Rapport de stage

13 Mai - 12 Juillet, 2013

Reconnaissance d'événements dans des bases vidéo

Event retrieval in large video databases

ROYER Amélie

Année académique 2012–2013
ENS Cachan - Antenne de Bretagne
Département Informatique et Télécommunications
$1^{\text{ère}}$ année

Encadrants :
JÉGOU Hervé
FURON Teddy
Équipe **Texmex**

Centre Rennes - Bretagne Atlantique

Résumé

La reconnaissance d'événements vidéo consiste à interroger une base de données par une vidéo requête d'un événement particulier, afin de retrouver toutes les vidéos correspondant à ce même événement.

Le problème principal est d'estimer la ressemblance entre deux vidéos. Mais il faut aussi garder une représentation compacte des vidéos et des temps de calcul raisonnables, puisque nous pouvons travailler sur des bases de données assez conséquentes.

Dans un premier temps, nous présenterons deux approches introduites dans [1], l'une considérant uniquement l'aspect visuel des vidéos, l'autre utilisant en plus l'ordre chronologique des images.

Nous étudierons ensuite une troisième méthode utilisant une représentation plus compacte des vidéos: les résultats obtenus sont similaires à ceux des approches précédentes, avec un gain en temps de comparaison et en espace mémoire; cependant la gestion des paramètres est plus complexe. Les implémentations des trois méthodes ont été réalisées en Matlab.

Mots-clés: Reconnaissance d'évènements, Descripteurs vidéos, Base de données vidéo.

Abstract

This internship's goal was to study the event retrieval task on large video databases: given a query video of a particular event, we want to retrieve every video of this same event in a database.

The main issue is to find an accurate way to compare videos. Moreover, we need to represent our videos in a compact form, and keep reasonable computation times: we work on large video databases, thus a need of efficiency.

First, we study two approaches introduced in [1]; the first one only uses the visual aspect of the videos, whereas the second one uses the chronological order of the frames in addition.

Next, we introduce a third event retrieval method, which uses a more compact description of our videos: it yields similar results to those of the previous approaches, with a benefit on memory usage and comparison time; however it requires fine tuning for the parameters. The three schemes were implemented using Matlab.

Keywords: Event retrieval, Video descriptors, Large video database.

Introduction

The event retrieval task aims at finding all the videos related to a very specific event in a large database. In this internship report, we only use video data (no audio, teletext, *etc.*).

Because we are dealing with videos, we have both visual and temporal informations. Visual data is efficiently transcripted with classic image descriptors often used in computer vision. And the chronological order of the frames is used to find an optimal alignment between videos: we assume that a good similarity measure is to estimate a visual similarity when the videos are optimally aligned.

In the first section, we introduce the dataset and the protocol used to evaluate the efficiency of our different methods. The next section describes two event retrieval algorithms, **MMV** and **CTE** that were introduced in [1]: we use these results as a reference for the event retrieval task. In the third section we present the main idea of our proposed model; it introduces a more compact description of the videos and can be used to obtain the same results than the above methods. We then explain the implementation in a more detailed way in Section 4. Finally the last section describes possible improvements, and it presents the final results obtained for each method.

1 Presentation of the Dataset

1.1 EVVE

The **EVVE** (**EV**ent **V**id**E**o) dataset contains 2975 videos retrieved from Youtube. Each video is associated with an event, for which it is labelled as positive or negative (see Appendix 6.1 for a detailed list of the events). Furthermore if a video is associated with an event E_i, it is implicitly negative for any different event E_j.

Definition 1.1 (Event)**.** *In that context, an event refers to a well localized (temporally and spatially) and very **specific** event (e.g. Obama's victory speech, Madonna's concert in Rome 2012, etc.).*

Finally, we distinguish the **query** videos (which are annotated as positive for their related event) from the **database** videos. The evaluation protocol is described in the next subsection.

Figure 1: The 13 events of the EVVE dataset

1.2 Event retrieval Protocol

The queries will be compared one by one to the whole database in order to evaluate the efficiency of the event retrieval scheme. For a query video q associated with an event E, the algorihm computes L_q, an **ordered** list of the database: it is sorted from the most similar to q, to the least similar one. We then use L_q to evaluate the **AP** (**A**verage **P**recision) of the algorithm for q. This is a scoring method often used in retrieval task; it takes account of the number of retrieved positive (*for* E) database videos and their ranking in the list (see Appendix 6.2 for details on computing the AP).

Finally, we compute the mean of the queries'APs (**mAP**) for each event. Our different event retrieval methods are evaluated on the basis of their mAP results (the higher, the better), their memory usage, and their computational time.

1.3 Describing the videos

We first process our videos to ensure they all meet a standard of 15 frames-per-second. Each frame of a video is then described as a single real vector called a **VLAD** descriptor [2]: this is an aggregation of **SIFT** local image descriptors [3] which are commonly used in computer vision. They are relatively invariant to point of view, luminosity and scale change. This robustness is preserved when they are merged into a **VLAD** descriptor.

These vectors are then compressed to the dimension $d = 512$. For this internship, the whole dataset was annotated, and the VLAD descriptors had already been processed.

From now on, we refer to a video V as a descriptor matrix $[\boldsymbol{v_1} \dots \boldsymbol{v_n}] \in \mathbb{R}^{d \times n}$, where $\boldsymbol{v_i}$ is the normalized VLAD descriptor of the i^{th} frame. The *similarity* between two frames is estimated by computing the dot product between their VLAD descriptors.

For the event retrieval task, our goal is to find a good way to compare videos; *i.e.* a score function, s, that estimates the similarity between two videos. It should be accurate and efficiently computed.

2 Comparing videos : initial approaches

In this section, we describe two methods for comparing videos that were introduced in [1] along with the EVVE dataset.

2.1 Visually

A first approach compares the videos based on their visual appearance, like we would do for two static images: we sum the descriptors of a video, which results in a single vector called **MMV** descriptor (**M**ean **M**ulti **V**lad). We use the dot product for comparison: formally, we define the score between two videos $\mathrm{v} = [\boldsymbol{v_1} \dots \boldsymbol{v_n}]$ and $\mathrm{w} = [\boldsymbol{w_1} \dots \boldsymbol{w_m}]$ as:

$$\boxed{s(\mathrm{v}, \mathrm{w}) \triangleq \langle \sum_{i=1}^{n} \boldsymbol{v_i}, \sum_{j=1}^{m} \boldsymbol{w_j} \rangle = \sum_{i,j} \langle \boldsymbol{v_i}, \boldsymbol{w_j} \rangle} \tag{1}$$

In short, we compare each frame with all those of the other video and then sum over the computed values. The chronological order of the frames is not taken into account using this metric, contrary to the following method.

2.2 Temporally

2.2.1 Frame-by-Frame Comparison

We now define a new score s(v, w), as the visual similarity between v and w when they are optimally aligned.

Formally we introduce $S_\delta(\mathrm{v}, \mathrm{w})$, the frame-by-frame comparison between two videos v and w of same length n, with v being shifted of $|\delta|$ frames ($\delta \in [\![-n+1; n-1]\!]$) to the right ($\delta > 0$) or to the left ($\delta < 0$):

$$\boxed{S_\delta(\mathrm{v}, \mathrm{w}) \triangleq \sum_{t=-\infty}^{+\infty} \langle \boldsymbol{v_{t-\delta}}, \boldsymbol{w_t} \rangle \text{ , with } \boldsymbol{v_i} = \boldsymbol{w_i} = 0 \text{ when } i \notin [\![0; n]\!].} \tag{2}$$

Finally the score between v and w is defined as $s(\mathrm{v}, \mathrm{w}) \triangleq \max_\delta S_\delta(\mathrm{v}, \mathrm{w})$.

