

Mémoire

Billal Boudjoghra

August 24, 2019

Contents

1	TODO vérif un frame vs une frame	4
2	TODO faire la distinction plan / transition de plan	4
3	TODO plus détaillé la partie détection de contours	4
4	Sportagraph	4
5	Introduction	6
6	Analyse d'image : état de l'art concernant la détection de replay	8
6.1	A Robust Replay Detection	8
6.1.1	Calcul du score de la luminance et filtrage des frames .	8
6.1.2	Recherche du logo template parmi les frames filtrés . .	8
6.1.3	Recherche des logo	9
6.1.4	Recherche des replays	9
6.1.5	Résultats des auteurs	9
7	TODO résultat auteurs pour robust replay	9
7.1	Mean Shift Based video Segment Representation And Appli- cations To Replay Detection	9
7.1.1	Algorithme	10
7.1.2	Résultats	10
8	Apprentissage automatique : les bases théoriques	12
8.1	Réseaux de neurones récurrents (RNN)	12
8.2	LSTM	12
8.3	CNN	13

8.3.1	Intéraction parcimonieuse	14
8.3.2	Partage de paramètres	15
8.3.3	Représentations équivariantes	15
8.3.4	Pooling	16
9	Apprentissage profond : état de l’art pour la reconnaissance d’action dans les vidéos	18
9.1	Two-Stream Convolutional Networks for Action Recognition in Videos	18
9.1.1	Classifieur spatial	18
9.1.2	Classifieur temporel	19
9.1.3	Méthode d’évaluation et résultats obtenus	19
9.2	Learning Spatiotemporal Features with 3D Convolutional Networks	21
9.2.1	C3D: Convolution et pooling 3D	21
9.2.2	Architecture et entraînement du réseau	22
9.2.3	Résultat pour la tâche de reconnaissance d’action	23
9.2.4	Conclusion	24
9.3	Beyond Short Snippets: Deep Networks for Video Classification	24
9.3.1	Approche	24
9.3.2	Architecture	25
9.3.3	Utilisation du flux optique	26
9.3.4	Résultat obtenu	26
10	Les approches proposées	28
10.1	Détection des plans	29
10.1.1	Online, Simultaneous Shot Boundary Detection And Key Frame Extraction For Sports Videos Using Rank Tracing	29
10.2	Analyse d’image - Première approche : ORB	30
10.2.1	Première expérimentation : 1 frame par shot	31
10.2.2	Deuxième expérimentation : 1 fenêtre de frame par shot:	32
10.3	Analyse d’image - Seconde approche : matching de contours	34
10.3.1	Mosaique de plan	34
10.3.2	Algorithme	34
10.4	Apprentissage profond - Première approche : convolution 3D	36
10.5	Flux optique	37
10.6	Apprentissage profond - Deuxième approche : convolution 2D sur des images de flux optique	38

10.7 Apprentissage profond - Troisième approche : convolution 3D sur des images de flux optique	38
11 Collecte des données & entraînement	38
11.1 Datasets	38
11.2 Collecte des données	39
11.3 Obtention des images de flux optiques	39
11.3.1 Image de flux optique : 2L canaux	40
11.3.2 Image de flux optique : somme des fluxs	41
11.4 Architecture du scrapper	41
12 Comparaison des approches	42
12.1 Procédure d'évaluation	42
12.2 Résultats pour ORB	43
12.3 Résultats pour la détection par luminance	43
12.4 Résultats pour l'approche par matching de contours	44
13 Appendice	47
14 Table des figures	48

1 TODO vérif un frame vs une frame

2 TODO faire la distinction plan / transition de plan

parfois je parle de plan alors que je ne regarde que la transition entre les plan (par ex, avec ORB)

3 TODO plus détaillé la partie détection de contours

rajouter des images :

1. le shot (avec les images couleurs)
2. shot => contours
3. contours => mosaic
4. match mosaic

4 Sportagraph

Sportagraph est une start-up filiale de Everspeed, fondée en 2013 et lancée en 2016 après trois années de recherche et développement (R&D). Son produit est un DAM (Digital Asset Manager) spécialisé dans les événements sportifs.

Le DAM permet aux utilisateurs de stocker et de gérer plusieurs types de fichiers (principalement des images, mais les vidéos et d'autres types sont aussi supportés).

L'équipe travaillant sur ce projet est composée de :

- Alex Macris : Chief Experience Officer
- Edouard Binchet : Chief Strategy and Operation Officer
- Régis Jean-Gilles : Lead Server Developer (mon tuteur d’alternance)
- Benoit Hozjan : SaaS operation manager
- Francois Jacquier : Lead Developer Front
- Khedidja Almherabet : Développeur front-end
- Mate Gyomrei : Développeur front-end
- Daniel Denke : Développeur iOS
- Zoltan Tarsoly : Test Automation Engineer
- Youva Ammaouche : Assistant chef de projet
- Stéphane : Développeur front-end
- Ashant : Développeur front-end

En tant que développeur back end, je suis chargé de rajouter de nouvelles fonctionnalités au DAM.

L’une des missions qui m’a été affectée est la détection des replays dans les vidéos de sport.

Cette fonctionnalité nous permettra de fournir à nos clients un résumé automatique des vidéos qu’ils publient sur notre plateforme.

5 Introduction

Au cours des dernières années, les techniques d'apprentissage automatique ont joué un rôle de plus en plus important dans les systèmes de reconnaissance automatique.

Les avancements dans le domaine de l'apprentissage profond et l'accès à un volume massif de données ont permis de mettre en place des solutions de plus en plus performantes.

En particulier, l'adoption globale des téléphones intelligents et de leur caméra intégrée a accru de manière exponentielle le nombre de vidéos disponibles sur Internet, à tel point qu'il est devenu impossible pour l'humain d'ingérer manuellement le contenu de toutes les vidéos disponibles sur la toile.

Dans cet étude, nous allons nous intéresser à la tâche de "video summarization" ou, en français, de récapitulation de vidéo.

Plus particulièrement, nous focalisons notre recherche sur les vidéos de sport.

Dans ces vidéos, notre objectif va être de parvenir à identifier les replays; car à partir de ces replays, nous serons capables de mettre en avant les moments importants du match.

Pourquoi les replays ? Un replay est la retransmission d'une action qui s'est déjà passée au cours d'une vidéo.

Les replays sont intéressants car ils sont un indicateur d'un moment fort dans une vidéo.

En effet, c'est l'équipe technique chargée du montage de la vidéo qui décide ou non de créer un replay pour une action, un replay est une annotation humaine sur une vidéo.

Typiquement, un replay sera incrusté dans la vidéo après une action importante comme, par exemple, un but ou un pénaltie.

Caractéristique des replays Les replays sont introduits et se terminent par un logo (cite:HaoPan2002).

Ces logos ont en général une apparence qui se démarque facilement des autres images dans la vidéo.

Les replays ne sont pas à confondre avec les ralentis.

Les ralentis sont un type particulier de replays où l'action est montrée de nouveau en *slow-motion*, mais tous les replays ne sont pas des ralentis.

C'est pourquoi, la vitesse de déplacement des objets dans l'image (pour détecter l'effet de *slow-motion*) n'est pas un bon critère pour la détection de replay.

L'objectif de notre recherche est double.

Dans un premier temps, nous mettons en place un système de détection de replays en utilisant uniquement les méthodes "classiques" tels que ORB (cite:Rublec2011).

Ensuite, nous comparons plusieurs approches d'apprentissage automatique pour la tâche de détection dans les vidéos.

Enfin, nous comparons l'efficacité des méthodes de vision par ordinateur "classiques" aux méthodes par apprentissage profond (deep learning).

6 Analyse d'image : état de l'art concernant la détection de replay

L'un des objectifs de notre recherche est de mettre en place un modèle de détection de replays n'utilisant que des techniques ne faisant pas intervenir d'apprentissage automatique.

Ce modèle servira de base de comparaison avec le modèle par apprentissage automatique que nous proposerons par la suite.

Afin d'avoir le meilleur modèle, nous avons comparé plusieurs approches.

La détection de replays est un domaine de recherche à part entière et les articles sont abondants (cite:HaOPan2002,LingYuDuan,Chu2015,Javed2019 ...).

6.1 A Robust Replay Detection

Cette approche cite:xull_{robustreplaydetecalgorsoccervideo} détecte les replays en trouvant les logos dans les vidéo.

Les logos sont trouvés grâce à la luminance. Nous savons qu'un logo est présent pendant 0.8 secondes soit 18 frames pour une vidéo de 24 FPS.

Tous les frames qui ont obtenu un score supérieur à un certain seuil sont considérés comme des candidats pour être le *logo template*.

Le *logo template* est le logo qui représente le mieux les logos dans la vidéo et c'est celui-ci qui va servir de référence pour trouver tous les logos.

6.1.1 Calcul du score de la luminance et filtrage des frames

L'idée est de parcourir toute la vidéo et de calculer pour chaque frame la différence de luminance qu'il y a entre ce frame et les 17 frames précédents.

Nous obtenons un score L_i pour chaque frame i dans la vidéo.

Tous les frames dont le score est inférieur à un certain seuil sont écartés, les autres vont servir à trouver le logo template.

6.1.2 Recherche du logo template parmi les frames filtrés

Le logo template est le frame qui représente le mieux tous les logos dans la vidéo.

Pour déterminer le logo template parmi les frames filtrés, l'algorithme K-means est utilisé pour séparer cet ensemble en deux ($K = 2$) en fonction de

la luminance moyenne des frames.

Pour trouver le logo template, nous allons chercher dans le cluster avec le centre de cluster le plus élevé, puis sélectionner le frame m minimisant la distance avec tous les autres frames du cluster.

$$d_{mn} = \sqrt{\sum_{i=0}^{\text{binsize}} \left[\frac{H_m(i)}{\sum_{j=0}^{\text{binsize}} H_m(j)} - \frac{H_n(i)}{\sum_{j=0}^{\text{binsize}} H_n(j)} \right]^2} \quad (4)$$

Figure 1: Formule de la distance entre deux frames dans le cluster label:calc-dist

6.1.3 Recherche des logos

Une fois que le logo template est déterminé, chaque logo trouvé en précédemment va être comparé avec le logo template.

