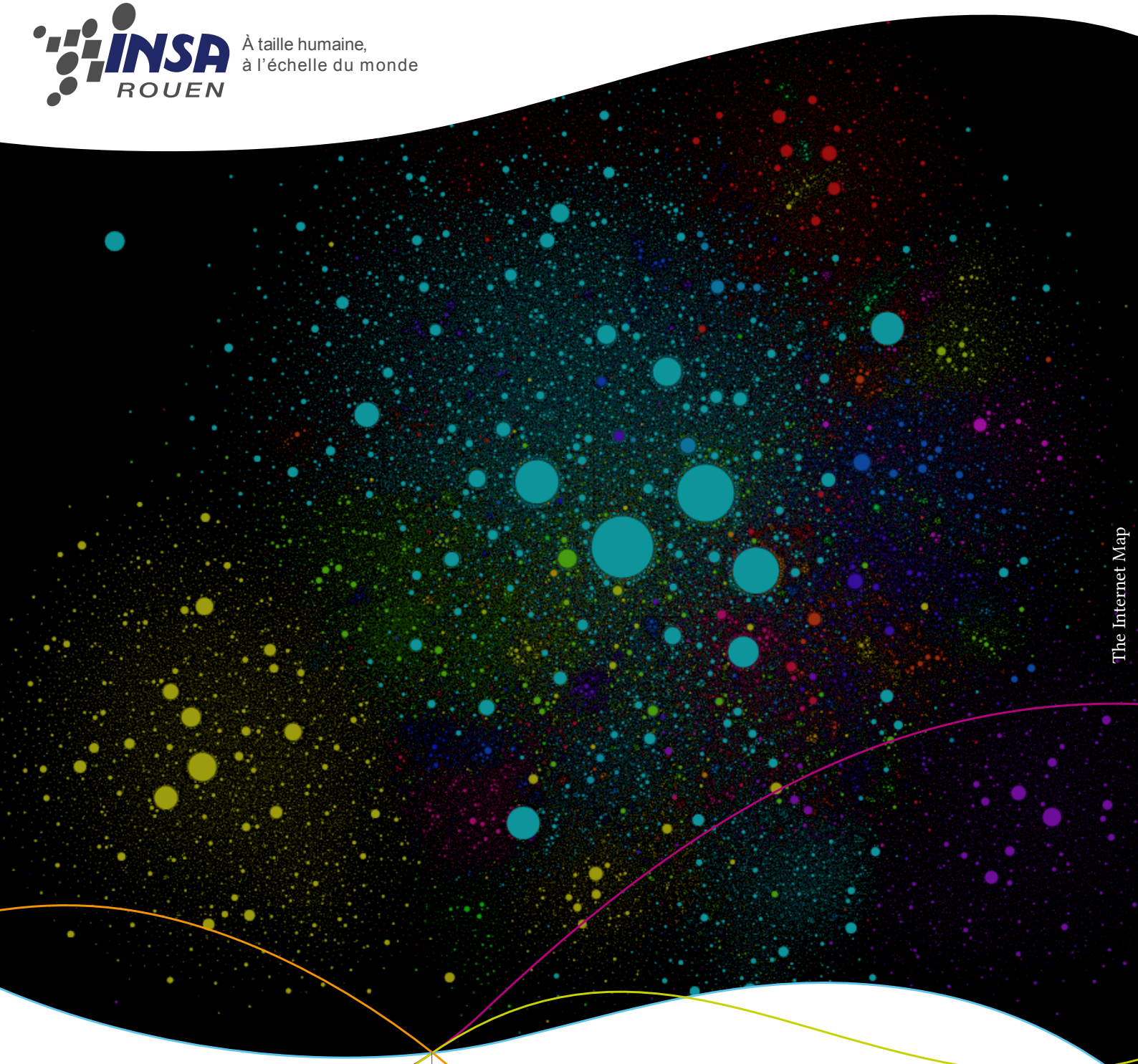
INSA ROUEN
À taille humaine,
à l'échelle du monde



The Internet Map

Projet d'EDTS

Institut National des Sciences
Appliquées de Rouen

Ya Huang
Rémi Mussard
Thomas Robert

Détection de rupture dans une vidéo

Rapport de projet d'EDTS

Table des matières

1 Introduction

Ce projet a pour but de détecter des ruptures de scènes dans une vidéo.

Pour ce faire, une phase d'extraction de caractéristiques des images sera faite. Tout d'abord, nous choisirons diverses caractéristiques afin de pouvoir les comparer au travers des différentes images de la vidéo. Quatre caractéristiques ont été retenues pour la comparaison des images.

Une fois les caractéristiques définies, nous passerons à l'étape de détection en elle même. Cette étape permettra de définir les méthodes permettant de comparer les caractéristiques des images, afin de vérifier s'il y a une rupture (changement de scène) dans la vidéo ou pas. Deux méthodes seront présentées.

Et enfin, une implémentation de la détection de rupture sera faite sous Matlab.

2 | Extraction de caractéristiques

2.1 Différents types de rupture

Ce projet a pour but de détecter les différents types de rupture listés ci-dessous :

- Changement d'image ;
- Fondu d'image.