2.2.2 Efficient Implementation

Call $\widetilde{\boldsymbol{v_i}}$ the i$^{\text{th}}$ line of the descriptor matrix of a video v. We rewrite $S_\delta(\mathrm{v}, \mathrm{w})$ as :

$$\begin{aligned}
S_\delta(\mathrm{v}, \mathrm{w}) &= \sum_{t=-\infty}^{+\infty} \langle \boldsymbol{v_{t-\delta}}, \boldsymbol{w_t} \rangle = \sum_{t=-\infty}^{+\infty} \sum_{i=1}^{d} v_{t-\delta,i} \cdot w_{t,i} = \sum_{i=1}^{d} \sum_{t=-\infty}^{+\infty} \widetilde{v}_{i,t-\delta} \cdot \widetilde{w}_{i,t} \\
&= \sum_{i=1}^{d} \sum_{t=-\infty}^{+\infty} \widetilde{v}_{i,t} \cdot \widetilde{w}_{i,t+\delta} \text{ , with the change of variable } t \leftarrow t - \delta \\
\implies S_\delta(\mathrm{v}, \mathrm{w}) &= \sum_{i=1}^{d} (\widetilde{\boldsymbol{v_i}} \circledast \widetilde{\boldsymbol{w_i}})(\delta) \text{ , where } \circledast \text{ is the cross-correlation operator.}
\end{aligned}$$

Call S the vector $S = [\dots S_\delta(\mathrm{v}, \mathrm{w}) \dots]$, $\delta \in [\![-n+1; n-1]\!]$. The previous lines implies that $S = \sum_{i=1}^{d} \widetilde{\boldsymbol{v_i}} \circledast \widetilde{\boldsymbol{w_i}}$. For computing efficiency we apply the **cross-correlation theorem**:

$$\boxed{s(\mathrm{v}, \mathrm{w}) = \max(S) = \max \left(\mathcal{F}^{-1} \left(\sum_{i=1}^{d} \mathcal{F}(\widetilde{\boldsymbol{v_i}})^* \odot \mathcal{F}(\widetilde{\boldsymbol{w_i}}) \right) \right)} \tag{3}$$

where $\odot$ is the component-wise product between vectors, and * the complex conjugate. This is efficiently computed using the fast Fourier transform (**FFT**) algorithm.

Remark 2.1 (Cross-Correlation Theorem)**.** The previous formula is the result of the **circular** cross-correlation theorem. The exact expression of (2) corresponds to the **linear** cross-correlation: it is obtained by 0-padding the lines of v and w to the size $2 \times n - 1$ before applying a circular cross-correlation.
In practice for efficiency reasons, we keep the circular expression of the cross correlation. In brief, it means that we compare our videos frame-by-frame with a side effect on the shifted video (see Figure 2).

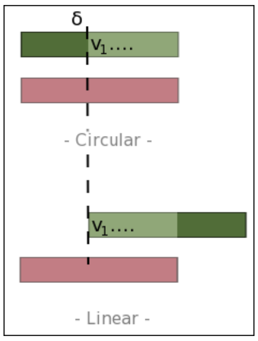

Figure 2: Cross-Correlation

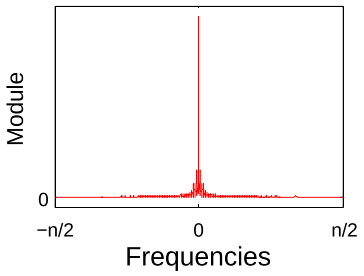

Figure 3: Fourier Transform on a line of a descriptor matrix

Moreover, the Fourier transforms computed here have high frequency components of low amplitude (see Figure 3). In practice we introduce a pruning coefficient, $\beta << 1$: we keep $\beta \times n$ low frequency vectors; the rest of the Fourier transform is approximated with zero. Besides, since our input signal is real, the Fourier transform is conjugate symmetric, thus we only need the information of the first $\frac{\beta \times n}{2}$ low frequency vectors.

We refer to this method as **CTE** (**C**irculant **T**emporal **E**ncoding) from now on.

2.2.3 Improving the precision of δ^*

Due to the existing correlation between consecutive frames in videos and because of the approximations we made to achieve computation effciency, the optimal alignment ($\delta^* = \operatorname{argmax}(S_\delta(\mathrm{v}, \mathrm{w}))$) is not always well localized: the score curves are noisy. For the same reason when plotting the self-similarity of a video, $s(\mathrm{v}, \mathrm{v})$, the curves are very "smooth" (see Figure 4). But the theoretical ideal result is a Dirac peak for $\delta = 0$.

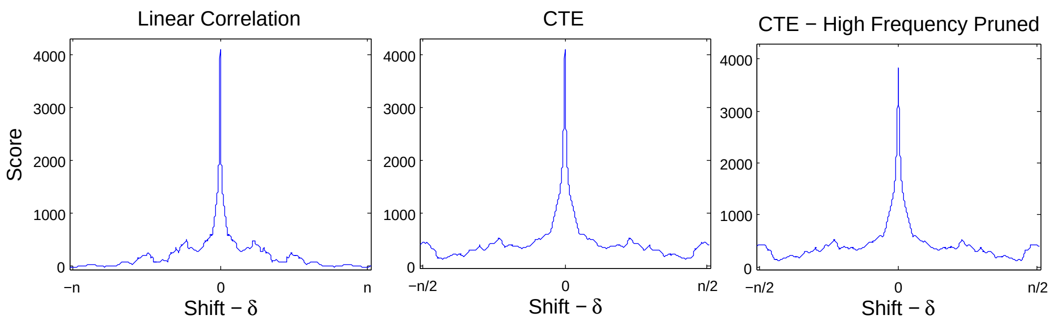

Figure 4: Self-similarity - Score curves for $s(\mathrm{v}, \mathrm{v})$

We can approach this Dirac behaviour by filtering the signal in the Fourier domain [4], We refer to this as the **regularization** step.

2.3 Combining the two approaches

A third method combines the visual and temporal aspects, this is done by adding both scores of **MMV** and **CTE**. Furthermore, it is efficiently computed: in fact a property of the Fourier transform is that $\forall \boldsymbol{x} = (x_1 \dots x_n) \in \mathbb{R}^n, (\mathcal{F}(\boldsymbol{x}))_0 = \sum x_i$. Hence, when computing the Fourier transforms of the lines of the descriptor matrix in (3) for CTE, the first column contains the sum of our descriptors, which we use to compute the MMV scores too.

2.4 Final results

Table 1 presents the mAP-results I obtained after implementing the previous methods using Matlab.

Method / Events	MMV	CTE	CTE (reg.) [1]	MMV+CTE (reg.)
1	0.5316	0.6243	**0.7447**	0.6607
2	0.3386	0.3355	0.3790	**0.3798**
3	0.0872	0.1071	**0.1265**	0.1136
4	**0.4554**	0.3062	0.3792	0.4467
5	0.2346	0.2946	**0.2969**	0.2736
6	0.2542	0.2468	0.2529	**0.2732**
7	0.1997	0.1515	0.1773	**0.2038**
8	0.1299	0.1138	0.1199	**0.1355**
9	0.1246	**0.2534**	0.2521	0.2124
10	**0.3669**	0.2201	0.2400	0.3424
11	0.2392	0.1398	0.1557	**0.2435**
12	0.7732	0.7525	0.7558	**0.7797**
13	0.6043	0.5794	0.6488	**0.6885**
avg-mAP	0.3338	0.3173	0.3484	**0.3657**

Table 1: Computed values of **mAP** for the 13 events, using MMV and CTE

We already have good results with **MMV** alone, which means by only comparing the videos from a global visual point of view. The **CTE** method yields significantly better **mAPs** when the event is really well localized in time (*e.g* Obama's victory speech, or the arrest of DSK): the videos for this type of events overlap temporally more often, so it is easier to align them. Finally we obtain the best results by combining both methods (**MMV+CTE** column), which shows that they are in fact complementary.

Moreover, these results show that the structure of the database itself has an influence on the mAPs. For example, the accuracy for event **12** is significantly better than for the other events. An hypothesis is that this event (***12** = Eruption of Strokkur Geyser in Iceland*) is difficult to mistake with another one. On the contrary, some events (*e.g.* Madonna's Concert, Shakira's Concert, Johnny Halliday's Concert) are very similar to one another: This may lead to more false-positive retrieved videos, hence lower mAPs.

[1]**CTE**(reg.) : (reg.) = with the regularization step

3 Comparing videos : an alternative

Beginning from this section, we study a new approach for comparing videos. We still use the chronological video information, but not as directly as in CTE. Moreover, we introduce more compact descriptors that are similar to those used in MMV.

3.1 A compact video representation

The basis idea is to re-use the **MMV** descriptor described in Section 2.1, but "break" the existing correlation between consecutive frame descriptors before summing: we first increase the dimension of the descriptors by 0-padding them; then we permute each descriptor differently, in order to separate the correlated coordinates that were on the same line.

Formally, let $\mathrm{v} = [\boldsymbol{v_1} \dots \boldsymbol{v_n}]$ and $\mathrm{w} = [\boldsymbol{w_1} \dots \boldsymbol{w_n}] \in \mathbb{R}^{d\times n}$ be two videos of same length n (*the shortest one being padded with* 0 *if needed*). We first define $L > d$, which is the new dimension of our descriptors; we 0-pad our matrices to this dimension (see Figure 5). Furthermore, we introduce a permutation matrix Q, and call ***ord*** its order (*i.e. ord* is the smallest strictly positive integer such that $Q^{ord} = Id_L$).

