IRIT @ TRECVid HLF 2009
Audio to the Rescue

Hervé Bredin[12] and Lionel Koenig[2] and Hélène Lachambre[2] and Elie El Khoury[2]
[1] Centre National de la Recherche Scientifique
[2] Institut de Recherche en Informatique de Toulouse

Abstract

This notebook paper describes the six runs submitted for the first participation of IRIT at TRECVid 2009 High-Level Feature Extraction task. They were submitted in an attempt to start answering two research questions:

1. Can acoustic information be of any help in this (historically) video-only task?
2. Are Support Vector Machines robust enough to deal with noisy and unbalanced datasets?

The six submitted runs can be described and compared as follows:

- Run 6 (**A_IRIT_V_Mono_6**) SVM-based late-fusion of visual descriptors
- Run 4 (**A_IRIT_AV_Mono_4**) SVM-based late-fusion of visual and audio descriptors
- Run 5 (**A_IRIT_V_Poly_5**) Same as run 6 except scores from other concepts are added during the late-fusion process
- Run 3 (**A_IRIT_AV_Poly_3**) Same as run 4 except scores from other concepts are added during the late-fusion process
- Run 1 (**A_IRIT_AV_BestAvg_1**) For each concept, uses the best of runs 3 and 4
- Run 2 (**A_IRIT_AV_BestMax_2**) Difference between runs 1 and 2 is explained in Section 4.3.

Taking into account the relatively poor performance of the six submitted runs (average precision ranges between 0.022 and 0.027), no definitive answer can be given to the first question: audio definitely helps for some concepts and is useless for others, and additional work has to be done on how to use SVM efficiently in this task.

1 Introduction

This notebook paper describes the six runs submitted for the first participation of the SAMoVA team of IRIT at TRECVid 2009 High-Level Feature Extraction task [**?**].

1.1 Notations

As for any supervised learning approach, our approach makes use of a training set. Because it is based on a two-steps training, the training set had to be divided into two parts $\mathcal{S}_1$ and $\mathcal{S}_2$ as described in Figure 1 – each of them being further divided into three disjoint cross-validation sets ($\mathbb{CV}_1$ to $\mathbb{CV}_3$).

Here is a quick list of notations used in the rest of the document. All concepts that are not clearly defined in this list will be thoroughly introduced in the document. This list is meant to be used as an easy reminder when reading the document.

- s_i is the i^{th} shot / $i \in [\![1, N_{\text{shots}} = N_s]\!]$.

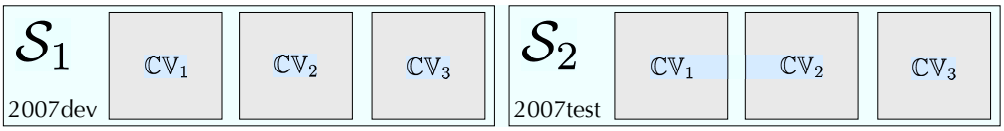

Figure 1: Partitioning the training dataset into two disjoint sets $\mathcal{S}_1$ and $\mathcal{S}_2$

- $y_i^c \in \{0,1\}$ is the c^{th} concept groundtruth annotation for shot s_i / $c \in [\![1, N_{\text{concepts}} = N_c]\!]$. $y_i^c = 1$ means that shot s_i contains the c^{th} concept, $y_i^c = 0$ means it does not.
- x_i^d is the d^{th} descriptor extracted from shot s_i / $d \in [\![1, N_{\text{descriptors}} = N_d]\!]$.
- SVM^{dc} is the SVM classifier for c^{th} concept, using $\{x_i^d\}$ descriptors as input.
- p_i^{dc} is the probability (as output by SVM^{dc}) for shot s_i to contain the c^{th} concept. In other words, $p_i^{dc} = p(y_i^c = 1 | x_i^d, \text{SVM}^{dc})$.
- $\delta^{dc} \in \{0,1\}$ is the *indicator of usefulness* of d^{th} descriptor for c^{th} concept detection.
- f_i^c is the late-fusion vector for shot s_i made from the *selective* concatenation of $\{p_i^{dc}\}$ / $\{d \in [\![1, N_d]\!] \, | \delta^{dc} = 1\}$.
- SVM^c is the SVM classifier for c^{th} concept, using $\{f_i^c\}$ vectors as input.
- p_i^c is the probability (as output by SVM^c) for shot s_i to contain the c^{th} concept. In other words, $p_i^c = p(y_i^c = 1 | f_i^c, \text{SVM}^c)$.
- f_i is the *cross-concept* late-fusion vector for shot s_i made from the concatenation of $\{f_i^c\}$ / $c \in [\![1, N_c]\!]$.
- SVM^{c*} is the SVM classifier for c^{th} concept, using $\{f_i\}$ vectors as input.

