

TP2: Découpage et indexation de vidéos

Analyse et Indexation d'images

ROB317

Yu WANG

November 7, 2021

1 Introduction

L'objectif de ce TP est de se familiariser avec le traitement de vidéos sur OpenCV, en particulier l'estimation du champ de vitesses apparentes (flot optique), et de réfléchir à la problématique de l'indexation automatique de vidéos, dont le but est de sectionner la vidéos en plans, en attachant à chaque plan une description textuelle, ainsi qu'une image représentative du plan. L'indexation de vidéos comprend donc classiquement les tâches suivantes :

- Déetecter les changements de plans dans la vidéo.
- Trouver pour chaque plan l'image la plus représentative.
- Associer à chaque plan une description textuelle.

Dans ce TP on restreindra la description textuelle à la nature du mouvement de la caméra et/ou des objets présents, en identifiant la nature du plan parmi les types (non exhaustifs) suivants :

- Plan fixe
- Panoramique horizontal (pan) ou vertical (tilt) — Rotation / axe optique
- Travelling horizontal, vertical, avant, arrière
- Zoom avant ou arrière

On pourra ajouter au type de plan une description du mouvement d'objets mobiles le cas échéant.

Remarque : Tous les résultats (code, images, vidéos, etc.) présentés dans ce rapport peuvent être trouvés sur ce GitHub : <https://github.com/Rescon/ROB317>

2 Histogramme de couleurs

2.1 Vidéos de codage Yuv

Yuv est un espace colorimétrique à trois composants. Y est une composante de la luminance, qui représente le niveau de gris, et (u, v) sont deux composantes de la chrominance, qui sont utilisées pour décrire la saturation des couleurs. u et v représentent respectivement le contraste Bleu/Jaune et le contraste Rouge/-Cyan.

Par rapport au RGB (RGB en anglais), Yuv occupe moins d'espace pour stocker les informations de couleur, et il peut répondre aux besoins de la télévision en noir et blanc et de la télévision couleur en même temps. Par conséquent, afin de maintenir la compatibilité, de nombreux fabricants choisissent de convertir l'espace colorimétrique RGB en espace colorimétrique Yuv . Si des graphiques doivent être affichés sur l'écran de l'ordinateur, ils peuvent être restaurés au format RGB . La formule de conversion de RGB et Yuv est comme Eq. (2.1) :

$$\begin{aligned} R &= Y + 1.403(V - 128) \\ G &= Y - 0.343(U - 128) - 0.714(V - 128) \\ B &= Y + 1.770(U - 128) \end{aligned} \quad (2.1)$$

En utilisant `cv2.VideoCapture` et `cv2.cvtColor`, nous avons réussi à lire chaque image de la vidéo et à les convertir en espace Yuv . Nous utilisons `cv2.compareHist` pour calculer la corrélation entre les histogrammes de deux frames consécutives, comme le montre la Figure 1.

Nous avons constaté que lorsque le plan ne change pas, il y a de petits changements continus dans l'histogramme. Mais lorsque l'objectif change, des changements soudains et discontinus apparaîtront sur l'histogramme. La chute soudaine de la corrélation dans la Figure 1 peut indiquer que les plans de la vidéo ont changé. Il est nécessaire de déterminer un seuil pour filtrer les réductions soudaines causées par les changements de scène plutôt que les changements d'objectif.

2.2 Vidéos monochromes

Pour la vidéo monochrome, il n'y a plus de canal u et de canal v , il n'y a donc aucun moyen d'afficher un histogramme 2d. À l'aide de la fonction `cv2.cvtColor`, nous avons pu convertir toutes les images de la vidéo en une image en niveaux de gris. En suivant les mêmes étapes que dans la sous-section précédente, nous obtenons la corrélation entre les histogrammes de deux images consécutives, comme le montre la Figure 2.

