Passage à l'échelle de la fouille vidéo basée sur la détection de copies

Sébastien Poullot [*#1], Michel Crucianu [*2], Olivier Buisson [#3]

[*]*Vertigo - Wisdom & CEDRIC, CNAM, 292 rue St Martin, 75141 Paris cedex 03, France*
[2]michel.crucianu(a)cnam.fr

[#]*Institut National de l'Audiovisuel, 94366 Bry-sur-Marne, France*
[1]spoullot(a)ina.fr, [3]obuisson(a)ina.fr

***Abstract*— La structuration automatique des grandes bases de données multimédia à partir du contenu non textuel a de nombreuses applications potentielles et permet de pallier à l'absence d'annotations riches et fiables. Dans le cas particulier des grandes bases vidéo, des liens très utiles entre séquences vidéo sont mis en évidence par la détection de copies par le contenu (DCPC). En simplifiant, on considère que deux vidéos sont en relation de « copie » si l'une est une version transformée de l'autre ou les deux ont un ancêtre commun. Avec une description du contenu et une mesure de similarité adaptées, cette relation peut s'exprimer comme une similarité entre vidéos. La fouille d'une base vidéo par DCPC peut alors être vue comme une *auto-jointure par similarité*. Nous abordons ici le problème difficile du passage à l'échelle de ce type d'opération. Nous introduisons une structure d'index adaptée, qui exploite les propriétés d'une nouvelle description, plus compacte, du signal vidéo (au niveau des trames clés). Les différentes étapes du processus de fouille basé sur cette structure sont ensuite décrites. La qualité de détection obtenue avec la méthode proposée est mesurée sur deux vérités terrain. Le passage à l'échelle est ensuite évalué sur des bases allant jusqu'à 10 000 heures de vidéo. Enfin, des résultats de l'application de cette méthode à des contenus issus d'un site Web2.0 sont illustrés.**

I. Introduction

Le récent essor des moyens de production et de diffusion de contenus multimédia entraîne la création et l'échange d'une très grande quantité de programmes. La structuration automatique de ces grands volumes de données par la fouille du contenu non textuel devient indispensable et constitue un défi majeur pour les prochaines années.

Pour des raisons essentiellement culturelles, le format court domine actuellement dans la nature des programmes audiovisuels produits (séries, reportages courts, clips musicaux, etc.) et dans la façon de les réaliser (plans courts et rapides, montages différents des mêmes séquences, etc.). Ces contenus font souvent l'objet de rediffusions ou sont réutilisés, en entier ou par fragments, dans d'autres programmes. Aussi, le grand public n'est plus un simple consommateur mais assume de plus en plus un rôle de producteur de contenus, bien souvent par le ré-agencement de contenus existants, parfois agrémentés de façon originale. En conséquence, les bases de programmes d'institutions comme l'INA ou de sites Web2.0 de partage de vidéos contiennent un nombre important de versions transformées des mêmes contenus de départ.

Fig. 1. Copie (à gauche) et contenu d'origine (à droite)

L'identification automatique, au sein d'une très grande base multimédia, des liens entre les différentes versions d'un même contenu est basée sur la détection de copies par le contenu (DCPC). On considère que deux vidéos sont en relation de « copie » si l'une est une version transformée de l'autre ou les deux ont un ancêtre commun. Bien entendu, toute transformation n'est pas acceptable, des contraintes sur la nature et l'amplitude des transformations sont généralement admises. La figure 1 illustre un tel exemple de copie.

La figure 2 décrit une situation typique dans la diffusion de programmes audiovisuels : des

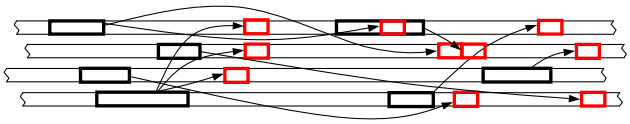

Fig. 2. Liens de contenu entre séquences vidéo issues de plusieurs flux audiovisuels

séquences vidéo issues d'une des chaînes sont transformées et retransmises ultérieurement par la même ou par d'autres chaînes. Ce sont ces liens de contenu qu'il faut retrouver dans l'archive qui stocke ces flux audiovisuels diffusés.

dans un contexte plus général, l'identification de liens de contenu dans une grande base vidéo peut avoir de multiples applications : la segmentation des programmes (par exemple par la détection de débuts et des fins d'émissions), l'aide à l'annotation ou la propagation d'annotations (entre les différentes versions d'une même séquence), le « nettoyage » des bases (par élimination des copies à l'identique), l'exploration de la base à travers les liens établis, etc. L'annotation manuelle des contenus, même lorsqu'elle est présente et fiable, ne peut pas répondre à bon nombre de ces besoins.

La fouille de bases vidéo à partir du contenu visuel a déjà été explorée sous différents angles, voir par exemple [1], [2], surtout pour trouver des liens entre séquences issues de bulletins d'information par l'identification de « presque duplicata » ([3], [4], [5], [2], [6]), notion plus contrainte — les transformations du contenu sont de faible amplitude — que celle de copie considérée plus haut. Mais les méthodes existantes sont confrontées aux problèmes du passage à l'échelle et de la généricité. En effet, pour que le traitement soit réalisé dans des délais raisonnables, les volumes traités doivent être assez restreints (pas plus que quelques centaines d'heures de vidéos), le contenu assez spécifique (journaux télévisés, publicité) et la notion de « presque duplicata » assez étroite (transformations peu perceptibles du contenu, découpage identique des séquences).

Notre objectif est de mettre au point une méthode *généraliste* qui *passe à l'échelle*, permettant ainsi de trouver des liens de contenu dans de très grandes bases, de toute provenance et sans contraintes sur la nature du contenu. Un exemple applicatif permet

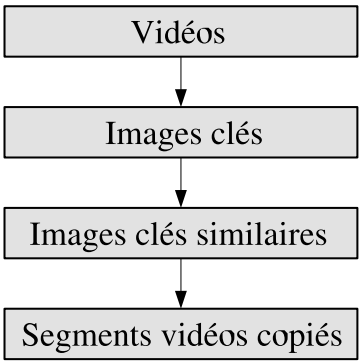

Fig. 3. Processus de fouille vidéo basé sur la détection de copies par le contenu

d'illustrer les exigences auxquelles nous souhaitons répondre, concernant le volume et la nature des contenus, ainsi que les transformations appliquées.

Considérons une recherche sur un site Web2.0 de partage vidéo avec le mot clé « Madonna », qui retourne 166 000 résultats. Les vidéos obtenues représentent environ 11 000 heures, volume nettement supérieur à ceux traités par les méthodes citées. De plus, les contenus retrouvés sont très hétérogènes : durée allant de quelques secondes à quelques dizaines de minutes, contenant des plans fixes (interviews) ou rapides (clips), tournés dans un studio ou à l'extérieur, etc. Il est nécessaire de pouvoir traiter de la même façon ces contenus, qu'ils soient très hétérogènes ou homogènes et spécifiques. Enfin, les transformations subies par les séquences vidéo sont de nature diverse et d'amplitude très variable. Les transformations appartiennent à plusieurs familles : photométriques (changement de contraste, gamma, passage en noir et blanc, ajout de bruit, flou), géométriques (recadrage, changement d'échelle, étirement, etc.), temporelles (accélération, ralentissement, ajout et suppression d'images) et de post-production (extraits, compilations, ajout de logos, de sous-titres, incrustations, bandes noires, réencodage, etc.). Dans l'ensemble, la copie intégrale (qui reprend intégralement la vidéo source) et la copie exacte (qui n'applique aucune transformation à la séquence reprise de la vidéo source) sont rares. Il est donc nécessaire de pouvoir lier entre eux des extraits vidéo plutôt que des vidéos entières, les liens de contenu doivent être trouvés à un niveau fin de segmentation (quelques secondes).

Avec une description du contenu et une mesure

de similarité adaptées, la relation de copie peut s'exprimer comme une similarité entre vidéos. La fouille d'une base vidéo par DCPC peut alors être vue comme une *auto-jointure par similarité*. Pour la réaliser, il est nécessaire de déterminer d'abord une représentation appropriée pour les séquences vidéo et une mesure de similarité associée. Par ailleurs, nous devons pouvoir identifier des copies d'assez courte durée (des extraits). Les « images clés » des vidéos seront ainsi considérées comme l'élément de base. Ces images clés sont définies par l'activité visuelle globale et apparaissent à un rythme moyen de une par seconde, typiquement lors de changements de plan, mouvements brusques, etc. La succession des images clés peut être vue comme un résumé précis d'une vidéo. L'identification de vidéos similaires sera basée sur la recherche de similarité entre images clés. Nous utiliserons donc une représentation des images clés et une mesure de similarité associée. Une fois trouvés les couples d'images clés similaires, les segments vidéo à mettre en correspondance peuvent être reconstitués. La figure 3 résume le processus global.