2.1.1 Changement d'image

Un changement d'image est construit avec deux images qui viennent de deux scènes différentes, sans fusion entre ces deux scènes :

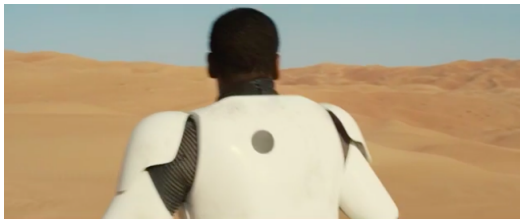


FIGURE 2.1 – Image avant la rupture

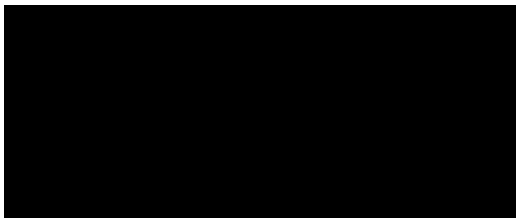


FIGURE 2.2 – Image après la rupture

2.1.2 Fondu d'image

Un fondu d'image est construit avec deux images qui viennent de deux scènes différentes, mais il y a fusion entre ces deux scènes :

FIGURE 2.3 – Image avant la rupture

FIGURE 2.4 – Image après la rupture

2.2 RGB

L'extraction de RGB utilise les trois couleurs de base pour construire une image ou une vidéo : Rouge, Vert et Bleu.

RGB est un paramètre de base d'une image, à l'aide de statistiques du nombre d'occurrences des différentes densités de couleur dans chaque frame (image), la méthode p-valeur peut être mise en place pour détecter les ruptures dans une vidéo.

2.2.1 Matrice RGB

Pour ce projet, une matrice de taille $768 \times NombreFrame$ est construite. Les 768 lignes contiennent le nombre d'occurrences des 256 densités des 3 différentes couleurs.

2.2.2 Niveau de Gris

Pour améliorer l'efficacité, le niveau de gris peut être utilisé aussi. Ce paramètre est obtenu en calculant la moyenne des trois différentes couleurs. En revanche cette solution peut réduire la précision de la détection.

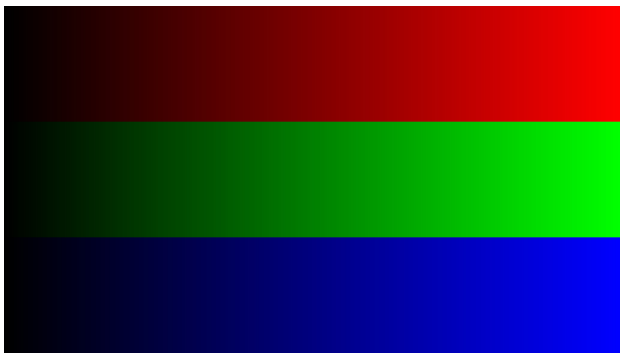


FIGURE 2.5 – Image après la rupture

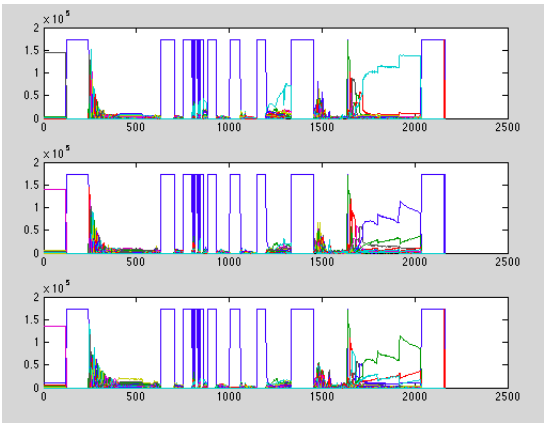


FIGURE 2.6 – Image avant la rupture

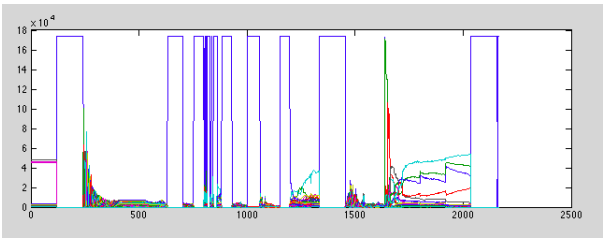


FIGURE 2.7 – Image après la rupture

2.3 YCbCr

YCbCr et Y'CbCr sont deux choses différentes. Pour ce projet, YCbCr est utilisé.

YCbCr est une manière de représenter l'espace colorimétrique en vidéo. YCbCr n'est pas un espace de couleur absolu, c'est une version compressée et biaisée de YUV (Y : la luminance, U et V : la chrominance). Y représente la luminance, qui représente la densité de la lumière. Cb et Cr sont les deux informations de chrominance. Cb est (Y - bleu), Cr est (Y - rouge).

2.3.1 Utilisation informatique

YCbCr est utilisé pour les images JPEG. Par comparaison avec RGB, ce modèle permet de réduire la taille d'une image. Il peut aussi augmenger l'efficacité de la détection de rupture.