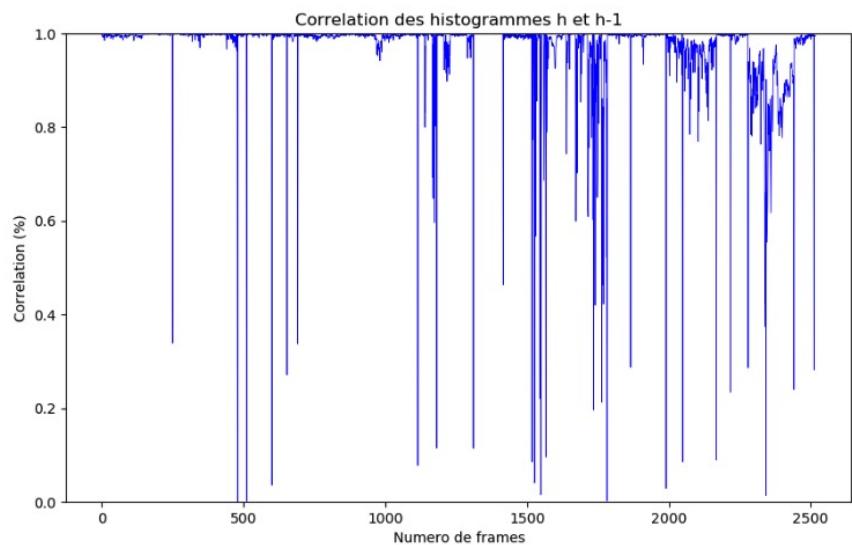


Figure 1: Corrélation entre les histogramme2d des deux frames successives

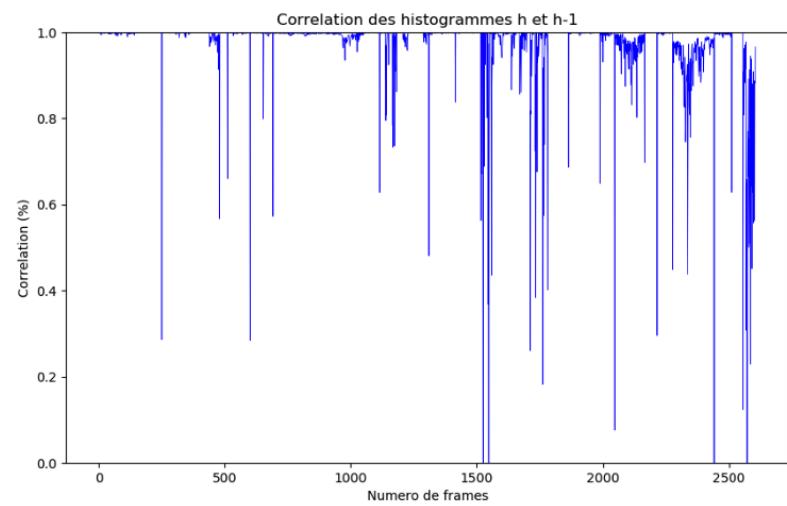


Figure 2: Corrélation entre les histogramme2d des deux frames successives

3 Flot optique et histogramme de vitesses

Tout dans le monde est en mouvement et des vitesses différentes et des directions différentes constituent un champ de mouvement. La projection du mouvement d'un objet sur une image est le mouvement d'un pixel, et sa vitesse instantanée est le flot optique.

La méthode de flot optique permet de déduire la vitesse et la direction du mouvement d'un objet en détectant les changements d'intensité des pixels d'une image dans le temps. L'information sur le mouvement peut être calculée en analysant la correspondance entre l'image précédente et l'image actuelle. Le flot optique consiste en l'étude du champ de mouvement apparent : à chaque pixel on associe un vecteur (v_x^t, v_y^t) qui représente la vitesse du pixel (x, y) à l'instant t .

Il existe deux types de méthodes de flot optique. Le flot optique sparse ne concerne que les points d'intérêt de l'image, tandis que le flot optique dense calcule le décalage de tous les points de l'image. La seconde est beaucoup plus coûteuse en termes de calcul que la première. Et dans cette section, nous analysons une des méthodes denses : Farnebäck.

3.1 L'algorithme *Dense-Optical-Flow.py*

La fonction utilisée pour calculer le flot optique est *cv2.calcOpticalFlowFarneback* qui utilise l'algorithme Farnebäck. Cette méthode nous permet de visualiser le mouvement des points dans la vidéo. Elle permet notamment de détecter la direction dans laquelle les points vidéo se déplacent. Pour des raisons de visualisation, il attribue des couleurs différentes selon le sens du mouvement. Dans la Figure 3, nous montrons les différentes couleurs : bleu pour le mouvement à la baisse, jaune pour le mouvement à la hausse, etc.

Dans le code nous avons:

```
1 flow = cv2.calcOpticalFlowFarneback(prvs,
2                                     next,
3                                     None,
4                                     pyr_scale = 0.5,
5                                     levels = 3,
6                                     winsize = 15,
7                                     iterations = 3,
8                                     poly_n = 7,
9                                     poly_sigma = 1.5,
10                                    flags = 0)
```

où:

- *prvs*: qui représente la première image en input.
- *next*: qui représente la deuxième image en input.

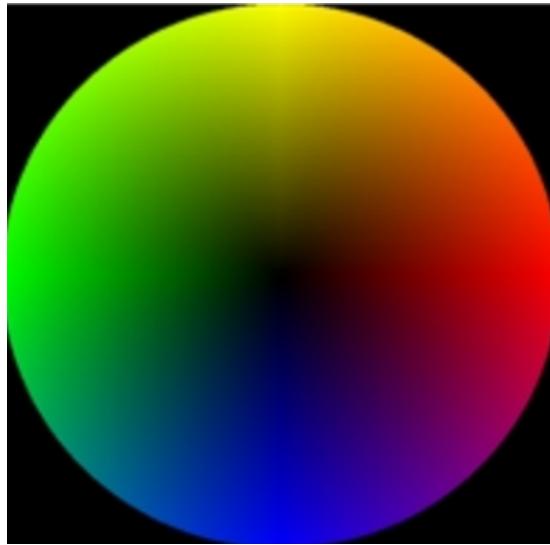


Figure 3: Couleurs utilisés pour indiquer la direction

- *flow*: qui est mis à *None* et fait référence à un flot de calcul de la même taille que *prvs* et de type *CV_32FC2*.
- *pyr_scale*: qui doit être inférieur à 1 et spécifier l'échelle de l'image utilisée pour construire la pyramide pour chaque image. Dans notre cas, nous avons une échelle *pyr_scale* = 0.5, ce qui signifie que chaque couche est 2 fois plus petite que la précédente.
- *levels*: qui est le nombre de couches dans la pyramide sélectionnée, y compris l'image initiale. Dans notre cas, nous avons *levels* = 3 qui signifie que la pyramide comporte deux autres couches dans la pyramide en plus de l'image initiale.
- *winsize*: qui indique la taille de la fenêtre de lissage. Une valeur plus grande augmente la robustesse de l'algorithme au bruit et fournit plus de moyens de détecter les mouvements rapides, mais d'un autre côté, cela peut aussi produire des champs de mouvement flous.
- *iterations*: qui représente le nombre d'itérations qui l'algorithme fait à chaque couche de la pyramide.
- *poly_n*: qui indique la taille du voisinage du pixel pour trouver l'expansion polynomiale dans chaque pixel. Des valeurs plus grandes rendent l'algorithme plus robuste et permettent d'estimer des images avec des surfaces plus lisses. D'un autre côté, cela permet d'obtenir un champ de mouvement plus flou.

- *poly_sigma*: qui représente l'écart type de la gaussienne utilisée pour calculer l'expansion polynomiale.
- *flags*: Cela dépend des différentes mesures d'erreur qui seront utilisées pour la valeur donnée. La variable peut prendre deux valeurs différentes : *OPTFLOW_USE_INITIAL_FLOW* utilisant l'estimation initiale effectuée dans *next* ou *OPTFLOW_FARNEBACK_GAUSSIAN* utilisant une fenêtre gaussienne carrée de taille *winsize*. Dans le second cas, l'algorithme donne des résultats plus lents mais est plus précis dans ses calculs. Dans notre cas, il est initialisé à zéro et n'est pas défini : nous allons donc copier *prev* dans *next* et considérer l'estimation initiale.