Then we shuffle each descriptor using different powers of Q. Even if they were modified, we still call the descriptors $[\boldsymbol{v_1} \dots \boldsymbol{v_n}]$ for notation convenience.

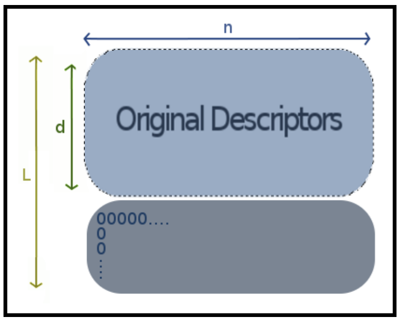

Figure 5: Descriptors are 0-padded

The final step is to sum over the rows of the descriptor matrix, so that our videos are now represented by a single vector each: $\overline{\mathbf{v}} \triangleq \sum_{i=1}^{n} Q^i \boldsymbol{v_i}$ and $\overline{\mathbf{w}} \triangleq \sum_{i=1}^{n} Q^i \boldsymbol{w_i} \in \mathbb{R}^L$. We refer to them as **compact descriptors**, provided that $L < n \times d$.

Remark 3.1 (Order)**.** In order to really apply a different permutation on each descriptor, we need *ord* $\geq n$.

3.2 Retrieving the temporal information

We expect that two descriptors permuted differently produce a very low dot product, because we are now working in high dimension with a significant proportion of zeros. On the contrary, two decriptors permuted the same way match "normally"; meaning they produce a high dot product if they are similar, and a low one if they are not.

We use this particularity in order to compare our videos frame-by-frame using only $\overline{\mathbf{v}}$ and $\overline{\mathbf{w}}$: in fact, as in MMV, computing $\langle \overline{\mathbf{v}}, \overline{\mathbf{w}} \rangle$ is the same as summing all the similarities between the (permuted) descriptors. And because of the previous remark, the similarities between unaligned frames should be very low.

Formally, we compute for all $\delta \in \mathbb{Z}$ the quantities:

$$\begin{aligned} S_\delta(\mathrm{v}, \mathrm{w}) &\triangleq \langle Q^\delta \overline{\mathbf{v}}, \overline{\mathbf{w}} \rangle \\ &= \langle \sum_{i=1}^{n} Q^{i+\delta} \boldsymbol{v_i}, \sum_{j=1}^{n} Q^j \boldsymbol{w_j} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \langle Q^{i+\delta} \boldsymbol{v_i}, Q^j \boldsymbol{w_j} \rangle \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \boldsymbol{v_i}, Q^{j-i-\delta} \boldsymbol{w_j} \rangle \end{aligned}$$

We then distinguish the dot products between descriptors permuted the same way (*i.e.* $j = i + \delta\ [ord]$), from the others:

$$\boxed{S_\delta(\mathrm{v},\mathrm{w}) = \sum_{\substack{j=i+\delta[ord]\\ i,j\in[\![1;n]\!]}} \langle \boldsymbol{v_i}, \boldsymbol{w_j}\rangle + \sum_{\substack{j\neq i+\delta[ord]\\ i,j\in[\![1;n]\!]}} \langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle}$$

The first term represents the comparison frame-by-frame between the video w and the video v being shifted of δ frames. Ideally, because of the permutation, the second term should be very small in comparison to the first summand. Furthermore, **CTE** can be seen as a subcase of this method, for which the second term would always be null.

Remark 3.2 (Order)**.** Because we have $2n-1$ possible alignments between the two videos:

- we need only $S_\delta(\mathrm{v},\mathrm{w})$ for $\delta \in [\![-n+1; n-1]\!]$.
- thus we need $ord \geq 2n-1$ to compute all needed values of $S_\delta(\mathrm{v},\mathrm{w})$

In the two next subsections, we study how we can use the values of $S_\delta(\mathrm{v},\mathrm{w})$ for computing a score, and we examine two possible candidates for the permutation Q.

3.3 Choice of the scoring method

Because we have an additional term that did not exist in CTE, we can wonder if taking the maximum of all possible alignments (*i.e.* the maximum of the $S_\delta(\mathrm{v},\mathrm{w})$ values) still yields the best possible score. Since the maximum is the infinite norm, we investigate the use of other p-norms in order to take account of this additional term. Besides, we only focus on finding matching parts in the videos: the negative alignment scores are irrelevant, so we give a null weight to the negative values. This results in a new scoring method:

$$s(\mathrm{v},\mathrm{w})_p = \sum_\delta S_\delta(\mathrm{v},\mathrm{w})^p \times \frac{sgn(S_\delta(\mathrm{v},\mathrm{w}))+1}{2}$$

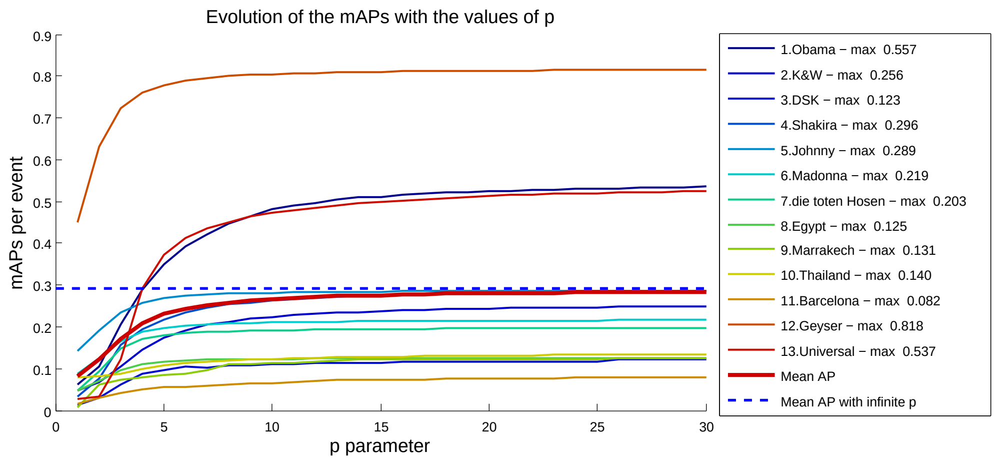

Figure 6: Evaluation of the mAPs for different p-norms.

The obtained curves strongly indicate that the mAPs values grow with p, attaining their maximum for $p=\infty$. Thus we once again define the score as $s(\mathrm{v},\mathrm{w}) = \max_{\delta\in[\![-n+1;n-1]\!]} S_\delta(\mathrm{v},\mathrm{w})$.

3.4 Choice of the permutation Q

3.4.1 An equivalent to CTE

A first idea was to totally suppress the second term in the expression of $S_\delta(\mathrm{v}, \mathrm{w})$, and only keep the alignment's term. This should yield the same results as **CTE**.

We therefore choose Q as a permutation that moves the coordinates of a vector by blocks of length d (see Figure 7). It ensures that when computing the dot products $\langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle$ non-null coordinates are always associated with zeros.

$$\begin{pmatrix} 0 & 0 & 0 & I_d \\ I_d & 0 & 0 & 0 \\ 0 & I_d & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Figure 7: Q - Matrix Form

So now, we simply have :

$$S_\delta(\mathrm{v}, \mathrm{w}) = \sum_{\substack{j=i+\delta[ord] \\ i,j\in[\![1;n]\!]}} \langle \boldsymbol{v_i}, \boldsymbol{w_j}\rangle \tag{4}$$

Finally we have to choose L, the new dimension of our descriptors, *i.e.* the length of our permutation. We have $ord = L/d$, thus because of Remark 3.2 we need $L \geqslant (2n-1)d$ to obtain all the alignment scores. Otherwise, we can choose to only follow Remark 3.1 and take for example $L = n \times d$, *i.e.* $ord = n$; because of the modulus on the indexes in (4), we obtain the same circularity as in CTE.

