INF8770 – Technologies multimédias

Automne 2024

Travail pratique #3 – Décomposition d'une séquence vidéo en prises de vue

Groupe 02 – Équipe A37

2221053 – Thomas Rouleau

2149244 – Justin Lefrançois

Soumis à : M. Yassine Achour

2024-12-02

Question 1 (/4) Identifier les difficultés propres aux séquences vidéo fournies pour leur analyse. Identifier les problèmes à résoudre. (Analyser les 2 séquences vidéo)

Ce rapport vise à analyser, identifier les principales difficultés associées à la détection des transitions (coupures et fondus enchaînés) et proposer une méthode de détection de transition permettant de résoudre ce problème pour deux vidéos fournies : une vidéo d'athlétisme, pour laquelle les vérités terrain sont connues, et une vidéo de soccer sans vérités terrain. La détection automatique des transitions est essentielle en vision par ordinateur, notamment pour la segmentation de contenu et l'analyse de scènes à des fins d'indexation. Cependant, cette tâche devient complexe dans des contextes comportant des mouvements rapides de caméra et d'objets ainsi qu'une forte corrélation de couleurs entre les prises de vue, comme c'est souvent le cas dans les vidéos de sports professionnels [1].

Plusieurs difficultés techniques impactent la précision des algorithmes de détection de transitions, particulièrement dans des vidéos dynamiques. Celles-ci peuvent être séparés en 5 catégories [2] :

1. **Soudaines variations de luminosité** : Les changements soudains de luminosité, comme ceux causé par des flashes, des effets d'éclairages, des changements de luminosité causé par l'environnement pour des événements extérieurs, peuvent être confondus avec des transitions de coupures [3].
2. **Séquences de faible luminosité** : Des séquences à faible luminosité ou des trames sombres créent des défis de détection de transitions, particulièrement dans des vidéos avec des variations d'éclairages multiples. La similarité visuelle entre les trames successives peut empêcher la détection correcte des transitions [2].
3. **Scènes avec arrière-plans comparables** : Les transitions entre des scènes ayant des arrière-plans comparables tel qu'un terrain de soccer, une piste d'athlétisme, un stade, etc. présentent un défie particulier car la forte similitude visuelle et de couleurs entraîne des détections manquées [2].
4. **Mouvements de caméra et d'objets** : Les mouvements de caméra et d'objets rapides ou complexes rendent difficile la distinction entre ceux-ci et les transitions réelles de scène [2].
5. **Changement dans de petites régions** : Les changements localisés, comme les images ou des parties de l'images sur deux scènes de mêmes plans avec caméra fixe peuvent être ignorés par la faible dissimilarité dans ces régions compliquant la détection [2].

L'analyse des difficultés principales pour chaque vidéo fournie sont présentés dans le tableau suivant (Tableau 1) :

Difficulté	Vidéo d'athlétisme	Vidéo de soccer
Variation de la luminosité	Aucune grande ou soudaine variation de luminosité susceptible de causer des erreurs de détection de transition.	Aucune grande ou soudaine variation de luminosité susceptible de causer des erreurs de détection de transition.
Séquences de faible luminosité	Absence de séquences de faible luminosité nuisibles à la détection des transitions.	Absence de séquences de faible luminosité nuisibles à la détection des transitions.
Scènes avec arrière-plans comparables	Certaines transitions entre plans avec des arrière-plans similaires rendent la détection plus difficile (e.g. : fondu 0:12s). La piste d'athlétisme présente une uniformité de couleur et de forme.	Plusieurs transitions impliquent des arrière-plans similaires, ce qui complique la détection (e.g. : coupure 0:12s). Cette uniformité revient fréquemment dû à l'uniformité de couleur et de forme du terrain de soccer.
Mouvement de caméra et d'objets	Les mouvements sont modérés, mais certains mouvements rapides pourraient générer des faux positifs.	Caméra rapide pour suivre les joueurs. Mouvements complexes des joueurs et du ballon pouvant provoquer des faux positifs.
Changement dans de petites région	Aucun plan avec caméra fixe, rendant cette difficulté absente.	Une transition (e.g. : 1:31s) entre deux plans presque fixes montre des différences minimes, principalement dans de petites régions (joueur effectuant un tir au but).

Tableau 1 : Analyse des types de difficultés techniques de détection des transitions en fonction de la vidéo

Autres difficultés remarquées

Dans la vidéo d'athlétisme, deux coupures successives en moins d'une seconde (à 0:17s) pourraient poser des défis selon la méthode de détection utilisée. Cette proximité temporelle risque d'affecter la précision ou l'efficacité des algorithmes.

Question 2 (/4) Proposer une méthode pour résoudre le problème. Donner son algorithme général. Justifier le choix de la méthode en fonction des problèmes identifiés à la question 1.

Méthode proposée : Combinaison de l'histogrammes de couleurs (HC) et de la détection des contours (DC)

Pour résoudre les difficultés identifiées dans la détection des transitions, nous proposons une méthode combinant deux techniques de détection : l'histogrammes de couleur (HC) et la détection des contours (DC). Cette approche exploite les forces de chaque méthode tout en atténuant leurs faiblesses respectives. La HC est efficace pour identifier les changements dans la distribution des couleurs, mais elle est sensible aux mouvements rapides ou aux arrière-plans similaires entre les trames [4]. La DC est robuste face aux mouvements rapides, mais elle peut être affectés par une complexité computationnelle accrue et une sensibilité excessive pouvant causer de faux positifs [5]. Nous supposons qu'en combinant leurs résultats, il serait possible d'obtenir une méthode plus précise et robuste pour détecter et classifier les transitions vidéo.

