



UNIVERSITÉ DE MONTPELLIER
Faculté des Sciences

Master 2 Informatique IMAGINE

Étude bibliographique et expérimentale d'un super lecteur vidéo

Effectué au **LIRMM** (équipe ICAR)
du 20 janvier 2025 au 11 juillet 2025

par **Benjamin Serva**

Encadrant Industriel : Olivier Strauss
Encadrant Universitaire : Madalina Croitoru

Table des matières

1	Introduction	3
2	Glossaire	4
3	Le Projet Numalyse	5
3.1	Introduction du Projet	5
3.2	L'équipe ICAR	5
3.3	Implication de l'équipe ICAR dans le projet Numalyse	5
4	Phase 1 : Conception Logiciel	7
4.1	État de l'art	7
4.2	Conception d'un logiciel v1	8
5	Phase 1 bis : Segmentation Automatique	16
5.1	Etat de l'art	16
5.2	Algorithme	17
5.3	Évaluation des méthodes	17
5.4	Validation par cas réel	21
6	Phase 2 : Extraction automatisée d'image clé	23
6.1	État de l'art	23
6.2	Méthodes sélectionnées	24
6.3	Détail des trois méthodes sélectionnées	25
6.4	Implémentation	26
6.5	Amélioration d'une méthode	31
6.6	Formule Final	35
6.7	Résultats	35
6.8	Création d'une variante	38
6.9	Élaboration d'un questionnaire	42
6.10	Bilan	43
7	Conclusion et perspectives	45
8	Annexes	46
8.1	Diagramme de Gantt	46
9	Bibliographie	47

1 Introduction

Ce stage a été réalisé dans le cadre de l’UE **Stages industriels – HAI002I**, au semestre 2 de la dernière année du master IMAGINE. Il s’est déroulé au LIRMM (Laboratoire d’Informatique, de Robotique et de Microélectronique de Montpellier), dans l’équipe ICAR, du 20 janvier 2025 au 11 juillet 2025.

Dans un environnement où la création et l’analyse des contenus audiovisuels prennent une place toujours plus importante, disposer d’outils de lecture et d’exploitation adaptés devient essentiel. Les œuvres cinématographiques, qu’elles soient destinées à un public de professionnels ou d’amateurs, se caractérisent par leur richesse et leur complexité temporelle. Pour en extraire toute la valeur, qu’il s’agisse d’une simple consultation, d’une annotation critique ou d’une étude approfondie des mouvements de caméra et des éléments de scène, il faut dépasser les fonctionnalités basiques des lecteurs vidéo classiques.

C’est dans ce contexte que s’inscrit la tâche 2 du projet NUMALYSE et, plus précisément, mon stage intitulé ”Étude bibliographique et expérimentale d’un super lecteur vidéo”. L’objectif global est de concevoir une première version d’un logiciel intégrant non seulement les fonctions de base d’un lecteur, mais aussi la capacité d’afficher simultanément plusieurs flux vidéo, d’extraire facilement des plans, des images fixes ou des extraits sonores, et de proposer une interface d’annotation fine. Au-delà de la simple navigation, ce ”super lecteur” vise à intégrer des modules d’analyse objective : segmentation de plan, détection et reconnaissance d’éléments ou de types de scènes, identification de personnages, et détection des mouvements de caméra, le tout potentiellement enrichi par des algorithmes d’intelligence artificielle.

Le stage se décompose en trois grandes phases :

- Étude bibliographique : Recenser et évaluer les méthodes et outils existants pour chacune des fonctions visées (lecture multi-flux, extraction de plans, annotation, analyse d’image et de mouvement).
- Développement du prototype : Concevoir et implémenter un logiciel prototype, base technique sur laquelle s’appuieront les travaux futurs du LIRMM.
- Extraction automatisée de *tag image* : Élaborer et expérimenter une méthode nouvelle permettant l’extraction automatisée de *tag image* pour n’importe quel plan.