La mesure de comparaison est la distance (figure ref:calc-dist) qu'il y a entre le frame et le template dans le cluster.

Tous les frames qui ont une distance inférieure à un certain threshold sont considérés comme des logos.

6.1.4 Recherche des replays

Une fois que les logos sont détectés, nous pouvons trouver les replays en cherchant les paires de logos éloignés de moins de 80 seconde (durée maximum d'un replay).

6.1.5 Résultats des auteurs

7 TODO résultat auteurs pour robust replay

7.1 Mean Shift Based video Segment Representation And Applications To Replay Detection

Dans cet article cite:LingYuDuan, les auteurs présentent une méthode permettant de détecter les replays.

L'idée est d'apprendre une base de représentation compressée de logos avec une méthode comme le spectral hashing (cite:NIPS2008₃₃₈₃), puis de se servir de cette base de données pour trouver les logo au début à la fin de replays.

7.1.1 Algorithme

```

L = []
R = []
Segmenter la vidéo en frame
Pour chaque frame f
    Calculer la représentation r_f de f
    Pour chaque représentation r dans la base de représentation:
        Si distance(r_f, r) < Seuil:
            Ajouter f à L
Pour chaque logo l dans L:
    Trouver le logo l' lui correspondant
    Ajouter (l, l') à R

```

La représentation des images est un hash obtenu par un algorithme de hashing d'image (spectral hashing).

La distance utilisée pour comparer le hash des images est la distance de Wasserstein .

7.1.2 Résultats

Match	Total	Correct	False Alarm	Recall	Precision
GER-KOR (25/06/02)	67	65	4	97.0%	94.2%
GER-BRA (30/06/02)	33	30	5	90.9%	85.7%
SEN-TUR (22/06/02)	48	46	4	95.8%	92.0%
SEN-FRA (31/05/02)	54	52	6	96.3%	89.7 %

Figure 2: Performance sur la tâche de détection de replay :label:mean-shift-res

Les résultats obtenus par les auteurs sont présentés dans la Figure ref:mean-shift-res.

Ceux-ci sont bons, mais l'ensemble de test n'est pas assez représentatif (seulement quatres vidéos).

Les avantages des représentations hashées pour les images sont les suivants :

- un hash est compacte (peu d'espace nécessaire pour les stocker)
- comparer des hash est rapide (comparer deux frames)
- chercher un hash dans une table de hachage est rapide (chercher un frame dans une base de données)

Pour ces raisons, cette approche est tout à fait adaptée à la reconnaissance de logo

8 Apprentissage automatique : les bases théoriques

Dans notre recherche, nous allons aborder plusieurs types de réseaux d'apprentissage automatique.

Nous allons présenter dans cette partie les principes fondamentaux à la bonne compréhension de ces derniers.

8.1 Réseaux de neurones récurrents (RNN)

Les RNN (Recurrent Neural Networks), ou réseaux de neurones récurrents (RNR) en français, sont capables de répéter leur couche cachée, en utilisant comme entrée la sortie de toutes les couches précédentes et de générer une sortie pour chaque couche.

Cela va leur permettre de prendre en entrée des séquences et de retourner des séquences.

En effet, pour une entrée $[e_1, e_2, \dots, e_n]$ et un initialiseur s_0 , le RNN va répéter n fois sa couche cachée, de telle sorte à générer une sortie s_1 associée à la couche 1 et à l'entrée (e_1, s_0) ; puis il va générer une sortie s_2 associée à la couche 2 et à l'entrée (e_2, s_1) , etc ...

Pour finir, nous aurons en sortie la séquence $[s_1, s_2, \dots, s_n]$.

Par exemple, appliqués à la génération de phrase, les RNN vont être capables de générer (mot par mot, ou n -gram par n -gram) des séquences de phrases de longueur arbitraire.

Pour apprendre un modèle, le RNN va avoir besoin d'un ensemble d'entraînement qui met en avant les propriétés qui nous intéressent dans le modèle.

La nature récursive de ces réseaux les rend particulièrement adaptés aux tâches de traitement du langage naturel ou pour traiter la temporalité.

8.2 LSTM

Les LSTM (Long Short Term Memory) sont un type de RNN **à portes (gated RNN)**.

Ces portes vont permettre de stocker l'information apprise par le réseau à travers le temps.

À la différence des RNN classiques, les LSTM sont capables d'oublier de

l'information grâce à leur **leaky unit** afin d'éviter une explosion ou une disparition du gradient.

Par exemple, si nous voulons entraîner un LSTM pour qu'il puisse reconnaître une action courte dans une vidéo, ce dernier n'a pas besoin d'enregistrer toutes l'information acquise depuis le premier frame, il lui suffit de connaître un voisinage de quelques frames.

La puissance de ces réseaux à portes est que c'est le réseau qui va apprendre à décider quand vider son état interne.

Concrètement, cela va leur permettre de pouvoir capturer des dépendances à long terme de manière bien plus efficace que les RNN classiques.

8.3 CNN

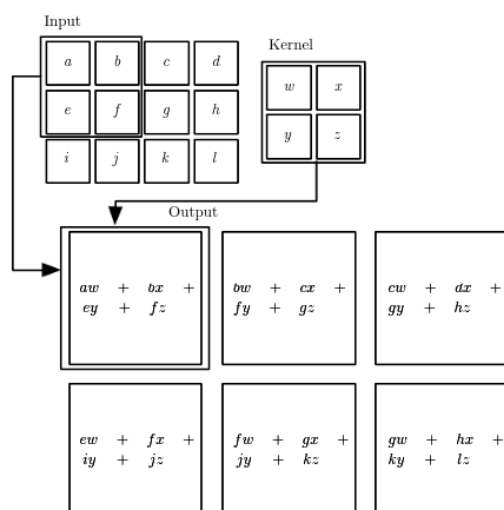


Figure 3: Opération de convolution label:convolution

Les CNN (Convolutional Neural Networks), ou réseaux de neurones convolutifs (RNC) en français, sont un type de réseau de neurones qui utilisent la convolution au lieu de la multiplication matricielle dans au moins une de leurs couches.

La convolution est une opération qui prend en argument l'entrée (typiquement un vecteur représentant une donnée) et un **noyau** (les paramètres qui vont être appris par le CNN) et renvoie une **carte de caractéristiques** (feature map).

Le noyau est une matrice qui va parcourir l'entrée et appliquer l'opération de convolution.

Pour parcourir l'entrée, celle-ci va être divisée en plusieurs matrices carrées de même taille que le noyau (par exemple 2x2 ou 6x6) en ajoutant si nécessaire du *padding* et du *striding*.

La fonction de convolution a trois caractéristiques importantes : l'**interaction parcimonieuse** ("sparse interaction"), le **partage de paramètres** et les **représentations équivariantes**.

La couche de convolution est généralement composée de la fonction de convolution suivie d'une fonction d'activation non linéaire (par exemple, ReLU ou tanh) et d'une fonction de **pooling**.

8.3.1 Interaction parcimonieuse

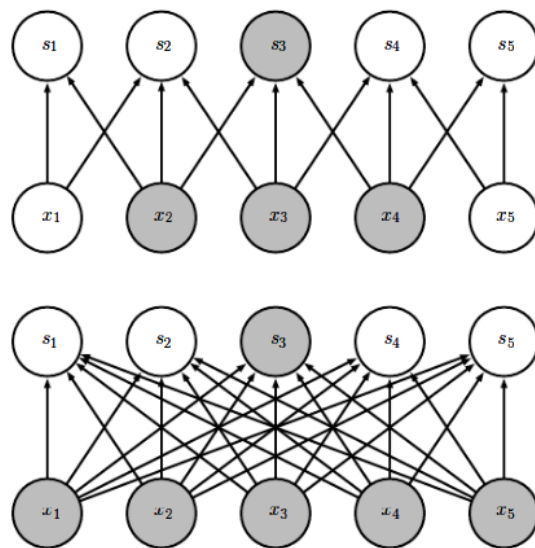


Figure 4: Interaction parcimonieuse (en haut), interaction non parcimonieuse (en bas) label:sparse-vs-dense

À la différence des réseaux classiques où toutes les sorties interagissent avec toutes les entrées, les réseaux à convolution ont des **interactions parcimonieuses**.

C'est à dire que la taille du noyau (donc de l'interaction avec l'entrée), est

plus petite que la taille de l'entrée.

Une image a une dimension en entrée de $c * l * w$ où c est le nombre de canaux de l'image (un seul pour une image en noir et blanc, trois pour une image en couleur), l la largeur en pixel de l'image et w la longueur en pixel de l'image. Une petite image couleur de dimension $100 * 100$ aura $100 * 100 * 3$ paramètres en entrée, ce qui provoque une explosion combinatoire avec les réseaux classiques qui n'ont pas d'interaction parcimonieuse car il faudra une connection entre chaque paramètre d'entrée et une entrée du réseau.

Un réseau de convolution, quant à lui, aura un noyau d'une dizaine ou d'une centaine de pixel qui parcourt l'image à la recherche de caractéristiques significatives comme des contours.

Cela signifie que l'interaction parcimonieuse permet aux CNN de stocker moins de paramètres que les autres types de réseau.

Par conséquent, ils ont donc besoin de moins de mémoire (pour la même tâche) et ont une meilleure efficacité statistique.

C'est l'une des raisons faisant que les réseaux à convolution sont très efficaces pour le traitement d'image.

8.3.2 Partage de paramètres

Dans un réseau classique, un poids (un paramètre) est associé à chaque paramètre d'entrée et ne sert qu'une fois.

Tandis que dans un réseau convolutif, le noyau utilisé par une couche de convolution est le même sur toutes les matrices représentant l'entrée.

Grâce à ce **partage des paramètres**, il n'y a que les poids du noyau à apprendre au lieu d'un poids pour chaque neurone d'entrée.

De plus, la taille du noyau est en général largement inférieure à celle de la couche d'entrée.

8.3.3 Représentations équivariantes

Une fonction est **équivariante** si, quand l'entrée change, la sortie change de la même manière.