1.2 Overview

Every run that was submitted by IRIT is a variation of a same generic and modular supervised learning approach. Its modules will be described in detail in the rest of the paper. We give here a quick overview of their interconnections.

In Figure 2, a large set of low-level descriptors are extracted from each shot s_i of the training set $\mathcal{S}_1$. They are denoted $\{x_i^d\}$, d varying from 1 (first type of descriptor) to N_d (last type of descriptor). For each high-level feature to be detected – we will call them *concept* in the rest of the document – groundtruth annotation $\{y_i^c\}$ is used in combination with each type of descriptor to train one support vector machine classifier per type of descriptor.

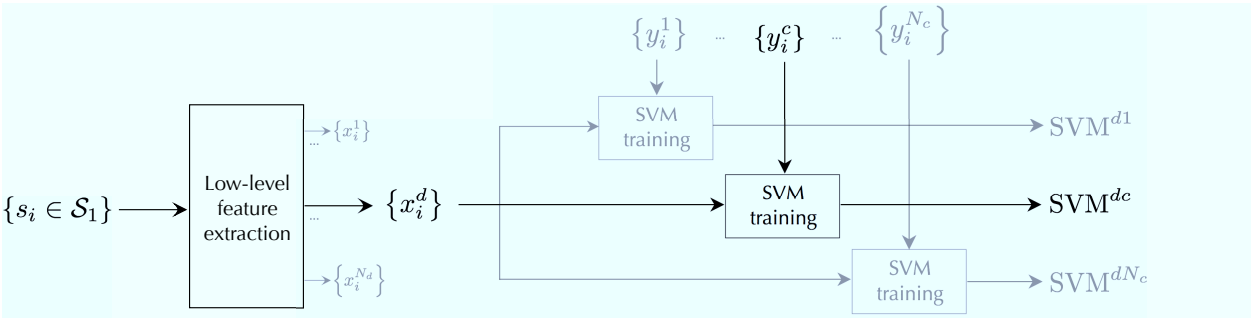

Figure 2: Training an SVM classifier for c^{th} concept using d^{th} descriptors on subset $\mathcal{S}_1$

In the end, for each concept, N_d SVM classifiers are available that can be applied later on descriptors extracted from the second half of the training set $\mathcal{S}_2$ to compute probabilities $\{p_i^{dc}\}$ for shot s_i to contain concept c. As shown in Figure 3, based on their indicator of usefulness $\{\delta^{dc}\}$ described in Section 4.1, some of these N_d probabilites (or scores) are selected and concatenated into late-fusion vectors $\{f_i^c\}$ used as input of another SVM training step that performs the actual fusion.

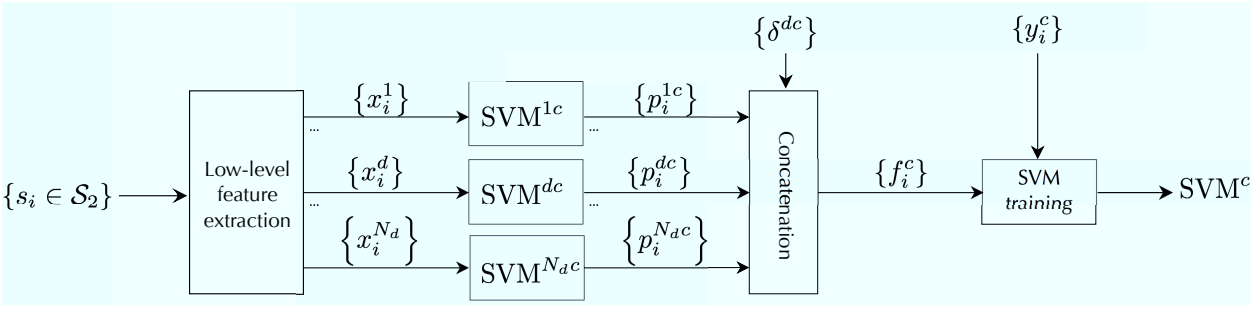

Figure 3: Training an SVM classifier for c^{th} concept using a late-fusion strategy on subset $\mathcal{S}_2$

For each shot of the test set $\mathcal{S}$, descriptors $\{x_i^d\}$ are extracted, their probabilites $\{p_i^{dc}\}$ are predicted based on $\{\text{SVM}^{dc}\}$, then fused and the final scores are computed using $\{\text{SVM}^c\}$.