Tout au long de mon stage, j’ai été accompagné par mes trois encadrants : Monsieur **Olivier Strauss**, Monsieur **William Puech** et Monsieur **Frédéric Comby**. Grâce à mes comptes rendus hebdomadaires et aux réunions bimensuelles au cours desquelles je présentais en détail l’avancement de mon travail, ils ont pu suivre mon projet de manière assidue et me guider efficacement.

2 Glossaire

- **Frame** : ça représente une image dans une vidéo.
- **Plan** : c'est un ensemble de frame qui est compris entre deux transitions.
- **Séquence** : c'est un ensemble de plan, par exemple une discussion entre deux protagonistes.
- **Tag Image** : cette appellation a été déterminé de façon commune pour nommer l'image la plus représentative d'une vidéo (non pas représentative de son contenu mais plutôt de ce qu'elle est).
- **SIFT** : Scale-Invariant Feature Transform, un algorithme pour détecter et décrire des points d'intérêt dans les images.
- **HSV** : Un espace colorimétrique qui décrit les couleurs en termes de Teinte (Hue), Saturation, et Valeur.

3 Le Projet Numalyse

3.1 Introduction du Projet

Le projet Numalyse est intitulé "Analyser les oeuvres cinématographiques et audiovisuelles en contexte numérique. Nouveaux outils, nouvelles pratiques." il a une durée de 4 ans et il est coordonnée par Loig LE BIHAN.

Ce projet est scindé en différentes tâches :

- Tâche 0 : Coordination Générale.
- Tâche 1 : Contextualiser, critiquer, enquêter.
- Tâche 2 : Concevoir de nouveaux outils logiciels.
- Tâche 3 : Expérimenter de nouvelles pratiques.
- Tâche 4 : Diffuser, valoriser.

3.2 L'équipe ICAR

L'équipe ICAR (Image & Interaction) dont le responsable est Olivier Strauss, regroupe des chercheurs des deux départements Robotique et Informatique autour de la thématique "image" et plus généralement des données visuelles. Elle est composée actuellement de 9 permanents, universitaires et CNRS mais compte aussi dans ses collaborateurs réguliers, plusieurs médecins hospitalo-universitaires du CHU utilisant l'imagerie médicale, des chercheurs en télédétection du laboratoire TETIS ou en modélisation pour l'agronomie du CIRAD.

L'équipe ICAR développe des thèmes de recherche associant l'interaction et le traitement des données visuelles telles que les images 2D, 3D, multi-spectrales (nD), les vidéos ou les séquences d'images nD+t et les objets 3D que ce soit sous forme de maillages 3D ou de modélisations paramétriques.

L'équipe est structurée suivant 4 axes de recherche :

- Analyse & traitement
- Sécurité Multimédia
- Modélisation & Visualisation
- Intelligence Artificielle pour les données visuelles

3.3 Implication de l'équipe ICAR dans le projet Numalyse

L'équipe ICAR a été sollicitée pour intervenir dans la tâche 2 pour la conception d'un super lecteur vidéo ainsi que pour développer de nouvelles méthodes IA.

Durant ce stage mes encadrants et moi avons fait des réunions mensuelles avec Monsieur LE BIHAN et ses nombreux collaborateurs pour d'une part présenter l'évolution du travail mais aussi pour mieux comprendre leur besoin, leur façon de travailler. Le défi principal était de se comprendre, mais aussi de répondre au mieux à leur besoin tous en leur faisant prendre conscience de l'enjeu de leur demande.

J'ai aussi pu assister à trois webinaires différents dans le cadre du projet:

1. 31 janvier 2025 : Laurent Tessier & Michaël Bourgatte, "Celluloid, un écosystème pour instrumenter la recherche sur les corpus audiovisuels."
2. 28 mars 2025 : Jacques Perconte, "Flux, vingt-cinq ans de danse avec les algorithmes de compression vidéo."
3. 16 mai 2025 : Philippe Boisnard, "Outils créés, outils détournés : Approches épistémologiques et sociotechniques des outils d'analyse contemporains."