Algorithme général :

1. **Détection des contours (DC)**
    a. Prétraitement : Convertir chaque trame en niveaux de gris.
    b. Extraction des contours :
        i. Calculer la magnitude des gradients en x et y à l'aide d'un filtre de Sobel.
        ii. Obtenir une image de contours basée sur les gradients.
    c. Amélioration des contours : Appliquer une dilatation avec un disque de rayon r pour renforcer les contours.
    d. Calcul des arêtes entrantes (ρ_{in}) et sortantes (ρ_{out}) :

$$\rho_{in} = 1 - \frac{\sum_{x,y} D(x,y,t-1)E(x,y,t)}{\sum_{x,y} E(x,y,t)}$$

$$\rho_{out} = 1 - \frac{\sum_{x,y} E(x,y,t-1)P(x,y,t)}{\sum_{x,y} E(x,y,t-1)}$$

        i. Où : *D* représente les contours dilatées, *E* les gradients de magnitude, *t* la trame actuelle et *t-1* la trame précédente
    e. Détection et classification des transitions:
        i. Coupure : Si $\rho=\max(\rho_{in}, \rho_{out})$ dépasse un seuil gaussien et présente une différence significative avec les valeurs récentes
        ii. Fondu: Si $\rho_{in} > \rho_{out}$ puis $\rho_{out} > \rho_{in}$ sur l'horizon d'événements

2. **Histogramme de couleurs (HC)**
    a. Calcul des histogrammes : Pour chaque trame I_t, calculer l'histogramme des couleurs H_t.
    b. Différences entre histogrammes : Calcul de distance euclidienne de deux trames successives H_t et H_{t-1}

$$D_{HD}(H_t, H_{t-1}) = \sum_{i=1}^{N} |H_t[i] - H_{t-1}[i]|$$

    c. Détection et classification des transitions :
        i. Coupure : Si D_{HD} dépasse un seuil t_c
        ii. Fondu : Si D_{HD} dépasse un seuil t_f
3. **Combinaison des scores**
    a. Appliquer indépendamment HC et DC à la même vidéo
    b. Fusion des fondus :
        i. Vérifier pour chaque fondu HC s'il existe déjà un fondu proche détecté par DC
        ii. Si aucun fondu est trouvé à proximité, alors l'ajouter à la liste des fondus
    c. Fusion des coupures :
        i. Combiner les coupures de HC et DC et les trier
        ii. Supprimer les coupures redondantes

Justification de la méthode sélectionnée

Cette méthode est adaptée aux défis identifiés, car :

- La HC permet une détection efficace dans des contextes où les couleurs changent significativement.
- La DC offre une robustesse face aux mouvements rapides et aux scènes complexes.
- Leur combinaison permet de réduire les fausses détections et d'améliorer la précision globale.

Les seuils (t_c,t_f,r) et les paramètres de dilatation sont ajustables pour s'adapter aux différentes caractéristiques des vidéos (résolution, fréquence d'images).

Question 3 (/4, 2 points par qualité) Implémenter la méthode proposée à la question 2. Intégrer obligatoirement dans le code informatique des librairies de traitement d'images/vidéos existantes (e.g. opencv). Expliquer brièvement comment l'algorithme implémenté, le langage et la librairie utilisés.

Deux algorithmes distincts pour la détection de transitions dans une vidéo ont été implémentés : un basé sur l'analyse des histogrammes de couleurs et l'autre sur la détection des contours avec l'utilisation d'un filtre de Sobel. Ces algorithmes ont été implémentés en Python et exploitent plusieurs libraires en particulier OpenCV pour le traitement des images.

Langage et librairies utilisées

- Langage : Python pour ses nombreuses bibliothèques spécialisées
- Librairie principale : OpenCV, utilisée pour le traitement des images (lecture vidéo, histogramme, gradients, filtre de Sobel, etc.)
- Autres librairies :
    - Numpy : Calculs vectorisés, manipulation matricielle, etc.
    - MatplotLib : Visualisation et génération de graphiques

Classe EdgeDetector (edgeDetection.py) : Algorithme basé sur la détection des contours

Cette classe s'appuie sur les gradients d'intensité pour détecter les bordures et transitions vidéo

- ***compute_gradients(frame)*** : Calculer les gradients d'intensité d'une image pour identifier les transitions visuelles (contours).
    - Utilise les filtres de Sobel pour calculer les gradients horizontaux (G_x) et verticaux (G_y) d'une image
    - Les gradients mesurent le changement d'intensité entre les pixels voisins, ce qui est utile pour repérer les contours
    - La magnitude du gradient est ensuite calculée pour chaque pixel :

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

- ***dilate_edges(edges, radius)*** : Dilater les bords détectés pour améliorer la détection des transitions.
    - Applique une opération de dilatation sur les bords détectés pour augmenter leur zone d'influence. Cela permet de rendre les transitions moins sensibles aux petites variations.
    - Le rayon de dilatation spécifié par l'utilisateur définit l'étendue de l'effet.

- ***detect_transitions(video_path)*** : Détecter les transitions (fondus ou coupures) dans une vidéo.
    - Analyser chaque image de la vidéo pour détecter des changements brusques dans les gradients d'intensité entre les frames consécutives.
    - Les transitions sont caractérisées par des changements rapides dans les gradients, où une variation importante de la magnitude des gradients entre deux frames successifs indique une coupure ou un fondu.
- ***get_transitions()*** : Retourner les résultats des transitions détectées.
    - La méthode renvoie les listes de fondues et de coupures détectées à partir des gradients calculés et des analyses des transitions vidéo.
- ***plot_edge_change_fraction(fps)*** : Tracer un graphique montrant l'évolution des fractions de changement d'arêtes (Rho) entre trames successives dans une vidéo, afin de visualiser les transitions (fondues et coupures) en fonction de la variation de ces fractions
- ***reset()*** : Réinitialiser les données internes pour une nouvelle exécution de détection.

Classe HistogramBasedDetector (histogramDetection.py) : Algorithme basé sur les histogrammes de couleurs

Cette classe utilise les histogrammes des canaux RGB pour détecter les transitions en calculant des distances entre histogrammes consécutif.

- ***calculate_histogram(frame)*** : Calculer un histogramme normalisé pour chaque canal RGB d'une image.
    - Chaque trame est décomposée en trois canaux (rouge, vert, bleu)
    - Pour chaque canal, un histogramme est créé en comptabilisant le nombre de pixels correspondant à chaque intensité de couleur. L'histogramme est ensuite normalisé (divisé par le nombre total de pixels), afin que la somme des fréquences soit égale à 1.
- ***calculate_distance(hist_a, hist_b)*** : Calculer la distance entre deux histogrammes de couleur pour mesurer les différences entre deux images consécutives.
    - La distance est calculée en utilisant la distance euclidienne entre les histogrammes normalisés $hist_a$ et $hist_b$:

$$d = \sqrt{\sum_{i=1} (hist_a - hist_b)^2}$$

    - Permet d'évaluer les changements dans les couleurs d'une image à l'autre, ce qui peut correspondre à des transitions vidéo.