Le résultat de cette fonction est une image de la même taille que *prev* et *next* dans la forme. Il montre le flot optique pour chaque pixel de *prev* en utilisant l'algorithme de Farnebäck, comme Eq. (3.1):

$$\text{prev}(y, x) \sim \text{next}(y + \text{flow}(y, x)[1], x + \text{flow}(y, x)[0]) \quad (3.1)$$

3.2 Le principe de l'algorithme Farnebäck

L'algorithme est basé sur une approximation entre deux images consécutives avec un voisinage particulier de pixels sous forme polynomiale, puis sur une estimation du déplacement à partir des coefficients d'expansion polynomiale. En particulier, nous allons relier le voisinage *x* du pixel dans la première image au polynôme, comme Eq. (3.2):

$$f_1(x, y) \approx x^T A_1 x + b_1^T x + c_1 \quad (3.2)$$

En supposant que dans le deuxième frame il y a eu un mouvement *d*, on associera le polynôme suivant au voisinage du même pixel mais déplacé d'une valeur *d*, comme Eq. (3.3):

$$f_2(x, y) = f_1(x - d) = (x - d)^T A_2(x - d) + b_2^T(x - d) + c_2 \quad (3.3)$$

avec:

$$\begin{aligned} A_2 &= A_1 \\ b_2 &= b_1 - 2A_1d \\ c_2 &= d^T A_1 d - b_1^T d + c_1 \end{aligned} \quad (3.4)$$

Si *A*₁ est inversible, nous pouvons obtenir finalement l'estimation de la distance *d* et donc du mouvement que l'image a effectué, comme Eq. (3.5):

$$d = -\frac{A_1^{-1}(b_2 - b_1)}{2} \quad (3.5)$$

Certaines approximations sont alors effectuées pour que l'erreur reste faible et pour avoir toujours une bonne estimation et ne pas être trop affecté par le bruit. L'avantage de cette approche est que l'algorithme travaille de manière itérative et à plusieurs échelles sur les estimations de déplacement. Dans notre cas, comme déjà expliqué, nous effectuerons trois itérations pour chaque échelle et nous réaliserons une analyse à trois échelles.

L'analyse itérative permet d'obtenir une meilleure précision dans le calcul, mais elle présente la faiblesse suivante : si le déplacement est trop important, il ne peut être saisi et, par conséquent, les itérations multiples ne seront pas pertinentes. Ce problème est résolu en effectuant le calcul à plusieurs échelles avec des résolutions différentes, de sorte que la méthode puisse gérer des mouvements plus importants. En revanche, cette méthode réduit la précision et augmente le coût de calcul. Les deux méthodes sont utilisées simultanément afin de pouvoir exploiter leurs avantages et minimiser leurs difficultés.

En résumé, l'algorithme peut estimer des vitesses élevées car il est réalisé au niveau multi-échelle et, grâce aux coefficients polynomiaux, il peut également calculer de flot dense en considérant tous les pixels de l'image à la fois afin de réduire le coût de calcul, qui ne sera calculé que la première fois et sera réutilisé par la suite.

3.3 L'histogramme 2D des composantes (V_x, V_y)

Nous nous demandons ici d'afficher les composantes horizontales et verticales du flot optique sous la forme d'une image. La fonction Farnebäck que nous allons utiliser renvoie deux tableaux bidimensionnels différents : l'un contenant les composantes du mouvement optique horizontal et l'autre les composantes du mouvement optique vertical.

Ainsi, nous traçons les deux composantes dans un histogramme à deux dimensions. La première chose que nous observons est que le changement de pixels dans l'histogramme est très peu significatif : pour cette raison, nous avons choisi d'éliminer la plupart des données statiques, qui ne représentent pas l'absence de changement d'une manière qui accorde plus d'importance au changement réel.

Nous avons observé que dans le cas d'un changement de scène, l'histogramme signalait une explosion dans toutes les directions : c'est-à-dire que l'on pouvait observer le graphique s'étendre dans toutes les directions. En effet, des changements importants se produisent dans toutes les directions, que nous mettons en relation avec les changements du plan. Par exemple, dans la vidéo présentée, nous avons un exemple de changement de scène : d'abord les visages, puis l'assombrissement du ciel. Nous avons tracé l'image de l'histogramme dans la Figure 4.

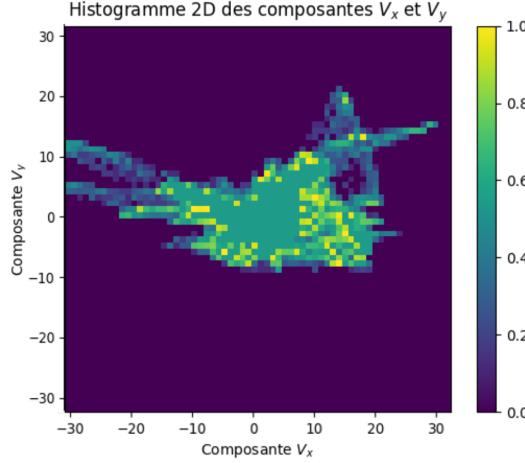


Figure 4: Histogramme qui reporte V_x et V_y

En outre, il nous a été demandé d'expliquer comment nous avons utilisé les informations fournies pour identifier l'histogramme pour le type de plan dans le graphique actuel. Pour ce faire, nous allons analyser la forme de l'histogramme 2d (V_x, V_y) pour les différents mouvements que subit la vidéo.

- Plan fixe
- Panoramique Horizontal (Pan)
- Panoramique Vertical (Tilt)
- Rotation (Roll)
- Travelling horizontal
- Travelling vers l'avant
- Zoom avant

Nous avons fait des recherches sur le WEB ou tourné des vidéos avec la caméra qui représentent différents mouvements de caméra et nous avons fourni une brève description de l'histogramme associé à un certain type de plan.