Dans la section II la fouille de données basée sur la détection de copies est exprimée sous forme de requêtes, en partant des vidéos pour arriver aux images clés. Pour traiter cette requête, une méthode généraliste pouvant passer à l'échelle est présentée. Un nouveau descripteur d'image clé, appelé Glocal, est introduit dans la section III ainsi qu'une mesure de similarité associée. Suit, dans la section IV, la présentation d'une structure d'index adaptée à ce descripteur et à la tâche de fouille. La section V est consacrée à la reconstruction des segments vidéos copies à partir des images clés similaires détectées. La qualité de détection obtenue avec la méthode proposée est mesurée sur deux vérités terrain dans la section VI-A. Le passage à l'échelle est ensuite évalué sur des bases allant jusqu'à 10 000 heures de vidéo dans la section VI-B. Enfin des résultats de l'application de cette méthode à des contenus issus d'un site Web2.0 sont illustrés dans la section VI-C.

II. Approche adoptée pour la fouille vidéo par DCPC

Rappelons qu'une vidéo est une succession d'images arrivant à rythme fixe, en général 25 fps (images par seconde). Les éléments accessibles sont ces images. Pour rendre le problème généralisable à toute base de données vidéo, il faut se limiter à un minimum d'attributs supplémentaires. Les autres attributs d'une image utilisés dans la suite sont

- l'identifiant unique de la vidéo à laquelle appartient l'image : ID,
- l'étiquette temporelle (*timecode*) de l'image par rapport au début de la vidéo : TC.

L'étiquette temporelle correspond au numéro de l'image par rapport au début de la vidéo. Pour la détection de segments vidéos copiés nous utilisons exclusivement les images clés. L'ensemble des contraintes et attributs étant décrits, le problème abordé peut être formulé de façon plus précise.

A. Formulation du problème

La fouille de données utilisant le détection de copies vidéo par le contenu peut être vue comme une auto-jointure par similarité sur une relation de vidéos, dans laquelle chaque vidéo a un identifiant (attribut `ID`) et une description du contenu visuel (représenté par l'attribut `VCONTENT`)

```
VIDEOS(ID, VCONTENT)
```

La recherche des vidéos similaires peut être exprimée sous la forme de la requête `R1` suivante :

```
Select V1.ID, V2.ID
From VIDEOS V1, VIDEOS V2
Where SIM(V1.VCONTENT, V2.VCONTENT) > Theta_v
```

où θ_v (`Theta_v`) est le seuil de similarité entre vidéos.

Généralement les applications imposent de trouver des copies de séquences vidéo de quelques secondes. Cela exige de comparer des séquences de quelques images clés au sein des vidéos de départ. Il est possible d'employer des descriptions de très courtes séquences vidéo (voir par exemple [7], [2]), avec une segmentation préalable ou une fenêtre glissante. Malheureusement, les traitements de post-production rencontrés ont souvent pour conséquence la disparition ou la duplication de quelques images dans les séquences, introduisant ainsi des distorsions supplémentaires significatives dans la comparaison directe des courtes séquences. C'est pourquoi nous resterons au niveau des images clés individuelles pour aborder le problème. Ces images clés sont donc préalablement extraites des vidéos disponibles et caractérisées par l'ID de la vidéo, l'étiquette

temporelle (relative au début de la vidéo, attribut Tc) et une description du contenu visuel (attribut FCONTENT)

```
KEYFRAMES(ID, Tc, FCONTENT)
```

Il faut trouver les images clés dont les descriptions du contenu visuel (FCONTENT) sont similaires, mais issues de vidéos différentes, et qui se suivent temporellement. Afin d'éviter la recherche d'une vidéo sur elle-même (qui générerait un grand nombre de résultats sans grand intérêt) et les doublons (la similarité étant symétrique), l'identifiant de la première image doit être inférieur à celui de la deuxième. Les ID résultants sont triés, ce qui permet de grouper les sous-ensembles d'images clés pour chaque couple d'ID possible et donc de trouver l'ensemble des images clés similaires entre deux vidéos. Enfin, un tri sur les étiquettes temporelles des images clés permet d'obtenir une partie de la continuité temporelle des détections des images clés. La recherche s'exprime par la requête R2 suivante :

```
Select KF1.ID, KF1.Tc, KF2.ID, KF2.Tc
From KEYFRAME KF1, KEYFRAME KF2
Where SIM(KF1.FCONTENT, KF2.FCONTENT) > Theta
And  KF1.ID > KF2.ID
Order by KF1.ID, KF2.ID, KF1.Tc, KF2.Tc Asc
```

où θ (Theta) est le seuil de similarité entre images clés.

A partir des couples d'images clés similaires ainsi identifiés, les séquences vidéo similaires sont reconstituées suivant l'algorithme présenté dans la section V. Ce processus entraîne aussi l'élimination de certaines fausses détections (images clés de similarité forte par la description mais visuellement éloignées). De plus, la redondance visuelle au sein d'une vidéo (images clés successives ressemblantes, plans très proches, etc.) peut lier une image clé d'une vidéo à plusieurs d'une autre vidéo ; les contraintes temporelle permettent d'identifier les bons segments copiés.

B. Complexité de la jointure

Le défi majeur de cette approche est le passage à l'échelle : sans index, la complexité de l'opération de jointure est quadratique dans le nombre d'images clés. De plus, dans les propositions existantes de méthodes de détection de copie, pour que la détection soit fiable chaque image clé est représentée par un nombre élevé (et variable) de signatures locales. Même avec une seule signature par image, sachant qu'il y a en moyenne 3000 images clés par heure de vidéo et que les bases à traiter contiennent plus de 10 000 heures, l'approche séquentielle directe induirait un nombre de calculs de similarité de $\frac{(10000\times3000)^2}{2} = 4,5 \times 10^{14}$. Il est donc nécessaire d'utiliser une structure d'index pour les descriptions des images clés, permettant d'accélérer la jointure par similarité. Cette structure doit limiter le nombre de calculs de similarité nécessaires en ne comparant que des images clés dont la similarité est supérieure à un seuil.

La section suivante présente dans le détail le descripteur qui sera utilisé pour représenter une image clé et une structure d'index adaptée au descripteur ainsi qu'aux transformations subies par une image lors de la génération d'une copie. Nous employons le terme de « descripteur » pour désigner une méthode particulière de description du contenu et le terme de « signature » pour désigner le vecteur qui décrit un point d'intérêt extrait d'une image (ou une image entière si la description est globale).

III. Description des images clés

Pour résoudre le problème de passage à l'échelle nous proposons un nouveau descripteur d'image clé qui génère des signatures compactes. Les signatures calculées sur l'ensemble des images clés d'une base vidéo sont ensuite indexées selon leur similarité. L'index employé est adapté aux caractéristiques des signatures. Le descripteur mis au point associe un vecteur (de dimension fixe) à chaque image clé, mais est obtenu à partir des signatures d'un ensemble de points d'intérêt détectés dans l'image.

En général, afin de décrire une image clé, il est possible d'utiliser plusieurs signatures locales ou une signatures globale. De nombreuses propositions existent pour ces deux grandes familles, voir par exemple [8], [9], [10], [11], [12], [13]. Nos précédents travaux [14], ainsi que d'autres travaux récents ([15], [16], [17]) ont montré que dans le cas de la DCPC les descripteurs locaux étaient les plus pertinents. Le principe est de calculer des descriptions spatio-temporelles dans chaque image clé autour de « points d'intêret » (coins de Harris, points de symétrie, etc.). L'utilisation de plu-

sieurs signatures locales de l'image permet une plus grande robustesse aux transformations que l'utilisation d'une signature globale décrivant l'image clé entière. En contrepartie, les signatures locales sont plus nombreuses et le processus de détection de copie demande une étape supplémentaire de décision (basée en général sur un vote) pour savoir si une image peut être considérée comme une copie de l'autre.

Pour combiner les avantages des deux types de description, la robustesse des descripteurs locaux aux transformations (pour la qualité de détection) et le volume de données réduit associé aux descripteurs globaux (pour le passage à l'échelle), nous avons mis au point un nouveau descripteur (appelé « Glocal ») dont le principe est d'intégrer les signatures locales d'une image clé dans une seule signatures très compacte, de taille fixe.

Dans [14], au maximum 20 points d'intérêt sont extraits de chaque image clé. Chaque point d'intérêt est décrit par un vecteur appartenant à $[0, 255]^{20}$. La méthode d'indexation employée était basée sur une approximation spatiale. Pour le descripteur Glocal ce principe est réutilisé, mais avec une approximation plus grossière. L'espace de description est découpé en deux de façon itérative, dimension après dimension, ce qui génère des « cellules » de taille égale (dont le volume est divisé par deux à chaque nouveau découpage). On peut ordonner et numéroter ces cellules (dans notre cas, selon une *space-filling curve*). A chaque signature locale est attribué le numéro de la cellule à laquelle elle appartient. L'ensemble des numéros des signatures d'une image clé constitue l'information portée par une signature Glocal. La signatures Glocal d'une image est un vecteur binaire dans lequel les bits correspondant aux positions (numéros) des signatures locales sont à 1. Ce vecteur sera en général creux, car il contient au maximum 20 bits à 1.