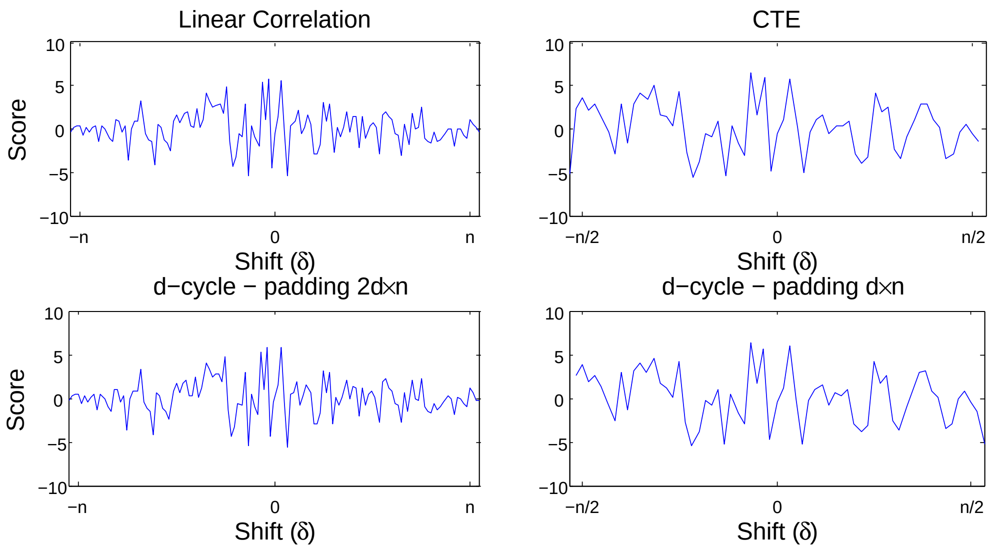

Figure 8: The linear correlation and CTE (*ie circular correlation*) curves are on the **first line**. The **second line** show the results obtained with the above described method.

However, this requires $L \geq d \times n$: Compared to CTE, the description of the videos takes the same space in memory and provides identical results. But it is still interesting to see we can achieve the same results.

3.4.2 Taking advantage of the high dimension : R1C

The idea is once again to reduce the $\sum\langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle$ term, but this time we want to keep a reasonable length for the padding.

First, we take advantage of the high dimension of the descriptors: we apply a random row permutation σ on our descriptor matrix, so that our non-null coordinates occupy the whole space provided by the padding length L. This permutation is only applied once.

Then, only after, we compute our compact descriptors using a simple cyclic permutation of step 1 as Q (More precisely, if $(\boldsymbol{e_i})_i$ is the canonical basis of $\mathbb{R}^L$, $\forall i, Q\boldsymbol{e_i} = \boldsymbol{e_{i+1[L]}}$).

In fact, the powers of such a permutation are easily and efficiently computed. Furthermore, when it comes to comparing two videos we can introduce the vector of scores $S = [\dots\langle Q^\delta \overline{\mathbf{v}}, \overline{\mathbf{w}}\rangle \dots]$ (such that $s(\mathrm{v}, \mathrm{w}) = \max(S)$). Because Q is a cyclic permutation, one can show that $S(\mathrm{v}, \mathrm{w}) = \overline{\mathbf{v}} \circledast_{circ.} \overline{\mathbf{w}}$. This is computed using Fourier transforms for efficiency, as in **CTE**.

We refer to this method as **R1C** (**R**andom permutation + **1**-**C**ycle).

In the next section, we further analyse the influence of the parameters on R1C. More particularly we show that the mAP results as well as memory usage and computation time mostly depends on the padding length L and the videos' length n.

4 Using R1C

4.1 Variance in R1C

Given two videos v and w, we established that the score $S_\delta(\mathrm{v}, \mathrm{w})$ contains a term corresponding to the frame-by-frame comparison in alignment δ. We want to estimate how close $S_\delta(\mathrm{v}, \mathrm{w})$ is to this term.

Therefore, in this subsection we determine the theoretical mean and variance of $S_\delta(\mathrm{v}, \mathrm{w})$ as a function of the padding length L, the videos' length n and some other parameters of lesser influence.

4.1.1 Theoretical expression of the variance

For reminder, we have, for a temporal alignment δ :

$$\begin{aligned} S_\delta(\mathrm{v}, \mathrm{w}) = \langle Q^\delta \overline{\mathbf{v}}, \overline{\mathbf{w}}\rangle &= \sum_{i=1}^{n}\sum_{j=1}^{n} \langle Q^{i+\delta}\boldsymbol{v_i}, Q^j \boldsymbol{w_j}\rangle \\ &= \sum_{\substack{j=i+\delta[ord] \\ i,j\in[\![1;n]\!]}} \langle \boldsymbol{v_i}, \boldsymbol{w_j}\rangle + \sum_{\substack{j\neq i+\delta[ord] \\ i,j\in[\![1;n]\!]}} \langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle \end{aligned}$$

Call $f_\delta(\mathrm{v}, \mathrm{w})$ the number of frames of v and w that match in the alignment δ : we can also extract from the first sum $f_\delta(\mathrm{v}, \mathrm{w})$ dot products equaling 1 (*i.e.* the dot products between perfectly matching frames). The other dot products, which should be much lower, are integrated to the second sum. That means we can write:

$$S_\delta(\mathrm{v}, \mathrm{w}) = \underbrace{f_\delta(\mathrm{v}, \mathrm{w})}_{\text{Matching Frames}} + \underbrace{\sum \langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle}_{\text{“Noise” term}}$$

Our goal is to study the dot products in the “noise term” as real random variables, in order to apply the central limit theorem to the whole sum.

Proposition 4.1 (Gaussian Distribution on the high dimensional unit sphere)**.** *Given $\boldsymbol{x}$ and $\boldsymbol{y}$ two **uniformly** distributed random **unit** vectors of dimension D, then for great values of D we have $\langle x, y\rangle \sim \mathcal{N}(0, \frac{1}{D})$.*

Lemma 4.2 (Projective Central Limit Theorem)**.** *Let $\boldsymbol{x} = [x_1 \dots x_D] \in \mathbb{R}^D$ be an uniformly distributed random vector of norm $\|x\| = \sqrt{D}$, then we have $x_1 \xrightarrow[D\to+\infty]{\mathcal{L}} \mathcal{N}(0,1)$.*

Proof. Lemma 4.2 - Let us study the probability density function of $x_1 \in \mathbb{R}$ (denoted by f). We use the fact that f is the derivative of F, the cumulative distribution function of x_1.

$$\begin{cases} F(t) & = \mathbb{P}(-\infty \leq x_1 \leq t) = 1 - \mathbb{P}(x_1 \geq t) \\ f(t) & = -\frac{d}{dt}\left(\mathbb{P}(x_1 \geq t)\right) \end{cases}$$

Furthermore, x is uniformly distributed on the sphere centered in 0 and of radius $\sqrt{D}$. Thus $\mathbb{P}(x_1 \geq t)$ can be computed as the ratio of the area of an hyperspherical cap of height $\sqrt{D} - t$ (*i.e.* the part of the sphere above the hyperplan $x_1 = t$), to the total area of the sphere (see Figure 9).

The expression of this ratio can be found in [5], and it results in :

$$\boxed{\mathbb{P}(x_1 \geq t) = \frac{1}{2} I_{\sin^2 \phi}\left(\frac{D-1}{2}, \frac{1}{2}\right)}$$

where I is the regularized incomplete beta function, and ϕ is the colatitude angle that defines the hyperspherical cap: it is the angle between the x_1-axis vector $(1, 0, \dots, 0)$, and any vector going from the center of the sphere to the basis of the cap.

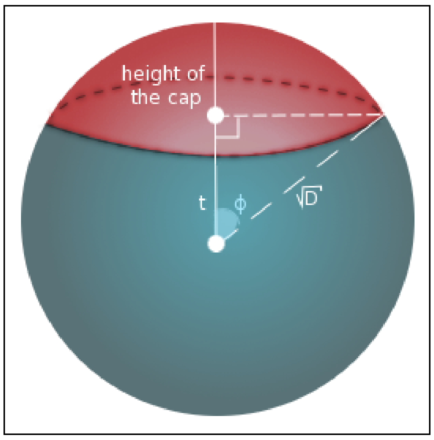

Figure 9: Schema for $D = 3$

The expression of ϕ is straightforward in dimension $D = 3$ and can be generalized to greater dimensions. We have $\sin^2 \phi = 1 - \cos^2 \phi = 1 - \frac{t^2}{D}$.