- ***detect_transition(current_histogram, frame_count)*** : Détecter des transitions (fondus ou coupures) en comparant les histogrammes consécutifs.
    - Compare l'histogramme actuel de l'image avec celui de l'image précédente.
    - Si la distance euclidienne entre les histogrammes dépasse un seuil prédéfini, une coupure est détectée. Si la différence est graduelle (c'est-à-dire qu'elle diminue lentement), un fondu est détecté.
- ***process_video(video_path)*** : Traiter chaque frame de la vidéo et détecter les transitions basées sur les histogrammes de couleur.
    - Lecture de la vidéo image par image.
    - Pour chaque image les histogrammes sont calculés et comparés avec l'image précédente pour détecter les transitions (fondues ou coupures).
- ***get_transitions()*** : Retourner les listes des coupures et des fondus détectés.
    - Les transitions sont retournées sous forme de listes, identifiant les moments où des changements significatifs dans les histogrammes ont été observés.
- ***plot_distances()*** : Tracer un graphique de la distance euclidienne entre les histogrammes consécutifs d'images dans une vidéo, afin de visualiser les transitions (fondues et coupures) détectées par comparaison d'histogrammes.
- ***reset()*** : Réinitialiser les données internes pour une nouvelle exécution de détection.

Classe Combiner (combiner.py)

Cette classe combine les résultats des deux méthodes afin d'améliorer la précision, diminuer les faux positifs et assurer une bonne spécificité.

- **prioritize_fades(edge_fades, histogram_fades)** : Prioriser les fondues détectées par EdgeDetector ou HistogramBasedDetector.
    - Compare les fondues détectées par chaque méthode en utilisant une tolérance. Si une fondu détectée par HistogramBasedDetector n'est pas déjà présente dans les fondues de EdgeDetector (selon la tolérance), elle est ajoutée à la liste des fondues priorisées.
- **merge_cuts(edge_cuts, histogram_cuts)** : Fusionner les coupures détectées par les deux méthodes.
    - Combine les coupures des deux méthodes et les trie par temps.
    - Supprime les coupures redondantes (celles qui sont trop proches les unes des autres, selon la tolérance spécifiée).
- **compare_results(edge_transitions, histogram_transitions)** : Comparer les transitions détectées pour identifier les résultats communs et spécifiques.
    - Compare les transitions détectées par les deux algorithmes. Si une transition détectée par EdgeDetector est proche d'une transition détectée par HistogramBasedDetector (selon la tolérance), elle est considérée comme une détection commune. Sinon, elle est classée comme spécifique à l'un ou l'autre des algorithmes.

- ***visualize_results()*** : Générer un graphique comparant les transitions détectées (fondues et coupures) à partir de deux méthodes (EdgeDetector et Histogram) avec les transitions de référence (Ground Truth). Le graphique affiche ces transitions en fonction du temps
- **process_video(video_path)** : Exécuter les deux algorithmes sur une vidéo et combiner les résultats.

Autre fonction

1. qaSoccer.py et qaAthletisme.py sont des scripts d'évaluation de la qualité des méthodes de détections des transitions. Celles-ci utilisent les métriques de précision et de rappel, des métriques couramment employées pour quantifier la performance des systèmes de classification [6]. La précision et le rappel sont définies par les équations suivantes :

$$Précision = \frac{Nb\ bonne\ transitions\ détectées}{Nb\ total\ transitions\ détectées}$$

$$Rappel = \frac{Nb\ bonne\ transitions\ détectées}{Nb\ total\ de\ transitions\ (gt)}$$

   Ces deux mesures sont essentielles pour comprendre l'efficacité d'un modèle, en particulier dans des situations où l'on cherche un équilibre entre identifier toutes les occurrences d'un événement (rappel) et éviter les fausses alarmes (précision). Le code sépare d'abord les vérités terrain et les détections en fonction du type d'événement (Coupure ou Fondu), puis compare les détections aux vérités terrain en utilisant une tolérance temporelle de 0,5 seconde pour déterminer les correspondances. Pour chaque type d'événement, les valeurs de précision et de rappel sont calculées et affichées pour chaque détecteur. Ces résultats sont ensuite visualisés à l'aide de diagrammes à barres.

Question 4 (/4) À partir du code informatique de la question 3, générer des résultats permettant d'évaluer la qualité de la solution. Utiliser des graphiques ou des tableaux.

Les vérités terrains, représentant les transitions de type coupure et fondu, ont été partiellement fournies pour la vidéo d'athlétisme et déterminées manuellement pour la vidéo de soccer ainsi que pour le reste de la vidéo d'athlétisme. Ces vérités terrains ont servi à évaluer la méthode de détection et de classification développée. La méthode « visualize_results() » de la classe Combiner a permis de comparer les vérités terrains avec les transitions détectées par l'algorithme proposé, tandis que la méthode « plot_distances() » de la classe « HistogramBasedDetector » a été utilisée pour visualiser les distances euclidiennes entre les histogrammes de couleurs en les superposant aux vérités terrains. Parallèlement, la méthode « plot_edge_change_fraction(fps) » de la classe « EdgeDetector » a généré des figures représentant les arêtes entrantes et sortantes, alignées avec les vérités terrains de chaque vidéo.

Les vérités terrains, soient les transitions de coupure et de fondus ont été partiellement fournis pour la vidéo d'athlétisme et déterminés manuellement pour la vidéo soccer et le reste de la vidéo d'athlétisme. Ces vérités terrains ont été utilisés afin d'évaluer la méthode de détection et de classification proposée. La méthode « visualize_results() » de la classe « Combiner » a été utilisé afin de comparer les vérités terrains aux transitions détectés par l'algorithme. La méthode « plot_distances() » de la classe « HistogramBasedDetector » a été utilisé afin de visualiser les distances euclidiennes entre les histogrammes de couleurs avec les vérités terrains. La méthode « plot_edge_change_fraction(fps) » de la classe « EdgeDetector » a été utilisé afin de générer des figures superposant les arêtes entrantes et sortantes avec les vérités terrains de chaque vidéo Les scripts qaSoccer.py et qaAthletisme.py ont permis de tester l'efficacité globale de la méthode combinée ainsi que celle de ses sous-algorithmes indépendants. Leur fonctionnement a été décrit dans la section précédente.