En terme mathématique, cela signifie que si $y = f(x)$ alors $g(y) = g(f(x))$. Les réseaux convolutifs sont équivariants à la translation.

Dans le cas de l'image, cela signifie que le déplacement des pixels n'a pas d'influence sur le réseau.

8.3.4 Pooling

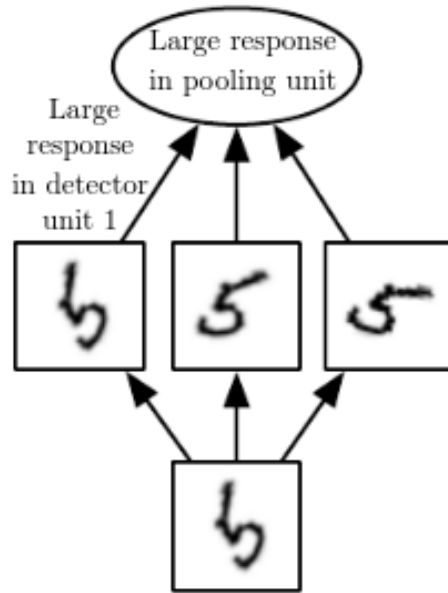


Figure 5: Pooling & invariance label:pooling

La fonction de pooling va modifier la sortie de la couche de convolution. Pour chaque valeur dans la carte des caractéristiques à la sortie de la convolution (après la fonction d'activation), la fonction de pooling va remplacer celle-ci en fonction de la valeur des cases voisines dans la carte.

Une fonction de pooling usuelle est max pooling, qui va renvoyer la plus grande valeur dans un voisinage rectangulaire.

L'utilité de la fonction de pooling est de rendre la représentation apprise par la couche de convolution **invariante** à de petites modifications sur l'entrée. Par exemple, dans le cas de la reconnaissance d'image, le réseau ne va pas chercher dans l'image en entrée les informations au pixel près.

Si le réseau a appris à détecter les visages, il n'a pas besoin de retrouver l'emplacement des yeux au pixel près, une position approximative de ceux-ci lui suffira.

Une autre utilité du pooling est de réduire la taille de la sortie de la couche de convolution.

Nous pouvons voir le pooling comme un résumé de la carte des caractéristiques obtenue par convolution.

9 Apprentissage profond : état de l'art pour la reconnaissance d'action dans les vidéos

Nous nous intéressons à l'état de l'art concernant la détection d'action dans les vidéos.

En effet, la transition d'un logo s'effectue sur plusieurs frames consécutifs; il y a donc une composante temporelle à notre recherche, et nous pouvons considérer la transition d'un logo comme une action.

9.1 Two-Stream Convolutional Networks for Action Recognition in Videos

Cet article est écrit par Karen Simonyan et Andrew Zisserman cite:DBLP:journals/corr/SimonyanZ14. Dans celui-ci, ils proposent de séparer la tâche de reconnaissance d'action dans les vidéos en deux parties : une composante spatiale et une composante temporelle.

La composante spatiale contient l'information concernant les objets dans la vidéo; tandis que la composante temporelle l'information sur les déplacements de ces objets et de la caméra.

A partir de ces observations, les auteurs proposent d'entraîner un classifieur spatial (Spatial stream ConvNet) et un classifieur temporel (Temporal stream ConvNet).

Ces classifieurs sont des réseaux de neurones convolutifs profonds.

9.1.1 Classifieur spatial

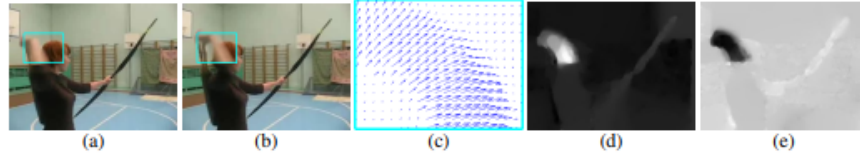
Ce réseau a une architecture de classifieur d'image classique.

Il va permettre de donner un indice fort pour la prédiction, car certaines actions sont très liées à certains objets.

De plus, la recherche dans le domaine de la classification est un domaine à part entière; toutes les avancées dans le domaine augmenteront l'efficacité de ce classifieur.

Il n'est pas nécessaire d'apprendre ce réseau "from scratch" (de zéro), les approches par transfer learning sont efficaces.

9.1.2 Classifieur temporel



Optical flow. (a),(b): a pair of consecutive video frames with the area around a moving hand outlined with a cyan rectangle. (c): a close-up of dense optical flow in the outlined area; (d): horizontal component d^x of the displacement vector field (higher intensity corresponds to positive values, lower intensity to negative values). (e): vertical component d^y . Note how (d) and (e) highlight the moving hand and bow. The input to a ConvNet contains multiple flows (Sect. 3.1).

Figure 6: Flux optique label:optical-flow label:opt-flow

L'innovation de l'article vient de l'introduction du classifieur temporel.

L'idée est de détecter le mouvement des objets dans la vidéo, car un mouvement est la représentation d'un objet dans le temps.

Les auteurs appellent leur approche "optical flow stacking" (empilement de flux optique).

Dans celle-ci, ils utilisent la méthode "optical flow" pour détecter le mouvement des objets entre des frames consécutifs.

Ils définissent aussi un hyperparamètre L qui définit la distance maximum entre deux frames pour lequel il faut calculer le flux optique.

Par exemple, si $L=5$, alors pour le frame t , il faudra calculer le flux entre le frame t et le frame $t+1$; entre $t+1$ et $t+2$; etc... jusqu'à $t+4$ et $t+5$.

Ainsi, des images représentant le flux optique entre les différents frames de la vidéo vont être générées.

Chacunes de ces images servira d'entrée au CNN (classifieur temporel).

9.1.3 Méthode d'évaluation et résultats obtenus

Le classifieur spatial est pré-entraîné avec ImageNet, tandis que le temporel est entraîné de zéro (car il n'y a pas de réseau déjà entraîné pour cette tâche). Les dataset utilisés pour l'entraînement et l'évaluation sont UCF-101 et

Table 4: Mean accuracy (over three splits) on UCF-101 and HMDB-51.

Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	87.9%	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	66.8%
Spatio-temporal HMAX network [11, 16]	-	22.8%
“Slow fusion” spatio-temporal ConvNet [14]	65.4%	-
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	88.0%	59.4%

Figure 7: Résultats obtenus par l’approche Two-stream model label:two-stream-res

HMDB-51, contenant à eux deux près de 20000 vidéos annotées.

Note Pour calculer la classe d’un frame à l’instant t , les auteurs proposent deux méthodes :

- fusion par la moyenne (by averaging) : $y_t = y_{t\text{spatial}} + y_{t\text{temporal}} / 2$
- fusion par SVM (by SVM) : un SVM multiclasse linéaire est entraîné pour prédire la classe à partir du softmax des scores L2-normalisés.

Les résultats (figure ref:two-stream-res) montrent l’efficacité de leur méthode par rapport aux autres approches état de l’art.

Nous pouvons voir que leur approche two-stream avec fusion SVM est la plus efficace sur le dataset UCF-101, et qu’elle a aussi de bons résultats sur HMDB-51.

Ce qui est le plus intéressant dans cet article, c’est l’amélioration qu’apporte l’ajout de la composante temporelle.

En effet, le classifieur d’image simple (spatial) n’a que 73.0% (UCF-101) et 40.5% (HMDB-51), tandis que le classifieur qui prend en compte l’image et la temporalité (two-stream model) atteint **88.0%** et 59.4%; ce qui est une nette amélioration.

Cet article nous a renforcé dans l’hypothèse qu’il est nécessaire d’étudier une vidéo non pas comme une suite d’images indépendantes, mais comme une suite de séquence avec un lien entre chaque élément de la séquence. La temporalité a une très grande importance pour l’analyse de vidéos.

9.2 Learning Spatiotemporal Features with 3D Convolutional Networks

Dans cet article cite:Tran₂₀₁₅, les auteurs proposent une approche pour apprendre les caractéristiques spatio-temporelles dans les vidéos grâce à un réseau de neurones à convolution.

Ils font l’hypothèse qu’un réseau avec une couche de convolution 3D qui prend en entrée une séquence d’images est capable d’apprendre efficacement les mouvements des objets dans les vidéos.

L’objectif est d’apprendre des caractéristiques qui soient :

- génériques : c’est à dire la capacité à représenter différents types de vidéos
- compactes : afin de pouvoir stocker un grand nombre de ces caractéristiques
- efficace (computationnellement): pour traiter les vidéos en temps réel
- simples : afin de fonctionner même avec les modèles simples (comme un classifieur linéaire)

9.2.1 C3D: Convolution et pooling 3D

Les auteurs appellent leur approche C3D (3D ConvNet).

Comparé aux réseaux à convolution 2D, C3D est capable de modéliser plus efficacement l’information spatio-temporelle grâce à la convolution et au pooling sur trois dimensions.

La convolution 2D appliquée à une image produira en sortie une image, la

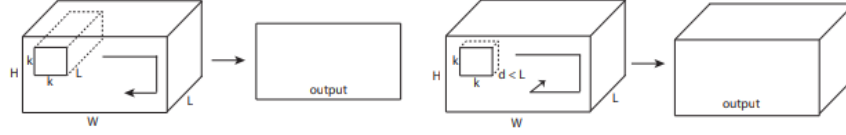


Figure 8: Convolution 2D sur une séquence d'images (gauche), convolution 3D sur une séquence d'images (droite) label:c3d-idea

convolution 2D appliquée à une suite d'images produira aussi une image. C'est pourquoi les réseaux à convolution 2D perdent l'information temporelle après l'opération de convolution. La convolution 3D permet, elle, de préserver cette information.

9.2.2 Architecture et entraînement du réseau

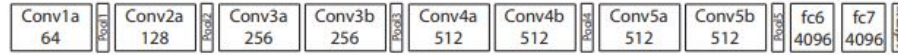


Figure 9: Architecture C3D label:arch-c3d

L'entrée de ce réseau est de dimension $c * l * h * w$ où c est le nombre de canaux des images (3 pour la couleur, 1 pour les images en noir et blanc), l le nombre d'images dans les séquences, h la longueur et w la largeur en pixel des images.