Moreover, $x \to I_x(a,b)$ is the cumulative distribution function of a beta distribution; Therefore, its derivative is the associated probability density function of the beta distribution. After calculus, with B being the Beta function, we have:

$$f(t) = -\frac{d}{dt}\left(\mathbb{P}(x_1 \geq t)\right) = \qquad \frac{1}{\sqrt{D}} \times \frac{1}{B\left(\frac{D-1}{2}, \frac{1}{2}\right)} \left(1 - \frac{t^2}{D}\right)^{\frac{D-3}{2}}$$

Because $D \to +\infty$, we can use the Stirling's approximation for the Beta function, and the usual limit $\left(1 - \frac{t^2}{D}\right)^{\frac{D-3}{2}} \xrightarrow[D\to+\infty]{} e^{-\frac{t^2}{2}}$. This results in:

$$\boxed{f(t) \xrightarrow[D\to+\infty]{} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}} \text{, the probability density function of } \mathcal{N}(0,1).}$$

Finally, by applying Scheffé's lemma, we deduce that $x_1 \xrightarrow[D\to+\infty]{\mathcal{L}} \mathcal{N}(0,1)$.

□

Lemma 4.3 (Unit Sphere and Rotation)**.** *Given* $\boldsymbol{x}$ *an unit vector and* $u_n = (1, 0, \dots, 0) \in \mathbb{R}^n$ *there exists a rotation* $\boldsymbol{R}$ *such as* $x = R \cdot u_n$*.*

Proof. Lemma 4.3 - Let $(\mathcal{H}_n)$ be the property to be proved.

- n = 2 Let $x = (x_1, x_2) \in \mathbb{R}^2$ an unit vector. Let R be the plane rotation of angle $\arccos(x_1)$.

  ie $R = \begin{pmatrix} x_1 & -x_2 \\ x_2 & x_1 \end{pmatrix}$. Then $Rx = u_2$: $(\mathcal{H}_2)$ **is true**.

- $n \in \mathbb{N}$ Let $n \geq 3$ and assume $(\mathcal{H}_{n-1})$ is true.

  Let $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$. Let us call $v = (x_{n-1}, x_n)$. Then, because $(\mathcal{H}_2)$ is true, there exists R_1 such that $R_1 v = \|v\| \cdot u_2$. We then extend R_1 to the nth dimension: $R_1 \leftarrow \begin{pmatrix} I_{n-2} & 0 \\ 0 & R_1 \end{pmatrix}$. Thus $R_1 x = (x_1, \dots x_{n-2}, \|v\|, 0)$.

  Let us introduce $x' = (x_1, \dots x_{n-2}, \|v\|)$.

  Then $\|x'\|^2 = \sum\limits_{i=1}^{n-2} |x_i|^2 + \|v\|^2 = \sum\limits_{i=1}^{n} |x_i|^2 = 1$. We can therefore use $(\mathcal{H}_{n-1})$ on x' : it exists a rotation R_2 which sends x' on u_{n-1}. Same as above, we extend R_2 to the nth dimension : $R_2 \leftarrow \begin{pmatrix} R_2 & 0 \\ 0 & 1 \end{pmatrix}$.

  And finally with $R = R_2 R_1$, we have $Rx = u_n$, thus $(\mathcal{H}_n)$ **is true for all** n.

□

Proof. **Proposition 4.1 -** First, we apply a rotation matrix $\mathbf{R}$ on $\boldsymbol{x}$ and $\boldsymbol{y}$ so that $Rx = x' = (1, 0 \dots, 0)$ and $Ry = y'$. Since a rotation leaves the dot product unchanged, we have $y'_1 = \langle x', y' \rangle = \langle x, y \rangle$.

By applying Lemma 4.2 on $\sqrt{D} \times y'$, which is an uniformly distributed vector on the sphere of radius $\sqrt{D}$, we have $\sqrt{D}\langle x, y \rangle = \sqrt{D} \times y'_1 \xrightarrow[D \to +\infty]{\mathcal{L}} \mathcal{N}(0, 1)$.

Hence when D is great enough, we can make the approximation $\sqrt{D}\langle x, y \rangle \sim \mathcal{N}(0, 1)$, thus $\langle x, y \rangle \sim \mathcal{N}\left(0, \frac{1}{D}\right)$.

□

We can now use this result on the score for a given alignment δ. For reminder, we have :

$$S_\delta(\mathrm{v}, \mathrm{w}) = f_\delta(\mathrm{v}, \mathrm{w}) + \sum \langle \boldsymbol{v_i}, Q^{j-i-\delta} \boldsymbol{w_j} \rangle$$

At this point we know that every dot product $\langle \boldsymbol{v_i}, Q^{j-i-\delta} \boldsymbol{w_j} \rangle$ follows a normal distribution $\mathcal{N}(0, \frac{1}{L})$. Finally, we assume that our dot products are independent in order to apply the **central limit theorem** to the sum.

Particular Case In the case $\delta = 0$, if the videos V and W share common frames, it appears that for some index $\langle Q^{i+\delta} \boldsymbol{v_i}, Q^j \boldsymbol{w_j} \rangle$ and $\langle Q^{k+\delta} \boldsymbol{v_k}, Q^l \boldsymbol{w_l} \rangle$ are the same variable. More simply, we have re-appearing, and thus non independent dot products when we are computing the score $S_0(\mathrm{v}, \mathrm{w})$ with v and w having some frames in common.

Therefore, in that case, we first extract the re-appearing dot products from the sum; formally, we introduce the set of indexes $I = \{k \in [\![1; n]\!], \boldsymbol{v_k} = \boldsymbol{w_k}\} = \{\sigma(1) \dots \sigma(q)\}$ where q is the cardinal of I. By doing so, we have:

$$S_0(\mathrm{v},\mathrm{w}) = f_0(\mathrm{v},\mathrm{w}) + 2\sum_{i=1}^{q}\sum_{j=i}^{q}\langle \boldsymbol{v_{\sigma(i)}}, Q^{j-i}\boldsymbol{w_{\sigma(j)}}\rangle + \sum \langle \boldsymbol{v_{\sigma(i)}}, Q^{j-i}\boldsymbol{w_{\sigma(j)}}\rangle$$

And all the appearing dot products are now independent and identically distributed variables following a normal distribution $\mathcal{N}(0, \frac{1}{L})$.

4.1.2 Experimental validation

We now compare the experimental results with the theory in different alignment situations.

Case 1: V and W do not match in alignment δ; In that case we have $f_\delta(\mathrm{v},\mathrm{w}) = 0$. Thus,

$$S_\delta(\mathrm{v},\mathrm{w}) = \sum_{i,j}\langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle$$

The dot products are n^2 independent identically distributed variables following a $\mathcal{N}(0, \frac{1}{L})$ distribution. By applying the central limit theorem we have :

$$\sqrt{n^2}\frac{\sum\langle \boldsymbol{v_i}, Q^{j-i-\delta}\boldsymbol{w_j}\rangle}{\frac{1}{\sqrt{L}}n^2} = \frac{S_\delta(\mathrm{v},\mathrm{w})}{\frac{1}{\sqrt{L}}\sqrt{n^2}} \sim \mathcal{N}(0,1)$$

We conclude that $S_\delta(\mathrm{v},\mathrm{w}) \sim \mathcal{N}\left(0, \frac{n^2}{L}\right)$

Case 2 : V and W match in alignment $\delta = 0$; In that case: $f_0(\mathrm{v},\mathrm{w}) \neq 0$ and $I \neq \varnothing$.

$$S_0(\mathrm{v},\mathrm{w}) = f_0(\mathrm{v},\mathrm{w}) + 2\sum_{i=1}^{q}\sum_{j=i}^{q}\langle \boldsymbol{v_i}, Q^{j-i}\boldsymbol{w_j}\rangle + \sum\langle \boldsymbol{v_i}, Q^{j-i}\boldsymbol{w_j}\rangle$$

By applying the same reasoning as above, we have that the first and second sum follow respectively a $\mathcal{N}\left(0, 4\frac{q(q-1)}{2L}\right)$ and $\mathcal{N}\left(0, \frac{n^2-q(q-1)-f_0(\mathrm{v},\mathrm{w})}{L}\right)$ distribution. And as the sum of two indepedent gaussian variables, $S_0(\mathrm{v},\mathrm{w}) \sim \mathcal{N}\left(f_0(\mathrm{v},\mathrm{w}), \frac{n^2+q(q-1)-f_0(\mathrm{v},\mathrm{w})}{L}\right)$.

Results After doing several measurements, I computed an estimation of the variance of $S_0(v,w)$ in the case where

- **v** and **w** are two randomly generated videos; *i.e.* Case 1, $\delta = 0$, $S_0(\mathrm{v},\mathrm{w}) \sim \mathcal{N}\left(0, \frac{n^2}{L}\right)$.
- **v** = **w**; *i.e.* Case 2 with $q = f_0(\mathrm{v},\mathrm{w}) = n$, $S_0(\mathrm{v},\mathrm{w}) \sim \mathcal{N}\left(n, \frac{2(n^2-n)}{L}\right)$.