La figure 4 présente un exemple, dans un espace de description simplifié, à deux dimensions. Les points indiquent les signatures locales, les lignes horizontales et verticales indiquent le découpage de l'espace. Les cellules formées sont numérotées en bas à droite. Les cellules grisées sont celles qui contiennent des signatures, leur position (numéro) est donc marquée par un bit à 1 dans la signature Glocal. Sur cette figure l'espace a été découpé 2 fois

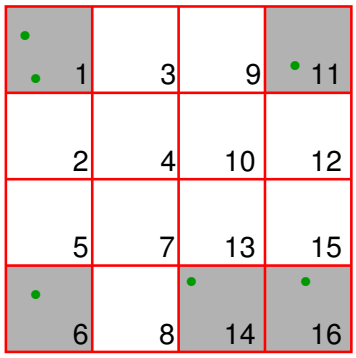

Fig. 4. Synthèse d'une signatures Glocal à partir des signatures locales dans un espace de description 2D, découpé à profondeur 4

selon chaque dimension, la profondeur d'indexation est donc $p = 4$ et la longueur du vecteur binaire résultant $l = 2^p = 16$. La signature Glocal pour cet exemple est **1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1**.

L'information portée par une signature Glocal correspondant aux approximations des positions des signatures locales, on peut craindre une forte perte d'information si trop de signatures locales parmi les 20 (au maximum) d'une image se retrouvent dans les mêmes cellules. Notre analysé montre qu'à une profondeur de découpage (d'indexation) $p = 8$ (soit $2^8 = 256$ cellules) les signatures des points d'intérêt d'une image se trouvent en moyenne dans 17,4 cellules distinctes, une signature Glocal contient donc en moyenne 17,4 bits à 1. Cette valeur est proche du nombre moyen de points d'intérêt extraits de chaque image clé (19,4), la perte d'information est de ce point de vue limitée. Un découpage plus fin augmente peu le nombre de bits à 1 dans les signatures. La figure 5 montre le rapport entre le nombre de bits à 1 dans la signature d'une image clé et le nombre de signatures locales utilisées pour la synthèse de celle-ci. Pour les profondeurs d'indexation employées ici, $p > 7$, le descripteur Glocal produit des signatures binaires creuses : le nombre de bits à 1 est beaucoup plus faible que le nombre de bits à 0.

Les signatures Glocal sont compactes par rapport aux ensembles de signatures locales dont elles sont issues. Une signature locale étaient codées sur 20 octets, une image clé demandait environ 400 octets de description dans [14]. Notons au passage que cette description était déjà compacte par rapport à des solutions basées sur PCA-SIFT [9], GLOH [18]

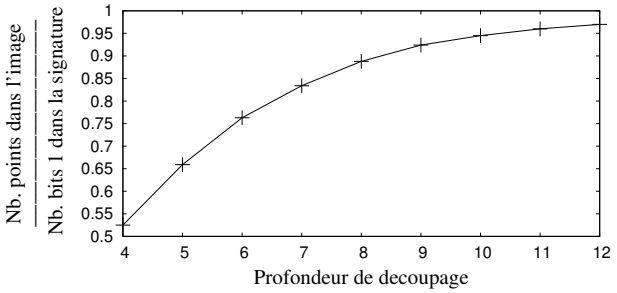

Fig. 5. Ratio entre le nombre de bits à 1 dans un descripteur Glocal et le nombre de points d'intérêt dans une image clé

ou SURF [19]). Pour Glocal on peut se contenter de conserver uniquement la position des bits à 1 (en nombre de 20 au maximum), ce qui prend au plus 40 octets (20×2) pour $p \leq 16$. Le plupart des signatures globales sont également plus gourmandes (> 100 octets).

Un vecteur Glocal étant binaire et creux, il est naturel d'utiliser le coefficient de Dice pour mesurer la similarité entre 2 tels vecteurs :

$$S_{\mathrm{Dice}}(\mathbf{g}_1, \mathbf{g}_2) = \frac{2\,|\mathcal{G}_1 \cap \mathcal{G}_2|}{|\mathcal{G}_1| + |\mathcal{G}_2|} \qquad (1)$$

où $\mathcal{G}_i$ est l'ensemble des positions à 1 dans la signature $\mathbf{g}_i$ et $|\cdot|$ la cardinalité d'un ensemble. Ce choix n'est toutefois pas restrictif. En effet, le coefficient de Dice est directement lié au coefficient de Jaccard $S_{\mathrm{Jaccard}}(\mathbf{g}_1, \mathbf{g}_2) = \frac{|\mathcal{G}_1 \cap \mathcal{G}_2|}{|\mathcal{G}_1| + |\mathcal{G}_2| - |\mathcal{G}_1 \cap \mathcal{G}_2|}$ par la relation $S_{\mathrm{Jaccard}} = \frac{S_{\mathrm{Dice}}}{2 - S_{\mathrm{Dice}}}$. De plus, comme le nombre de positions à 1 est presque la même pour toutes les signatures, $|\mathcal{G}_i| \approx n$, le coefficient de Dice est presque équivalent au coefficient de chevauchement, $S_{\mathrm{overlap}}(\mathbf{g}_1, \mathbf{g}_2) = \frac{|\mathcal{G}_1 \cap \mathcal{G}_2|}{\min\{|\mathcal{G}_1|, |\mathcal{G}_2|\}} \approx S_{\mathrm{Dice}}(\mathbf{g}_1, \mathbf{g}_2)$, et peut aussi être lié à la distance de Hamming par $d_{\mathrm{Hamming}} \approx 2n(1 - S_{\mathrm{Dice}})$.

Nous avons mené une étude de l'impact des transformations dues au processus de copie sur la similarité entre les signatures des images clés. La similarité entre signatures issus de vidéos originales et de copies a été observée. Les copies ont été produites artificiellement (100 heures de vidéo comprenant les modifications suivantes : changement de contraste, de gamma et d'échelle, ainsi que ajout de bruit). L'étude montre qu'il faut considérer des valeurs supérieures à 0,40 pour le coefficient de Dice afin de retrouver 90% des images clés copiées (pour une profondeur de découpage $p = 8$).

IV. Auto-jointure par similarité entre images clés

Il convient à présent de choisir une méthode d'indexation pour des vecteurs Glocal afin de limiter le nombre de similarités à calculer lors de l'opération de jointure.

A. Méthodes d'indexation candidates

Différentes méthodes d'indexation proposées pour la recherche par similarité ou pour la jointure par similarité ont été passées en revue pour évaluer leur compatibilité avec les caractéristiques du descripteur Glocal et avec les volumes de données à traiter. La quasi-totalité des méthodes mentionnées sont approximatives.

Dans des méthodes comme OMEDRANK [20] ou PvS [21], les signatures vectorielles sont projetées sur des droites de direction aléatoire. Chaque droite est partitionnée et chaque signature est référencée dans le *bucket* correspondant à chaque droite. Le principe du hachage sensible à la similarité (*locality-sensitive hashing*, LSH, [22]) est assez proche. Ces méthodes semblent peu adaptées aux vecteurs à composantes binaires, comme les signatures Glocal.

Miller, dans [23], propose un arbre 256-aire pour stocker de longs vecteurs binaires (8192 bits). Ceux-ci étant découpés en octets, le chemin de la racine à une feuille (1024 noeuds) représente une signature. Pour comparer une nouvelle signature à celles présentes dans la base, on parcourt l'arbre en calculant un « taux d'erreur » (issu de la distance de Hamming) à chaque niveau, entre le début du vecteur requête et les noeuds dans lesquels on se trouve. Lorsque ce taux d'erreur est trop fort, on arrête la recherche dans la branche. Le résultat est l'ensemble des feuilles atteintes. Mais avec des signatures très creuses (comme celles issues de Glocal) le taux d'erreur augmenterait faiblement lors du parcours de l'arbre. Comme les détections se font avec un coefficient de Dice de 0.45, pour chaque signature il faudrait calculer en moyenne près de la moitié de la distance à la moitié des éléments de la base (la moitié des noeuds de l'arbre), soit $\frac{1}{4}$ des calculs de la méthode séquentielle directe.

Oostveen et Kalker [24] suggèrent d'utiliser des listes inversées sur des vecteurs binaires. Une signature de 960 bits est découpé en 30 vecteurs de 32 bits. La signature est alors insérée dans 30

positions différentes dans une liste inversée ayant pour entrée des entiers codés sur 32 bits. Une signature est comparée à un autre si elles ont en commun au moins un des 30 sous-vecteurs, une hypothèse forte étant donné le caractère arbitraire du découpage. La recherche peut être très rapide mais la qualité difficile à contrôler. La méthode est inadaptée aux signatures creuses : le grand nombre de 0 impliquerait un mauvais équilibre de la liste inversée et de nombreuses comparaisons entre des signatures trop peu similaires.