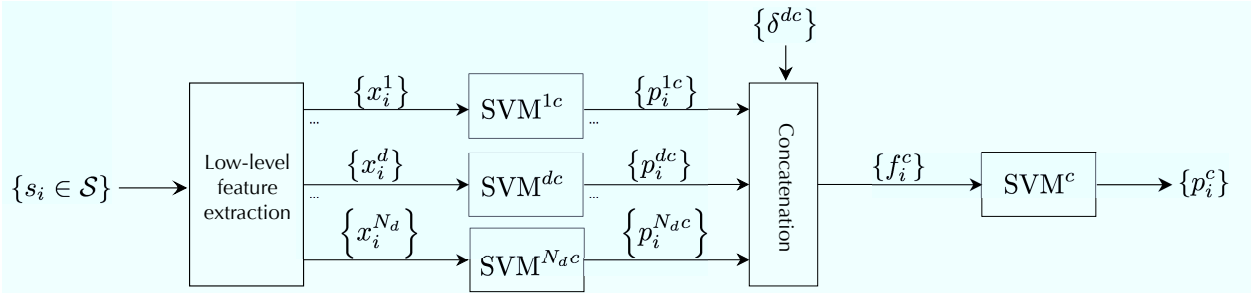

Figure 4: Applying classifiers on test set $\mathcal{S}$

2 Low-level feature extraction

2.1 Visual descriptors

For our first participation to the TRECVid HLF task, a very limited set of visual descriptors were available. All of them were extracted from one keyframe per shot, as shown in Figure 5. Keyframes selection was provided by the collaborative annotation effort [2].

Two out of three visual descriptors are based on local keypoints detected *a la* SIFT using the *VLFeat* open source library [10]. From these keypoints, either Lowe's SIFT descriptors [9] or hue histograms are extracted. Eventually, each keyframe is described as a 250-dimensional TF-IDF vector.

OpenCV open source library [1] is used to perform face detection, leading to the extraction of a 3-dimensional vector containing the number of detected faces in the keyframe, the size of the largest face and the keyframe area covered by faces.

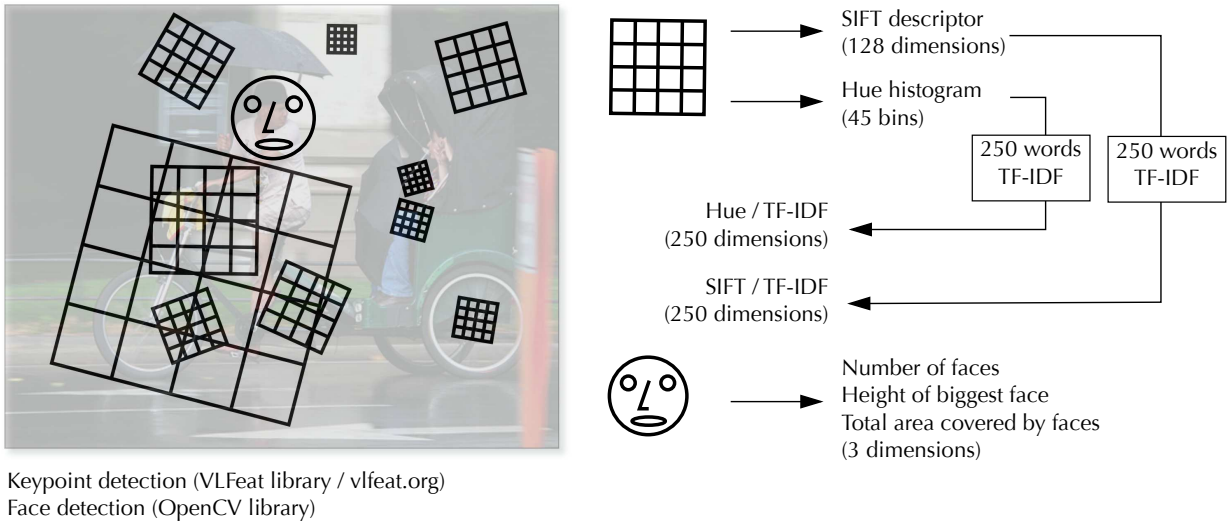

Figure 5: A limited number of visual descriptors

2.2 Acoustic descriptors

As far as low-level feature extraction is concerned, most of the effort was put into investigating how audio descriptors can help in this task. Therefore, a large set of audio descriptors were extracted, the list of which is given here.

2.2.1 Definitions

Unless stated otherwise, audio descriptors are computed on a 20 ms sliding window with a 10 ms overlap. In the following definitions, N is the number of audio samples contained in the signal window and acoustic samples are denoted $\mathtt{x}_n$ with $n \in [\![1, N]\!]$.

Energy is defined as the log-energy of the audio signal: $E = \log \sum_{n=1}^{N} \mathtt{x}_n^2$

Zero-Crossing Rate is the rate of sign-changes along the audio signal. It is directly linked to the frequency of the signal, and can be computed as follows: $\mathrm{ZCR} = \frac{1}{2} \sum_{n=1}^{N} |\mathrm{sign}(\mathtt{x}_{n-1}) - \mathrm{sign}(\mathtt{x}_n)|$

4 Hz modulation The 4Hz modulation of energy [6] aims at capturing the 4Hz syllabic rate. Therefore, the energy in 40 frequency subbands is filtered with a 4Hz band-pass filter. Energy is summed for all channels. The modulation is obtained by computing the filtered energy variance (in dB) over one second of signal. This parameter is higher for speech signal than it is for music or noise.

