific concepts.

For the interactive search task, we coupled the results generated during the high-level task with methods to facilitate efficient and productive interactive search. Our system allowed for several "expansion" methods based on different image representations. The main differences between this year's system and last year's was the availability of many more expansion methods and a "temporal zoom" facility which proved invaluable to answering the many action queries in this year's task. We submitted just one run, `I_C_2_VGG_I_1_1`, which came second overall with an mAP of 0.328, and came first in 5 queries.

1 High-level Feature Extraction

For the high-level feature task, we used two generic methods which were run for all topics and used more specialized methods for particular topics. 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 an SVM 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 representations. The first uses a standard bag-of-words representation and the second concatenates this bag-of-words representation with a topic-based LDA representation.

1.1.1 Bag of visual word representation

The first method uses a *bag of (visual) words* [29] representation for the frames, where positional relationships between features are ignored. This representation has proved successful for classifying images according to whether they contain visual categories (such as cars, horses, etc) by training an SVM [10]. Here we use the kernel formulation proposed by [33].

Figure 1: An example of Hessian-Laplace regions used in the bag of words method. Left: original image; right: sparse detected regions overlaid as ellipses.

Features and bag of words representation. We used Hessian Laplace(HL) [21] interest points coupled with a SIFT [20] descriptor. This combination of detection and description generates features which are approximately invariant to an affine transformation of the image, see figure 1. These features are computed for all reduced i-frames. The "visual vocabulary" is then constructed by running unsupervised K-means clustering over both the training and test data. The K-means cluster centres define the visual words. We used a vocabulary size of $K = 10,000$ visual words. The SIFT features in each reduced i-frame are then assigned to the nearest cluster centre, to give the visual word representation, and the number of occurrences of each visual word is recorded in a histogram. This histogram of visual words is the bag of visual words model for that frame.

Topic-based representation We use the Latent Dirichlet Allocation [5, 16] model to obtain a low dimensional representation of the bag-of-visual-words feature vectors. Similar low dimensional representations have been found useful in the context of unsupervised [26, 28] and supervised [6, 25] object and scene category recognition, and image retrieval [17, 27]. We pool together both TRECVid training and test data in the form of 10,000 dimensional bag-of-visual words vectors and learn 20, 50, 100, 500 and 1,000 topic models. The models are fitted using the Gibbs sampler described in [16].

These representations are concatanated into a single feature vector, each one independantly normalized, such that the bag-of-words and the individual topic representations are each given equal weight. This approach was found to work best using a validation set taken from the training data.

SVM classification. To predict whether a keyframe from the test set belongs to a concept, an SVM classifier is trained for each concept. Specifically, a kernel SVM with χ^2 kernel

$$K(p,q) = e^{-\alpha\chi^2(p,q)}$$

where

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

Figure 2: Four frames from the beginning of a shot, and the pedestrians detected in them. Every fourth frame is shown.

is used. The parameter α in the kernel function is set to be an estimate of the average χ^2 distance between training images. We used the SVM-light [18] package.

The positive and negative training examples are obtained using the collaborative annotation over the 2007 training data [3]. All shots positively labelled in the annotations plus a random selection of negative examples were used for training.

SVM parameters (slack variables plus error weights for the misclassified positive/negative examples) were determined using a validation set.

1.2 Feature-specific Approaches

In addition to the generic approaches previously described, we also used specific, tailored methods for the following features: "Face", "Person", "Walking/Running", "Car" and "Truck". These were found to significantly outperform the generic approaches on these categories.

1.2.1 Pedestrian Detection

We describe here the approach used to detect pedestrians, track them over time, and to classify the tracks as either walking/running or standing.

We start by detecting *pedestrians* in every frame separately, by using the detector of Dalal and Triggs [11]. This approach slides a window over the image at various locations and scales and classifies each as either a pedestrian or not. The classifier is based on the spatial distributions of oriented gradients. Figure 2 shows detections on three frames from the beginning of a shot.