L'architecture conseillée par les auteurs est 8 couches de convolution et 5 couches de pooling, ainsi que 2 couches complètement connectées et la fonction softmax pour la couche de sortie.

Le kernel recommandé par les auteurs est $3 * 3 * 3$ avec un pas (stride) de $1 * 1 * 1$ pour toutes les couches de convolution.

Toutes les couches de pooling sont max pooling avec une taille de kernel $2 * 2 * 2$ (sauf pour la première qui est $1 * 2 * 2$) avec un stride $2 * 2 * 2$ (sauf pour la première qui a un stride de $1 * 2 * 2$).

Pour finir avec l'architecture, les deux couches complètement connectées ont 4096 sorties.

Ce réseau va être entraîné de zéro par descente du gradient à partir de séquences d'images annotées.

Le taux d'apprentissage est de 0.003 et est divisé par 10 toutes les 4 epoch. L'entraînement s'arrête après 16 epoch.

Après l'entraînement, le réseau peut être utilisé comme un extracteur de caractéristiques pour des tâches d'analyse vidéo.

Pour se faire, la vidéo va être découpée en des clips de 16 frames (avec 8 frames de chevauchement entre deux clips consécutifs).

Ensuite, chacun de ces clips va être passés au réseau et l'avant dernière couche complètement connectée (fc6) va contenir les caractéristiques du clip.

Qu'est-ce que ce réseau apprend ? Ce réseau apprend à se focaliser sur l'image des premiers frames, et à traquer leur déplacement dans les frames suivants.

9.2.3 Résultat pour la tâche de reconnaissance d'action

Method	Accuracy (%)
Imagenet + linear SVM	68.8
iDT w/ BoW + linear SVM	76.2
Deep networks [18]	65.4
Spatial stream network [36]	72.6
LRCN [6]	71.1
LSTM composite model [39]	75.8
C3D (1 net) + linear SVM	82.3
C3D (3 nets) + linear SVM	85.2
iDT w/ Fisher vector [31]	87.9
Temporal stream network [36]	83.7
Two-stream networks [36]	88.0
LRCN [6]	82.9
LSTM composite model [39]	84.3
Conv. pooling on long clips [29]	88.2
LSTM on long clips [29]	88.6
Multi-skip feature stacking [25]	89.1
C3D (3 nets) + iDT + linear SVM	90.4

Figure 10: Résultats pour l'approche Learning Spatiotemporal Features with 3D Convolutional Networks (C3D) comparés à d'autres approches état de l'art label:c3d-res

Ces résultats (figure ref:c3d-res) ont été obtenus par les auteurs pour la tâche de reconnaissance d'action sur le corpus de vidéo UCF101.

Nous voyons que l'approche par réseau à convolution 3D est la plus efficace.

9.2.4 Conclusion

Dans cet article, les auteurs ont adressé le problème de la temporalité dans les vidéos.

Ils ont montré qu'un réseau à convolution 3D est capable de modéliser l'information temporelle et spatiale simultanément, et donc d'obtenir de meilleurs résultats que les réseaux à convolution 2D sur plusieurs tâches d'analyse de vidéos.

De plus, cette approche est très élégante car elle fonctionne sans ajout artificiel de caractéristiques (comme le flux optique), c'est le réseau qui va se charger d'apprendre ce dont il a besoin pour apprendre la temporalité.

9.3 Beyond Short Snippets: Deep Networks for Video Classification

Dans cet article cite:JoeYueHeiNg2015, les auteurs proposent d'utiliser une architecture hybride à base de CNN et de RNN (LSTM) pour l'analyse vidéo. Leur objectif est d'apprendre des dépendances à long terme dans les vidéos, d'où l'utilisation d'un LSTM.

Les CNN sont des réseaux particulièrement efficaces pour analyser les frames des vidéos, c'est le CNN qui va se charger de la composante spatiale de la vidéo.

Les LSTM va servir à apprendre la composante temporelle.

9.3.1 Approche

L'objectif des auteurs est d'apprendre des dépendances à long terme dans les vidéos.

Les réseaux à convolution sont très efficaces pour l'analyse d'image; mais leur coût computationnel est très élevé.

C'est pourquoi, il n'est pas possible de se servir d'un CNN pour apprendre les dépendances à long terme; en effet, il faudrait que le réseau prenne en entrée toute la vidéo (ou bien une grande partie), ce qui n'est pas possible avec la puissance de calcul actuelle.

Les auteurs font l'hypothèse que tous les frames dans la vidéo ne sont pas utiles, et qu'il est judicieux de ne garder qu'un sous-ensemble des frames de la vidéo; ils proposent donc de ne traiter qu'un frame par seconde. L'intérêt de ne garder qu'un sous ensemble des images de la vidéo et qu'il va donc être possible de traiter la vidéo sur une plage temporelle plus large pour le

même coût en calcul.

Néanmoins, ne regarder qu'une seule image par seconde dans la vidéo ne préserve pas le déplacement des pixels entre l'instant t et l'instant $t + 1$ (en seconde) et donc l'information du mouvement des objets.

Pour palier à ce problème, le flux optique (de la même manière que cite:DBLP:journals/corr/SimonyanZ14 est calculé entre les frames adjacents.

Ainsi, l'information temporelle et l'information spatiale sont préservées, tout en ne traitant qu'une seule image par seconde, ce qui réduit beaucoup le coût de calcul.

Pour apprendre les dépendances qu'il y a entre les frames, un LSTM est utilisé; celui-ci va traiter les vidéo comme des séquences d'images et va apprendre à prédire la classe de la vidéo en fonction de ces séquences.

9.3.2 Architecture

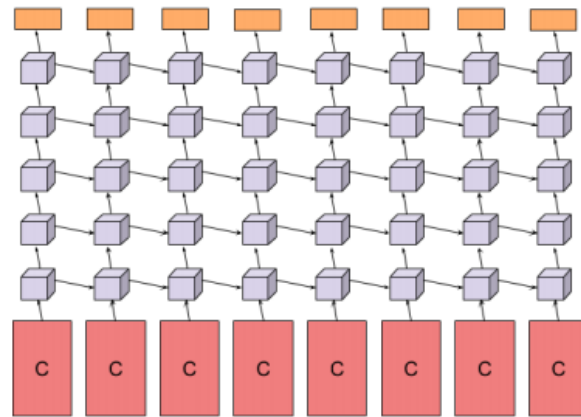


Figure 11: Architecture du LSTM label:arch-lstm

Comme pour cite:DBLP:journals/corr/SimonyanZ14, un réseau à convolution va être utilisé pour extraire les caractéristiques visuelles de la vidéo. Les architectures utilisées pour ce réseau sont GoogLeNet et AlexNet.

L'architecture proposée pour le LSTM est présentée en ref:arch-lstm.

La sortie du CNN est traitée par propagation avant à travers cinq couches de LSTM.

La couche de sortie du LSTM est munie de la fonction softmax et prédit une

classe à chaque étape.

Les paramètres du réseau à convolution et de la couche de sortie du LSTM sont partagés pour toutes les étapes.

9.3.3 Utilisation du flux optique

Le flux optique encode l'information des déplacements des objets dans la vidéo.

Comme pour l'approche cite:DBLP:journals/corr/SimonyanZ14, les images de flux vont être pré-calculées et servir lors de l'entraînement du CNN.

9.3.4 Résultat obtenu

Method	3-fold Accuracy (%)
Improved Dense Trajectories (IDTF)s [23]	87.9
Slow Fusion CNN [14]	65.4
Single Frame CNN Model (Images) [19]	73.0
Single Frame CNN Model (Optical Flow) [19]	73.9
Two-Stream CNN (Optical Flow + Image Frames, Averaging) [19]	86.9
Two-Stream CNN (Optical Flow + Image Frames, SVM Fusion) [19]	88.0
Our Single Frame Model	73.3
Conv Pooling of Image Frames + Optical Flow (30 Frames)	87.6
Conv Pooling of Image Frames + Optical Flow (120 Frames)	88.2
LSTM with 30 Frame Unroll (Optical Flow + Image Frames)	88.6

Figure 12: Résultat obtenu pour l'approche combinant CNN et LSTM (LSTM with 30 Frame Unroll) label:cnn_{lstm}

Les auteurs évaluent leur approche sur le dataset Sports-1M et UCF-101 sur la tâche de classification de vidéos.

Nous pouvons voir que leur approche obtient les meilleurs résultats (88.6% contre 88.0% pour cite:DBLP:journals/corr/SimonyanZ14).

Ces résultats sont intéressants car à la différence des autres approches, celle-ci est capable de prédire une classe pour des morceaux de vidéos pouvant aller jusqu'à deux minutes (contre quelques secondes pour les autres). De plus, les meilleurs résultats sont là aussi obtenus en utilisant le flux optique, confirmant l'hypothèse faite par [cite:DBLP:journals/corr/SimonyanZ14](#) que ce dernier est nécessaire pour le traitement des vidéos.

Dans notre cas, cette approche n'est pas la plus adaptée. En effet, nous souhaitons reconnaître les logos dans les vidéos, or un logo ne dure pas plus longtemps que quelques secondes et les dépendances à long-terme que le LSTM va apprendre ne nous intéressent pas.

Néanmoins, une variante de celle-ci où le LSTM reçoit en entrée une séquence de frames consécutifs (et pas une séquence formée d'un frame par seconde) pourrait avoir de bons résultats pour la tâche de détection de logos.

10 Les approches proposées

L’objectif de notre recherche est de détecter les replays dans les vidéos de sport

Pour détecter les replays, nous faisons les hypothèses suivantes :

- un replay a un logo de début (I)
- un replay a un logo de fin (II)
- les logos de début et de fin sont les mêmes (III)
- les logos ont une forme facilement reconnaissable qui se distingue des autres images dans la vidéo (IV)
- un replay dure entre 2 et 90 secondes (V)

Nous proposons plusieurs approches permettant de détecter les logos de replay dans les vidéos de sport.