Pour faire la conversion RGB vers YCbCr il faut utiliser les formules suivantes :

$$\begin{cases} Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \\ Cb = -0.1687 \times R - 0.3313 \times G + 0.5 \times B + 128 \\ Cr = 0.5 \times R - 0.4187 \times G - 0.0813 \times B + 128 \end{cases}$$

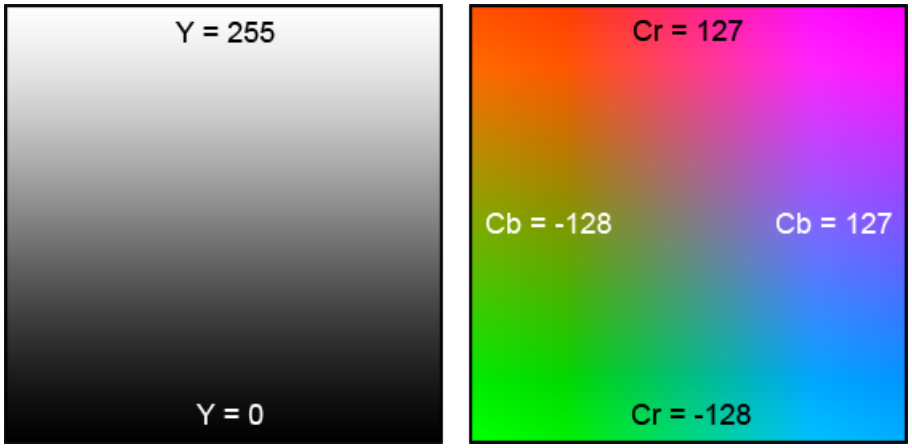


FIGURE 2.8 – YCbCr

Pour faire la conversion YCbCr vers RGB il faut utiliser les formules suivantes :

$$\begin{cases} R{=}Y + 1.402 \times (Cr - 128) \\ G{=}Y - 0.34414 \times (Cb - 128) - 0.71414 \times (Cr - 128) \\ B{=}Y + 1.772 \times (Cb - 128) \end{cases}$$

2.3.2 Point fort

YCbCr est une méthode similaire à RGB, mais propose une solution plus efficace. À l'aide de YCbCr, nous pouvons aussi augmenter efficacité de la détection de rupture.

2.4 Texture

Dans le cadre de la détection de rupture, une autre caractéristique intéressante à exploiter est la texture des images. Différentes approches sont possibles pour représenter mathématiquement une texture. Pour ce projet, nous nous sommes intéressés à une approche statistique, avec la matrice de cooccurrence.

2.4.1 Définition d'une texture

En traitement d'image, on défini une texture comme étant *une région spatiale d'une image présentant une organisation spatiale homogène des niveaux de luminance.* Une texture consiste en une reproduction de motifs de base dans l'espace et dans plusieurs directions. Ces motifs (ou primitives) sont organisés les uns par rapport aux autres de manière aléatoire.

D'après cette définition, on peut considérer une texture comme une structure à deux dimensions :

- La première dimension représente les primitives formant la texture ;
- La seconde dimension représente l'organisation spatiale de ces primitives entre elles.

On recense deux types de texture, les macrotextures et les microtextures.

- Une **macrotexture** est une texture structurée, pour laquelle on peut extraire facilement un motif de base et les lois d'assemblage des primitives entre elles. Par exemple, la texture d'un mur de briques est une macrotexture ;
- Une **microtexture** est une texture aléatoire, avec un aspect désorganisé, mais qui donne une

FIGURE 2.9 – Sable non zoomé

FIGURE 2.10 – Sable zoomé

impression visuelle relativement homogène. On peut prendre l'exemple d'une vue aérienne sur un champs.

Cependant, il est parfois difficile de classer une texture dans une catégorie ou l'autre. En fonction de la résolution de l'image ou du zoom effectué sur celle-ci, on va préférer classer la texture dans l'une ou l'autre des catégories. On peut prendre l'exemple du sable, dans le cas où aucun zoom n'est effectué (2.9), il est tentant de classer cette texture en tant que macrotexture, alors que dans le cas où un zoom est effectué (2.10), il est plus probable de classer cette texture en tant que microtexture.

Il est important de retenir qu'une texture est une région d'une image pour laquelle il est possible de définir une fenêtre telle que pour toute translation de cette dernière, on retrouve le même motif. Une texture peut être décrite spatialement ou statistiquement. C'est cette dernière solution que nous tâcherons d'expliquer dans le cadre de ce projet, avec la matrice de cooccurrence.

2.4.2 Matrice de cooccurrence

En analyse de texture, cette méthode statistique donne des résultats relativement bons. Cette méthode consiste à compter le nombre de paires similaires de pixels ayant une distance d entre eux. Cette méthode se base sur le niveau de gris des pixels, allant de 1 à 8. On prend également en compte la direction de la paire de pixels avec un angle θ. Cette angle varie généralement entre 0 et $180degres$. Les pixels d'une paire sont généralement séparés par une distance $d = 1$ pixel.

Si on prend les paramètres suivants, $d = 1, \theta = 0$, on obtient une paire de pixel comme suit :

$$\begin{cases} pix_1{=}(x_1, y_1) \\ pix_2{=}(x_2, y_2) \end{cases}$$

avec $x_2 = x_1 + 1$ et $y_2 = y_1$.