Les vérités terrains, obtenues par analyse humaine et synthétisées dans les tableaux 2 et 3, présentent les transitions manuellement identifiées pour les vidéos d'athlétisme et de soccer respectivement. Ces données ont servi de référence pour valider la performance des algorithmes testés. Les figures générées offrent donc une évaluation visuelle et quantitative de la qualité de la solution.

Temps (s)	Type de transition
0.50	F
3.63	C
6.88	C
9.58	F
11.38	F
17.08	C
17.62	C
24.63	C
32.63	C
41.67	F
46.83	C
49.46	C
52.33	C
55.54	C

Tableau 2 : Vérités terrain de la vidéo d'athlétisme. F : Fondu, C : Coupure

Temps (s)	Type de transition
2.30	C
4.50	C
8.80	C
10.70	C
12.03	C
13.10	C
14.66	C
19.16	F
19.53	C
21.43	C
27.33	C
32.46	C
33.96	C
37.70	C
38.70	C
45.26	C
46.73	C
49.23	F
52.60	C
55.40	C
56.66	C
61.20	F
61.40	C
66.00	C
71.00	C
72.63	C
77.70	C
79.70	C
85.20	C
87.03	C
89.40	C
91.33	C
93.43	C
93.76	C
99.33	C

Tableau 3 : Vérités terrain de la vidéo de soccer. F : Fondu, C : Coupure

Les figures 1 et 2 illustrent les arêtes entrantes (ρ_{in}) et sortantes (ρ_{out}) entre chaque trame des vidéos d'athlétisme et de soccer, respectivement. Ces valeurs sont calculées à l'aide des formules présentées dans la section 2, en utilisant la méthode de détection des contours. Le seuil gaussien de détection y est également représenté, définissant le critère de considération pour les transitions de type coupure.

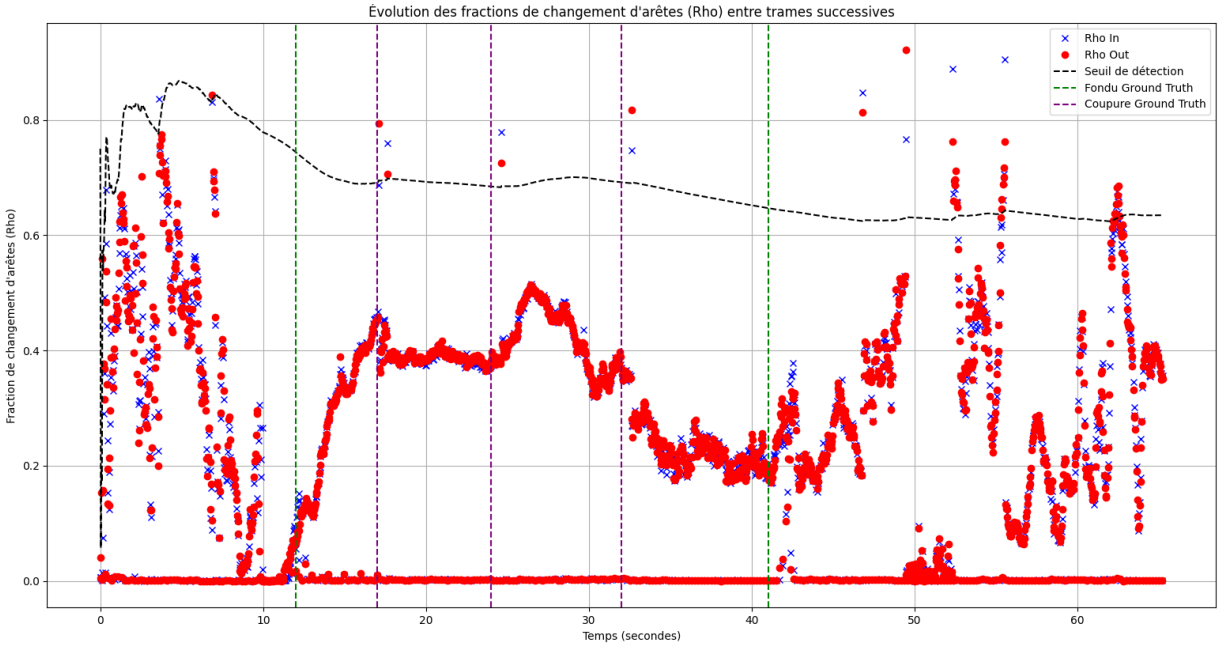

Figure 1 : Arrêtes entrantes (ρ_{in}) et sortantes (ρ_{out}) en fonction du temps(s), déterminées par la méthode de détection des contours pour la vidéo d'athlétisme.