Plan Fixe

Dans la Figure 5, l'histogramme de plan fixe est un point au centre car il n'y a aucun mouvement horizontal ou vertical.

Panoramique Horizontal (Pan)

Dans la Figure 6, l'histogramme de panoramique horizontale est un cône avec un axe horizontal (ici à droite). Le cône est situé au centre de l'histogramme.

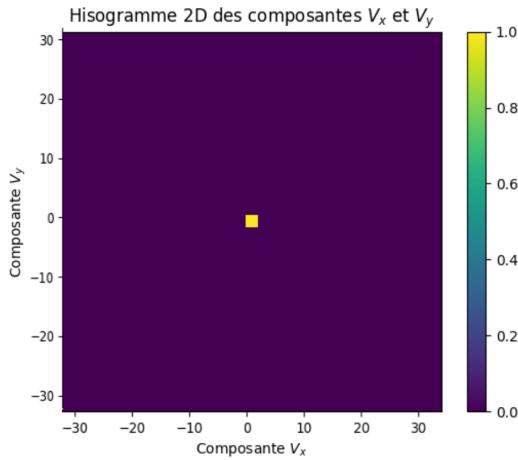


Figure 5: Histogramme pour un mouvement de Plan Fixe

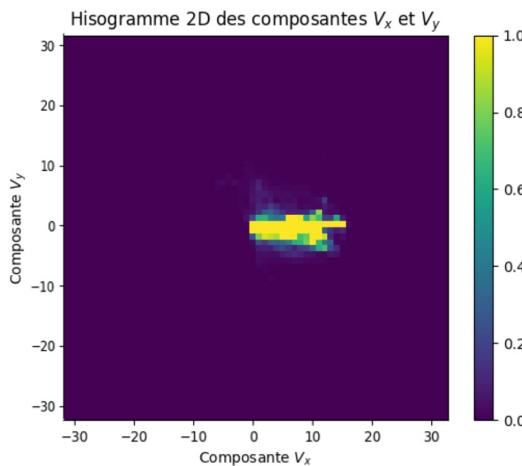


Figure 6: Histogramme pour un mouvement de Panoramique Horizontal

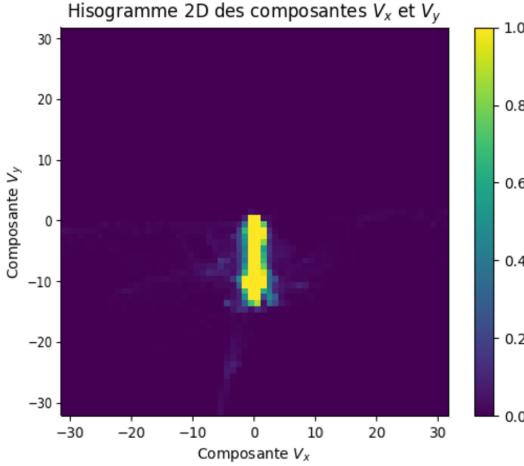


Figure 7: Histogramme pour un mouvement de Panoramique Vertical

C'est logique car dans le mouvement de panoramique horizontale (droite), la moitié des pixels en haut se déplacent vers la gauche et le haut et la moitié des pixels droits se déplacent vers la gauche et le bas, ce qui nous donne une forme de cône vers la droite.

Panoramique Vertical (Tilt)

Dans la Figure 7, l'histogramme de panoramique vertical est un cône avec un axe vertical (ici à bas). Le cône est situé au centre de l'histogramme. C'est logique car dans le mouvement de panoramique horizontale (bas), la moitié des pixels gauche se déplacent vers la gauche et le haut et la moitié des pixels droits se déplacent vers la droit et le haut, ce qui nous donne une forme de cône vers le bas.

Rotation

Dans la Figure 8, pour le mouvement de rotation, l'histogramme sera caractérisé par un grand nombre de points, car les pixels se déplacent dans la direction de la rotation. Nous saurons donc quels mouvements sont petits (et ont un petit vecteur vitesse) car ils sont proches du centre de rotation et quels mouvements sont plus importants (et auront un vecteur vitesse plus grand) car ils sont éloignés du centre de rotation. De plus, les déplacements seront dans toutes les directions, nous aurons donc un histogramme circulaire.

Travelling horizontal

Dans la Figure 8, Le mouvement de travelling horizontal de la caméra nous fournit un histogramme utilisant la ligne horizontale comme axe du point du nuage. La cohérence entre le mouvement de la caméra et le mouvement des pixels a été vérifiée dans cet histogramme.