Le pixel pix_2 est donc le voisin de droite du pixel pix_1. La matrice de cooccurrence est une matrice carrée $n \times n$, avec n le niveau de gris (égale à 8 dans notre cas). Elle va recenser le nombre de pixels ayant un niveau de gris $n2$, situé à la droite d'un pixel dont le niveau de gris est $n1$. Les lignes de la matrice de cooccurrence représentent les pixels de référence et les colonnes, les pixels situés à droite du pixel de référence. Soit l'image I dont les pixels ont les niveaux de gris suivants :

$$I = \begin{pmatrix} 1 & 8 & 1 & 8 \\ 2 & 1 & 1 & 8 \\ 5 & 4 & 3 & 7 \end{pmatrix}$$

La matrice de cooccurrence de I sera donc :

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

La valeur $C(1, 8) = 3$ signifie qu'il y a trois fois un pixel de niveau de gris 8 à la droite d'un pixel de niveau de gris 1.

2.4.3 Détection de rupture et matrice de cooccurrence

Voyons à présent en quoi la matrice de cooccurrence permet de détecter des ruptures dans une vidéo. Dans une vidéo, deux images consécutives seront relativement proches l'une de l'autre, excepté en cas de rupture. La matrice de cooccurrence va permettre d'obtenir une description de la texture de l'image. Entre deux images d'une même scène, la texture ne devrait pas changer et on devrait avoir deux matrices de cooccurrence très proches. Il faut rappeler que la texture d'une image est basée sur la répétition de motifs, et donc, dans une scène, on devrait retrouver les mêmes motifs. Dans notre cas, dans une même scène, les niveaux de gris ne devraient pas énormément évoluer d'une image à l'autre.

La méthode des matrices de cooccurrence possède certains avantages, mais également quelques inconvénients.

2.4.4 Avantages

Les avantages de cette méthode sont les suivants :

- Simple à mettre en oeuvre ;
- Relativement robuste aux changements d'intensité d'une couleur ;
- Relativement robuste aux transitions avec un effet de fondu.

2.4.5 Inconvénients

Les inconvénients de cette méthode sont les suivants :

- Peu robuste aux changement d'angles de caméra ;
- Peu robuste à l'apparition de nouveaux objets sur une scène.

2.5 Flux Optique

Et enfin, pour détecter des ruptures dans une vidéo, on va se baser sur le flux optique et plus particulièrement, sur le gradient des images.

2.5.1 Définition du flux optique

Le flux optique (ou flot optique) représente le mouvement apparent des objets présents sur une image. Cette méthode permet d'estimer le mouvement entre deux images consécutives, à l'instant t et $t + \Delta t$, à partir de la variation de la luminance. La luminosité est supposée constante entre deux images consécutives (séparées par un temps Δt) pour un même pixel, qui se serait déplacé de $(\Delta x, \Delta y)$. Ainsi, on obtient la relation suivante qui doit être vérifiée pour la grande majorité des pixels : $I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$, avec I, l'intensité lumineuse.

En supposant que le mouvement est petit et constant, on peut écrire la précédente équation sous forme différentielle, ce qui nous donne :

$$\frac{dI(x, y, t)}{\Delta t} = \frac{\partial I}{\partial x}.\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}.\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} = 0$$

Soit :

$$I_x.V_x + I_y.V_y = -I_t$$

avec V_x et V_y, le flux optique (ou les composants (x, y) de la vitesse) et I_x, I_y et I_t les dérivées partielles de l'intensité de l'image.

On se retrouve ainsi, avec une équation à deux inconnues (V_x, V_y), que l'on ne peut résoudre telle quelle. Différentes méthodes existent pour résoudre cette équation, en ajoutant diverses contraintes. Avant de voir deux de ces méthodes, voyons d'abord comment estimer la dérivé de l'image.

2.5.2 Matrice gradient

Pour calculer le gradient de l'intensité d'une image, on va se servir du filtre de Sobel. Ce filtre sert en traitement d'image à détecter des contours. Pour ce faire, le gradient de l'intensité de chaque pixel est calculé. Cela indique la direction de la plus forte variation du clair vers le sombre.

Le filtre de Sobel se base sur des matrices de convolution. Une matrice subit une convolution avec l'image, afin de calculer une estimation des dérivées horizontales et verticales. En règle générale, on utilise les matrices de convolution suivantes :

$$C_1 = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \text{ et } C_2 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Ainsi, on va calculer les dérivées partielles de l'image, de tel sorte que G_x et G_y représentent respectivement, les approximations en x et en y :

$$G_x = C_1 * Img \text{ et } G_y = C_2 * Img$$

avec Img la matrice intensité de l'image.

Les gradients horizontaux et verticaux peuvent être combinés pour avoir une approximation de la norme du gradient G :

$$G = \sqrt{G_x^2 + G_y^2}$$

Dans ce projet, c'est de cette caractéristique dont nous nous sommes servis, la norme du gradient, à savoir G. D'autres filtres que Sobel existent pour calculer une estimation du gradient d'une image. Ces autres filtres se basent également sur des matrices de convolution, de taille diverse, avec des valeurs différentes.