Spectral statistics are high-order statisitics on spectral power density of each signal window. Among them are spectral mean, variance and kurtosis.

Mel-Frequency Cepstral Coefficients are descriptors extracted from the power spectral density of the signal. They are commonly used in applications dealing with speech recognition or speaker authentication.

Voicing percentage $V_{\%}$ represents the harmonicity of the signal window. It is computed as the ratio between the harmonic power and the power of each signal window [8].

Pitch is estimated using the YIN algorithm [4], which is based on the cumulative mean normalized difference function defined as follows: $\mathrm{cmnd}(\tau) = \sum_{n=-\infty}^{+\infty} (\mathtt{x}_n - \mathtt{x}_{n+\tau})^2$.

In case of a T-periodic signal, cmnd is minimum for $\tau = T$. If cmnd(τ) reaches a value below a predefined threshold, the signal window is set as pitched and the pitch value is the index of the first minimum of the cmnd function. The pitch value is set to zero otherwise.

Vibrato is a periodic oscillation of the fundamental frequency of musical signals. For the singing voice, its rate is between 4 and 8 Hz. We compute it on the fundamental frequency estimated with the YIN algorithm. This parameter is supposed to be high in presence of singing voice.

2.2.2 Making sense of signal-level acoustic descriptor at shot level

Most of these audio descriptors are extracted every few milliseconds, using a sliding temporal window (every 10 ms typically). Therefore, the way the collaborative annotation was performed (a shot is labelled as positive even though the feature is visible for only a fragment of the shot) makes the raw audio descriptors difficult to use for training a binary classifier: Should we consider as positive samples all audio descriptors extracted from a positively annotated shot even though most of them would be considered as negative in a less coarse annotation?

This is an interrogation that we tried to avoid by using basic statistical properties of the audio descriptors rather than the audio descriptors themselves. As shown in the right part of Figure 6, each shot can be described as the dimension-wise mean, variance, minimum or maximum of the audio descriptors it *contains*.

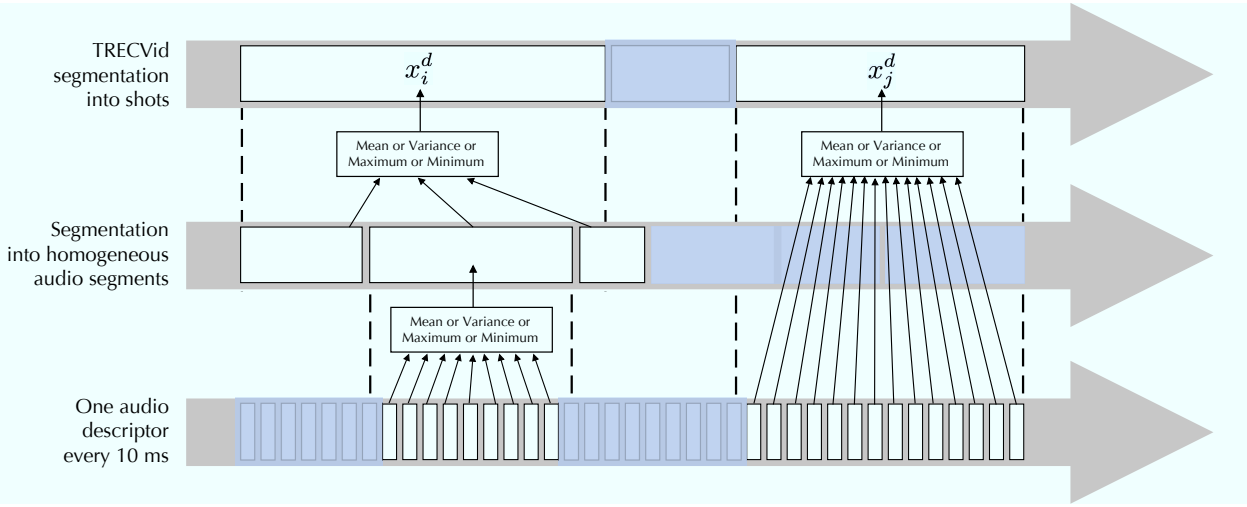

Figure 6: Three levels of description

However, these basic transformations do not always make sense on the whole duration of a shot. Therefore, we propose to use an additional segmentation step based on the sole acoustic information. The audio stream is segmented into homogeneous zones, where each zone ideally corresponds to one single audio source. Different music tones, speakers, noises or silence should be split into different segments. The algorithm was first introduced in the context of speaker diarization [5] and uses a combination of the Generalized Likelihood Ratio (GLR) and the Bayesian Information Criterion (BIC).