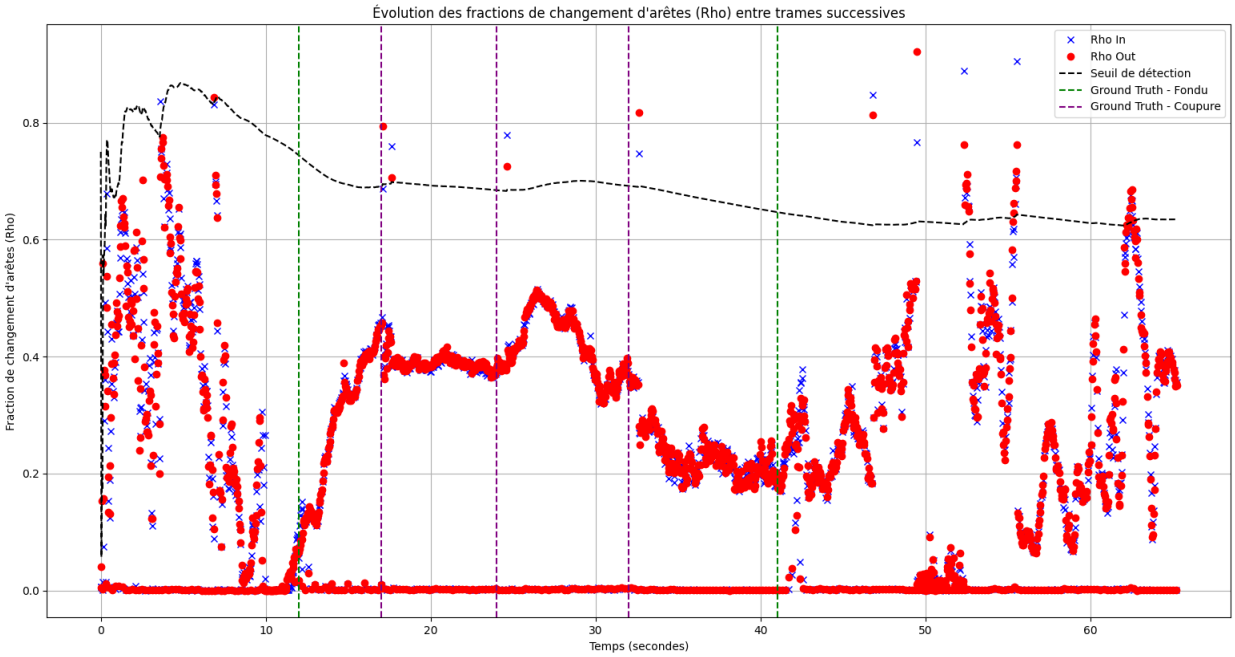

Figure 2 : Arrêtes entrantes (ρ_{in}) et sortantes (ρ_{out}) en fonction du temps(s), déterminées par la méthode de détection des contours pour la vidéo de soccer.

Les figures 3 et 4 présentent la distance euclidienne entre les histogrammes des trames consécutives pour les vidéos d'athlétisme et de soccer, respectivement. Ces distances sont obtenues grâce à la méthode des histogrammes de couleurs.

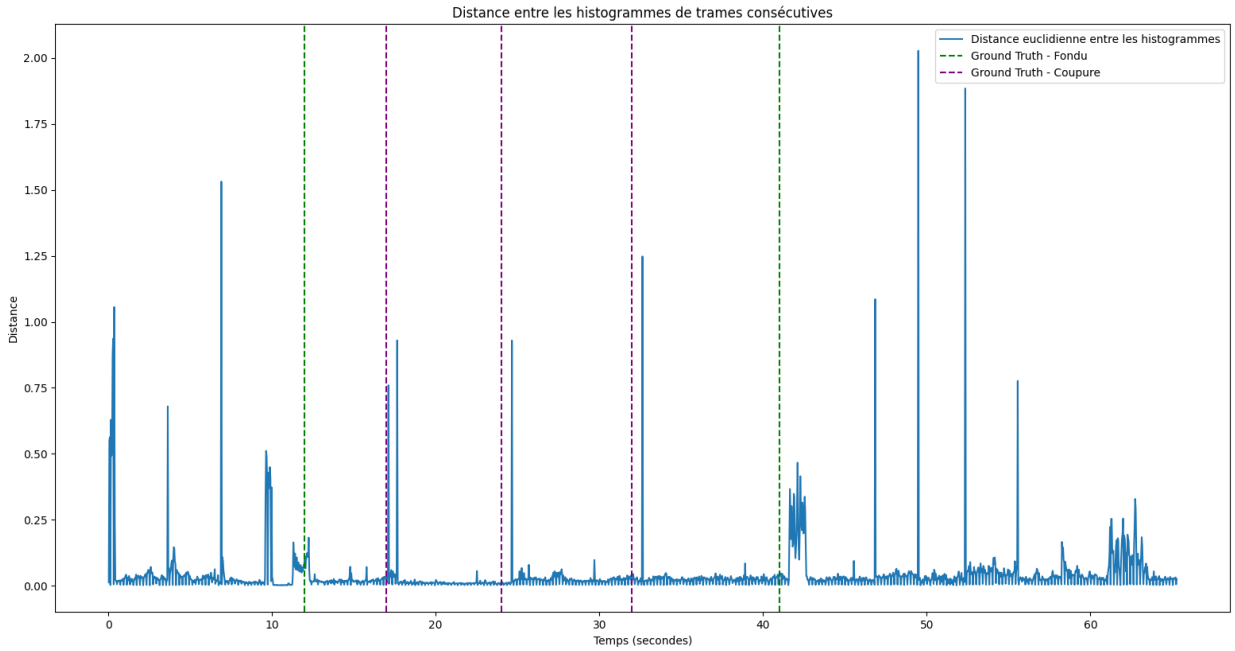

Figure 3 : Distance euclidienne entre les histogrammes des trames consécutives en fonction du temps (s), obtenue par la méthode des histogrammes de couleurs pour la vidéo d'athlétisme.