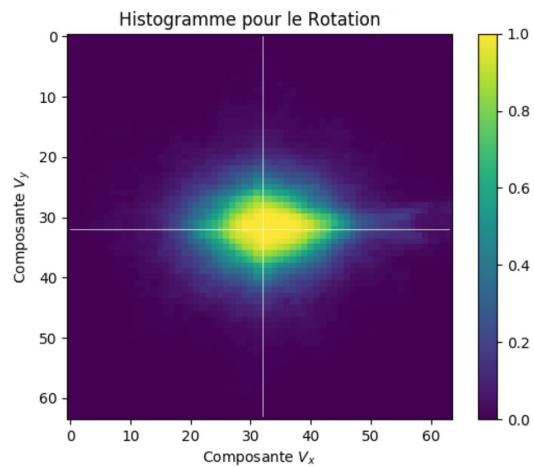


Figure 8: Histogramme pour un mouvement de Rotation

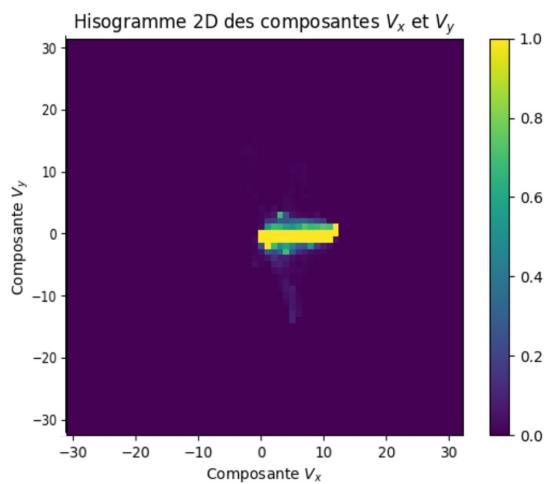


Figure 9: Histogramme pour un mouvement de Travelling horizontal

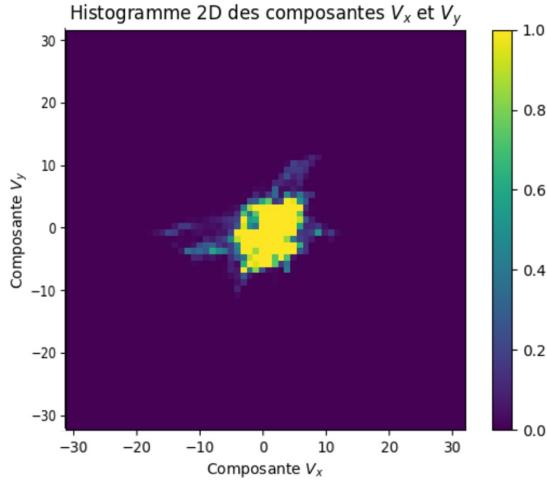


Figure 10: Histogramme pour un mouvement de Travelling vers l'avant

Travelling vers l'avant

Dans la Figure 10, le travelling vers l'avant produit un nuage de points circulaire avec un rayon moyen. Cela signifie que tous les pixels de l'image sont en mouvement. En particulier, les pixels de gauche se déplaceront vers la gauche, les pixels de droite se déplaceront encore vers la droite, les pixels du bas vers le bas, etc. Cela explique la présence de nuages de points dans toutes les directions.

Zoom avant

Dans la Figure 11, le zoom avant donne également un nuage de points circulaire, mais avec un rayon plus court. Cela s'explique par le fait que notre vision est réduite à un seul objet (que nous mettons à l'échelle) et que le déplacement sera donc plus petit par rapport au mouvement de déplacement.

La détection du type de plan

Sur la base des différents types d'histogrammes mentionnés ci-dessus, nous pouvons facilement identifier les différents types de plans. En particulier, nous envisageons de diviser l'histogramme obtenu en différentes parties, que nous additionnerons afin de pouvoir identifier dans quelle direction et avec quelle intensité l'histogramme se déplace. Nous mettrons en œuvre cette méthode dans la Q6 et nous discuterons des difficultés et des limites rencontrées avec cette approche.

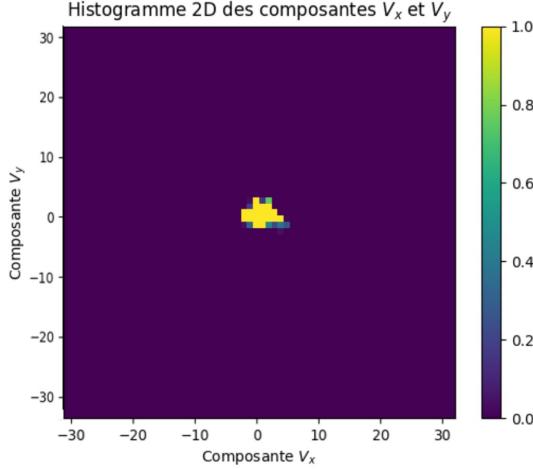


Figure 11: Histogramme pour un mouvement de Zoom avant

4 Découpage et Indexation

Le découpage et l'indexation des vidéos sont des tâches très importantes dans le traitement des vidéos. Dans cette section, nous décrivons le processus de découpage de la vidéo en plans (Section 4.1), la recherche de l'image clé pour ce plan (Section 4.2) et enfin l'identification du type de plan (Section 4.3).

4.1 Technique de découpage en plans

Afin d'effectuer la détection de différents plans, nous devons choisir un seuil qui peut différer d'un nombre entier du pic qui représente réellement le changement de plan.

Pour ce faire, nous avons envisagé d'unifier les deux méthodes développées précédemment (détection à l'aide d'un histogramme, qui se réfère au champ Yuv et utilise le flot optique) et d'ajouter une troisième méthode capable d'exploiter les changements de caractéristiques invariants de la couleur. La troisième méthode s'est avérée nécessaire après avoir testé notre algorithme sur la vidéo Vertigo, qui présentait plusieurs changements de couleur, notamment lorsque des problèmes sont apparus avec la méthode impliquant le champ Yuv .

En outre, l'utilisation de trois méthodes différentes pour détecter les changements prévus nous permet d'être plus robustes dans notre algorithme, et l'utilisation de décisions de vote majoritaire nous permet de tirer parti des trois méthodes différentes. Dans les trois méthodes, nous effectuons une corrélation entre l'image actuelle et l'image précédente, et nous classons les pics négatifs comme des changements de plan si la valeur de corrélation est inférieure à un certain

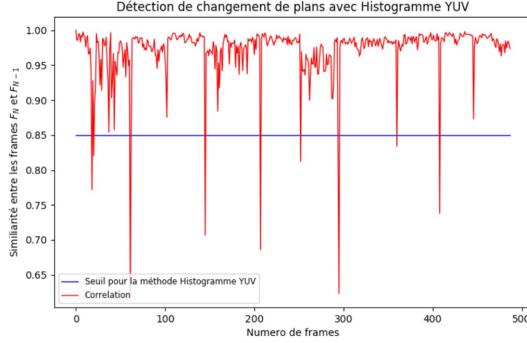


Figure 12: Corrélation des histogrammes lié à Yuv

seuil σ . La valeur finale de cette comparaison sera une valeur binaire (0 si la méthode ne détecte pas de changement de plan, 1 si la méthode détecte un changement de plan), comme décrit ci-dessous, où $Corr$ indique que la corrélation est entre les images t et $t - 1$, comme Eq. (4.1)

$$f(x) = \begin{cases} 0, & Corr(I, I_{t-1}) > \sigma \\ 1, & Corr(I, I_{t-1}) \leq \sigma \end{cases} \quad (4.1)$$

En général, nous pouvons avoir deux scénarios différents : si nous connaissons les caractéristiques de la vidéo avant de l'analyser, nous pouvons simplement choisir les pondérations des trois méthodes pour sélectionner l'utilisation de la méthode la plus appropriée aux caractéristiques de la vidéo. Si nous ne connaissons pas la vidéo a priori, alors nous appliquerons chacune des trois méthodes séparément et nous appliquerons la méthode du vote pour choisir si elle représente un changement de plan. Comme on pouvait s'y attendre, cela nous permet de tirer parti des trois méthodes et de réduire le risque d'erreur d'appréciation.