La méthode All Pairs de [25] est adaptée aux signatures binaires creuses et les comparaisons expérimentales montrent sa supériorité par rapport à partEnum [26] (qui est néanmoins une version *exacte* de la jointure par similarité). Mais les performances ne sont évaluées que sur des signatures de très grande dimension (plus de 1000000) très creuses (entre 1000 et 20 bits à 1). L'utilisation sur des signatures beaucoup plus courtes, avec un nombre pratiquement identique de bits à 1, rendrait globalement inopérantes des optimisations qui jouent un rôle important dans les performances de cette méthode.

SASH [27] montre de bonnes performances pour les vecteurs de dimension moyenne à très grande, plus ou moins creux. SASH construit un graphe à niveaux avec les éléments stockés dans la base (un noeud est un élément). Les connexions entre niveaux sont établies récursivement, suivant les k plus proches voisins approximatifs. Ces connexions permettent de diriger la recherche vers des voisins potentiellement proches. La méthode a été définie pour un stockage en mémoire vive du graphe entier, ce qui ne permet pas dans notre cas de traiter une base de plusieurs milliers d'heures.

RBV [28] emploie une indexation redondante (comme LSH [22], OMEDRANK [20], PvS [21]) basée cette fois sur le processus de recherche. En raison de la structure utilisée, toutes les données de travail doivent résider en mémoire vive, ce qui limite la taille de la base. Aussi, le modèle de requête employé est peu adapté à des signatures à composantes binaires.

Plusieurs des méthodes mentionnées font appel à une indexation redondante, qui est bien adaptée à la problématique de la fouille de données. En effet, la redondance rend possible une segmentation de la base suivant le critère de similarité et la réalisation des comparaisons de façon indépendante dans chaque segment. La redondance permet également de paralléliser facilement la fouille pour un meilleur passage à l'échelle.

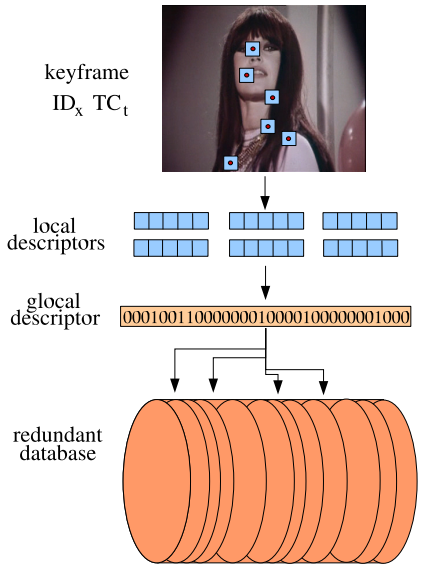

Fig. 6. Construction de la base redondante de signatures

Aucune des méthodes mentionnées ne satisfait entièrement nos exigences, certaines exploitent une indexation redondante (LSH, OMEDRANK, PvS, RBV) mais ne sont pas adaptées aux signatures binaires, ou alors traitent des signatures binaires mais ne permettent pas d'atteindre les volumes de données attendus (SASH) ou font des hypothèses qui ne sont pas vérifiées par nos données (All Pairs). La proposition de [24] laisse penser qu'une méthode inspirée de l'indexation textuelle peut être intéressante à condition de définir une segmentation mieux adaptée de la base. En nous appuyant sur cette idée, nous avons mis au point une nouvelle méthode d'indexation de signatures binaires creuses, approximative et utilisant la redondance, adaptée aux caractéristiques du descripteur Glocal et au problème de fouille. Cette méthode est décrite dans le section suivante.

B. Méthode proposée

L'étape la plus coûteuse lors de la fouille vidéo par DCPC est l'identification des liens de similarité entre les images clés. Si $\mathcal{D}$ est la base de signatures Glocal et θ le seuil de similarité au dessus duquel

une image clé est considérée comme étant une version transformée de l'autre, alors la fouille devrait retourner l'ensemble

$$\mathcal{K}_\theta = \{(\mathbf{g}_i, \mathbf{g}_j) \,|\, \mathbf{g}_i, \mathbf{g}_j \in \mathcal{D}, i < j, S_{\text{Dice}}(\mathbf{g}_i, \mathbf{g}_j) > \theta\} \quad (2)$$

Cela correspond à une auto-jointure par similarité sur la base, opération dont la complexité serait $O(N^2)$ sans index (où N est le nombre de signatures dans la base).

Nous souhaitons pouvoir fouiller des volumes de 10 000 heures de vidéo ($\geq 30 \times 10^6$ signatures Glocal) ou plus. Afin de rendre possible l'utilisation à des échelles encore plus grandes, la solution proposée doit être parallélisable.

C. Principe de l'indexation

L'indexation redondante que nous proposons est inspirée des listes inversées et du hachage. La méthode a pour but de diviser la base de signatures Glocal en sous-bases en faisant en sorte que les signatures Glocal dont la similarité est supérieure à un seuil se retrouvent ensemble dans au moins une sous-base. Une auto-jointure est ensuite réalisée dans chaque sous-base. Dans une grande base, les signatures ne sont pas regroupées dans des *clusters* compacts et bien séparés, qui pourraient constituer les sous-bases ; les différentes sous-bases doivent donc se recouvrir partiellement afin de garantir que toutes les paires de signatures similaires sont identifiées par les auto-jointures dans les sous-bases. Une redondance excessive est toutefois source d'inefficacité. Par ailleurs, chaque sous-base doit tenir en mémoire afin d'éviter les accès disque lors de l'opération de jointure.

La figure 6 résume les étapes nécessaires à la construction de cette base redondante, de l'extraction des signatures dans les images clés à l'insertion dans la base. La suite décrit la façon de construire les sous-bases et de les remplir de signatures Glocal.

Nous appellerons « mot » chaque position dans une signature Glocal ; pour une profondeur d'indexation $p = 8$ nous avons 256 mots possibles. Une sous-base (*bucket*) est définie par un ensemble de positions de bits à 1, appelé « phrase » ci-dessous ; la sous-base sera une liste inversée (la liste des signatures qui contiennent une phrase spécifique). Pour chaque signature Glocal on construit un ensemble de phrases à partir des positions des bits à 1. La signature sera stockée dans chacune des sous-bases correspondant aux phrases de cet ensemble. Il reste à préciser (i) à quelle profondeur l'espace de description doit être partitionné pour définir les signatures Glocal, (ii) quelle longueur doivent avoir les phrases, (iii) combien de phrases doivent être employées pour indexer une signature Glocal et (iv) comment choisir ces phrases.

D. Estimation du gain potentiel

Soit N le nombre total de signatures Glocal dans la base. Si la similarité était déterminée pour chaque paire de signatures de la base, $\frac{N(N-1)}{2} \approx \frac{N^2}{2}$ calculs seraient nécessaires pour l'auto-jointure. En considérant une base indexée selon le schéma précédemment décrit, avec des signatures obtenues à une profondeur p et des phrases de longueur l, le nombre de sous-bases possible est $C^l_{2^p}$. Comme vu précédemment, le nombre de mots (nombre de bits à 1) par signature, noté M, est proche de 20, pratiquement le même pour toutes les signatures et stable pour $p \geq 8$. Il y a alors C^l_M phrases de l mots possibles par signature. Si tous les bits peuvent prendre la valeur 1 avec la même probabilité, alors toutes les sous-bases ont la même taille, égale à $\frac{NC^l_M}{C^l_{2^p}}$. Le nombre de calculs de similarité devient $\left(\frac{NC^l_M}{C^l_{2^p}}\right)^2 / (2C^l_{2^p}) = \frac{(NC^l_M)^2}{2C^l_{2^p}}$. Le gain en nombre de calculs sera donc :

$$a = \frac{C^l_{2^p}}{(C^l_M)^2} \quad (3)$$

Pour un vecteur binaire creux comme une signature Glocal, le temps de calcul de similarité entre 2 vecteurs peut être considéré fixe et indépendant de p (pour p entre des limites assez larges). Comme le montre la Fig. 7 pour $l \in \{1, \ldots, 6\}$ et $p \in \{8, \ldots, 20\}$ (avec $M = 20$), le gain donné par (3) augmente avec l et p. La redondance étant de $C^l_{2^p}$, elle augmente lorsque l varie de 1 à 10, puis diminue. En prenant pour l des valeurs entre 15 et 20, la similarité minimale entre les signatures d'une même sous-base serait de $\frac{2\times15}{20+20} = 0,75$, selon (1), ce qui abaisserait fortement le taux de rappel (le nombre de copies retrouvées). Pour cette raison nous emploierons plutôt des phrases courtes. La taille de l'index a également de l'importance ; cette taille est de NC^l_M, elle augmente donc avec

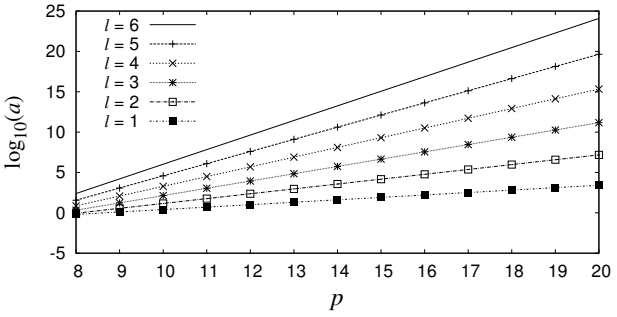

Fig. 7. Impact de p et l sur le gain potentiel

l lorsque l varie de 1 à 10, puis diminue. Ceci encourage la sélection pour l de valeurs proches soit de 1, soit de 20. Une « étude des collisions » entre signatures copiées nous aidera dans le choix de ces paramètres.