En premier lieu, nous proposons deux approches n’utilisant que des algorithmes d’analyse d’images classiques (flouttage, filtre de Canny, ORB, ...) :

- la première se sert de l’algorithme ORB (cite:Ruble2011) et de l’algorithme K-Means
- la seconde utilise la détection de contours pour trouver les images avec des contours similaires dans la vidéo

Ensuite, nous présentons deux approches par apprentissage profond :

- la première utilise un réseau à convolution 3D sur une séquence d’images (similaire à cite:Tran2015)

- la seconde utilise un réseau à convolution 2D sur des images représentant le flux optique des objets dans la vidéo

10.1 Détection des plans

Les approches que nous proposons itèrent sur tous les frames de la vidéo, à la recherche des logo pouvant se trouver au début et à la fin des replays.

Si nous faisons l'hypothèse qu'un replay entraînera toujours un changement de plan, alors au lieu de rechercher les logos parmi tous les frames de la vidéo, nous pouvons réduire la recherche à tous les frames qui sont entre deux plans.

C'est pourquoi nous allons chercher une méthode permettant de détecter les changements de plan dans les vidéos.

10.1.1 Online, Simultaneous Shot Boundary Detection And Key Frame Extraction For Sports Videos Using Rank Tracing

Cette méthode est proposée par W. Abd-Almageed en 2008 cite:AbdAlmageed2008.

Chaque frame est converti en HSV et les histogrammes H, S et V sont calculés.

Un vecteur est formé pour chaque frame à partir de ces histogrammes.

Ensuite, une matrice M de dimension $N * L$, représentant une fenêtre de N frames va être formée à partir de ces vecteurs, où L est la taille des histogrammes et N la taille de la fenêtre.

L'algorithme SVD (singular value decomposition) va être appliqué sur M . $M = U W V$, où W est la matrice de valeurs singulières.

Les diagonales de la matrice W comportent des poids S ordonnés de manière décroissante.

Le premier poids S_1 est le poids maximal. Ces poids représentent l'information contenue dans le vecteur V .

Nous allons assigner un rang à la matrice M , ce rang va être égal au nombre d'éléments s dans S tel que $s/S_1 > \text{threshold}$. Le rang va être calculé pour chaque fenêtre de frames dans la vidéo.

Si le rang d'une fenêtre est plus grand que le rang de la fenêtre avant elle, alors le contenu visuel de la fenêtre est différent de la fenêtre précédente. À l'inverse, si le rang est inférieure à la fenêtre précédente, alors le contenu visuel se stabilise. S'il est de 1, alors c'est stable.

Le début d'un frame est celui qui maximise le rang parmi les fenêtres environnantes.

Cette méthode pour trouver les plans dans une vidéo est très efficace, et nous servira tout au long de la suite de notre recherche.

En effet, avant de segmenter la vidéo en plan, nous comparions N frames, où N peut être aussi grand que 400000 (pour des vidéos de 120 minutes à 60 fps).

Il est impensable d'utiliser un algorithme en $O(N^2)$, par exemple en comparant tous les frames entre eux, avec un N aussi grand.

Après avoir segmenter la vidéo en plan, nous obtenons un N' au alentours de 2000 pour une vidéo de 120 minutes à 60 fps.

Nous pouvons donc nous permettre d'utiliser des algorithmes plus complexes que sans la segmentation en plan.

De plus, la segmentation en plan réduit le champs de recherche des frames logo, et donc le nombre de faux positifs.

10.2 Analyse d'image - Première approche : ORB

Dans cette approche, nous cherchons à reconnaître les logos dans les vidéo.

Pour ce faire, nous optons pour une approche de clustering.

L'idée est de séparer la vidéo en deux groupes : un groupe pour les frames logo, et un autre groupe pour les frames non-logo.

Il existe plusieurs manières de représenter une image, par exemple :

- une matrice $w * l * c$ où c est le nombre de canal (3 pour une image couleur, 1 pour une image en noir et blanc), w et l la largeur et la longueur (resp.) de l'image en pixel

- un histogramme des couleurs RGB
- un histogramme HSV

Néanmoins, aucune de ces méthodes ne permet de vectoriser l'image de tel sorte qu'un algorithme comme K-Means obtienne de bons résultats, ces caractéristiques ne sont pas assez discriminantes.

De plus dans notre cas, nous souhaitons regrouper non pas des frames (images) mais des plans (séquence d'images).

OpenCV implémente l'algorithme KMeans. Celui-ci permet de regrouper les objets similaires en fonction de leurs caractéristiques.

Dans notre cas, il va nous permettre de créer deux groupes d'images : logo / non logo.

L'avantage de KMeans est qu'il est très rapide et assez efficace dans la plupart des cas.

C'est l'un des algorithmes de clusterisation les plus utilisés.

Pour fonctionner, KMeans a besoin que les objets que l'on souhaite séparer en groupe soit sous forme de vecteurs et que ces vecteurs soient discriminants. C'est à dire qu'un frame logo ne doit pas être proche dans l'espace d'un frame non-logo.

Dans cette partie, nous utilisons l'algorithme ORB pour extraire des caractéristiques à partir pour chaque plan dans la vidéo.

10.2.1 Première expérimentation : 1 frame par shot

Ici, nous ne récupérons les caractéristiques que d'une seule image par plan.

L'algorithme est le suivant :

- Récupérer le frame à la fin de chaque shot
 - nous obtenons $|S|$ frame
- Pour chaque frame, calculer ses features (orb ou akaze)
 - Nous obtenons $|S|$ vecteurs

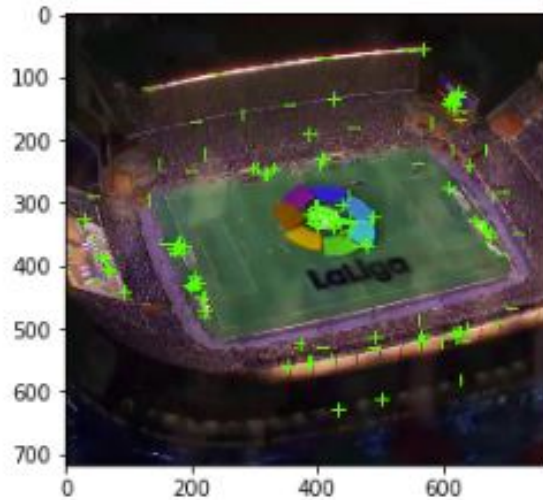


Figure 13: Caractéristiques extraites par ORB (en vert), pour une seule image par plan

- Utiliser KMeans avec $K=2$ pour séparer les vecteurs en deux groupes
 - le groupe le plus petit est le groupe des logo

10.2.2 Deuxième expérimentation : 1 fenêtre de frame par shot:

L'approche précédente ne récupère qu'une seule image par plan.

Or, les logos sont présents sur plusieurs frames consécutifs.

Dans cette approche, nous récupérerons plusieurs images consécutives pour chaque plan, et nous extrayons les caractéristiques pour chaque frame (voir figure ref:orb-window).

L'algorithme est le suivant :

- Récupérer W frames pour chaque shot, les regrouper en une fenêtre
 - nous obtenons $|S|$ fenêtres de dimension W , où W est le nombre de frame
- Pour chaque fenêtre, calculer les features de chacun de ses frames (ORB)
 - Nous obtenons un vecteur de dimension $|S*W|$
- Utiliser KMeans avec $K=2$ pour séparer les vecteurs en deux groupes
 - le groupe le plus petit est le groupe des logo

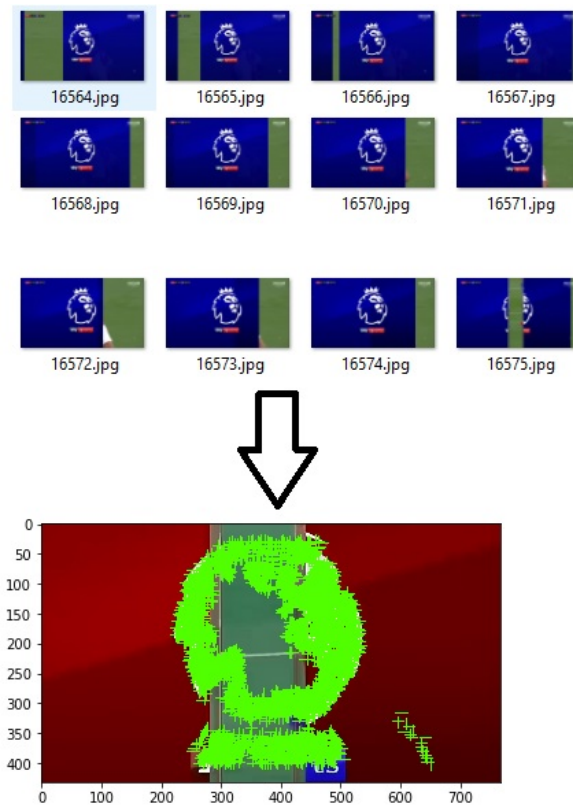


Figure 14: Caractéristiques extraites par ORB (en vert) pour un plan (aggrégat des caractéristiques de chaque image dans le plan) label:orb-window

10.3 Analyse d'image - Seconde approche : matching de contours

Dans cette approche, nous allons chercher les frames qui ont des formes en commun dans la vidéo.

En effet, d'après l'hypothèse III, il est fort probable que si un frame à l'instant t a beaucoup de formes en commun avec un frame à l'instant t' , avec $2 < t' - t < 90$ (hypothèse V), alors il y a un logo à l'instant t et un logo à l'instant t' , et un replay entre t et t' .

Dans cette méthode, nous utilisons là aussi le découpage en plan (cite:AbdAlmageed2008) pour réduire la zone de recherche des logos de début et de fin des replays.

L'idée est de chercher, pour chaque plan, si il existe un autre plan dans son voisinage tel qu'ils ont des contours en commun dans plusieurs de leurs frames.

10.3.1 Mosaïque de plan

Pour des raisons d'optimisation, il n'est pas viable de comparer chaque frame de chaque plan avec chaque frame des plans voisins.