As shown in the left part of Figure 6, each acoustically homogeneous segment is described as the mean, variance, minimum or maximum of the audio descriptors it *contains*. Then, each TRECVid shot is described as the mean, variance, minimum or maximum of the audio segment descriptors it contains.

3 Support Vector Machines training

In order to train and perform the actual classification (i.e. to decide whether a shot contains a concept or not), *libSVM* [3] implementation of the SVM classifier with RBF kernel was used. Parameters tuning (cost and gamma) is based on a three-fold cross-validation grid search.

3.1 *David & Goliath* – SVM with unbalanced datasets

One main problem with 2-class SVMs is that they do not always behave at their best when training sets are unbalanced. Figure 7 illustrates this unwanted behavior. 10.000 negative and 50 positive samples are

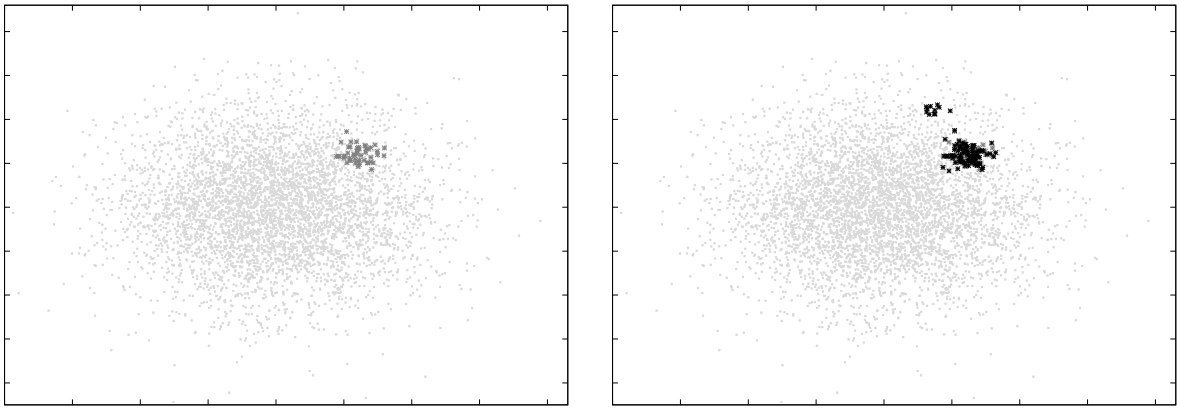

Figure 7: Left – 10.000 negative (**light gray** •) and 50 positive (**dark gray** •) samples drawn from two normal 2D distributions. Right – negative support vectors (**black** •) computed by SVM with RBF kernel.

drawn from two normal distributions. Applying SVM as it is leads to the extraction of negative support vectors that are very close to positive samples (if not coinciding with them). The resulting binary classifier therefore tends to classify every new unknown sample as negative.

Table 1 provides a quick overview of the training set balance between positive, negative and un-annotated shots in the development set provided by NIST for the TRECVid HLF 2009 task. It appears that, for some concepts, negative samples are 1000 times more numerous than positive ones. This table also uncovers an additional potential problem: there might not be enough positive samples to accurately train an efficient SVM classifier.

Concept ID	001	002	003	004	005	006	007	008	009	010
Positive (%)	1.66	0.72	0.02	0.05	0.17	0.61	0.03	0.13	0.02	2.52
Negative (%)	97.96	97.91	99.63	99.61	98.55	99.24	98.93	99.72	99.81	96.89
No annotation (%)	0.38	1.36	0.35	0.34	1.27	0.15	1.04	0.15	0.17	0.58
Concept ID	011	012	013	014	015	016	017	018	019	020
Positive (%)	0.03	0.31	0.05	1.00	3.21	0.08	0.93	0.70	0.46	1.47
Negative (%)	99.53	99.32	99.56	98.70	93.34	99.47	98.25	98.34	97.21	97.80
No annotation (%)	0.44	0.37	0.39	0.30	3.44	0.45	0.83	0.96	2.33	0.73

Table 1: Distribution of the training samples for each concept. Only 74 positive samples (out of 43161) are available for the 9th feature. At best, for the 15th feature, 1502 positive samples are available.

3.2 *Slimming down Goliath* – Iterative removal of negative support vectors

In order to address the unbalance problem, we used an iterative process aiming at removing selected samples from the dominant negative class (i.e. shots that do not contain the concept) in order to improve the overal efficiency of SVM prediction. Figure 8 illustrates this algorithm when applied on simulated 2-D datasets.