At this point, the system is not aware that there are two persons in the shot, and doesn't know yet how they evolve over time. For this purpose, we associate detections over time using a graph clusterer [15] 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 3.

Once we have obtained the person tracks, we want to determine whether the person is walking/running or simply standing still. We have devised two cues for this task. The first cue primes tracks lasting for many frames, and includ-

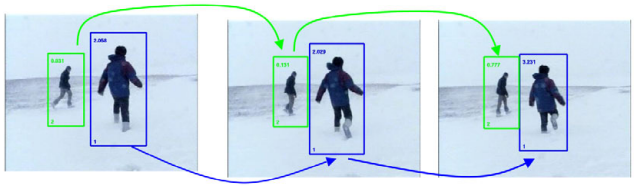


Figure 3: The detections for the frames of figure 2, associated over time.

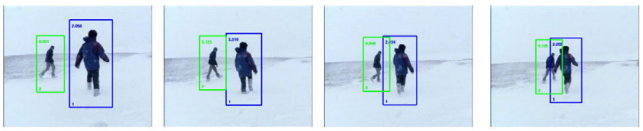


Figure 4: A longer view on the tracks of figure 3. Shown frames 1,10,20, and 30.

ing high-scored detections. The primary effect of this cue is to weed out false-positive tracks, not corresponding to a person. Figure 4 shows a longer, 30-frame interval of the shot in figure 3. The score of a detection is assigned by the initial detector [11] and is displayed on the top-right corner.

In many shots the camera tracks a walking person, making it appear static. In other shots, a static person appears moving due to camera motion. Hence, absolute displacement in the image plane is not a reliable cue for walking/running. We devised a better cue, which checks whether the pixel patch inside the detection bounding-box changes over time. This *appearance* cue effectively spots static pedestrians. In more detail, we describe the appearance of a detection with the spatial colour histograms of [7] and compute change as their variance over the duration of the track. Figure 5 illustrates this idea on a close-up over a track from figure 4.

1.2.2 Face Detection

This section describes our face detection approach (for details see [1] which presents a real-time version of the soft-

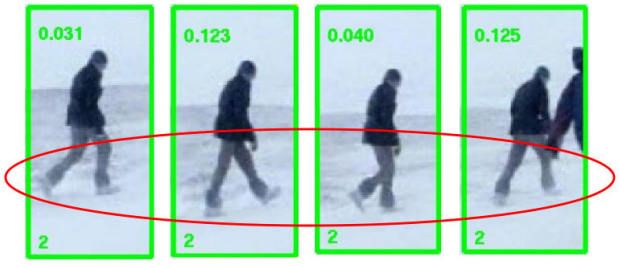


Figure 5: Close-up on one of the tracks of figure 4. The moving legs causes changes in the appearance descriptor over time. This cues indicates that the person is walking/running.

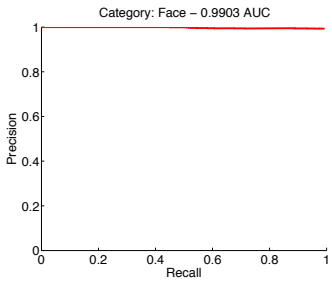
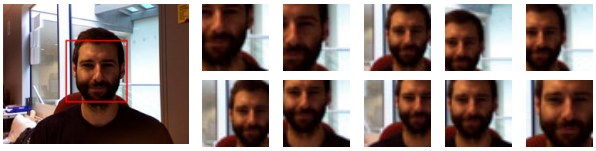


Figure 6: Face precision-recall curve for the highest ranked 2000 shots in the training dataset. The area under the curve (AUC) score for this curve is 0.9903



(a) Face detection (b) Training samples

Figure 7: **Training the regressor from face detections**. Given a set of input faces from the face detector, (a), a set of 192 training pairs (b) are artificially sampled over the state-space of face poses for training.

ware 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). See figure 6 for details.