As shown by Figures 10 and 11, the experimental results match the expected variance.

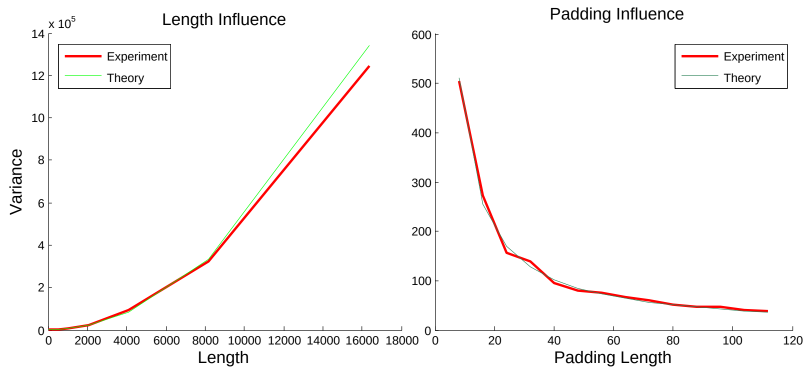

Figure 10: Case 1 - Variance as function of n (left) and L (right)

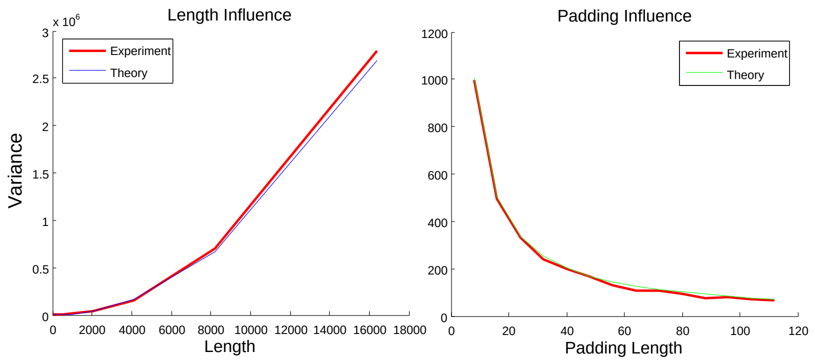

Figure 11: Case 2 - Variance as function of n (right) and L (left)

Remark 4.4. For too high values of L, the variance tends to be higher than expected. This probably comes from the fact that we made the assumption of uniformly distributed vectors of dimension L; But this depends on the permutations applied on the descriptors, and therefore on d and n:

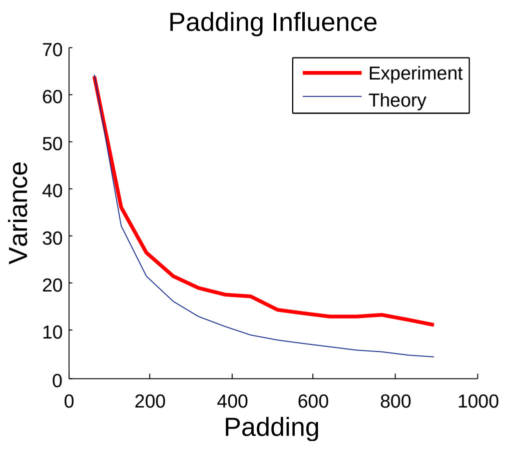

Figure 12: Example ($d = 8$ and $n = 64$): problems begin to appear for $L \sim 150$

4.2 Implementation of R1C

Pre-Treatment of the descriptors Using Matlab, the pre-computing part (*i.e.* computing the sums $\sum Q^i \boldsymbol{v_i}$) is very slow (several minutes for a video). To address this problem, this part was written in **C**, and interfaced with Matlab using a **mex** file. Moreover I used an implementation similar to that of sparse matrices, so that we do not have to manipulate the full $L \times n$ matrix when computing the permutations:

- Input : A descriptor matrix, V (size $d \times n$) and a random row permutation σ (size $\mathbf{L}$).
- Output : The corresponding compact descriptor, $\overline{\mathbf{v}} = \sum Q^i(\sigma \cdot \boldsymbol{v_i})$, of size $\mathbf{L}$.
- Implementation:
  1. First, we allocate an integer array RowsIndexes, of size $d \times n$. The number rowsIndexes[i;j] represents the line at which the number V[i;j] would be located in the fully 0-padded descriptor matrix: we only work on the line indexes.
  2. Then, both rows permutations (the random one: σ, and the cyclic ones: Q^i) are applied by changing the values in RowsIndexes :
     $\text{RowsIndexes}[i;j] \leftarrow \underbrace{\sigma(\text{RowsIndexes}[i;j])}_{\text{Random part}} + \underbrace{j}_{\text{Cyclic part}}$
  3. Finally we need to sum every non null coordinate in the ouput vector compact. It is easily done by browsing V and updating Compact as follow : $\overline{\mathbf{v}}$[rowsIndexes[i,j];j] += V[i;j]

Comparing the videos As stated in Section 3.4.2, we compute the score vector $S = [\dots S_\delta(\mathrm{V}, \mathrm{W}) \dots]$ with a single circular cross-correlation between the compact descriptors.

This is efficiently computed using the FFT algorithm. Furthermore, in order to reduce the gaussian noise existing in S , we apply a low-pass filter on the Fourier transforms by suppressing a part of their high frequency components.

Summary

- 1 - For every video V (database and query) we precompute $\overline{\mathbf{v}} = \sum_{i=1}^{n} \boldsymbol{v_i}$. For efficiency reasons, we directly store their Fourier transforms.
- 3 - At query time, given a query V, for every database video W we compute : $S = \mathcal{F}^{-1}\left(\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{w}})\right)$
- 4 - Finally we compute $s(\mathrm{V}, \mathrm{W}) = \max(S)$
- 5 - We use the $s(\mathrm{V}, \mathrm{W})$ values to compute a mAP for the query V. Steps 3-5 are repeated for each query.

4.3 Spatial and Temporal Analysis

The next step is to compare the temporal and spatial use of CTE and R1C. Both methods have a step of pruning high frequencies therefore we introduce the parameter β, which represents the proportion of low frequency components kept. We denote with a “ ’ ” the values that have been modified by the pruning.

R1C

1. Pre-computing : Using the method described above, we only browse the arrays V and RowsIndexes, thus a temporal complexity $O(n \times d)$. At the end of this step, we have for each video a vector of size L (possibly pruned).

2. Comparing : It only requires a cross-correlation, *i.e.* in the Fourier domain, a component-wise multiplication between two vectors and an inverse FFT $(O(L' + L' \log(L'))$.

CTE

1. Pre-computing : We map the videos to the Fourier domain $(O(n \log(n)))$. At the end of this step we have for each video a matrix of size $n' \times d$.

2. Comparing : It requires $d \times n'$ component wise multiplications (and as much additions), and a final inverse Fourier transform after re-adding the 0-values corresponding to high frequencies $(O(d \times n' \log(n') + n \log(n)))$.

	Memory	Time(PreComputing)	Time(Comparison)
R1C	L'	$n \times d$	$L' \log(L')$
CTE	$d \times n'$	$n \times \log(n)$	$d \times n' \log(n')$

Table 2: Complexity Summary (Predominant Term) per video

The main difference between the two methods is that the spatial and time complexity in CTE depends on the videos' length, whereas in R1C (except for the pre-computing) it only depends on the fixed parameters L and d. Thus, the efficiency depends a lot of the repartition of the videos' length in the database. In practice, the more long videos there is in the database, the more R1C is efficient compared to CTE.

Observations on EVVE - 2975 videos On the EVVE Dataset, R1C is better when it comes to memory usage and comparison time. But it requires a longer pre-computing time. Furthermore during pre-computing R1C needs to allocate space for the array ROWSINDEXES (even if it is discarded afterwards).

	Database size	Time(PreComputing)	Time(Comparison)
R1C	974M	34s/event	5.9s/query
CTE	3.282G	17s/event	6.8s/query

Table 3: Average values observed on EVVE for the final simulations (mAPs in Table 4)

In the next section, we study four steps that can be applied on R1C (such as a regularization step) to improve its obtained mAPs. And we conclude this next section by a table recapitulating the best score achieved by each method.