Pour réduire encore plus le nombre de calculs, il faut limiter la redondance. Il est possible de diminuer à la fois le nombre de sous-bases et la cardinalité de chaque sous-base en affectant chaque signature à une faible part des sous-bases auxquelles elle devrait appartenir. Pour cela, il suffit de définir des règles pour sélectionner les phrases à utiliser. Toutefois, la réduction de la redondance ne doit pas être excessive, sous peine d'obtenir comme résultat final une faible part de $\mathcal{K}_\theta$. Le procédé de filtrage proposé est décrit dans la section suivante.

E. Sélection des phrases

Les règles que nous formulons ici sont les suivantes : créer les phrases en utilisant des bits à 1 voisins (pas de bits à 1 entre ces bits), un bit à 1 sur deux (des bits à 1 espacés de un bit à 1), un bit à 1 sur trois et un bit à 1 sur quatre. Par exemple, pour la signature **1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1**, les positions des bits à 1 sont 1, 6, 11, 14 et 16. Si on considère les phrases de longueur $l = 2$, les phrases de bits à 1 voisins sont 1-6, 6-11, 11-14 et 14-16, les phrases utilisant un bit à 1 sur deux sont 1-11, 6-14 et 11-16 et celles utilisant un bit à 1 sur trois sont 1-14 et 6-16. Le choix de ces phrases est fait de façon à remplir le plus homogène possible les sous-bases. La réduction du nombre de phrases est alors très significative, pour $l = 3$ on obtient 48 phrases choisies parmi les 1120 possibles. Cependant, si cette réduction est trop forte, le risque de perdre des détections est élevé.

Pour estimer la pertinence de ces règles et évaluer si un aussi petit nombre de phrases (par rapport au nombre possible) est suffisant, nous avons réalisé une étude de collision, utilisant notre base de copies artificielles, déjà mentionné dans la section III. Cette analyse nous permet par ailleurs de choisir un p et un l selon nos exigences de qualité des résultats.

F. Analyse des collisions

Nous estimons la probabilité de collision entre la signature d'une image clé originale et celui d'une image clé copiée ayant subi quelques transformations. La collision traduit le fait que ces deux signatures se retrouvent au moins une fois ensemble dans une sous-base (leur similarité est alors calculée). Les résultats apparaissent dans la figure 8 pour $p \in \{7, 8, 9\}$. La figure 9 montre la probabilité de collision entre deux signatures prises au hasard (mais n'appartenant pas à une même séquence vidéo), cette probabilité doit être aussi basse que possible pour éviter des calculs inutiles. Sur chaque graphique apparaissent 4 courbes, la plus à gauche correspond à la règle de construction de phrases n'utilisant que les bits 1 voisins (notée « 0 »), la seconde reprend cette règle et ajoute la règle « un bit à 1 sur deux » (notée « 0_1 »), la troisième ajoute la règle « un bit à 1 sur 3 » (notée « 0_1_2 ») et la quatrième ajoute la règle « un bit à 1 sur 4 » (notée « 0_1_2_3 »). Ainsi on a {phrases 0} $\subset$ {phrases 0_1} $\subset$ {phrases 0_1_2} $\subset$ {phrases 0_1_2_3}. L'abscisse note le nombre de phrases générées par les règles. Sur chacune des courbes apparaissent des points, chacun correspond à une longueur de phrase de 2 à 8 : la point le plus haut de chaque courbe correspond aux phrases de longueur 2, le suivant aux phrases de longueur 3, etc.

Étant donné que nous voulons une probabilité de collision forte entre la signature d'une image et celle de sa copie, et faible entre signatures prises au hasard, les figures 8 et 9 indiquent que le choix de phrases de longueur $l = 3$ avec des signatures à profondeur $p = 8$ en utilisant la règle « 0_1_2 » présente un bon compromis, un rappel élevé (probabilité de collision de 0,9 entre la signature d'une image et celle de sa copie) et une bonne précision (probabilité de collision $< 0,01$ entre signatures prises au hasard). Notons que cette précision est sous-évaluée, deux signatures devant en plus présenter

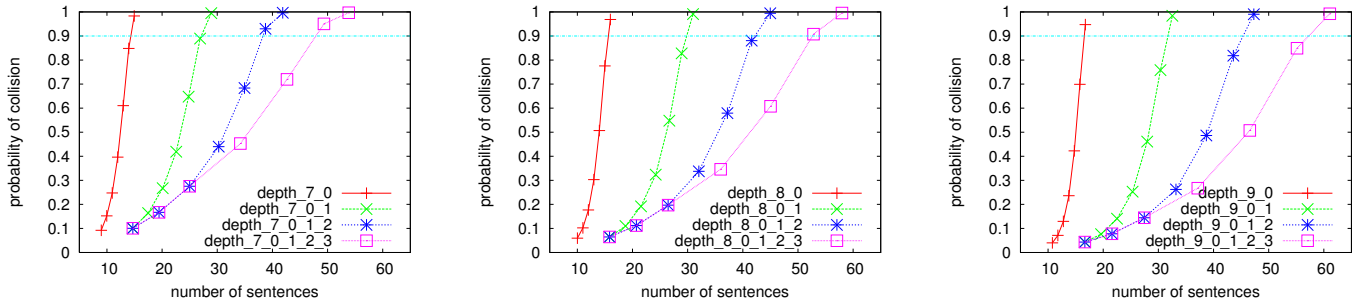

Fig. 8. Probabilité estimée de collision entre la signature Glocal issue d'une image clé originale et de sa version transformée, à profondeur d'indexation 7 (à gauche), 8 (au milieu) et 9 (à droite)

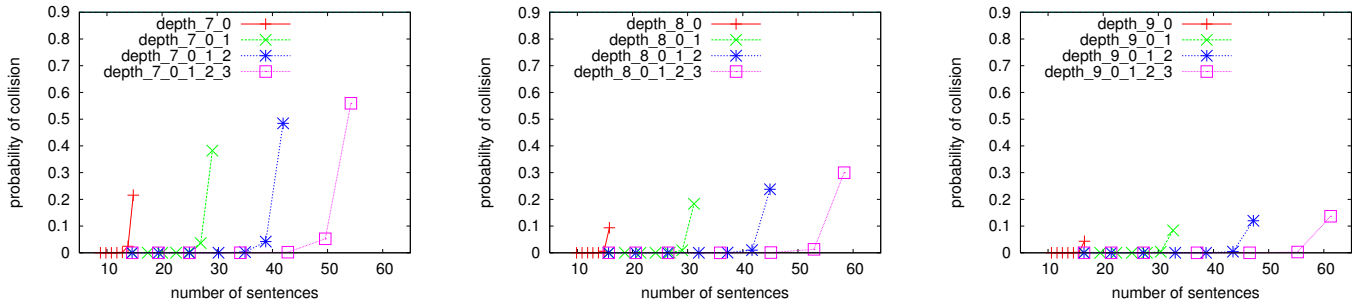

Fig. 9. Probabilité estimée de collision entre signatures Glocal choisies au hasard, à profondeur d'indexation 7 (à gauche), 8 (au milieu) et 9 (à droite)

une similarité supérieure à θ pour pouvoir passer le filtrage. Ajoutons enfin que pour une application de fouille nous favoriserons la précision au rappel, car la présence de nombreuses fausses détections enlèverait tout intérêt à cette approche.

L'algorithme 1 résume les deux étapes vues jusqu'à présent, l'indexation des images clés suivie de la jointure par similarité. En sortie, les paires d'images clés sont triées.