Face detection and tracking. The first stage of processing is frontal face detection which is done using the Viola-Jones cascaded face detector [31]. 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 [32]. 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 (see figure 7). 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 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

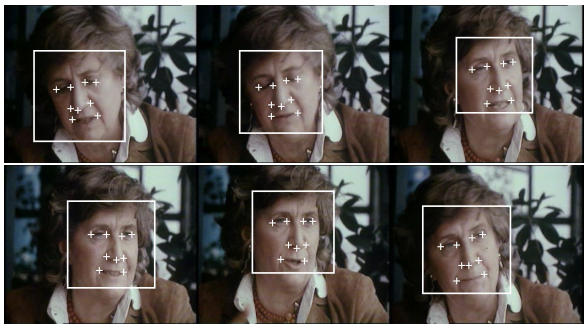


Figure 8: Face track with localized facial features. Tracking and feature localisation performs well even with changes of pose and expression.

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 [14], and further details can be found in [13, 12]. Figure 8 shows a face track with the respective feature localization, showing that the features can be located with high reliability despite variability in pose, lighting and facial expressions.

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

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

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.

1.2.3 The Exemplar Model for Cars/Trucks

The model for each class consists of a set of exemplars obtained from regions of interest (ROIs) around the object class instances in the training images. Each exemplar represents the spatial layout of visual words and edge directions in the region using a hierarchical spatial histogram. The spatial correspondence between an exemplar and a target image region can then be assessed by a level-weighted distance [19] between the histograms representing the exemplar and target. Figure 10 illustrates this correspondence. Example detections are shown in figure 9. Implementation details are given in [9].

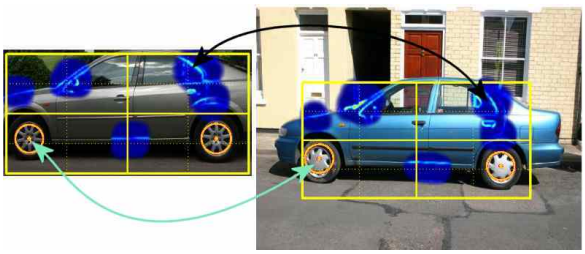


Figure 10: An exemplar image and a corresponding class instance in a car side training set. The hierarchical representation and cost function measure the spatial correspondence between sparse visual words and dense edge distributions. Some corresponding visual words and edges are highlighted.

Learning the exemplar model Suppose we know the model, and wish to detect a class instance in a target image. This can be done by a search for a ROI in the target image that matches well with one of the exemplars, i.e. as a minimization of the distance between the exemplars and target region as the target region is varied. We define the following cost function to measure this similarity:

$$C_{D} = \sum_{X} (d(X^w, Y^w)) + \alpha\,(d(X^e, Y^e)) + \beta \frac{(A-\mu)^2}{\sigma^2} \qquad (2)$$

where X^w and X^e are the hierarchical spatial histograms of visual words and edge directions, respectively, in the exemplars, and Y is similarly defined for the target image ROI. The sum is over the set of exemplars X of the model. A is the aspect ratio of the target region, μ and σ are the aspect ratio average and variance, respectively, of the exemplar ROIs. The cost C_{D} is a weighted sum of three terms: the pair wise distance between the visual words of the target region and exemplar, the pair wise distance between the edge directions of the target region and exemplar, and a cost for the aspect ratio of the target region deviating from the average aspect ratio. The distance function used is defined below.

The detection problem involves finding the target region that minimizes C_{D}. Examples are shown in figure 9We now turn to learning the exemplar set model from training images. Suppose we are given a set $\mathcal{T}$ of N training images, we wish to find the region in each training image which best matches with regions in the other training images. These regions will define the exemplar set. This is equivalent to the detection problem above, where now we must learn the regions in all images simultaneously. The cost function is then a sum of distances between all pairs of training examples

$$C_{L} = \sum_{X \in \mathcal{T}} \sum_{Y \in \mathcal{T}} (d(X^w, Y^w)) + \alpha\,(d(X^e, Y^e)) + \beta \frac{(A-\mu)^2}{\sigma^2} \qquad (3)$$

and we wish to find the region in each training image such that C_{L} is minimized.