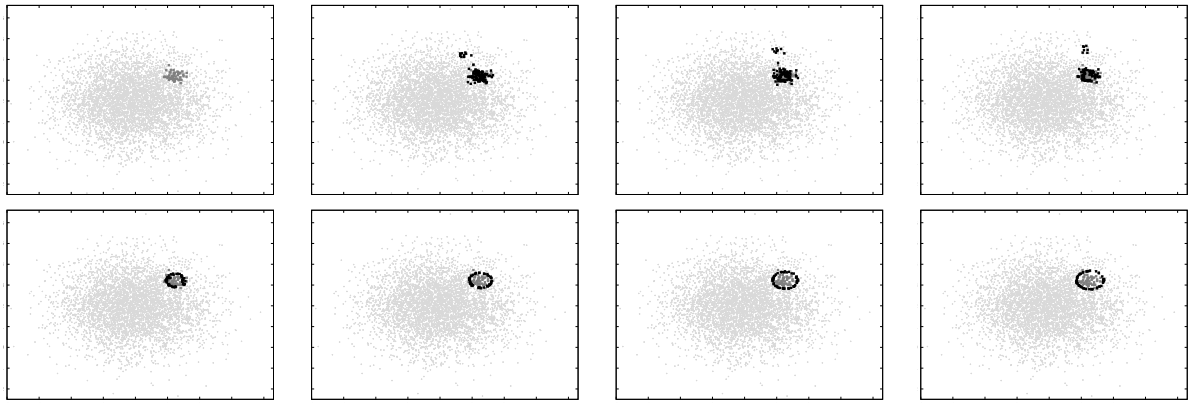

Figure 8: Evolution of negative support vectors for the first 7 iterations. Left to right, top to bottom.

First, SVM is trained using the whole training set. Support vectors (**black** •) from the (dominant) negative class (light gray •) are removed from the training set and SVM is trained again using the resulting training set. Using the cross-validation paradigm (training on 2 out of the 3 $\mathbb{CV}_i$ sets and testing on the other one), average precision is estimated after each iteration of the algorithm. The latter stops when at least 10 iterations were performed and a local maximum was reached. For instance, in Figure 9, the optimal number of iterations is set to 5.

This approach was used for the training of both mono-descriptor (Figure 2) and fusion (Figure 3) SVMs.

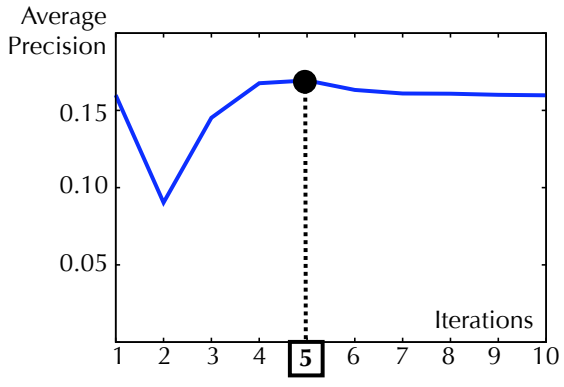

Figure 9: Average precision w.r.t. iterations

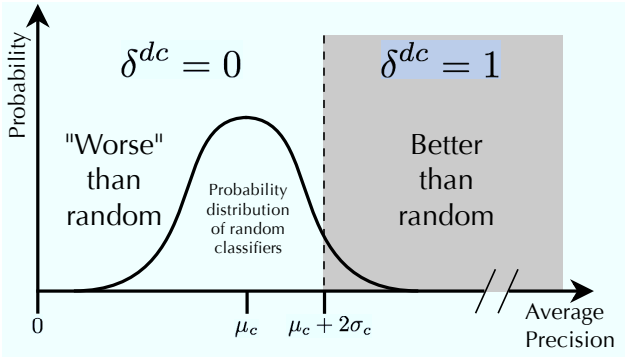

Figure 10: Indicator of usefulness

4 Late fusion

As mentioned in Section 1.2, all six runs are based on the late fusion of scores given by each type of descriptors.

4.1 Indicator of usefulness δ^{dc}

For a given concept, some low-level descriptors happen to be very useful while others are completely useless. Some of them might even be confusing for the final fusion stage. In order to estimate how well a given descriptor performs, we compare its performance to the one of a random classifier (a random value generator that outputs a score between 0 and 1 for each shot).

As shown in Figure 10, based on one hundred runs of a random classifier, it is possible to obtain the probability distribution of scores of a random classifier for each concept c. A descriptor d is said to be *useful* ($\delta^{dc} = 1$) if it performs better than most of the random classifier runs (i.e. its average precision on the cross-validation set is higher than $\mu_c + 2\sigma_c$). It is said to be *useless* otherwise ($\delta^{dc} = 0$). Only *useful* descriptors are concatenated into the fusion vectors $\{f_i^c\}$.