4.1.1 Corrélation des histogrammes lié à Yuv

Comme la Figure 12, cette méthode est appliquée aux vidéos 1 et 5 avec un seuil $\sigma = 0.85$. Dans la vidéo numéro 1, la majorité des plans sont détectés. L'un des inconvénients de cette méthode est qu'elle n'est pas robuste aux changements soudains de luminosité ou de vitesse (on peut le constater entre les images 1500 et 2000 de la vidéo 1). Dans le cas de la vidéo 3, cette méthode n'est pas optimale car elle est très sensible aux changements de couleur et souvent les changements soudains de couleur donnent des signaux qui indiquent des changements incorrects dans le métrage.

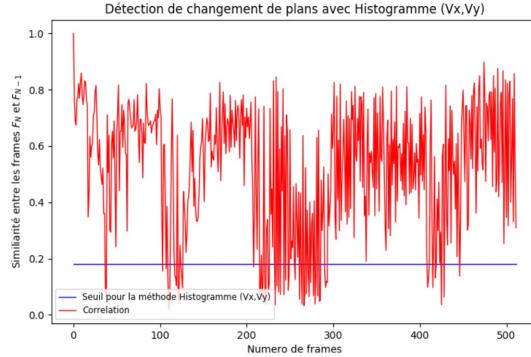


Figure 13: Corrélation des histogrammes lié à Flot Optique

4.1.2 Corrélation des histogrammes lié à Flot Optique

Comme la Figure 13, pour cette méthode, nous utiliserons l’information donnée par la vitesse à laquelle les pixels se déplacent dans l’image. Nous avons choisi une valeur seuil de 0,18. Par rapport à la méthode précédente, nous la trouvons moins sensible aux changements soudains de la position des pixels, mais d’un autre côté, il y a toujours le problème de la couleur, avec de grands changements de vitesse à la fin de la vidéo 1 (images 2000 -2500) et dans la vidéo 3. De plus, lorsque nous changeons le plan par fondu, c’est-à-dire lorsque le changement de plan est progressivement achevé, nous ne pouvons pas détecter de bons résultats : cela est dû à la lenteur de la transition et au fait que ces changements ne se reflètent pas bien dans l’histogramme du flot optique. Un autre problème que nous avons trouvé dans la vidéo 5 est que cette méthode ne détecte pas les changements lorsque la caméra a plusieurs mouvements d’inclinaison. Un autre inconvénient de cette méthode est qu’elle est plus longue à exécuter que les deux autres méthodes (elle est très inefficace).

4.1.3 Corrélation des histogrammes lié à Hog

Comme la Figure 14, nous avons choisi d’utiliser l’histogramme Hog. Le descripteur Hog peut décrire l’apparence et la forme locale des objets dans une image par la distribution de l’intensité du gradient ou l’orientation des contours. Par conséquent, c’est un histogramme qui est moins affecté par la couleur. Cela nous a permis de profiter de son invariance aux changements de géométrie et de luminosité. Un problème que nous avons remarqué avec cette approche est que dans la vidéo 3, les changements de plans sont incorrectement détectés lorsque de nombreux détails changent rapidement de position.

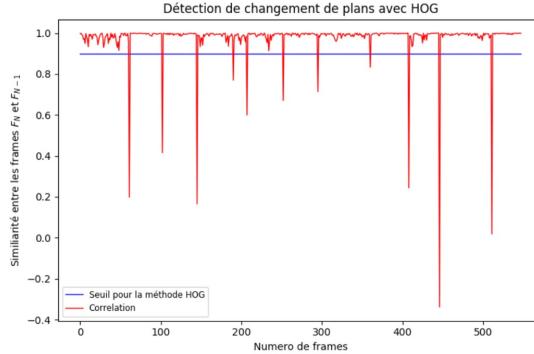


Figure 14: Corrélation des histogrammes lié à Hog

4.1.4 Conclusion

Ainsi, notre algorithme détecte les changements de plans dans la vidéo proposée et affiche sur le terminal les numéros d'images initiales et finales d'un plan donné. De plus, la méthode de vote nous permet d'obtenir des performances satisfaisantes sans connaître les caractéristiques de la vidéo.

4.2 Une mesure d'extraction d'image-clef représentative d'un plan

Dans la section précédente, nous avons réussi à diviser la vidéo en plusieurs plans et pour obtenir une image plus claire du montage, nous devons extraire une image clé de chaque plan.

Avant de présenter notre méthode d'extraction des images clés, nous commençons par définir une image clé. Selon nous, une image clé est une image de la vidéo qui donne une bonne représentation des caractéristiques du métrage, comme le type de plan, le changement de pixels, etc. L'analyse des images clés peut faciliter l'étude des séquences vidéo.

Nous avons en tête deux méthodes d'extraction d'images clés : la méthode intermédiaire et la méthode du point d'intérêt.

- **Méthode intermédiaire :** Dans cette méthode, nous prenons l'image avec le intermédiaire $\frac{t_{min}+t_{max}}{2}$, mais ce n'est pas nécessairement l'image la plus représentative.
- **Méthode des points d'intérêt :** Dans cette méthode, nous allons calculer le nombre de points d'intérêt pour chaque image de chaque plan. L'image présentant le plus de points d'intérêt sera sélectionnée comme image clé pour cette prise de vue.