C'est pourquoi, pour chaque shot deux images sont générées.

Chaque image est de dimension $I * I * \text{width} * \text{height}$ où I est le nombre de frames dans le shot.

Ces images sont des matrices d'images qui vont permettre de comparer rapidement deux shot.

La première matrice a un décalage d'un frame par ligne, la seconde n'a pas de décalage.

Pour comparer deux shot (figure ref:mosaic), il suffit d'appliquer un ET binaire entre les matrices des mosaïques, puis de calculer la longueur des contours dans cette matrice (les images sont des matrices).

10.3.2 Algorithme

- Pré traitement sur les shots

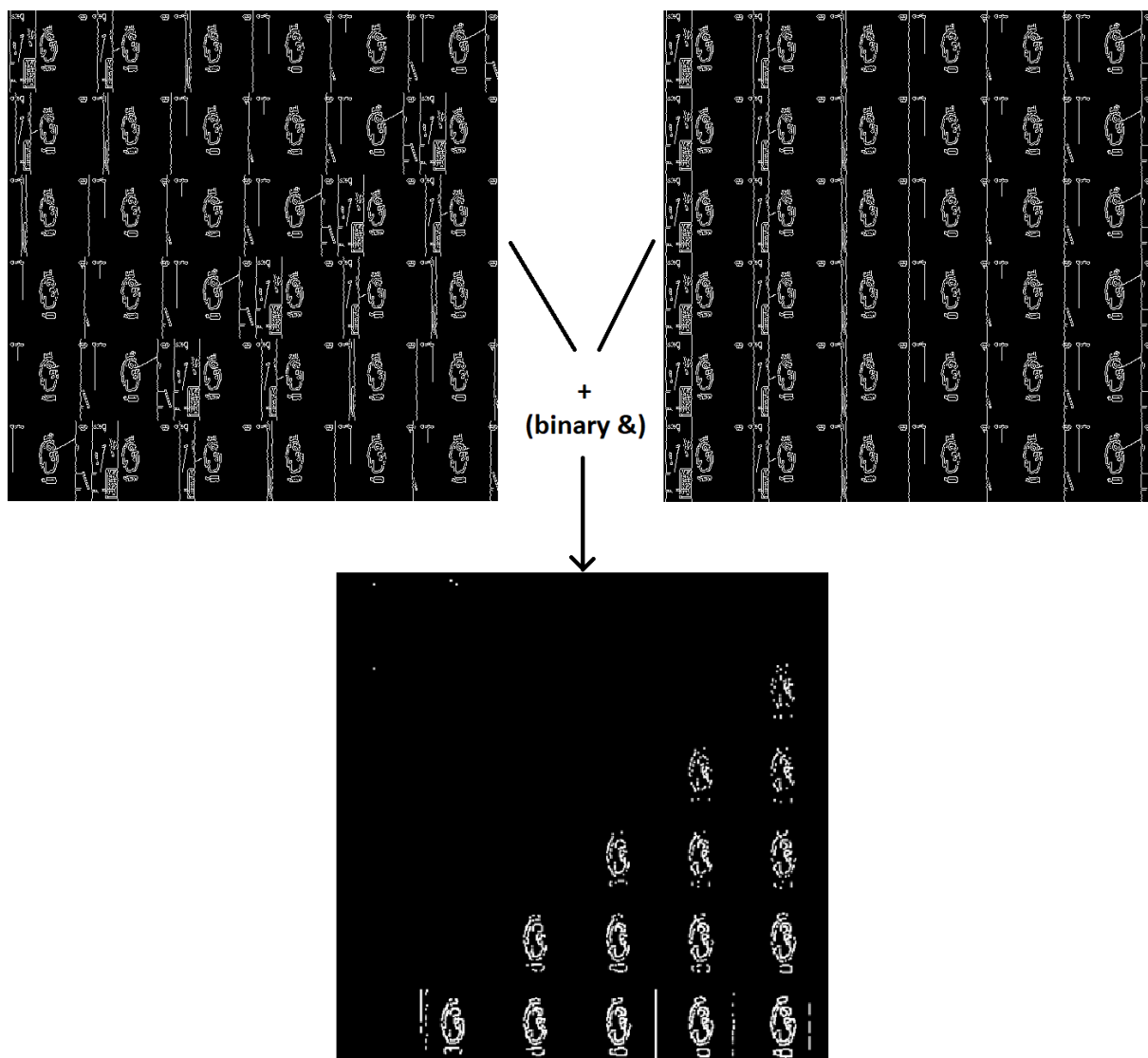


Figure 15: ET binaire (en bas) appliqué à une mosaïque de plan à l'instant t (à gauche) et à une mosaïque de plan à l'instant t' (à droite) label:mosaic

1. Redimensionner
 2. Cropper
 3. Supprimer le background (s'étendre la dessus)
 4. Détecter le contour (Canny Edge Detection)
 5. Génération des mosaïques
- Pour chaque mosaïque de plan S_A :
 - Pour chaque mosaïque de plan S_B après S_A :
 1. Contour_commun = C_A & C_B
 2. Contours_diff = Détection du contour de Contour_commun (cv2.findContours)
 3. Résultat = Ne garder que les contours qui sont assez longs ($|\text{contours}| > K$)
 4. Si Résultat > Seuil : alors S_A et S_B sont des logos potentiels
 - Pour chaque logo potentiel LP :
 1. Le comparer avec les autres logo L' (même procédure qu'en 2)
 2. Si au moins 2 logo L' match, alors LP est un logo
 - Trouver les replays grâce aux logos

Notre algorithme est sensible au plan fixe et aux images avec beaucoup de bruits (ces images ont beaucoup de contours détectés par l'algorithme de détection de contours).

Beaucoup de ces faux-positifs peuvent être filtrer lors du pré-traitement sur les plans, notamment en rajoutant du blur ou en supprimant le background, néanmoins, nous ne sommes pas parvenus à filtrer 100% des faux-positifs.

10.4 Apprentissage profond - Première approche : convolution 3D

Dans cette approche, nous allons implémenter une méthode similaire à cite:Tran₂₀₁₅. L'idée va être d'entraîner un réseau à convolution avec une couche de convolution 3D.

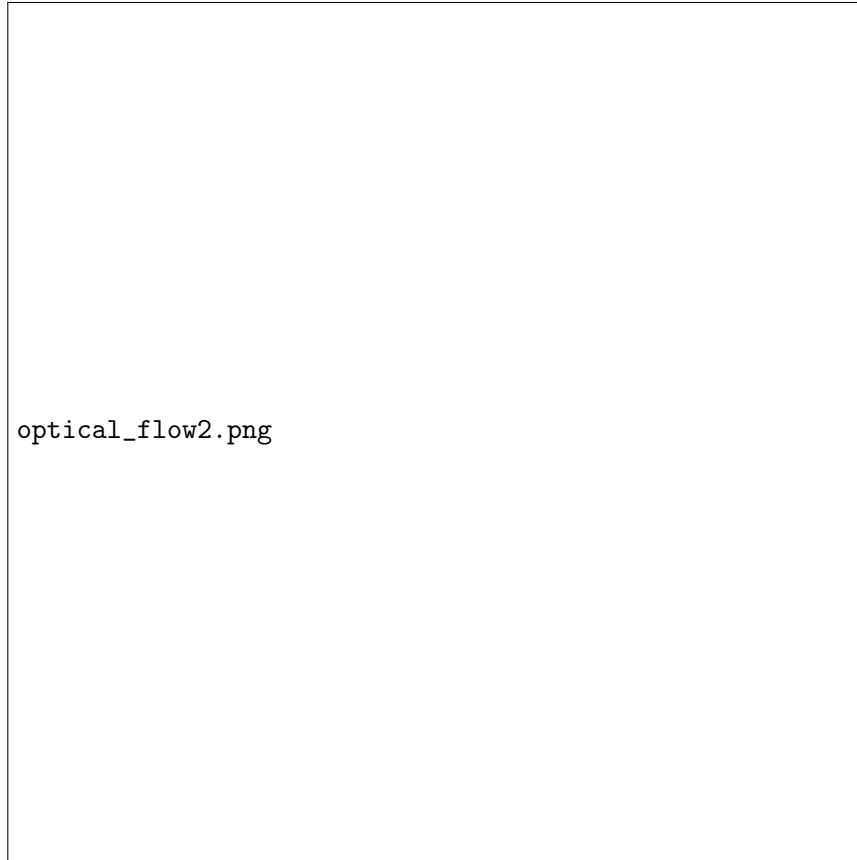
Cette couche va prendre en entrée une séquence d'images et va prédire une classe logo ou non-logo.

Pour réduire la zone de recherche, seulement les séquences d'images entre deux plans (détectées avec cite:AbdAlmageed₂₀₀₈) seront traitées.

Lors de l'entraînement, le réseau va apprendre à partir de séquences d'images labellisées (logo/non-logo).

Comme pour cite:Tran₂₀₁₅, nous pensons que la convolution 3D va permettre au réseau d'apprendre l'apparence des logos mais aussi leur animation dans la vidéo.

10.5 Flux optique



Le flux optique décrit le déplacement des objets entre deux frames consécutifs, ce mouvement peut être causé par un déplacement de la caméra ou de l'objet lui-même.

Le flux optique est une matrice $X * Y * D$, où X et Y sont l'axe des abscisses et des ordonnées (resp.) dans l'image, et D une droite décrivant le déplacement du pixel à la position (x,y) entre le frame t et le frame $t + 1$.

De la même manière que pour `cite:wang11action`, nous allons traquer le déplacement des objets grâce au flux optique.

Il existe deux types d'algorithmes de calcul du flux optique :

- sparse/creux : seulement les déplacements de certains points d'intérêt vont être traqués (méthode de Lucas-Kanade)

- dense : le déplacement de tous les points dans l'image vont être traqués (algorithme de Gunner Farneback)

Le calcul du flux optique dense est plus couteux mais à l'avantage de ne pas nécessiter de déterminer les points d'intérêt dans l'image avant de calculer le flux optique.

10.6 Apprentissage profond - Deuxième approche : convolution 2D sur des images de flux optique

Dans cette approche, nous allons entraîner un classifieur d'image de flux optique.