4.2 *Mono-concept* vs. *cross-concept* fusion

This type of fusion is said to be *mono-concept* as selected fused scores into $\{f_i^c\}$ are part of larger set of scores $\{p_i^{dc}\}$ all resulting from SVMs that were trained based on the sole annotation of a given concept c. In other words, for a given concept c and a shot s_i, f_i^c will not depend on whether s_i contains concept c' or not.

However, we can definitely imagine some concept c (*space shuttle*, for instance) for which knowing that another concept c' (*cow*, for instance) is present or not would help a lot in the detection of the original concept c. In our example, it is very unlikely to see both a space shuttle and a cow in the same shot. Therefore, we also investigated this *cross-concept* notion by replacing each fusion vectors f_i^c by f_i which is the concatenation of all f_i^c ($c \in [\![1, N_c]\!]$).

Runs 3 and 5 are submissions using *cross-concept* fusion while runs 4 and 6 make use of *mono-concept* fusion. However, we noticed during the preparation of our submissions that, for some concept, cross-concept would help and for some others, it would not. Therefore, based on this knowledge (obtained by cross-validation on the training set), we created runs 1 and 2 by selecting for each concept the best approach.

4.3 Switching $\mathcal{S}_1$ and $\mathcal{S}_2$

During the training process, datasets $\mathcal{S}_1$ and $\mathcal{S}_2$ can easily be exchanged, thus leading to the availability of two classifiers per concept. They both output a score measuring how likely a shot contains a concept. We investigated two ways of combining them by either averaging (runs 1 and 3 to 6 in Table 2) or taking their maximum value (run 2) after an additional step of *tanh* normalization [7].

5 Discussion

Table 2 shows the performance of a collection of selected descriptors on the development set. One interesting observation is that there is no *universally best* descriptor that would perform best for all concepts. Even though SIFT descriptors have best performance on average, they are outperformed by audio descriptors for some specific concepts: for instance, pitch for #3 *infants*, energy variance for #6 *flying airplanes* or voicing percentage for #7 *musical instruments*.

Was audio helpful? In order to answer this question, we can compare cross-concept runs 3 (audio and visual descriptors, AV) and 5 (visual descriptors only, V) or mono-concept runs 4 (AV) and 6 (V). In a nutshell, the AV cross-concept run outperforms its V counterpart for 9 concepts out of 20, and the AV mono-concept run outperforms its V counterpart for only 6 concepts out of 20. AV never outperforms V for 9 concepts out of 20: #2 *chair*, #4 *traffic intersection*, #5 *doorway*, #6 *airplane flying*, #8 *bus*, #10 *cityscape*, #15 *hand*, #18 *boat* and #20 *singing*. Note, however, that our best-performing run is an audiovisual one (though it is agreed that the difference is most probably not statistically significant).

Was cross-concept fusion helpful? Similarly, we can compare mono- and cross-concept runs 3 and 4 or 5 and 6. In a nutshell, the cross-concept AV run outperforms its mono-concept counterpart for 9 concepts out of 20, and the cross-concept V run outperforms its mono-concept counterpart for 7 concept out of 20. Cross-concept runs never outperforms mono-concept runs for 7 concepts out of 20: #2 *chair*, #5 *doorway*, #10 *cityscape*, #11 *bicycle*, #12 *telephone*, #13 *person eating*, #18 *boat*. Note that our best-performing run is a mono-concept run.

Did fusion work as expected? An in-depth analysis of the separate performance of each descriptor is planned in order to check the behavior of our fusion approaches.

References

[1] *OpenCV: Open Computer Vision Library*. Software available at `http://sourceforge.net/projects/opencvlibrary/`.

[2] Stéphane Ayache and Georges Quénot. Video corpus annotation using active learning. *Advances in Information Retrieval*, pages 187–198, 2008.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] Alain de Cheveigné and H Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917, April 2002.

[5] Elie El Khoury, Christine Senac, and Régine André-Obrecht. Speaker Diarization: Towards a more Robust and Portable System. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, Hawaii, USA*, pages 489–492. IEEE, 2007.

[6] T. Houtgast and J. M. Steeneken. A Review of the MTF Concept in Room Acoustics and its Use for Estimating Speech Intelligibility in Auditoria. *Journal of the Acoustical Society of America*, 77(3):1069–1077, 1985.

[7] Anil Jain, Karthik Nandakumar, and Arun Ross. Score normalization in multimodal biometric systems. *Pattern Recognition*, 38(12):2270 – 2285, 2005.