V. Mise en correspondance des séquences vidéo

Au terme de l'indexation redondante, nous pouvons procéder à la jointure par similarité (voir la requête `R2`). Pour cela, la similarité est calculée dans chaque sous-base, entre toutes les signatures d'identifiant (ID) différent. Avec les choix réalisés dans la section précédente, la borne inférieure pour la similarité est de 0,15 (voir (1)), ce qui est assez bas. Même si le nombre de calculs à été fortement réduits, il faut encore filtrer pour éliminer des fausses détections. On fixe donc un seuil de similarité θ au-delà duquel les similarités seront acceptées (voir section VI). Les étiquettes temporelles $(\mathrm{Tc_{x,n}}; \mathrm{Tc_{y,m}})$ des signatures présentant une similarité supérieure à θ sont alors stockées dans un fichier identifié par $(\mathrm{ID_x}; \mathrm{ID_y})$ (les vidéos dont sont issues ces deux signatures) et ensuite triées ; la complexité du tri est $O(N_k \log N_k)$, où N_k est le nombre de paires stockées dans le fichier k.

Fusion temporelle. La dernière étape de la fouille vidéo par détection de copies est la reconstitution des couples de segments vidéo à partir des couples d'images clés. Dans chacun des fichiers créés à l'étape précédente et identifiés par par $(\mathrm{ID_x}; \mathrm{ID_y})$, on part des segments unitaires que sont les couples d'images clés et on essaie de les prolonger en exploitant les autres couples. Pour cela, sont vérifiées plusieurs contraintes qui caractérisent la cohérence temporelle des images au sein des deux vidéos.

La première contrainte est le caractère séquentiel, traduit par une double condition d'antériorité des images clés d'un couple par rapport aux images du couple qu'on cherche à lui ajouter. La seconde

Algorithm 1 Indexation et jointure par similarité

Require: DG : ensemble des signatures Glocal
Ensure: LP : liste des paires de (ID,TC)

1: *// Indexation redondante*
2: **for all** $DG_i \in DG$ **do**
3: Déterminer ensemble de phrases PH_i
4: **for all** $PH_ij \in PH_i$ **do**
5: Déterminer adresse sous-base SB_k
6: **if** SB_k n'existe pas **then**
7: Créer SB_k
8: **end if**
9: Insérer DG_i dans sous-base SB_k
10: **end for**
11: **end for**
12: *// Jointure par similarité*
13: *// SB : ensemble des sous-bases*
14: **for all** $SB_i \in SB$ **do**
15: **for all** $DG_{ij} \in DG_i$ **do**
16: **for all** $DG_{ik} \in DG_i, k > j$ **do**
17: Calculer $S_{ijk} = S_{\text{Dice}}(DG_{ij}, DG_{ik})$
18: **if** $S_{ijk} > \Theta$ **then**
19: Ajouter $(DG_{ij}.id, DG_{ij}.tc)$ $(DG_{ik}.id, KF_{ik}.tc)$ à LP
20: **end if**
21: **end for**
22: **end for**
23: **end for**
24: *// Tri*
25: Trier $LP = \{\langle (DG_{1i}.id, DG_{1i}.tc), (DG_{2i}.id, DG_{2i}.tc) \rangle\}$ par $DG_{1i}.id$, $DG_{2i}.id$, $DG_{1i}.tc$, $DG_{2i}.tc$ croissants

contrainte est la proximité temporelle : il ne faut pas lier des couples d'images clés si l'écart temporel est trop important, mais on autorise l'absence d'un ou plusieurs couples dans la séquence. Enfin, pour à la fois éliminer des aberrations temporelles et compenser les effets dus à la non reproductibilité du détecteur d'images clés entre la vidéo originale et une copie (les images clés n'ont pas toujours la même étiquette temporelle en cas de copie non exacte), on ajoute une dernière condition, qui impose une certaine homogénéité des écarts temporels entre les images clés d'un même couple (une borne supérieure sur la gigue acceptée).

L'algorithme 2 présente les trois contraintes au

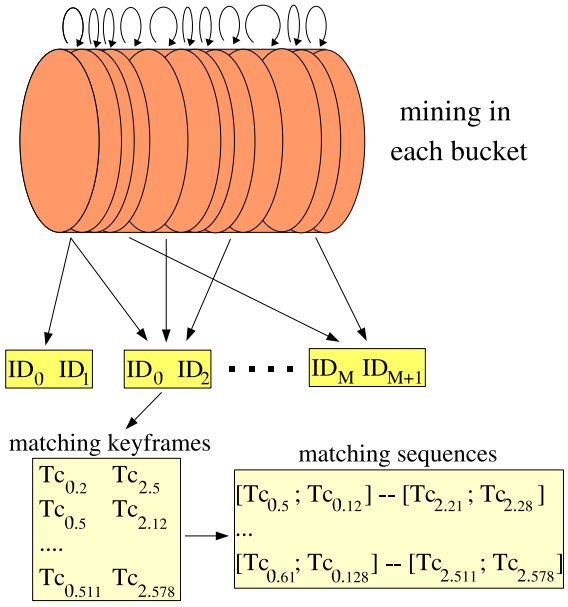

Fig. 10. Mise en correspondance des séquences vidéo

sein du processus de reconstitution des paires des segments homologues. Th_d est le seuil du délai et Th_h est le seuil d'homogénéité. Les fichiers sont classés par paires d'étiquettes temporelles croissantes (voir la requête `R2`), la complexité de cette étape est donc linéaire dans le nombre de paires d'images clés résultant de l'étape précédente.

La figure 11 présente une possibilité de parallélisation de l'ensemble du processus. L'étage le plus haut, $V_1 \dots V_{NV}$, représente l'ensemble des vidéos à fouiller. La première étape, le calcul des signatures Glocal, est parallélisable (étage 2). Le calcul des sous-bases destinées à recevoir les signatures et l'insertion dans ces sous-bases (étage $B_1 \dots B_{NB}$) est également parallélisable, ainsi que le calcul des similarités dans chacune des sous-bases (étage $SSS_1 \dots SSS_{NB}$). Les paires d'images clés sont réparties dans les fichiers correspondant à une paire de ID (étage $SRV_1 \dots SRV_{NV}$) ; la reconstitution des segments en correspondance (étage $TCV_1 \dots TCV_{NV}$) peut être réalisée en parallèle dans chacun de ces fichiers et les résultats écrits sur disque (étage $RV_1 \dots RV_{NV}$).

VI. Évaluation expérimentale

Pour juger de la pertinence de notre approche, nous avons d'abord calibré sur une vérité terrain le seuil de similarité θ. Dans cette vérité terrain nous

Algorithm 2 Détection des séquences liées

Require: LP : liste des paires de (ID,TC)
Ensure: SEG : liste de paires de segments

for all $\langle DG_{1i}.tc, DG_{2i}.tc\rangle \in LP$ **do**
  for all SEG_j in SEG **do**
    $EC_1 = DG_{1i}.tc - SEG_{1j}.tcFin$
    $EC_2 = DG_{2i}.tc - SEG_{2j}.tcFin$
    // double antériorité
    if $((DG_{1i}.tc \leq SEG_{1j}.tcFin)$ **and** $(DG_{2i}.tc \leq SEG_{2j}.tcFin))$
    and // proximité
    $((EC_1 \leq Th_d)$ et $(EC_2 \leq Th_d))$
    and // homogénéité
    $((EC_1 - EC_2) \leq Th_h)$ **then**
      $SEG_1j.tcFin = DG_1i.tc$
      $SEG_2j.tcFin = DG_2i.tc$
    else
      Ajouter $\langle DG_1i.tc, DG_2i.tc\rangle$ à SEG
    end if
  end for
end for

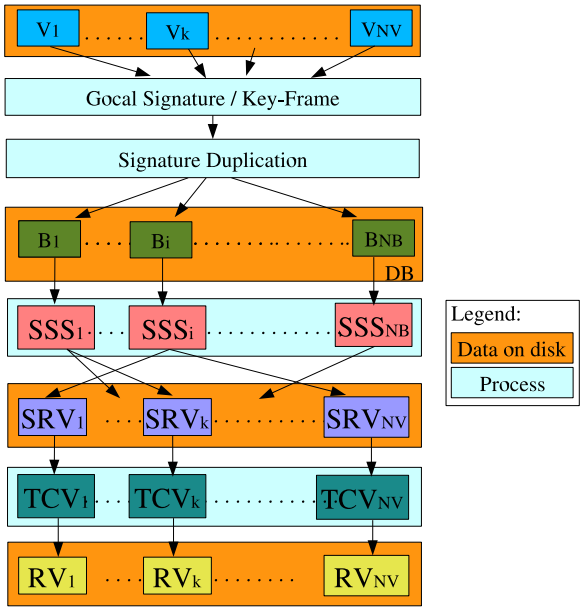

Fig. 11. Parallélisation complète du processus de fouille. Seuls les résultats intermédiaires doivent être fusionnés

connaissons *a priori* tous les liens de copie. Le seuil θ est fixé de façon à avoir un bon rappel (on retrouve une grande partie des liens existants), mais surtout une précision forte (on ne trouve pas de faux liens).