2.5.3 Détermination du flux optique par la méthode de Lucas-Kanade

Connaissant maintenant les dérivées partielles de l'image, nous pouvons maintenant chercher à résoudre l'équation du flux optique. Le principe de la méthode de Lucas-Kanade est le suivant : Au lieu d'estimer le déplacement d'un seul pixel, on va chercher à estimer la vitesse d'un groupe de pixels voisins. Cette méthode suppose que le déplacement au voisinage d'un pixel p est petit et approximativement constant, entre deux images consécutives. Ainsi, l'équation du flux optique peut être considérée vraie pour tous les pixels situés dans une région centrée sur le pixel p. On passe d'une équation à deux inconnues à un système de n équations à deux inconnues. n étant le nombre de pixel composant la région choisie. On peut ainsi écrire notre système de la manière suivante :

$$\begin{cases} I_x(p_1).V_x + I_y(p1).V_y = -I_t(p_1) \\ I_x(p_2).V_x + I_y(p2).V_y = -I_t(p_2) \\ \vdots \quad = \quad \vdots \\ I_x(p_n).V_x + I_y(pn).V_y = -I_t(p_n) \end{cases}$$

avec $p_1..p_n$ les pixels situés dans la région choisie.

Ce système est surdéterminé. Il est possible de le résoudre avec la méthode des moindres carrés :

$$V = (A^\intercal A)^{-1} A^\intercal b$$

avec

$$A = \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix}, V = \begin{pmatrix} V_x \\ V_y \end{pmatrix} \text{ et } b = \begin{pmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{pmatrix}$$

2.5.4 Corrélation de phase

Une seconde méthode pour déterminer le flux optique consiste à utiliser la corrélation de phase. Cette méthode utilise le domaine fréquentiel pour avoir une estimation de la translation observée entre deux images consécutives. Cette méthode suppose que l'on dispose de deux images à comparer.

Pour être plus précis avec cette méthode, on va calculer le flux optique sur des régions de l'image et pas sur l'image entière (comme dans la méthode précédente). Soit les images img_1 et img_2. On va passer ces images dans le domaine fréquentiel grâce à la transformée de Fourier :

$$G_1 = fft(img_1), G_2 = fft(img_2)$$

On calcule ensuite le spectre de puissance croisé en faisant une multiplication terme à terme de la transformée de Fourier de l'image 1 et du conjugué de la transformée de Fourier de l'image 2. On normalise ensuite ce produit :

$$R = \frac{G_1 \circ G_2^*}{|G_1 \circ G_2^*|}$$

Suite à quoi, on repasse dans le domaine temporel en appliquant une transformée inverse de Fourier :

$$r = fft\_inv(R)$$

A partir de la transformée inverse, on recherche le maximum. Les indices en x et en y de ce pic correspondent aux translations observées en x et en y :

$$(\Delta x, \Delta y) = argmax_{(x,y)}(r)$$

2.5.5 Détection de rupture et flux optique

Le flux optique, qui représente le mouvement des objets d'une image, va permettre de comparer deux images consécutives d'une séquence vidéo. Dans une même scène, les mouvements des objets seront continus et donc, entre deux images consécutives, on aura un mouvement quasiment constant. Dans le cas d'une rupture, les deux images auront un flux optique suffisamment différent pour détecter cette rupture.

Cependant, dans notre cas, les résultats ont été meilleurs avec la matrice gradient. Ceci est sans doute dû au fait que le gradient permet de détecter les contours dans une image. Ainsi, d'une image à l'autre, sur une même scène, on retrouve les mêmes contours. Alors que pour le flux optique, lorsqu'il y a une rupture, l'estimation de ce dernier est faussée et on ne détecte pas forcément la rupture.

2.5.6 Avantages

Les avantages de cette méthode sont les suivants :

- Relativement robuste aux mouvements ;
- Relativement robuste aux changements d'intensité de couleur (comme le bleu du ciel).

2.5.7 Inconvénients

Les inconvénients de cette méthode sont les suivants :

- Peu robuste aux transitions avec un effet de fondu ;
- Peu robuste lors de l'arrivé ou de la disparition d'un objet dans la scène.

2.6 Récapitulatif sur les caractéristiques

Comme il a été dit auparavant, chacune des caractéristiques extraites des images ont leurs avantages et leurs inconvénients. En fonction du type de rupture à détecter, il sera préférable d'utiliser l'une ou l'autre des caractéristiques, bien que certaines des caractéristiques présentées ici sont très proches et détecte le même type de rupture. Cependant, dans un cas plus général, on ne connaît pas *a priori* les types de rupture qui composent une vidéo. Il conviendrait donc de combiner ces différentes caractéristiques, afin d'avoir un spectre de détection plus grand. Toutefois, il faut prendre en compte les faux-positifs. En effet, certaines des caractéristiques ont tendance à détecter des ruptures là où il n'y en a pas.

Le choix des caractéristiques est donc important pour la détection de rupture dans une vidéo.

3 | Détection de rupture

3.1 Introduction

Cette partie présentera les deux méthodes de détection de rupture implémentées : une méthode par dérivée filtrée utilisant la p-valeur décrite par [Bertrand11] et [Herault14], et une méthode de détection de changement de noyau décrite par [Desobry05].

Cette détection de rupture sera appliquée sur une matrice de signal X contenant n lignes d'« observations », une par image de la vidéo, et p colonnes de caractéristiques, 1 pour chaque caractéristique extraite de la vidéo.