[8] Lionel Koenig, Corinne Mailhes, Régine André-Obrecht, and Serge Fabre. A continuous voicing parameter in the frequency domain. In *13th International Conference on Speech and Computer*, St Petersburg, Russia, 6 2009.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[10] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

Concept ID	001	002	003	004	005	006	007	008	009	010	
	011	012	013	014	015	016	017	018	019	020	

Cross-validation on the training set (average-precision @ 2000)

											Avg.
SIFT	.002	**.005**	.003	**.307**	**.013**	.003	**.004**	.000	**.016**	**.011**	.050
	.239	.001	**.238**	**.005**	**.030**	.002	**.090**	**.022**	**.013**	.001	
Hue	.000	**.006**	.001	**.276**	.002	**.006**	**.004**	.001	**.008**	**.004**	.038
	.200	.000	**.217**	.001	**.006**	.002	**.011**	**.003**	**.008**	.001	
Face	.000	**.004**	.001	**.005**	.001	.000	.001	.001	.000	.001	.006
	.005	.002	.002	.001	.004	.002	.001	.002	**.090**	.001	
MFCC	.001	.002	.001	**.279**	**.010**	.001	.002	.001	**.012**	**.005**	.039
(avg.)	**.242**	.002	**.200**	**.006**	.005	.001	.001	.002	**.011**	.002	
MFCC	.000	.001	.001	**.270**	.002	.000	**.005**	.000	.000	**.007**	.038
(var.)	**.226**	.002	**.226**	.000	.005	.002	.001	.003	**.012**	**.004**	
Pitch	.002	.001	.001	.002	.003	.003	**.004**	.001	.000	**.013**	.003
(avg.→avg.)	.001	.001	.003	.000	.003	.001	.002	**.009**	.003	.001	
$V_{\%}$	.001	.001	.001	.001	.003	.002	**.025**	.001	.001	.001	.003
(avg.→avg.)	**.004**	.000	.001	.002	.005	.000	.002	**.003**	.002	**.004**	
{ Pitch, $V_{\%}$ }	.001	.001	**.005**	**.005**	.001	**.005**	.002	.001	.000	.000	.003
	.001	.001	**.020**	.001	.004	.000	**.003**	.002	**.005**	.001	
Energy	.001	.001	.001	**.004**	.002	**.046**	.003	.000	.001	.001	.004
(var.→avg.)	.001	.002	.001	.000	.004	.000	.002	**.006**	.004	.001	
Vibrato	.000	.001	.001	.001	.001	.000	.002	.000	.000	**.004**	.001
(mean)	.001	.000	.001	.001	.004	.001	.001	.001	**.005**	.001	
ZCR	.001	.002	.001	.001	.002	**.006**	**.003**	.000	.000	**.010**	.003
(var.→avg.)	.001	.000	.000	.000	**.006**	.001	.003	**.015**	**.005**	.001	
Spect. stats	**.007**	.002	.002	.001	**.004**	.000	.002	.001	.000	.000	.002
(avg.→avg.)	.001	.001	.000	.002	.002	.001	.001	.001	.002	.003	

(avg.) means average on TRECVid shots. (var.) means variance on TRECVid shots.
(var.→avg.) means variance on audio segments, followed by average on TRECVid shots.

Best	.007	.006	.005	.307	.013	.046	.025	.001	.016	.013	.058
descriptor	.242	.002	.238	.006	.030	.002	.090	.022	.090	.004	

Actual performance on test set (average-precision @ 2000)

											Avg.
Run 1	.001	.001	.000	.043	.012	.000	.006	.000	.011	.005	.026
AV_BestAvg	.017	.013	.209	.009	.009	.008	.064	.030	.083	.000	
Run 2	.001	.002	.000	.044	.012	.000	.005	.000	.009	.005	.023
AV_BestMax	.015	.013	.207	.007	.008	.008	.058	.032	.043	.000	
Run 3	.001	.001	.001	.044	.008	.001	.006	.000	.011	.002	.022
AV_Poly	.017	.000	.208	.009	.009	.008	.069	.006	.049	.000	
Run 4	.000	.002	.000	.043	.012	.000	.004	.000	.008	.005	.027
AV_Mono	.024	.013	.209	.001	.026	.005	.064	.030	.083	.002	
Run 5	.001	.004	.000	.047	.015	.005	.004	.001	.006	.002	.025
V_Poly	.013	.000	.206	.000	.051	.001	.060	.023	.056	.002	
Run 6	.002	.004	.001	.046	.023	.003	.002	.000	.021	.007	.025
V_Mono	.017	.000	.206	.001	.030	.008	.063	.030	.043	.002	

Best run	.002	.004	.001	.047	.023	.005	.006	.001	.021	.007	.030
	.024	.013	.209	.009	.051	.008	.069	.032	.083	.002	

Table 2: Comparison of the efficiency of a few selected visual and audio descriptors on the training set (upper part). Actual results obtained by our six runs (lower part). .000 means descriptor (or run) performs best for the corresponding concept. **.000** (bold font) means that $\delta_{dc} = 1$, i.e. the corresponding descriptor was selected during the late fusion step.