Figure 9: Top ten car results from the test set, showing the cars correctly detected and localized.

Thus, learning the model involves: (i) automatical location of the exemplar regions from the training set; and (ii) selecting the value of the parameters α, β and learning the parameters μ and σ.

Distance functions. It is well known that distances may be strongly corrupted by the presence of an outlier, i.e. in this case an example image not containing an instance of a category object, or a missed detection. Instead of histogram intersection we use a (squared) χ^2 distance since then a single training image has a limited influence on the model. This follows from the fact that the cost function (3) is additive and the contribution of each exemplar is bounded by a constant. So,

$$d(x, y) = (\chi^2(x, y))^2$$

. In our experiments, the sum of squared χ^2 distances outperformed the sum of χ^2 distances as well as the Jensen-Shannon divergence.

1.3 Merging lists

To fuse the ranked lists, generated using the different methods, we used a weighted Borda count method [2], which assigns votes to each candidate depending on its rank in the list. These votes are then accumulated over each list to fuse and the final rank is generated by sorting the candidates in non-increasing order over a weighted sum of the votes, where the weights quantify some measure of relative confidence between the multiple sources. The weightings for each concept are determined from a validation set, taken from the training data.

2 Interactive Search

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 second with an mAP score of *0.328*. We compare our score to other teams in figure 11.

Our expansion-search algorithms allow us to search for images with the same object (particular object search), similar textural layout (bag-of-words, spatial bag-of-words and LDA), similar colour layout (spatial colour histograms) or which are nearly identical (near-duplicates).

2.1 Particular object search

Some of the queries for TRECVid this year can be partially answered by using real-time, human selected object search over the corpus. For example, searching for the distinctive pattern of piano keys in answering the query "Find shots of one or more people playing musical instruments such as drums, guitar, flute, keyboard, piano, etc." allows us to find more shots containing pianos.

Here we describe an implementation of the "Video Google" approach [29, 24, 8] for the TRECVid corpus. The aim is to retrieve shots containing a specific object despite changes in scale, viewpoint and illumination. The visual query is specified at runtime by outlining the object in an example image.

For the visual features, we use Hessian-Affine [22] regions, which are invariant to changes in object illumination, scale, rotation and viewpoint. We then use SIFT descriptors [20] to describe each elliptical region in the image. The SIFT descriptors for these appearance regions are vector quantized into visual words, to give us a visual words repre-

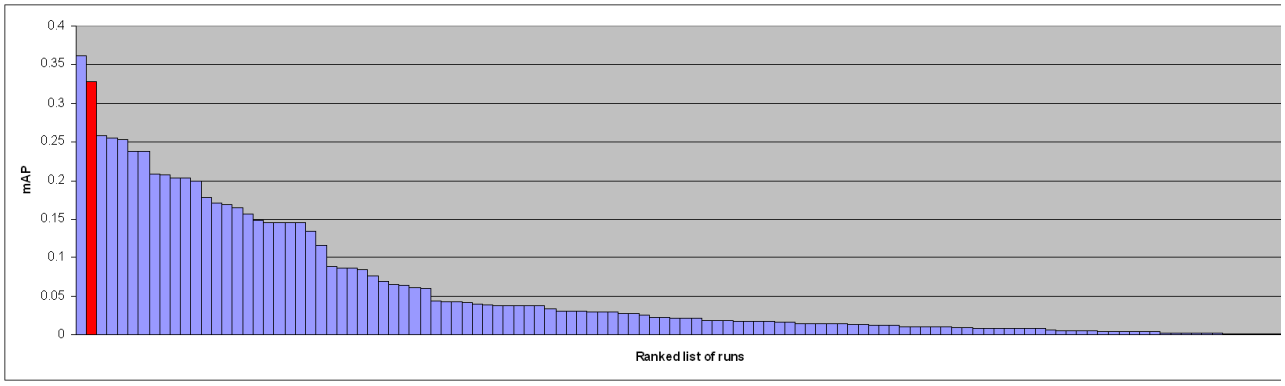


Figure 11: Ranked list of all TRECVid runs for interactive search, with our single submission marked in red.