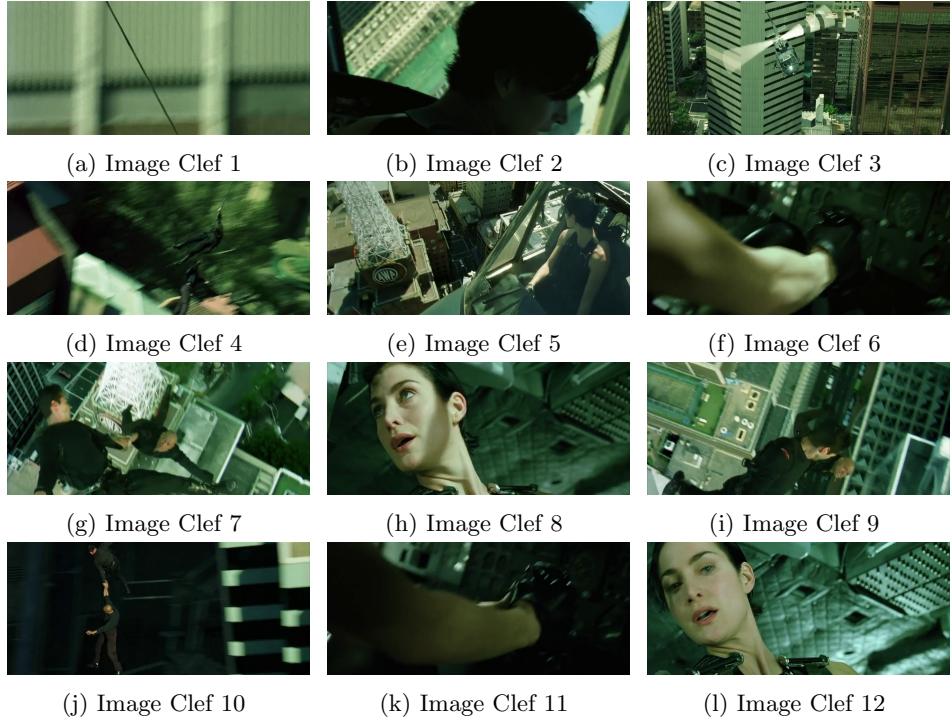


Figure 15: Images-clé obtenues par la méthode intermédiaire de la vidéo 5

Nous les avons implémentés avec succès dans le script Sparse_Optical_Flow en utilisant `cv2.goodFeaturesToTrack` et `cv2.calcOpticalFlowPyrLK`, et nous pouvons facilement calculer le nombre de points d'intérêt pour chaque image de la vidéo.

Comme le montrent la Figure 15 et la Figure 16, les résultats sont satisfaisants, car les images clés sélectionnées représentent la scène de la vidéo. Bien que les images clés obtenues par les deux méthodes soient légèrement différentes, même si le premier plan change très rapidement, pour les autres clips, les images clés correspondent très étroitement aux autres clips de la vidéo.

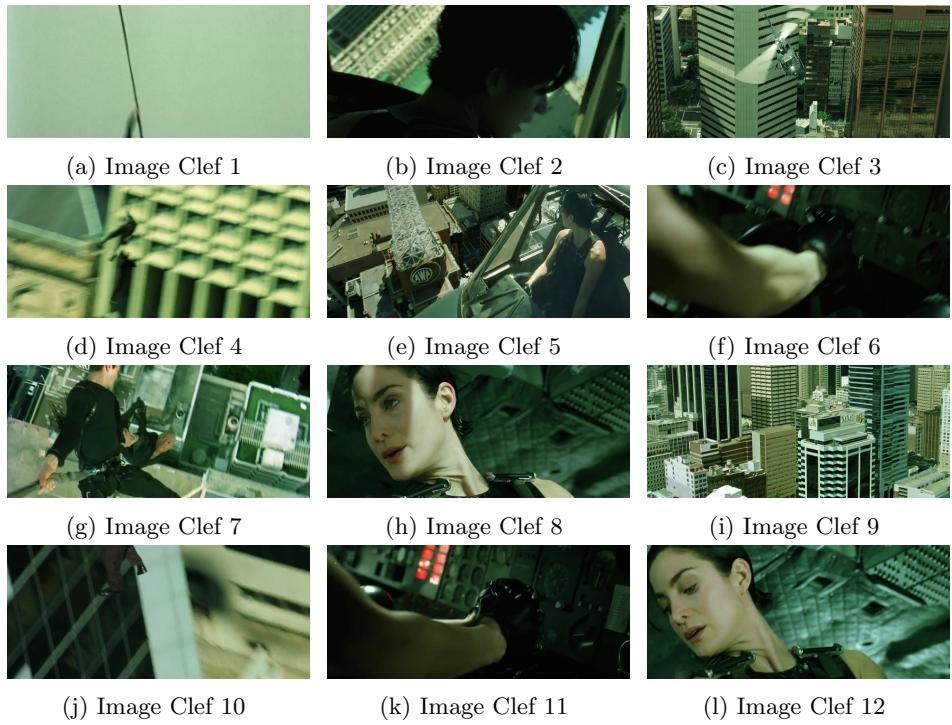


Figure 16: Images-clef obtenues par la méthode des points d'intérêt de la vidéo
5

4.3 Une technique d'identification de type de plan

Afin d'identifier les différents types de plans, comme déjà vu en Q3, nous avons calculé un histogramme du flot optique moyen des images représentant les scènes. Nous allons donc fournir un histogramme pour chaque scène de la vidéo. Nous divisons ensuite ce plan en quatre parties et pour chaque partie nous allons faire la somme afin de visualiser où se trouvent la plupart des pixels. Grâce à une série de jugements conditionnels, nous sommes en mesure de détecter différents types d'avions. Nous parvenons à distinguer trois grands types de plans : ceux associés aux mouvements de rotation, ceux associés aux mouvements de translation (translation et mouvement horizontal ou inclinaison et mouvement vertical), et ceux associés à la mise à l'échelle ou au déplacement vers l'avant.