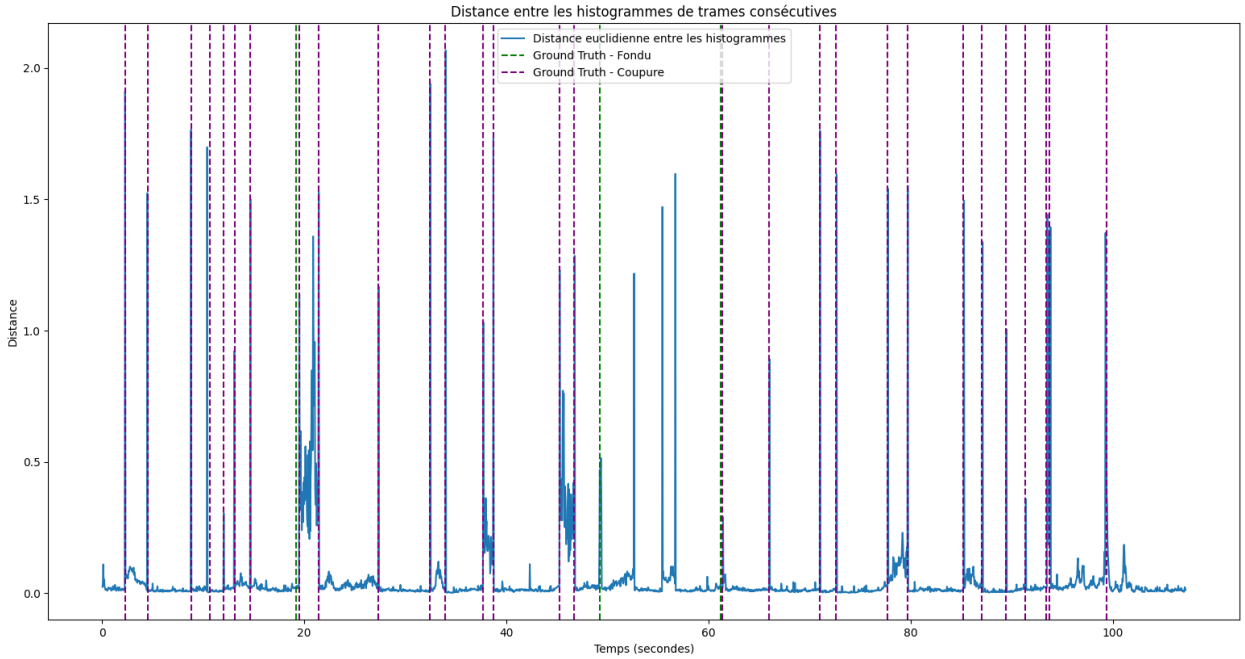

Figure 4 : Distance euclidienne entre les histogrammes des trames consécutives en fonction du temps (s), obtenue par la méthode des histogrammes de couleurs pour la vidéo de soccer.

Les figures 5 et 6 montrent les transitions détectées par les méthodes de détection des contours et des histogrammes de couleurs, superposées aux vérités terrains correspondantes. Ces figures sont générées par la classe « Combiner ».

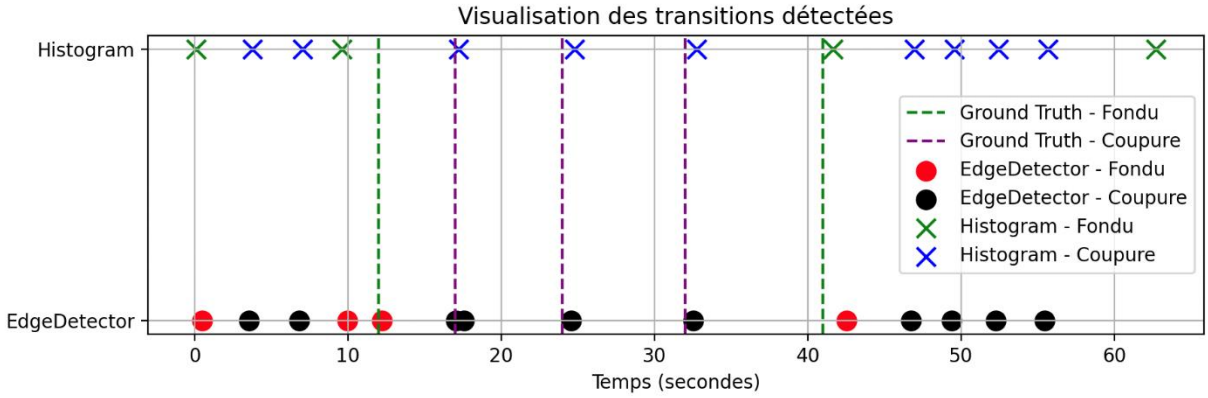

Figure 5 : Types de transitions détectées par la méthode des histogrammes de couleurs et par la méthode de détection des contours, superposés aux vérités terrains pour la vidéo d'athlétisme.

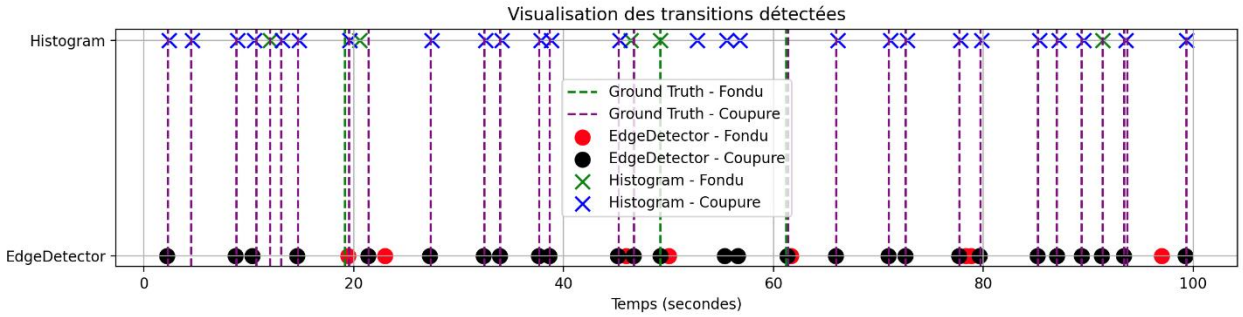

Figure 6 : Types de transitions détectées par la méthode des histogrammes de couleurs et par la méthode de détection des contours, superposés aux vérités terrains pour la vidéo de soccer.

Enfin, les figures 7 et 8 présentent les métriques de précision et de rappel pour les méthodes de détection des contours, des histogrammes de couleurs, ainsi que pour la méthode combinée. Les équations et scripts utilisés pour ces calculs ont été détaillés dans la section précédente.