L'objectif est de capturer l'information du mouvement des objets dans des séquences d'images grâce au flux optique, puis de projeter cette information dans un plan 2D (donc des images) afin de pouvoir entraîner un classifieur à partir de ces représentations 2D.

Là aussi, nous n'allons traiter que les frames qui sont entre deux plans (cite:AbdAlmageed2008).

10.7 Apprentissage profond - Troisième approche : convolution 3D sur des images de flux optique

11 Collecte des données & entraînement

11.1 Datasets

L'objectif de la collecte de données est d'obtenir les ensembles de données suivants :

- Dataset non logo
- Dataset logo

Chaque élément du dataset contient :

- une séquence de 20 images représentant la séquence (pour le classifieur d'image à convolution 3D)
- une séquence 38 images représentant une seule image (avec 38 canaux) de flux optique par la méthode cite:DBLP:journals/corr/SimonyanZ14 (pour le classifieur de flux optique à convolution 2D)
- une image représentant la somme des flux optiques (pour le classifieur de flux optique à convolution 2D)
- une séquence 19 images représentant (pour le classifieur de flux optique à convolution 3D)

11.2 Collecte des données

L'approche par matching de contours convient tout à fait pour former notre ensemble de données de logo.

En effet, elle est :

- rapide : une vidéo au format 100x100 de 200000 frames va être traitée en moins de cinq minutes, et en moyenne une cinquantaine de logos (séquence d'images pendant laquelle un logo apparaît) sont extraits par vidéo
- précise : il est possible de modifier les paramètres pour que l'algorithme ne renvoie presque pas de faux-positifs ($>1\%$)

11.3 Obtention des images de flux optiques

Pour obtenir les images de flux optiques, nous allons utiliser notre algorithme de détection de logo par matching de contours afin d'obtenir les séquences d'images comprises entre deux plans.

Pour chacune de ces séquences, nous allons calculer son flux optique. Le problème est qu'il n'est possible de ne calculer le flux optique qu'entre deux

frames; or, une séquence s'étale sur plus que deux frames (dans notre cas, 20 frames).

Afin d'obtenir une seule image pour le flux optique de la séquence, il existe plusieurs méthodes.

11.3.1 Image de flux optique : 2L canaux

Dans cite:DBLP:journals/corr/SimonyanZ14, les auteurs proposent de représenter le flux optique d'une séquence d'images comme une image avec 2L canaux, où L est le nombre d'images dans la séquence. Les canaux pairs représentent le déplacement vertical des objets, et les canaux impairs le déplacement horizontal. Ainsi, chaque image qui servira d'entrée au classifieur sera de dimension $w*h*2L$ où w et h sont la longueur et la largeur en pixels et L le nombre d'images dans la séquence. Ces images n'auront pas un contenu visuel identifiable pour l'humain, mais seront comprises par un réseau à convolution. Concrètement, cela revient à stocker deux images de flux pour chaque image de la séquence; la première image est l'information horizontal de déplacement, la seconde l'information vertical de déplacement.

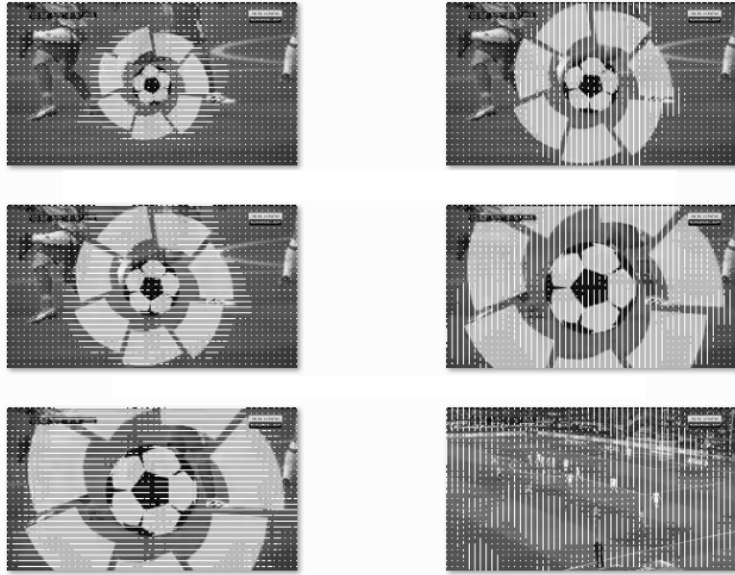


Figure 16: Obtention du flux optique pour une séquence, avec séparation du flux vertical (droite) et horizontal (gauche) label:flux-trajectory

11.3.2 Image de flux optique : somme des fluxs

Une autre méthode pour obtenir une image de flux optique à partir d'une séquence de flux optique est de calculer la somme de tous les flux optiques.



Figure 17: Obtention du flux optique pour une séquence par somme de tous les fluxs de la séquence

11.4 Architecture du scrapper

Nous voulons une architecture de scrapping qui soit distribuée.

En effet, le traitement d'une vidéo prend en moyenne cinq minutes, il est donc nécessaire de pouvoir traiter plusieurs vidéos en même temps.

Pour mettre en place cette architecture, nous avons créé une image Docker englobant le programme de détection de logos par matching de contours et qui upload les frames des séquences logo ainsi que leur flux optique sur la solution de stockage de GCP (storage).

L'image est stockée sur le Container Registry de GCP, et va être lancée dans un container.

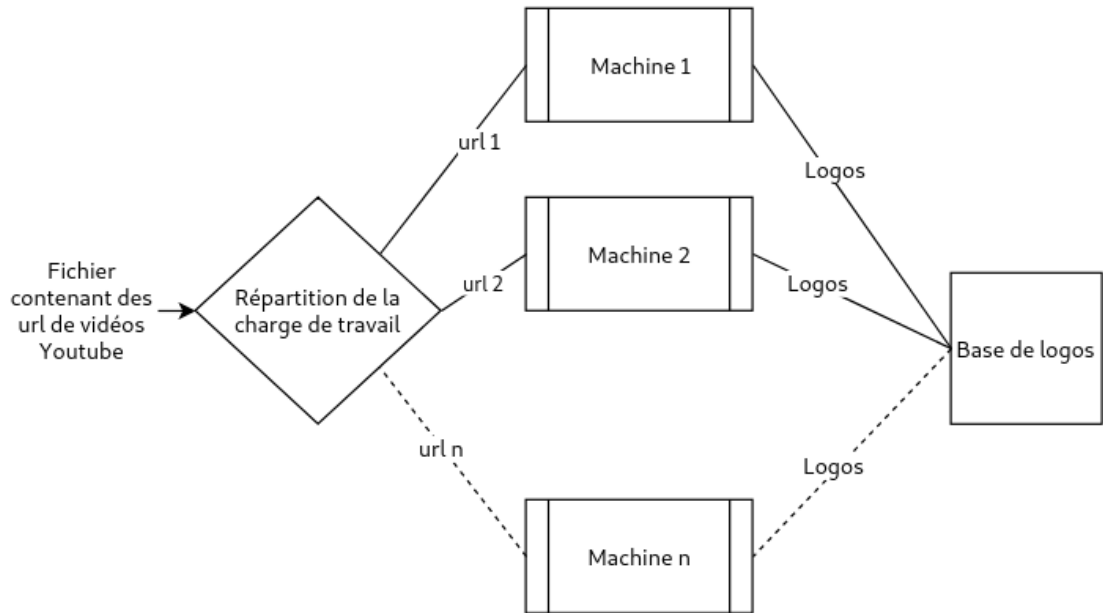


Figure 18: Architecture proposé pour le scrapper de logos

12 Comparaison des approches

Dans cette partie, nous allons évaluer chaque approche sur le même ensemble d'évaluation.

Puis nous allons comparer les résultats obtenus et enfin nous allons conclure sur l'efficacité des différences approches.

12.1 Procédure d'évaluation

L'ensemble d'évaluation est le suivant : une vidéo de ligue 1 de 150k frames, une vidéo de ligue 1 de 80k frames, une vidéo de liga et enfin une vidéo de premier league.

Dans toutes les expérimentations, la vidéo est découpée en plans (plan). Soit S l'ensemble des plans.

Feature extracted by ORB on :	Football:PL 150k frames 48 logos	Football:Ligue 1 80k frames 16 logos	Football:Ligue 1 150k frames 34 logos	Football:Liga 300k frames 36 logos	Tennis:AusOpen 400k frames 32 logos
1 frame / shot	29.73% 68.75%	14.94% 40.63%	30.86% 73.53%	19.19% 52.78%	11.36% 46.88%
1 frame window / shot	55.10% 84.38%	31.51% 71.88%	34.67% 76.47%	34.62% 75.00%	14.29% 53.13%
Process time	2700 s	1600 s	2700 s	4000 s	5300 s

Score : Precision (left), Time (middle, in italic), Recall (right)

Figure 19: Résultats obtenus pour le clustering avec les caractéristiques extraites par ORB label:orb-res

12.2 Résultats pour ORB

Les résultats (ref:orb-res) sont mauvais sur toutes les vidéos.

Le gros inconvénient de ces méthodes est le caractère arbitraire de la sélection du groupe de logo après le clustering.

En effet, nous faisons l'hypothèse qu'il y aura toujours plus de non-logo que de logo, ce qui n'est pas toujours vrai.

De plus, l'algorithme d'extraction de caractéristiques ne marche pas toujours très bien, ce qui fausse évidemment le clustering par la suite.

Néanmoins, il est intéressant de remarquer que les résultats avec la fenêtre de frame sont toujours meilleurs que les résultats avec un seul frame.