> *Ce chapitre est accompagné de démonstrations, qui prennent la forme de blocs de code Matlab dans le fichier* Projet.m. *Des paragraphes identiques à celui-ci indiqueront quels blocs de code correspondent aux sections du rapport.*

3.2 Prétraitement des caractéristiques

Avant de se lancer dans la détection, il peut être intéressant de regarder ce que contient la matrice de caractéristiques X. On se propose de la visualiser sous la forme d'une image représentée par la figure 3.1a.

Notons que cette première matrice contient des blocs de caractéristiques superposés verticalement (RGB puis YCbCr puis coocurence, etc.), et que chacun de ces blocs a été normalisé entre 0 et 1 afin que les caractéristiques soient comparables entre elles.

On peut voir sur la figure qu'à première vue, la matrice a l'air quasiment creuse, comme si tout valait 0. En réalité, cette perception est due au fait que quelques composantes de chacune des caractéristiques prend des très grandes valeurs. Par exemple sur une image noire, certaines caractéristiques prennent des très grandes valeurs comparativement aux autres.

On se propose donc de prétraiter cette matrice X par une fonction de normalisation appliquée à chaque valeur de la matrice.

> *Les blocs 1 & 2 de la démonstration permettent de tester diverses fonctions de normalisation dans une figure interractive.*

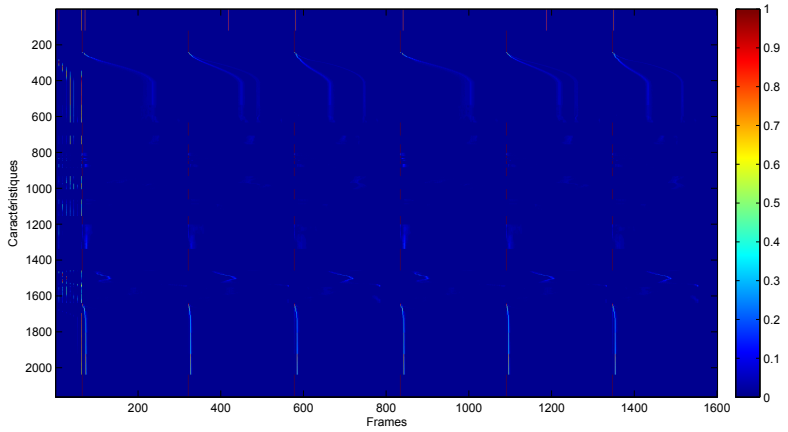
Finalement, on retiendra la fonction de normalisation $\log(10^9 X + 1)$ donnant le résultat visible en figure 3.1b, sur laquelle les valeurs des caractéristiques apparaissent bien plus clairement.

3.3 Principe de la dérivée filtrée

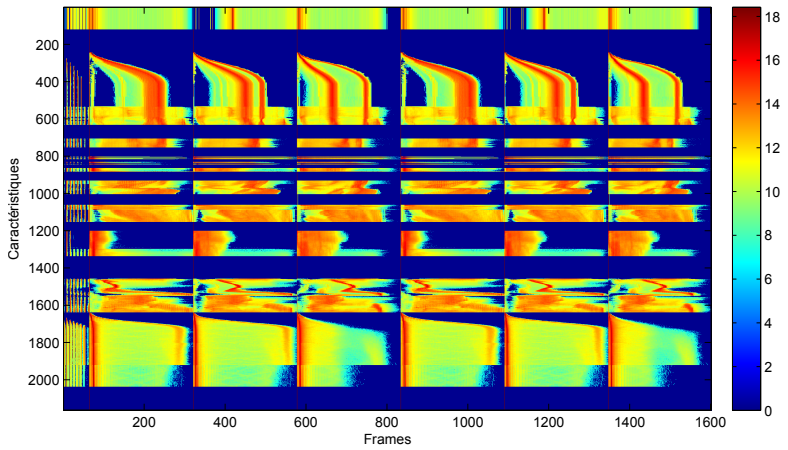
3.3.1 Méthode de la dérivée filtrée

Le principe de la méthode utilisée est de parcourir le signal temporel X avec deux fenêtres de taille A_1 et A_2 respectivement avant et après une position k (de $k - A_1$ à $k - 1$ et de k à $k + A_2 - 1$).

On calcule ensuite, pour chaque position k, une valeur de distance $D(k)$ entre des indicateurs $\hat{\theta}$ calculés sur chaque fenêtre.



(a) Matrice X originale



(b) Matrice X prétraitée

FIGURE 3.1 – Matrices X avant et après prétraitement

On détermine ensuite les zones de rupture comme étant les zones telles que $|D(k)| > C$, avec C un seuil fixé, puis on détermine pour chaque zone de rupture $\mathcal{K}_i$ la position k_{r_i} exacte de la rupture comme étant $k_{r_i} = \arg\max_{k \in \mathcal{K}_i} |D(k)|$.

3.3.2 Calcul de C de façon probabiliste

Afin de déterminer le seuil C, il est possible d'utiliser une méthode probabiliste utilisant la p-valeur. En posant $M = \max_k |D(k)|$, et l'hypothèse $\mathcal{H}_0$ où il n'y a pas de rupture, on veut C tel que $\mathbb{P}(M > C \mid \mathcal{H}_0) < p$ avec p fixé.