Une fois cette calibration faite, le passage à l'échelle de l'approche est examiné en utilisant des bases de 2 000, 5 000 et 10 000 heures de vidéo issues de l'INA. Sont ensuite présentés visuellement les résultats produits sur l'ensemble des vidéos obtenues en réponse à une requête par mot clé faite sur un site Web2.0.

Dans toutes ces expérimentions Th_d est fixé à 100 images (environ 4 secondes) pour d'éviter une fragmentation trop forte des résultats de détection. Th_h est fixé à 15 images (environ 0,6 secondes) pour compenser les modifications temporelles de post-production et les imperfections du détecteur d'images clés. Sont conservées uniquement les liens entre séquences comprenant plus de $Th_l = 4$ images clés, les séquences plus courtes correspondant souvent à de fausses alarmes.

A. *Qualité de détection*

La première vérité terrain utilisée contient 30 heures de vidéos originales, appartenant à l'INA, et 20 minutes de copies, découpées en 20 extraits trouvés sur un site Web2.0. Tous les programmes originaux contiennent au moins une séquence copiée, plus ou moins courte et plus ou moins attaquée, dans les 20 extraits. Tous les liens de contenu sont connus. La base de données étant petite, l'indexation est faite à une profondeur $p = 8$.

Des expérimentations antérieures [14] avaient montré que si la moitié des signatures locales étaient conservés entre 2 images clés, on pouvait les considérer comme copies. Ce qui laisse penser qu'un coefficient de Dice de 0,50 ferait un bon seuil. Ce résultat est proche du seuil de 0,45 au-delà duquel on retrouve 90% des images copiées (voir la section III). Etant donné qu'il est préférable d'avoir un minimum de fausses alarmes, les premiers tests sont faits avec un seuil de 0,50. On obtient alors 46 vrais positifs (vraies copies trouvées), 6 faux négatifs (vraies copies non trouvées) et 30 faux positifs (séquences qui ne sont pas des copies mais sont détectées), ce qui donne un rappel de 0,88 et une précision de 0,60. Cette précision étant trop basse par rapport à nos exigences, le seuil est augmenté à 0,55, avec pour conséquence 44 vrais positifs, 8 faux négatifs et 2 faux positifs, donc un rappel de 0,84 pour une précision de 0,95. La faible diminution du taux de rappel permet d'augmenter

la précision à un niveau très satisfaisant. De plus, les faux négatifs correspondent à des séquences très courtes, très bruitées ou fortement compressées (les artéfacts MPEG dominent). De plus, un seuil plus elevé permet d'avoir moins d'images clés pour la dernière étape de reconstruction des séquences, qui sera donc plus rapide.

Ce seuil a été validé sur une autre vérité terrain, celle de la base publique d'évaluation de détection de copies[1] de CIVR 2007. Il s'agit d'une base de 80 heures avec 2 lots de requêtes, le premier contenant des copies intégrales, le deuxième des copies d'extraits. Sur les 2 lots réunis, avec un seuil de similarité de 0,55, le rappel obtenu est de 0,8 et la précision de 0,96. Comme dans ce cas toutes les données peuvent loger en mémoire, les opérations de fouille prennent 48 secondes.

B. Passage à l'échelle

Afin d'évaluer le temps nécessaire pour la fouille de grands volumes de vidéo, trois bases de 2 000, 5 000 et respectivement 10 000 heures ont été créées à partir de vidéos prises au hasard dans la base de l'INA. Ces bases ont été indexées à une profondeur $p = 9$. La table I montre les temps nécessaires pour chaque étape du processus : création de la base d'images clés, jointure par similarité sur les images clés et reconstitution des segments homologues. L'étape de construction de la base comprend le calcul des signatures Glocal à partir des signatures des points d'intérêt et la création des sous-bases. Les accès disque représentent 95% du temps de construction.

TABLE I
TEMPS REQUIS POUR LA FOUILLE DE 3 BASES VIDÉOS

Taille de base	2 000 h	5 000 h	10 000 h
Nb. images clés	$5,8 \times 10^6$	$14,5 \times 10^6$	$28,7 \times 10^6$
Construction base	2h35min	3h38min	7h00min
Auto-jointure images	5h40min	14h59min	55h
Liens entre segments	1h15min	7h15min	17h35min

Pour comparaison, si aucun index n'avait été employé, la jointure par similarité sur 10 000 heures aurait pris environ un an. La méthode proposée demande un peu plus de 3 jours pour accomplir le processus global. En utilisant le système de détection de copies de [14] pour filtrer les images clés, 20 jours auraient été nécessaires pour l'ensemble du processus, et avec une précision résultante beaucoup plus faible.

C. Résultats de la fouille d'une base Web2.0

Nous avons construit une base à partir de vidéos récupérées (pour évaluation) d'un site Web2.0 afin d'illustrer l'exemple cité en introduction. Il s'agissait de faire de la fouille en cherchant les occurrences des vidéos de Madonna. En raison de contraintes opératoires, seules les 925 premières vidéos issues de la requête « Madonna » ont pu être récupérées et transcodées vers un format exploitable. Le volume total de la base résultante est de 63 heures. Comme dans ce cas toutes les données peuvent loger en mémoire, l'étape de fouille (calcul des similarités et reconstitution des segments) dure seulement 42 secondes.

A partir des résultats, un graphe a été construit. Chaque noeud est un segment d'une vidéo ayant au moins une copie dans la base. Le graphe des segments, avec 477 noeuds et 1978 arêtes, est présenté dans la figure 12. Les plus courtes séquences détectées contiennent seulement 4 images clés. Pour trois sous-graphes, des noeuds sont illustrés par des images clés extraites. La clique du sous-graphe A correspond à des versions transformées de la même vidéo, la queue contenant des versions très transformées du même clip plus de nouveaux éléments. Le plus grand graphe connexe C, occupant le centre, contient une multitude de segments de clips du récent *single* « 4 minutes ». On trouve une multitude de montages des clips, quelques versions officielles, mais surtout des versions remaniées par des admirateurs. Le graphe B fait apparaître des segments rarement utilisés de ce clip.

Les modifications les plus fréquentes sont d'abord les remontages, avec des plans courts de quelques secondes, ré-agencés de multiples fois, la fin pouvant devenir le début, etc. On trouve aussi des changements d'échelle ou de ratio de l'image. Une autre modification très utilisée est la compression temporelle, certaines versions du même clip sont 15% plus courtes que la référence (qui fait 4 minutes). Un cas extrême fait apparaitre un extrait de concert de 1 minute 20 et une version de 1 minute (compression temporelle de 25%). Bien sûr,