12.3 Résultats pour la détection par luminance

	Video					
Video width / Video height / Luminance threshold	Football : Premier League (150k frames)	Football: Ligue 1 (80k frames)	Football: Ligue 1 (150k frames)	Football: Liga (300k frames)	Tennis: Australia Open (400k frames)	American football: NFL (*) (200k frames)
100,100, 50000	>100, 5/48, 150	45, 0/16, 70	>100, 0/34, 200	>100, 0/36, 150	>100, 12/32, 280	X
100,100, 75000	>100, 12/48, 150	21, 0/16, 70	58, 0/34, 200	43, 0/36, 150	>100, 12/32, 280	X
100,100, 100000	52, 32/48, 150	7, 0/16, 70	16, 0/34, 200	23, 12/36, 150	53, 15/32, 280	X
100,100, 125000	12, 48/48, 150	4, 0/16, 70	5, 0/34, 200	13, 36/36, 150	33, 32/32, 280	X
100,100, 150000	5, 48/48, 150	2, 2/16, 70	2, 3/34, 200	4, 36/36, 150	23, 32/32, 280	X

Score : False Positive (FP), False Negative (FN), Time (rounded)

(*) Didn't work because in NFL games, the logo at the and at the beginning of a replay do not match

Après avoir implémenté la méthode de détection par luminance (cite:xu11_{robustreplaydetectionalgorithmsoccervideo}), nous avons constaté que celle-ci n'est pas efficace et ne fonctionne pas sur notre ensemble d'évaluation.

Cette approche est trop dépendante de la luminance et elle ne parvient pas à détecter les logos peu lumineux.

De plus, les bases mathématiques (notamment la manière de choisir le cluster et la mesure de distance) sont quelque peu douteuses.

Enfin, celle-ci dépend trop du paramètre "seuil de luminance" affectant les logos détectés.

Le seuil de luminance fournis par les auteurs ne produit pas de bons résultats sur toutes les vidéos.

Nous n'avons pas réussi à trouver une valeur pour le seuil de luminance qui obtienne universellemnt de bons résultats.

Un seuil à 100000 détecte les logos de Ligue 1 mais pas les logos de Liga.

Un seuil de 75000 détecte les logos de Liga et de Ligue 1, mais laisse passer trop de faux positifs.

Les logos de Premier League quant à eux ne sont pas tous détectés avec un seuil à 50000, alors que ce seuil accepte un grand nombre de faux positifs.

Pour conclure, cette approche n'est pas celle qui va nous permettre de mettre en place un système de détection de replays robuste et efficace.

12.4 Résultats pour l'approche par matching de contours

Minimum score / Video width / Video height / Window size / Minimum logo segment length / Number of frame before removed / Number of frame after removed / Crop ratio	Football : Premier League 150k frames 48 logos	Football: Ligue 1 80k frames 16 logos	Football: Ligue 1 150k frames 34 logos	Football: Liga 300k frames 36 logos	Tennis: Australia Open 400k frames 32 logos
With delete BG + dilate contour	<i>200 seconds</i>	<i>100 seconds</i>	<i>150 s</i>	<i>300 s</i>	<i>200 s</i>
5000,100,100,5,11,2,2,0.64	100.00%	100.00%	93.75%	100.00%	65.63%
5000,100,100,5,11,0.4,0.64	100.00%	100.00%	93.75%	100.00%	65.63%
	100.00%	100.00%	100.00%	100.00%	85.71%
				71.88%	88.24%
				71.88%	94.12%
Without delete BG + Gauss contour / mosaic	<i>340 s</i>	<i>160 s</i>	<i>150 s</i>	<i>290 s</i>	<i>350 s</i>
10000,100,100,10,85,0,0,0.81 (gaussian blur mosaic + contourDiff)	97.96%	50.00%	100.00%	62.50%	94.12%
	100.00%	56.25%	100.00%	62.50%	94.12%
With delete BG + gauss contour					
Gauss contour / mosaic	<i>300 s</i>	<i>180 s</i>	<i>150 s</i>	<i>290 s</i>	<i>300 s</i>
+ group shot by seconds (rounded)					
+ deleteBG = 0.8 sec					
10000,100,100,10,10,1,1,0.81	95.92%	55.56%	97.14%	62.86%	97.06%
	97.92%	62.50%	100.00%	68.75%	97.06%
With delete BG + gauss mosaic					
+dilate (2) contourDiff	<i>300 s</i>	<i>280 s</i>	<i>330 s</i>	<i>320 s</i>	<i>600 s</i>
+ deleteBG = 0.8 sec					
10000,100,100,10,20,1,1,0.81	97.96%	100.00%	91.89%	75.00%	97.06%
	100.00%	100.00%	100.00%	75.00%	97.06%
With delete BG + gauss mosaic					
+dilate (2) contourDiff	<i>430 s</i>	<i>260 s</i>	<i>640 s</i>	<i>260 s</i>	<i>1620 s</i>
+ deleteBG = 0.8 sec					
+ group shot by half-second					
+ saveWindowSize relative to fps					
X*1000,100,100,X,20,1,1,0.81	97.96%	100.00%	91.89%	75.00%	91.89%
X*1500,100,100,X,10,1,1,0.81	93.75%	93.75%	85.29%	86.11%	100.00%
		100.00%	100.00%	68.75%	100.00%

Score : Precision (left), Time (middle, in *italic*), Recall (right)

Figure 20: Résultats obtenus label:res-match-contour

Les résultats obtenus sont présentés en ref:res-match-contour.

Après avoir fait plusieurs essais avec des paramètres différents à chaque fois, nous ne sommes pas parvenus à trouver une combinaison telle qu'il n'y ait aucun faux positif et aucun faux négatif.

Néanmoins, si le seuil d'acceptance des logos est assez élevé, l'algorithme ne retourne aucun faux positif.

Ceci nous servira par la suite pour obtenir des séquences d'images logo dans les vidéos.

Concernant le temps d'exécution, celui-ci est relié presque entièrement à la taille de la vidéo donnée en entrée, ainsi qu'à la taille des mosaïques.

Les limitations de notre méthode sont les suivantes :

- Dans certaines vidéos, il n'y a pas de logo pour les replays (simple fondu)
- Dans certaines vidéos, les logo de début et fin de replay ne sont pas les mêmes.
- Dans certains vidéos, il y a des logo au début des replays, mais pas de logo à la fin des replays (un simple fondu remplace le logo).

Cette méthode présente les avantages suivants :

- elle est rapide
- très peu de faux positifs

Néanmoins, elle n'est pas adaptative.

En effet, elle n'est pas capable de détecter les replays si le logo n'est pas le même au début qu'à la fin; car elle n'apprend pas ce qu'est un logo, elle sait juste trouver les images avec des contours similaires.

Puisque cette méthode est rapide et ne renvoie presque pas de faux positifs, elle nous servira par la suite pour créer un ensemble de données d'image avec des logo de replay.

Cette méthode servira aussi de *baseline* pour évaluer l'efficacité des modèles par apprentissage profond.

13 Appendice

- Clustering : procédé permettant de regrouper des éléments
- Histogramme : représentation d'une image en fonction de ses canaux de couleurs (rouge, vert, bleu)
- Frame : une image à l'instant t d'une vidéo
- Shot : un plan
- FPS : frame per second / image par seconde
- Cropper : sélectionner une partie continue des pixels l'image
- RNN : Recurrent Neural Networks, ou réseaux de neurones récurrents (RNR) en français
- CNN : Convolutional Neural Network, réseaux de neurones convolutifs (RNC) en français
- LSTM : Long Short Term Memory

14 Table des figures

ref:calc-dist Xu, W., & Yi, Y., A robust replay detection algorithm for soccer video, IEEE Signal Processing Letters, 18(9), 509–512 (2011). <http://dx.doi.org/10.1109/lsp.2011.2161287>. Equation (4)

ref:mean-shift-res Duan, L., Xu, M., Tian, Q., & Xu, C., Mean shift based video segment representation and applications to replay detection, 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, (), (). <http://dx.doi.org/10.1109/icassp.2004.1327209>. Table 1

ref:sparse-vs-dense Goodfellow, I., Bengio, Y., & Courville, A., Deep Learning (2016), : MIT Press. cite:Goodfellow-et-al-2016 Figure 9.2

ref:optical-flow Simonyan, K., & Zisserman, A., Two-stream convolutional networks for action recognition in videos, CoRR, abs/1406.2199(), (2014). Figure 2

ref:flux-trajectory Wang, H., Klaser, A., Schmid, C., & Liu, C., Action recognition by dense trajectories, In , CVPR 2011 (pp.) (2011). : . Equation 2

ref:c3d-res Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M., Learning spatiotemporal features with 3d convolutional networks, 2015 IEEE International Conference on Computer Vision (ICCV), (), (2015). <http://dx.doi.org/10.1109/iccv.2015.510>. Table 3

ref:opt-flow Simonyan, K., & Zisserman, A., Two-stream convolutional networks for action recognition in videos, CoRR, abs/1406.2199(), (2014). Figure 2

ref:two-stream-res Simonyan, K., & Zisserman, A., Two-stream convolutional networks for action recognition in videos, CoRR, abs/1406.2199(), (2014). Table 4

ref:arch-c3d Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M., Learning spatiotemporal features with 3d convolutional networks, 2015 IEEE International Conference on Computer Vision (ICCV), (), (2015). <http://dx.doi.org/10.1109/iccv.2015.510>. Figure 3

ref:arch-lstm Ng, J. Y., Hausknecht, M., Vijayanarasimhan, S., Vinyals,

O., Monga, R., & Toderici, G., Beyond short snippets: deep networks for video classification, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (), (2015). <http://dx.doi.org/10.1109/cvpr.2015.7299101>. Figure 4

ref:convolution Goodfellow, I., Bengio, Y., & Courville, A., Deep Learning (2016), : MIT Press. Chapitre 9. Figure 9.1

ref:pooling Goodfellow, I., Bengio, Y., & Courville, A., Deep Learning (2016), : MIT Press. Chapitre 9. Figure 9.9

ref:c3d-idea Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M., Learning spatiotemporal features with 3d convolutional networks, 2015 IEEE International Conference on Computer Vision (ICCV), (), (2015). <http://dx.doi.org/10.1109/iccv.2015.510>. Figure 1

ref:cnn_lstm Ng, J. Y., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G., Beyond short snippets: deep networks for video classification, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (), (2015). <http://dx.doi.org/10.1109/cvpr.2015.7299101>. Table 7

bibliographystyle:unsrt

bibliography:summary.bib