Pour estimer C, on calcule des estimations de réalisation de M sous l'hypothèse $\mathcal{H}_0$ en permutant le signal temporel X (cela force en quelque sorte $\mathcal{H}_0$) et en calculant $\hat{M}$ sur ce nouveau signal. Ces réalisations $\hat{M}$ permettent d'estimer la distribution de M. En supposant qu'il s'agisse d'une loi normale, on peut estimer ses paramètres $\hat{\mu}$ et $\hat{\sigma}$, et on peut ainsi estimer le seuil C en fonction d'une valeur fixée de p.

3.4 Changement de noyau

La méthode de détection de changement de noyau est un cas particulier de la méthode de la dérivée filtrée où la distance $D(k)$ est calculée comme étant une distance entre les paramètres ρ_i et α_i de deux *one-class SVM* calculés respectivement sur chaque fenêtre (les $\hat{\theta}$ estimés sur chaque

fenêtre).

Dans ce cas, pour chaque valeur de k, la distance est la suivante :

$$
\begin{aligned}
D &= \frac{\widehat{c_1 c_2}}{\widehat{c_1 p_1} + \widehat{c_2 p_2}} \\
\widehat{c_1 c_2} &= \arccos\left(\frac{\alpha_1^\top K_{12} \alpha_2}{\sqrt{\alpha_1^\top K_{11} \alpha_1}\sqrt{\alpha_2^\top K_{22} \alpha_2}}\right) \\
\widehat{c_i p_i} &= \arccos\left(\frac{\rho_i}{\sqrt{\alpha_i^\top K_{ii} \alpha_i}}\right)
\end{aligned}
$$

avec i et j le numéro du *one-class SVM*, α_i les variables duales, ρ_i le biais et K_{ij} la matrice de Gram avec les données i et j.

3.5 Implémentation

Ces méthodes de détection de rupture ont été appliquées à la première bande-annonce du film *Star Wars 7*. Cette section présentera les résultats de détection de rupture obtenus sur la matrice de caractéristiques prétraitée de cette vidéo, visible en figure 3.1b.

3.5.1 Distances

La méthode de la dérivée filtrée a donc été essayée avec deux distances et estimateurs différents :

1. Somme des différences absolues entre moyennes ($\hat{\theta}_i = \bar{X}_{k,A_i}$, $D(k) = \sum |\hat{\theta}_1 - \hat{\theta}_2|$)
2. Différence de noyaux

Ces distances seront nommées *distance moyenne* et *distance SVM* par la suite.

3.5.2 Choix des fenêtres A_i

Les premiers paramètres à fixer sont des tailles des fenêtres avec lesquelles on parcourt le signal.

Pour cela, nous avons choisi de réaliser une figure interactive permettant de visualiser la distance obtenue avec la *distance moyenne*, en fonction des fenêtres. Ce résultat est affiché sur un graphique indiquant les vraies ruptures dans la vidéo (étiquetées manuellement) afin de pouvoir comparer la distance aux ruptures.

Les fenêtres ont donc été choisies manuellement en deux passes. Tout d'abord on fait varier A_1 et on impose $A_2 = A_1$. Puis on fixe A_1 avec la valeur choisie précédemment (dans notre cas $A_1 = 20$) et on fait varier A_2 afin de choisir la meilleure valeur (dans notre cas $A_2 = 28$).

> *Le code permettant de choisir A_1 correspond au bloc 3 de la démonstration, et celui de A_2 au bloc 4.*

3.5.3 Choix de C

Normalement, il est possible de choisir la valeur C par la méthode présentée précédemment. Cependant, dans notre cas, compte tenu du nombre de caractéristiques et d'images dans la vidéo, la