[1] `http ://www-rocq.inria.fr/imedia/civr-bench/`

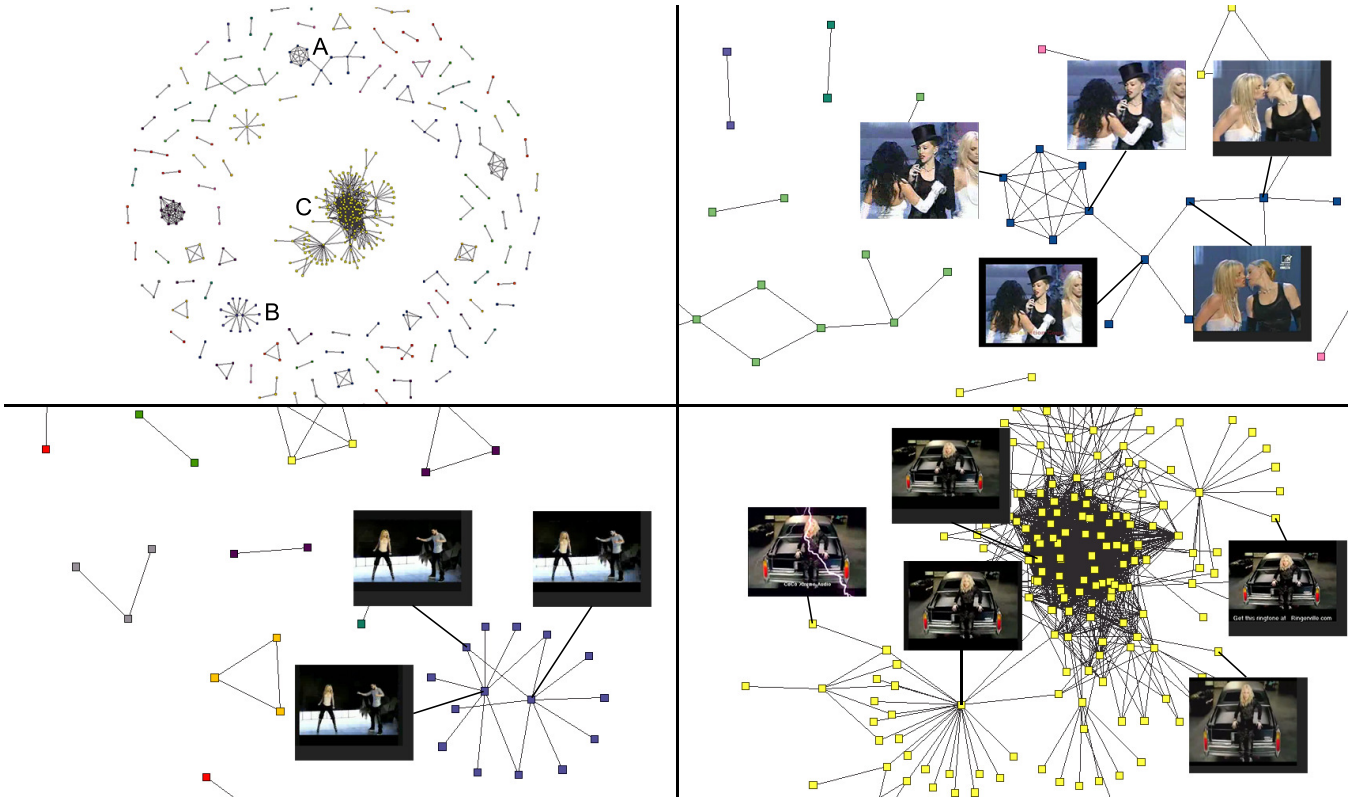

Fig. 12. Graphe trouvé (en haut à gauche) pour les 925 vidéos extraites d'un site Web2.0 (pour évaluation) et illustrations pour les sous-graphes A, B et C

on trouve également les occurences avec différents logos et incrustations des chaînes les ayant diffusés.

Avec les attributs que l'on peut mettre sur les arêtes (longueur, fréquence, similarité, etc.), cette représentation graphique peut devenir un puissant outil de navigation visuel dans les bases vidéo.

VII. Conclusion

La multiplication du nombre de copies transformées dans les bases de données vidéo motive la fouille de ces bases par des méthodes de détection de copie basées sur le contenu vidéo. En exprimant cette fouille comme une auto-jointure par similarité sur l'ensemble des images clés extraites des vidéos, nous proposons ici une solution qui passe à l'échelle et offre un bon taux de détection (ou rappel) ainsi qu'une très bonne précision. Nous avons introduit un descripteur (Glocal) qui associe un vecteur unique et compact à chaque image clé, obtenu à partir des signatures d'un ensemble de points d'intérêt détectés dans l'image. Une méthode d'indexation adaptée au descripteur et à la tâche de fouille a ensuite été proposée. Il devient ainsi possible de réaliser l'opération d'auto-jointure par similarité de façon approximative dans des délais acceptables sur de grandes bases vidéo. On trouve les différentes occurrences d'un extrait (transformé) à un niveau de granularité fin, de quelques secondes. Les performances de la méthode ont été mises en évidence sur des bases allant jusqu'à 10 000 heures. La méthode est en outre parallélisable facilement et de façon efficace, ce qui permet d'aller encore plus loin dans le passage à l'échelle.

Remerciements

Ce travail s'inscrit dans le cadre du projet Sigmund financé par l'Agence Nationale pour la Recherche. Les remarques et suggestions de Michel Scholl, ainsi que celles des rapporteurs anonymes, ont permis d'améliorer de façon significative la présentation. Remerciements également à Jerôme Thièvre (INA) pour l'outil d'affichage et de navigation dans les graphes.

REFERENCES

[1] S. Satoh, M. Takimoto, and J. Adachi, "Scene duplicate detection from videos based on trajectories of feature points," in *Proceedings of the international workshop on Multimedia Information Retrieval (MIR'07)*. New York, NY, USA : ACM, 2007, pp. 237–244.

[2] J. M. Gauch and A. Shivadas, "Finding and identifying unknown commercials using repeated video sequence detection," *Computer Vision and Image Understanding*, vol. 103, no. 1, pp. 80–88, 2006.

[3] S. Satoh, "News video analysis based on identical shot detection," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'02)*, 2002, pp. 69–72.

[4] F. Yamagishi, S. Satoh, and M. Sakauchi, "A news video browser using identical video segment detection," in *PCM (2)*, ser. Lecture Notes in Computer Science, K. Aizawa, Y. Nakamura, and S. Satoh, Eds., vol. 3332. Springer, 2004, pp. 205–212.

[5] X. Wu, W.-L. Zhao, and C.-W. Ngo, "Near-duplicate keyframe retrieval with visual keywords and semantic context," in *Proceedings of the 6th ACM international Conference on Image and Video Retrieval (CIVR'07)*. New York, NY, USA : ACM, 2007, pp. 162–169.

[6] X. Wu, A. G. Hauptmann, and C.-W. Ngo, "Practical elimination of near-duplicates from web video search," in *Proceedings of the 15th international conference on Multimedia*. New York, NY, USA : ACM, 2007, pp. 218–227.

[7] A. Hampapur, K. Hyun, and R. M. Bolle, "Comparison of sequence matching techniques for video copy detection," in *Proc. Conf. on Storage and Retrieval for Media Databases*, M. M. Yeung, C.-S. Li, and R. W. Lienhart, Eds., December 2002, pp. 194–201.

[8] E. Mortensen, H. Deng, and L. Shapiro, "A SIFT descriptor with global context," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005, pp. 184–190.

[9] Y. Ke and R. Sukthankar, "PCA-SIFT : A more distinctive representation for local image descriptors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, vol. 02. Los Alamitos, CA, USA : IEEE Computer Society, 2004, pp. 506–513.

[10] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 530–535, 1997.

[11] A. Joly, C. Frélicot, and O. Buisson, "Discriminant local features selection using efficient density estimation in a large database," in *Proc. 7th ACM SIGMM intl. workshop on Multimedia Information Retrieval (MIR'05)*. New York, NY, USA : ACM Press, 2005, pp. 201–208.

[12] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE *Transaction on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[13] T. Gevers and A. W. M. Smeulders, "Content-based image retrieval : An overview," in *Emerging Topics in Computer Vision*, G. Medioni and S. B. Kang, Eds. Prentice Hall, 2004, ch. 8.

[14] S. Poullot, O. Buisson, and M. Crucianu, "Z-grid-based probabilistic retrieval for scaling up content-based copy detection," in *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR'07)*, Amsterdam, The Netherlands, July 2007, pp. 348–355.

[15] S.-A. Berrani, L. Amsaleg, and P. Gros, "Robust content-based image searches for copyright protection," in *Proc. 1st ACM intl. workshop on Multimedia Databases (MMDB'03)*. New Orleans, USA : ACM Press, 2003, pp. 70–77.

[16] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa, "Robust voting algorithm based on labels of behavior for video copy detection," in *Proceedings of the 14th annual ACM international conference on Multimedia*. New York, NY, USA : ACM Press, 2006, pp. 835–844.

[17] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford, "Video copy detection : a comparative study," in *Proceedings of the 6th ACM international conference on Image and video retrieval (CIVR'07)*. New York, NY, USA : ACM, 2007, pp. 371–378.

[18] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[19] H. Bay, T. Tuytelaars, and L. J. V. Gool, "SURF : Speeded up robust features," in *Proceedings of the European Conference on Computer Vision (ECCV'06)*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951. Springer, 2006, pp. 404–417.

[20] R. Fagin, R. Kumar, and D. Sivakumar, "Efficient similarity search and classification via rank aggregation," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD'03)*. New York, NY, USA : ACM, 2003, pp. 301–312.

[21] H. Lejsek, F. H. Ásmundsson, B. T. Jónsson, and L. Amsaleg, "Scalability of local image descriptors : a comparative study," in *Proceedings of the 14th annual ACM international conference on Multimedia*. New York, NY, USA : ACM, 2006, pp. 589–598.

[22] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999, pp. 518–529.

[23] R. M. A. Miller M. L. and C. I. J., "Audio fingerprinting : nearest neighbor search in high dimensional binary spaces," in *IEEE Workshop on Multimedia Signal Processing*, 2002.

[24] J. Oostveen, T. Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Proc. 5th Intl. Conf. on Recent Advances in Visual Information Systems (VISUAL'02)*. London, UK : Springer-Verlag, 2002, pp. 117–128.

[25] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in *Proceedings of the 16th international conference on World Wide Web (WWW'07)*. New York, NY, USA : ACM, 2007, pp. 131–140.

[26] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in *Proceedings of the 32nd international conference on Very Large Data Bases (VLDB'06)*. VLDB Endowment, 2006, pp. 918–929.

[27] M. E. Houle and J. Sakuma, "Fast approximate similarity search in extremely high-dimensional data sets," in *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*. Washington, DC, USA : IEEE Computer Society, 2005, pp. 619–630.

[28] J. Goldstein, J. C. Platt, and C. J. C. Burges, "Redundant bit vectors for quickly searching high-dimensional regions," in *Deterministic and Statistical Methods in Machine Learning*, 2004, pp. 137–158.