Nous avons défini différents seuils en fonction des tests effectués. En particulier, si la caméra se déplace vers la droite, nous aurons un histogramme vers la gauche, ce qui nous permettra de savoir que la somme de la partie gauche sera supérieure à la somme de la partie droite. S'il y a également des mouvements diagonaux, notre algorithme fera la distinction entre les mouvements vers les quatre coins du champ de vision. Si la vidéo était limitée aux mouvements horizontaux ou verticaux, nous obtiendrions de bons résultats : nous cherchons donc à distinguer les groupes contenant des mouvements de translation, d'inclinaison, de mouvement horizontal et les groupes de mouvements combinés. en faisant tourner et en mettant à l'échelle les mouvements. Nous nous concentrerons ensuite sur la distinction entre la mise à l'échelle et la rotation qui ont des histogrammes très similaires. Leurs histogrammes sont caractérisés par des cercles de différents rayons, en fonction de l'échelle et de la vitesse de rotation.

Au début, nous avons constaté que pour le mouvement de rotation, le rayon du cercle dans l'histogramme était plus grand que le rayon de l'échelle. Ensuite, en testant la robustesse de notre algorithme sur plusieurs vidéos, nous avons remarqué que nous pouvions mieux utiliser les informations données par la magnitude et l'angle du vecteur vitesse. Pour la détection de l'échelle et de la rotation, nous avons combiné les informations de direction et de magnitude obtenues grâce aux informations données par la fonction Farnebäck pour le calcul du flot optique.

Nous allons décrire le processus suivi dans ce qui suit.

- Nous regroupons les vecteurs qui ont un angle fini entre certains intervalles définis. Dans notre exemple, nous avons choisi d'utiliser 24 intervalles représentant les intervalles suivants : $[0 - 15, 15 - 30, \dots, 345 - 360]$.
- Une fois les valeurs des bins définies, nous construisons deux histogrammes : l'un du nombre de vecteurs dans chaque intervalle lié au bin, et l'autre de la somme des magnitudes des vecteurs liés dans l'intervalle. Après la construction, chaque histogramme a été normalisé, comme le montrent la

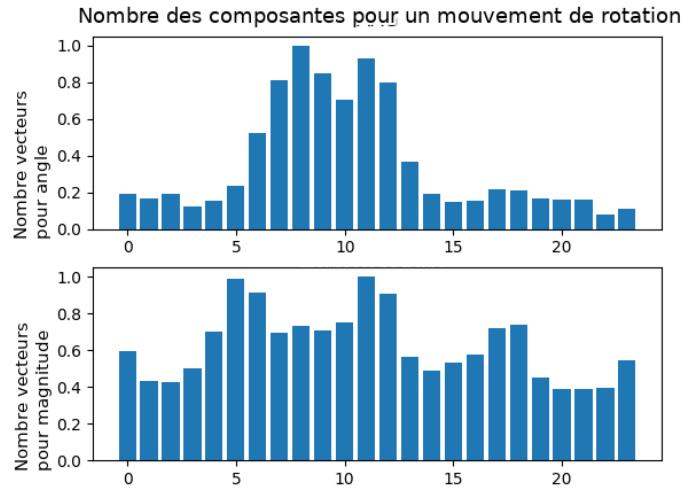


Figure 17: Corrélation des histogrammes lié à Hog

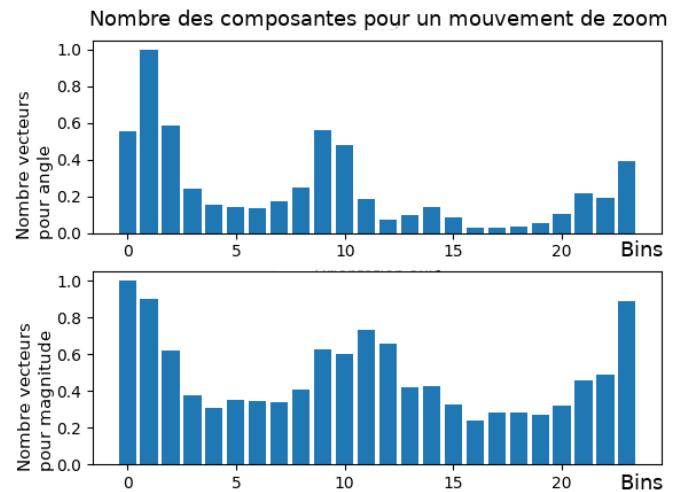


Figure 18: Corrélation des histogrammes lié à Hog

Figure 17 et la Figure 18.

- Pour vérifier la nature du mouvement (mise à l'échelle ou rotation), les 5 blocs principaux de l'histogramme d'orientation ont été pris. Nous avons calculé la $compara_{ang}Mag = \frac{1}{5} \sum_{i=1}^5 abs(H_m - H_a)$, où H_m est le vecteur correspondant aux 5 bins les plus importants appartenant à l'histogramme de magnitude et H_a est le vecteur correspondant aux 5 bins les plus importants appartenant à l'histogramme de direction.
- Après avoir calculé $compara_{ang}Mag$, nous utilisons cette valeur comme paramètre pour distinguer le mouvement de mise à l'échelle du mouvement de rotation, car selon les tests effectués sur les extraits de rotation et de mise à l'échelle, nous avons constaté que pour la rotation les valeurs de $compara_{ang}Mag$ sont toujours inférieures à un seuil, alors que pour le zoom elles sont toujours supérieures à ce seuil.

Nous remarquons que pour le mouvement de rotation, la valeur est plus grande par rapport à la valeur associée au mouvement d'échelle, ce qui s'explique par le fait qu'en rotation, les pixels se déplacent par rapport à un point fixe. Nous aurons donc des pixels qui bougeront plus, parce qu'ils éloignent du centre de rotation, et d'autres qui bougeront moins, parce qu'ils sont plus proches du centre de rotation, mais dans la mise à l'échelle, les pixels se déplacent dans la même direction et avec la même magnitude, c'est-à-dire que la différence entre les deux quantités est faible.