5 Improvement on R1C

5.1 Regularization

Similarly to CTE, if we introduce the vector $S = [\dots S_\delta(\mathrm{v}, \mathrm{w}) \dots] = [\dots \langle Q^\delta \overline{\mathbf{v}}, \overline{\mathbf{w}} \rangle \dots]$, we previously noticed that $S = \overline{\mathbf{v}} \circledast_{circ.} \overline{\mathbf{w}} = \mathcal{F}^{-1}\left(\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{w}})\right)$. We therefore can apply the same regularization step as in Section 3.

The idea is to have a Dirac-like peak for alignment $\delta = 0$ when comparing a video with itself, by adding a filtering stage. We add a Fourier Filter A in the score formula ($S_A = \mathcal{F}^{-1}\left(A \odot \mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{w}})\right)$), and we want to achieve :

$$S_A = [1, 0 \dots 0] \implies A \odot \mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{v}}) = [1 \dots 1]$$
$$\implies A = \frac{1}{\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{v}})} \text{ (Component-wise division)}$$

And we add a regularization parameter λ, for preventing the denominator to be null.

Finally, we extend the filter's expression for the score S between two videos $\mathrm{v} \neq \mathrm{w}$. We choose to use a symmetric expression for the filter: $A_{sym} = \frac{1}{\sqrt{\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{v}})} \times \sqrt{\mathcal{F}(\overline{\mathbf{w}})^* \odot \mathcal{F}(\overline{\mathbf{w}})} + \lambda}$.
And we finally have :

$$\boxed{S = \frac{\mathcal{F}^{-1}\left(\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{w}})\right)}{\sqrt{\mathcal{F}(\overline{\mathbf{v}})^* \odot \mathcal{F}(\overline{\mathbf{v}})} \times \sqrt{\mathcal{F}(\overline{\mathbf{w}})^* \odot \mathcal{F}(\overline{\mathbf{w}})} + \lambda}}$$

In the two next sections we study the influence of the scoring metric, and we introduce a parameter for noise reduction in order to further improve the results.

5.2 Using the p-norms

In Section 3.3 we introduced another scoring method that used the different p-Norms of the vector S instead of simply taking the maximum. After the regularization step, this have a positive impact: there exists a norm that achieves better results than the maximum (see Figure 13).

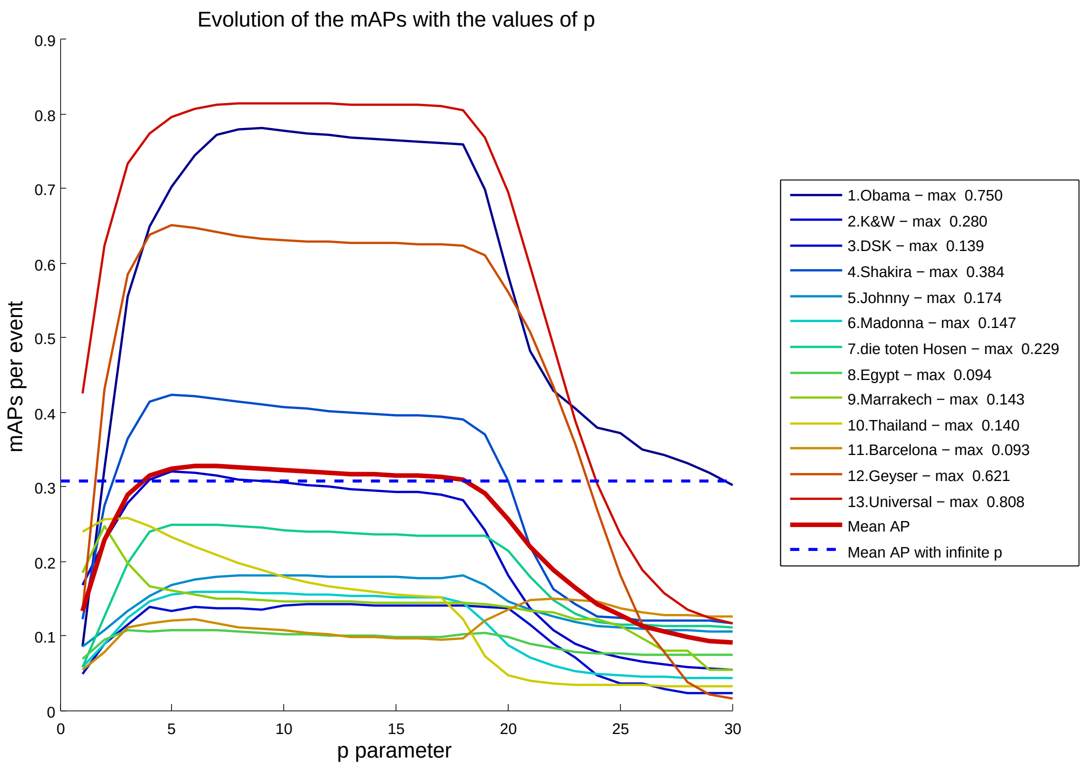

Figure 13: Evolution of the mAPs for different p-Norms with R1C(reg.)

5.3 Choosing a non-random permutation

Using the R1C method, the results fluctuate because of the random permutation σ generated at the beginning. Therefore I tried to find a deterministic permutation in order to replace this undesirable random aspect.

Given a descriptor $\boldsymbol{v_i} \in \mathbb{R}^d$, that we 0-pad to a dimension L, we search a permutation σ of size L such that the d non-null coordinates in $\sigma \cdot \boldsymbol{v_i}$ occupy the whole dimensionality. The problem does not have an exact solution, therefore I constructed the permutations based on a simple model. We introduce two arrays :

- Steps of size d that represents the different steps between the non-null coordinates after being permuted. In practice we choose a fixed initial step $\boldsymbol{st}$ and Steps $= [\dots, k \times st, \dots]$, $k \in [\![1; d]\!]$.
- Index $= [\![1, d]\!]$ (initially). It represents the new order of the non null coordinates after the permutation.

Furthermore, in order to have a less "ordered" structure, we apply very simple permutations both on Steps and Index (For example : "For each odd index i, switch array(i) and array(end-i)").

Finally, the permutation σ is constructed as follow :

$$\boxed{\sigma = [\dots \text{INDEX}(j), \underbrace{0, 0, \dots 0}_{\text{Steps(j) } 0}, \text{INDEX}(j+1) \dots]}$$

for j going from 1 to d.

With this scheme, I could find some permutations that achieved good results. However this is not an "exact" method, and we do not know if this permutation would achieve good results on a totally different database. We refer to this method as **D1C** (**D**etermined permutation + **1**-**C**ycle).

5.4 Final Results

The Table 4 contains the final best results obtained with the R1C with regularization and D1C with regularization methods, that were presented above.

	CTE	R1C [2]	D1C [2]	MMV+CTE	MMV+D1C
	norm ∞	norm 5	norm 3	norm ∞	norm ∞
1	**0.7447**	0.7032	0.7218	**0.6607**	0.6307
2	**0.3790**	0.3195	0.3322	**0.3798**	0.3559
3	0.1265	0.1389	**0.1518**	0.1136	**0.1402**
4	0.3792	0.4216	**0.4249**	0.4467	**0.4716**
5	**0.2969**	0.1761	0.2203	**0.2736**	0.2452
6	**0.2529**	0.1584	0.2127	**0.2732**	0.2451
7	0.1773	**0.2496**	0.2451	0.2038	**0.2453**
8	0.1199	0.1076	**0.1421**	0.1355	**0.1457**
9	**0.2521**	0.1559	0.2180	0.2124	**0.2512**
10	0.2400	0.2196	**0.3053**	0.3424	**0.3535**
11	0.1557	0.1218	**0.1680**	**0.2435**	0.2266
12	0.7558	**0.7834**	0.7513	0.7797	**0.8005**
13	0.6488	0.6474	**0.6863**	0.6885	**0.7123**
avg-mAP	0.3484	0.3233	**0.3523**	0.3657	**0.3711**

Table 4: Comparison of the final results

In the last column, we also computed the results for **MMV+D1C**; it significantly improves the results. But the MMV descriptors have to be computed separately from the compact descriptors (even if in practice the additional memory and time needed for computing the MMV scores are negligible).

On average, we observe that D1C yields a little better results than those in Table 1, and it uses less memory for a lower average comparing time. However, D1C has a lot more parameters to control.

On one hand, finding optimal parameters is more difficult and it requires fine tuning. On the other hand, it is interesting to have a fully parameterizable method since the results always depend at least on the database. Thus the flexibility of D1C could be useful in order to adapt the method to different datasets.