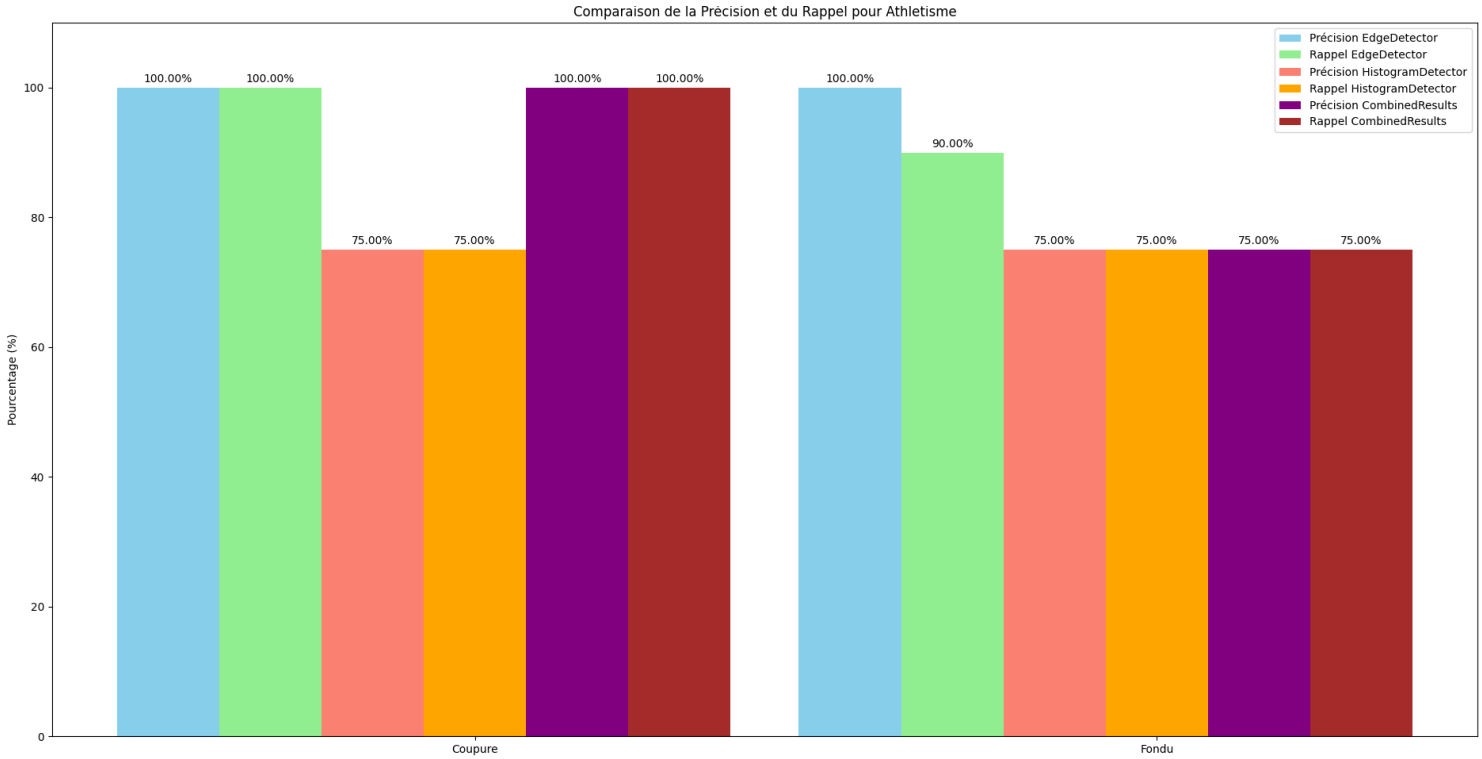

Figure 7 : Précision (%) et rappel (%) pour la vidéo d'athlétisme, selon les transitions de type coupure et fondu, pour les trois méthodes analysées

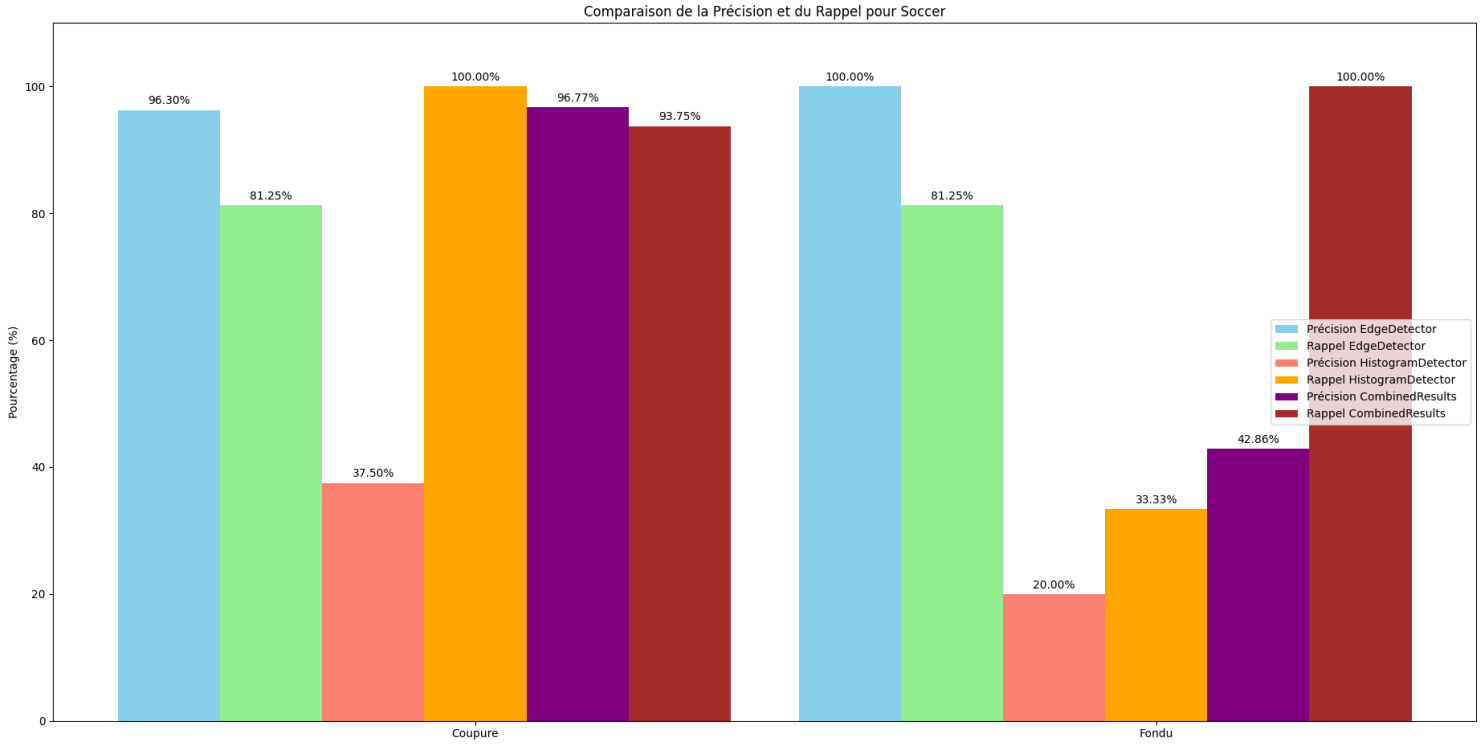

Figure 8 : Précision (%) et rappel (%) pour la vidéo de soccer, selon les transitions de type coupure et fondu, pour les trois méthodes analysées

Question 5 (/4, 2 points par qualité) Analyser les résultats obtenus à la question 4 de manière critique et identifier les limites de la solution proposée.