sentation for each i-frame. With this representation, standard efficient text retrieval methods [4] can be employed to enable object retrieval in a Google-like manner.

The vector quantization is carried out using an approximate K-means method [24] which allows us to use very large, highly discriminative visual vocabularies. For the interactive search task, the user was able to search using vocabularies of size $K = 10,000$, $K = 100,000$ and $K = 1,000,000$. This search was coupled with a fast spatial re-ranking method [24] to improve retrieval quality.

2.2 (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 [19]. We re-used our topic-based representation based on LDA, to provide the user with a third expansion method.

We also implemented a global gradient orientation descriptor similar to SIFT [20] 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 [23], with an L_2 distance for measuring similarity.

2.3 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 [7]. This used a bag-of-words representation coupled with a min-hash search algorithm to quickly compute an approximate set overlap score.

2.4 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. LDA concept rankings.
3. Specific "People Walking/Running" rankings.
4. Face detections.
5. Pedestrian detections.
6. Particular object expansions.
7. Texture expansions.
8. Colour expansions.
9. Near-duplicate detections.

Inspired by the *CrossBrowser* approach [30], the main interface view contains two axes (see figure 17). 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.

A difference from last year's interface was a "temporal zoom" 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 proved vital for answering the many action queries in this year's competition. Additionally, every shot on the screen could be

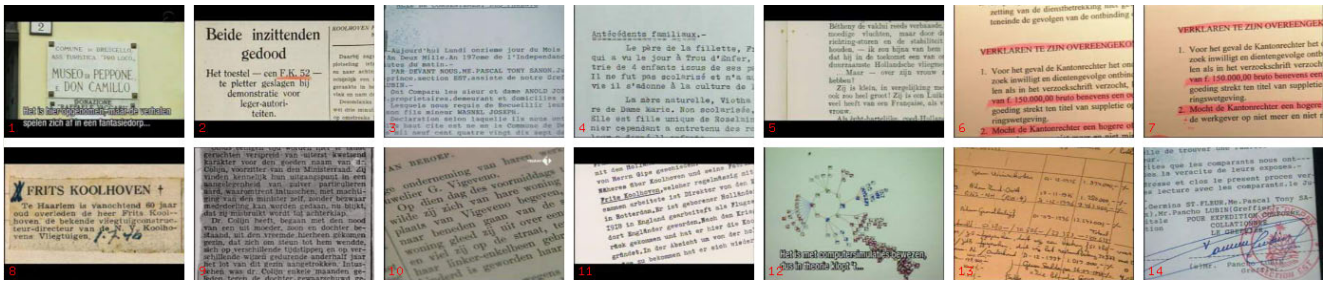


Figure 12: BOW expansion on “text”. The query image is labelled 1, with the top 14 results shown.