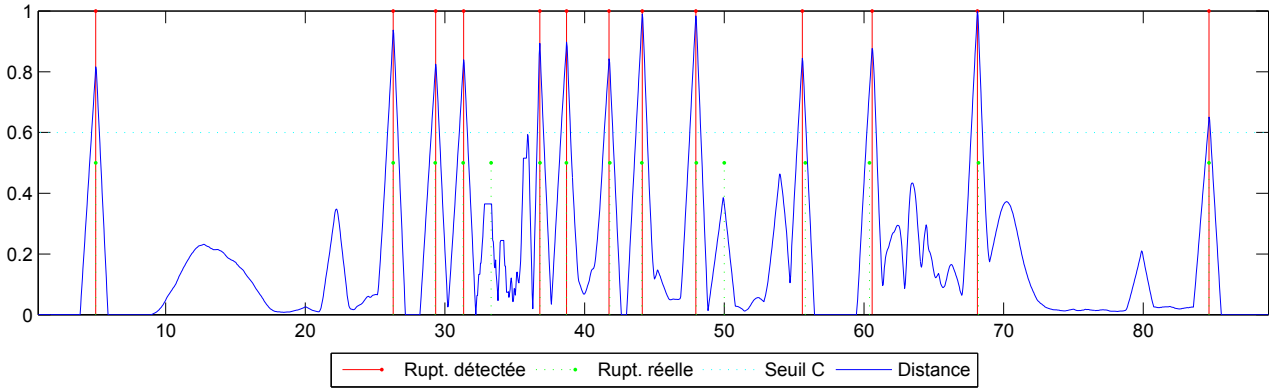


FIGURE 3.2 – Résultat avec la *distance moyenne*

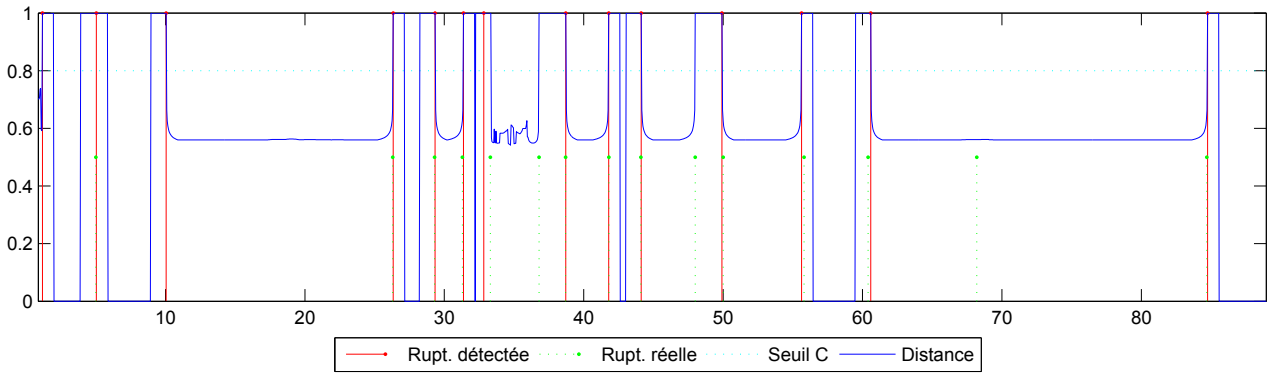


FIGURE 3.3 – Résultats avec la *distance SVM*

permutation aléatoire des images de la vidéo demande beaucoup de ressources (5 minutes environ pour 200 permutations).

Or, cette méthode nécessite énormément de permutations pour être efficace. En effet, avec les 200 permutations effectuées, la méthode nous donne pour $p = 0$ (valeur extrême qui ne devrait engendrer aucun faux-positif) un $C = 0,46$ avec la *distance moyenne*, alors même que ce seuil entraîne beaucoup de faux positifs et que le seuil idéal se situe plutôt autour de $C = 0,6$ d'après les graphiques de distance vus durant le choix des A_i.

On choisira donc C manuellement, avec $C = 0,6$ pour la *distance moyenne* et $C = 0,8$ pour la *distance SVM*.

> *Le bloc 5 de la démonstration montre l'évolution de C en fonction de p, permettant de voir que les valeurs de C ne « montent » pas suffisamment haut.*

3.5.4 Résultats

On obtient finalement les résultats de détection représentés en figure 3.2 pour la *distance moyenne* et la figure 3.3 pour la *distance SVM*. On observe quelques erreurs mais le résultat est globalement satisfaisant.

> *Ces résultats peuvent être observés dans des figures interractives permettant de visualiser la vidéo autour de chaque rupture. Ce code correspond aux blocs 6 et 7.*

4 Conclusion

Ce projet aura été l'occasion pour nous de réaliser du traitement de vidéo sous forme de détection de rupture.

Nous avons ainsi commencé par la phase classique d'extraction de caractéristiques de la vidéo, une phase que l'on retrouve dans la plupart des problèmes de *Machine Learning* sur des images ou des vidéos, où il est nécessaire d'extraire l'information du signal d'origine pour simplifier et améliorer le processus de traitement et baisser la dimensionnalité du problème *(curse of dimensionality)*.

Ce fut donc pour nous l'occasion d'être confronté à ce problème, qui nous a permis d'extraire des histogrammes RGB et YCbCr, mais également des matrices de coocurrence et le gradient de l'image.

Nous avons également été confronté à la nécessité de prétraiter la matrice de caractéristiques afin de faire ressortir les faibles valeurs et atténuer les fortes valeurs.

Cette phase a été suivie de la détection de rupture utilisant deux types de distance, une distance « simple » qu'est la somme des différence des moyennes, et une distance plus complexe basée sur les paramètres de *SVM one-class*. Les deux ayant fourni des résultats assez bons.

Les axes d'amélioration de notre projet se situent sans doute dans l'essai d'autres caractéristiques qui pourraient peut-être être plus robustes aux variations à l'intérieur d'une scène. Concernant les méthodes de détection de rupture, nous n'avons pas trouvé de méthode alternative importante à celle de la dérivée filtrée. En revanche, il pourrait être intéressant de travailler sur des indicateurs $\hat{\theta}$ et des distances D différentes et qui permettraient une meilleure détection de rupture.

5 Bibliographie

[Bertrand11] Pierre Raphael Bertrand, Mehdi Fhima, and Arnaud Guillin. Off-line detection of multiple change points by the filtered derivative with p-value method. *Sequential Analysis*, 30(2) :172–207, Avril 2011.

[Desobry05] Frédéric Desobry, Manuel Davy, and Christian Doncarli. An online kernel change detection algorithm. *Signal Processing, IEEE Transactions on*, 53(8) :2961–2974, 2005.

[Herault14] Romain Hérault. Cours « Derivée filtrée par la méthode p-valeur », INSA Rouen, Décembre 2014.

[Loosli05] Gaëlle Loosli, Sans-Goog Lee, and Stéphane Canu. Context changes detection by one-class SVMs. 2005.