L'objectif de ce rapport était d'identifier les difficultés spécifiques à deux séquences vidéo (athlétisme et soccer) afin de développer et de tester une méthode combinée de détection des transitions basée sur l'histogramme de couleur (HC) et la détection des contours avec le filtre de Sobel (DC). Nous avons évalué la précision et le rappel pour chaque type de transition (coupure et fondu) afin de valider la pertinence et la robustesse de notre approche.

Pour la vidéo d'athlétisme, la méthode combinée a atteint un taux de précision et de rappel de 100 % pour les transitions de type coupures, égalant ainsi la performance de la méthode HC seule et surpassant celle de la méthode DC seule, qui se limite à 75 %. En revanche, pour les transitions de type fondu, la méthode combinée affiche des taux de rappel et de précision de 75 %, inférieurs à ceux de la méthode HC seule, qui obtient respectivement 90 % et 100 %. Ces résultats indiquent que la combinaison des deux approches n'apporte pas de bénéfices significatifs pour les fondus. Cependant, il convient de noter que la vidéo d'athlétisme contient un faible nombre de transitions (10 coupures et 4 fondus), ce qui réduit la robustesse statistique des résultats. De plus, des variations mineures dans le moment exact de détection des transitions peuvent avoir un impact disproportionné sur les taux de rappel et de précision, particulièrement pour les transitions progressives comme les fondus.

Dans le cas de la vidéo de soccer, les résultats offrent une perspective plus détaillée. Pour les transitions de type coupures, la méthode combinée atteint une précision de 96,77 % et un rappel de 93,75 %, surpassant légèrement la méthode HC seule (96,30 % de précision et 81,25 % de rappel) et nettement la méthode DC seule, qui obtient une précision de 37,50 % mais un rappel maximal de 100 %. Pour les transitions de type fondu, la méthode combinée affiche une précision de 42,86 % et un rappel de 100 %, tandis que la méthode HC seule atteint une précision supérieure de 100 %, mais un rappel légèrement inférieur de 81,25 %. La méthode DC seule présente des performances faibles pour les fondus, avec une précision de 20 % et un rappel de 33,33 %. Ces résultats montrent que la méthode combinée est particulièrement robuste pour les transitions brusques (coupures), mais qu'elle est pénalisée pour les transitions progressives (fondus), probablement en raison d'une intégration insuffisamment optimisée des contributions des descripteurs HC et DC.

Ces observations mettent en évidence plusieurs limites importantes. Premièrement, le faible nombre de transitions dans certaines vidéos, comme celle d'athlétisme, réduit la robustesse statistique des métriques de performance, particulièrement pour les transitions de type fondu. Deuxièmement, la méthode combinée semble mieux adaptée aux transitions brusques qu'aux transitions progressives, ce qui pourrait s'expliquer par une faible sensibilité de la détection des contours aux variations graduelles. Enfin, les différences dans les moments de détection des transitions ont un effet amplifié sur les mesures de rappel et de précision, introduisant une sensibilité indésirable dans les résultats. Par exemple, comme

illustré dans la figure 5, le fondu à 41,67 s est correctement détecté par la méthode DC, mais avec un retard par rapport au moment réel. Ce type d'erreur pourrait être attribué à une implémentation suboptimale de la méthode DC et à sa sensibilité réduite aux transitions progressives. La complexité plus élevée de l'implémentation de DC, par rapport à HC, pourrait également avoir impacté la performance globale de la méthode combinée.

En conclusion, bien que la méthode combinée proposée démontre une efficacité notable pour les transitions de type coupures, elle présente des limites importantes pour les transitions de type fondu. Elle constitue néanmoins une excellente approche pour des vidéos comportant un grand nombre de coupures, grâce à ses taux élevés de précision et de rappel. Toutefois, la méthode HC, plus simple à implémenter, se révèle être un choix compétitif, particulièrement pour des vidéos présentant de nombreux fondus ou une combinaison des deux types de transitions. Ces résultats mettent en évidence la nécessité d'ajuster et d'enrichir la méthode combinée pour améliorer sa polyvalence et son adaptabilité aux caractéristiques variées des transitions vidéo. Une telle démarche contribuerait à renforcer sa robustesse et à élargir son champ d'application.

Bibliographie

[1] S. Tippaya, T. Tan, M. Khan, and K. Chamnongthai, "A study of discriminant visual descriptors for sport video shot boundary detection," 2015.

[2] S. H. Abdulhussain, A. R. Ramli, M. I. Saripan, B. M. Mahmmod, S. A. R. Al-Haddad, and W. A. Jassim, "Methods and Challenges in Shot Boundary Detection: A Review," *Entropy,* vol. 20, no. 4, p. 214, 2018. [Online]. Available: https://www.mdpi.com/1099-4300/20/4/214.

[3] T. Kar, P. Kanungo, S. N. Mohanty, S. Groppe, and J. Groppe, "Video shot-boundary detection: issues, challenges and solutions," *Artificial Intelligence Review,* vol. 57, no. 4, p. 104, 2024/03/30 2024, doi: 10.1007/s10462-024-10742-1.

[4] J. Mas and G. Fernandez, "Video Shot Boundary Detection Based on Color Histogram.," 2003.

[5] H. Wei Jyh and N. N. King, "High accuracy flashlight scene determination for shot boundary detection," *Signal Processing: Image Communication,* vol. 18, no. 3, pp. 203-219, 2003, doi: https://doi.org/10.1016/S0923-5965(02)00139-X.

[6] S. Somnugpong and K. Khiewwan, "Content-based image retrieval using a combination of color correlograms and edge direction histogram," 2016.