Conclusion

To conclude on the different methods presented above: the good results of **MMV** attests the robustness of the SIFT/VLAD descriptors system. **CTE** shows that the temporal information can be useful when the events are well localized in time. Finally **R1C/D1C** is a more vectorial approach that achieves a gain in memory usage and comparison time with similar mAPs results. But it depends on more parameters.

To go further, it could be interesting to use different evaluation methods for this task. Computing of the mAP is a relative measure: in practice with retrieval tasks, the user is only interesed in the 10/20 first results (*e.g.* a Google search). But here, we computed the AP on the whole database. Furthermore, during measurement we mostly focused on maximizing the average mAP. But we noticed that some parameters could positively influence an event, when lowering the scores for another one. It could be interesting to create a classification of the events using these informations, in order to optimize the algorithms.

Finally, we could use more of the inherent information contained in the video. In the Appendix 6.3 we describe how we can use the shot boundary information on this task. But we could also use audio data for example in order to compute a more accurate similarity measure between video.

[2] Order of magnitude for R1C and D1C : $n_max \sim 65000$; $L \sim 50n_{max}$; $\beta = \frac{1}{32}$

References

[1] J. Revaud, M. Douze, C. Schmid, and H. Jégou, “Event retrieval in large video collections with circulant temporal encoding,” Mar. 2013.

[2] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” pp. 3304–3311, 2010.

[3] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[4] J. Henriques, R. Caseiro, P. Martins, and J. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7575 of *Lecture Notes in Computer Science*, pp. 702–715, Springer Berlin Heidelberg, 2012.

[5] S. Li, “Concise formulas for the area and volume of a hyperspherical cap,” *Asian Journal of Mathematics and Statistics*, vol. 4, no. 1, pp. 66–70, 2011.

6 Appendices

6.1 List of the events in the EVVE Dataset

Events	Queries	Data	Positive Data
1. Obama's Victory Speech	14	75	29
2. Kate and William's Wedding	44	188	88
3. Arrest of DSK	9	79	19
4. Shakira's Concert	19	174	39
5. Johnny Halliday's Concert	87	401	174
6. Madonna's Concert	51	171	104
7. Die toten Hosen's Concert	32	207	64
8. Egyptian riots	36	99	72
9. Marrakech Bomb Attack	4	110	10
10. Thailand Flood	73	157	148
11. Barcelona Riots	13	149	27
12. Strokkur Geyser	215	498	431
13. Jurassic Park Ride	23	57	47
Total	620	2375	1252

Table 5: Details of the EVVE dataset

6.2 Computing an average precision

Given **D** a set of data labelled as positive or negative in relation to a query q ($\mathrm{D} = \mathrm{D}^+ \cap \mathrm{D}^-$), the event retrieval algorithm computes L_q, a sorted list of **D** in decreasing similarity order. Intuitively, the algorithm would be optimal if all the videos of D^+ appear at the beginning of L_q. The average precision takes account of the repartition of the videos of D^+ in L_q.

Formally, $\forall v \in D^+$ we call $k(v)$ the index of v in L_q. Then, we compute $Pre_q(v)$, the precision of the sub-list $L_{q_{|v}} = L_q[1 \dots k(v)]$, which is defined as the ratio $Pre_q(v) = \frac{card(D^+ \cap L_{q_{|v}})}{card(L_{q_{|v}})}$. It is the number of positive videos retrieved so far, divided by the total number of retrieved videos so far.

Finally, the expression of the average precision for the query **q** is given by:

$$\boxed{\mathrm{AP}(q) = \frac{\sum\limits_{v \in D^+} \mathrm{Pre}_q(v)}{\mathrm{card}(D^+)}}$$

6.3 Potential Improvement : Using the Shot Boundaries

6.3.1 Motivation

The basis idea was to sum the descriptors belonging to the same shot: a whole shot is now represented by a single vector. The resulting descriptors are less correlated, and they still provide a good visual description of our videos. Furthermore, this method is robust against resampling.

6.3.2 Detecting the Shot Boundaries

Let $\mathrm{V} = [\boldsymbol{v_1} \dots \boldsymbol{v_n}] \in \mathbb{R}^{d \times n}$ be a video of length n. For each time $t \in [\![2, \dots, n-1]\!]$, we first compute the value $s_\mathrm{V}(t) = \langle \boldsymbol{v_{t-1}}, \boldsymbol{v_{t+1}} \rangle$: it is the similarity between the frames $\boldsymbol{v_{t-1}}$ and $\boldsymbol{v_{t+1}}$.

A first approach is to define the shot boundaries as every time t for which $s_V(t)$ is inferior to a threshold α. The inconvenients of this method are its strong depedency on the parameter α, and its bad precision when the shot boundary is not neat (*e.g* fade transitions).

To address these problems,we browse the frames by groups of 10 (assuming that there would never be more than one shot change in that lapse of time). This prevents detecting duplicate shot boundaries when the transition is not neat.

Besides, we compute the variance of the $s_V(t)$ values in theses groups. If that variance is greater than a certain threshold, we assume there is a shot change in that intervall; the time t for which $s_V(t)$ is minimal in the group is marked as a shot boundary.
By using the variance, the idea is to look for group of frames that contains a brutal variation in the similarities, it doesn't depend on a threshold value for the similarity. See Figure 14 for a comparison of both methods.

`http://www.youtube.com/watch?v=-5vWztiU7aI` (First minute)

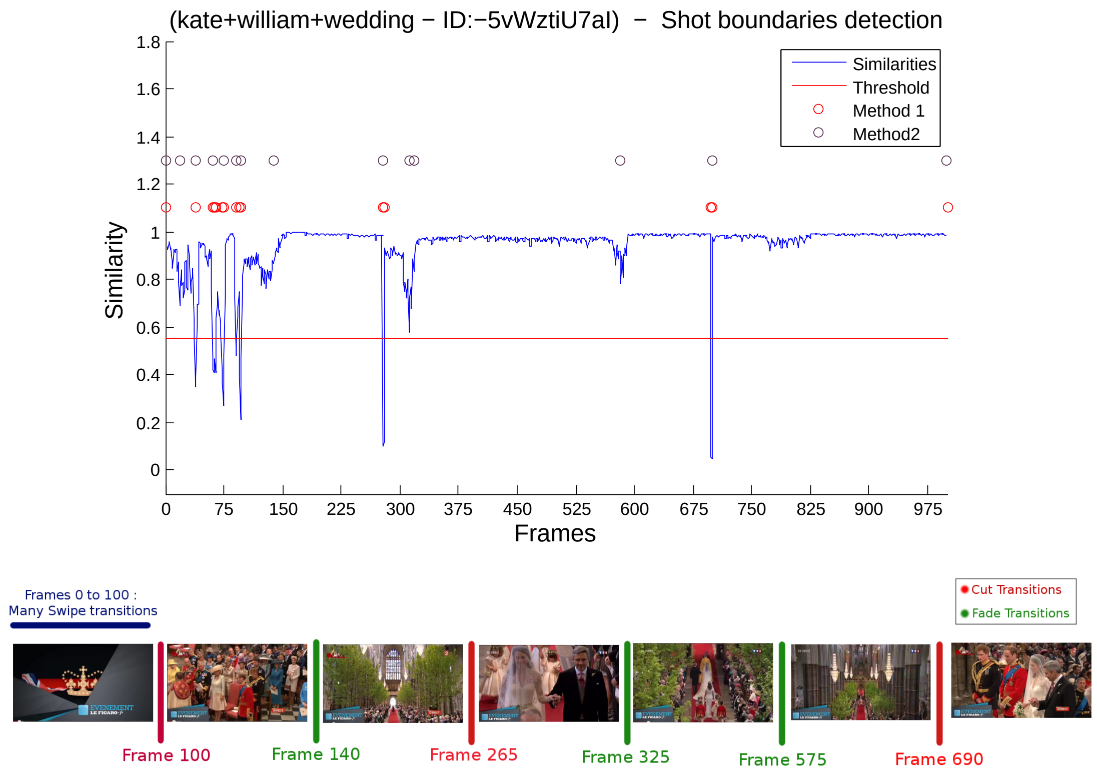

Figure 14: Comparison of both methods - Overall, the cut transitions are well detected by both methods. Finally the fade transitions (around 300th and 600th frame) are only detected by the second approach.

6.3.3 Application on the EVVE dataset

I didn't have much time to try this method, but the first results were not convincing enough (final mAP around 0.275), and the precomputing time is longer. However, because the treated videos are shorter, we can afford to use shorter padding lengths. Comparing two videos is therefore much faster (around 1s per query in the final simulations).