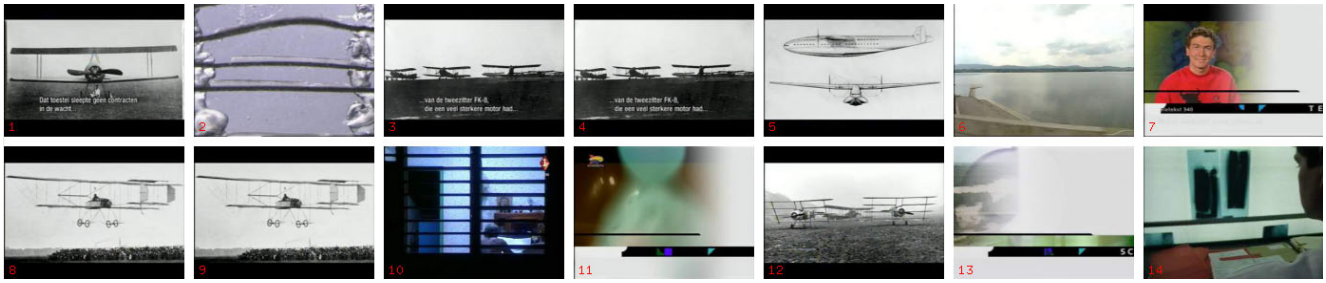


Figure 13: Spatial BOW expansion on “airplane”.

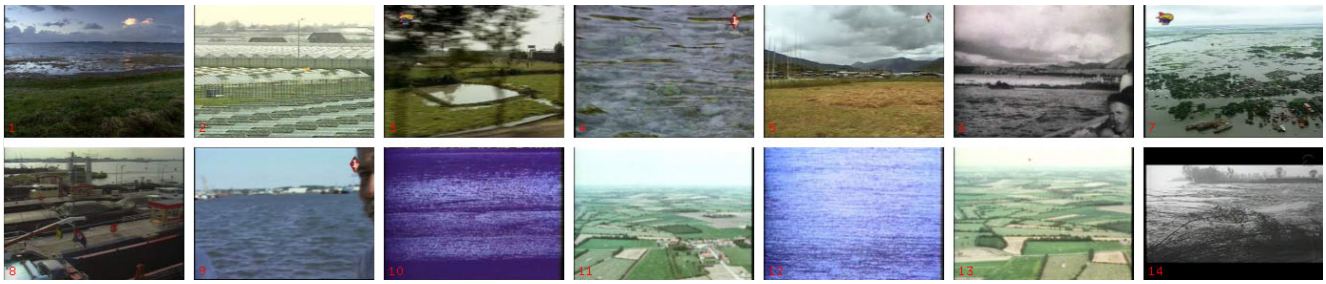


Figure 14: LDA (topic-based) expansion on “waterfront”.

Figure 15: Colour expansion on “mountain”.

Figure 16: Texture expansion on “crowd”.

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. Additionally, this year, we incorporated external image search into the system. This allows the user to drag and drop new images into the interface which can then be 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. We also used Google Images as a source of external images for example in the "Find shots with sheep or goats" query.

References

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

[2] J. Aslam and M. Montague. Models for metasearch. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284. ACM Press, 2001.

[3] Stéphane Ayache and Georges Quénot. Evaluation of active learning strategies for video indexing. *Image Commun.*, 2007.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, ISBN: 020139829, 1999.

[5] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *J. Machine Learning Research*, 3:993–1022, Jan 2003.

[6] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via plsa. In *Proc. ECCV*, 2006.

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

[8] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007.

[9] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[10] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

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

[12] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. In *Proceedings of the British Machine Vision Conference*, 2006.

[13] 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.

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

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

[16] T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[17] E. Hoerster, R. Lienhart, and M. Slaney. Image retrieval on large-scale image databases. In *CIVR*, 2007.

[18] T. Joachims. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[19] 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.

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

[21] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. ECCV*. Springer-Verlag, 2002.

[22] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

Figure 17: 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 seperate panes.

[23] 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.

[24] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[25] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica, T. Tuytelaars, and L. Van Gool. Modeling scenes with local descriptors and latent aspects. In *Proc. ICCV*, pages 883–890, 2005.

[26] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[27] J. Sivic. *Efficient Visual Search of Images and Videos*. PhD thesis, University of Oxford, 2006.

[28] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proc. ICCV*, 2005.

[29] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477, Oct 2003.

[30] 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.

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

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

[33] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: An in-depth study. Technical Report RR-5737, INRIA Rhône-Alpes, Nov 2005.