Deux d'entre eux mon étaient utile car c'était la présentation de leur logiciel et donc j'ai pu m'en inspirer et avoir un partage de leur expérience sur les erreurs à ne pas faire etc ...

4 Phase 1 : Conception Logiciel

Avant même de commencer à concevoir un logiciel, il est essentiel de s'informer sur les solutions existantes. Dans cette optique, j'ai effectué une recherche sur les logiciels de lecture vidéo les plus répandus, tout en consultant également la liste des outils suggérés comme exemples dans les missions du projet Numalyse. Cette phase de recherche m'a permis de réaliser un état de l'art des outils les plus courants et les plus répandus dans ce domaine. L'objectif était d'identifier les fonctionnalités pertinentes, les limites des solutions existantes, ainsi que les besoins non couverts, afin d'orienter au mieux la conception du logiciel.

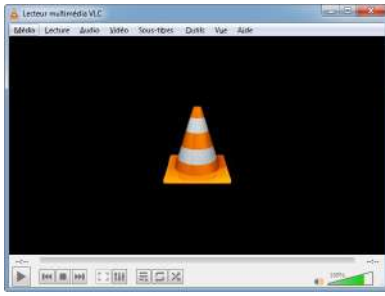
4.1 État de l'art

Pour faciliter la lecture et la comparaison entre ces logiciels, j'ai synthétisé les résultats de cette recherche sous forme de tableau récapitulatif, mettant en évidence leurs principales caractéristiques.

Logiciel	Plateforme	Langages	API	IA	Payant
VLC	Multi-plateforme	C, C++, Obj-C	✓	✗	✗
Fonctionnalités : Lecture robuste, streaming, effets audio/vidéo, sous-titres avancés					
QuickTime	macOS, Windows	C	✓	✗	✗
Fonctionnalités : Lecture fluide, édition simple, chapitres/sous-titres, codecs Apple					
Screen	macOS	C++, Python	✓	✗	✓(176€)
Fonctionnalités : Annotation vidéo, comparaison multiple, zoom, calques, grille, intégration API					
IINA	macOS	Swift	✗	✗	✗
Fonctionnalités : Interface moderne, PiP, Touch Bar, raccourcis clavier, basé sur MPV					
MPV	Multi-plateforme	C	✓	✗	✗
Fonctionnalités : Interface minimale, HDR, scripting Lua/Python, accélération matérielle					
DaVinci Resolve	Multi-plateforme	C++, Python	✓	✓	✓(Studio)
Fonctionnalités : Montage pro, IA (visage, dialogue, tri), étalonnage, collaboration multi-utilisateur					

Table 1: Comparaison synthétique des logiciels de traitement vidéo

Pour compléter cette analyse, une illustration présente les interfaces des logiciels étudiés afin de visualiser et s'inspirer des choix ergonomiques utilisés.



(a) VLC



(b) QuickTime



(c) Screen



(d) IINA



(e) MPV



(f) DaVinci Resolve

Figure 1: Interfaces des différents logiciels comparés

4.2 Conception d'un logiciel v1

4.2.1 Objectif de la conception

Chaque personne a sa propre manière d'analyser un film, même si certaines pratiques sont largement partagées. Comme nous avons pu le voir dans l'état de l'art, il existe une grande variété de logiciels de lecture vidéo, dont certains proposent des fonctionnalités utiles pour l'analyse cinématographique. Cependant, aucun ne s'avère véritablement complet.

L'objectif de ce logiciel est donc de regrouper l'ensemble de ces pratiques afin de proposer une solution à la fois complète et robuste pour l'analyse de films. Il vise également à toucher un large public dans le domaine de l'analyse cinématographique.

De nombreux projets de logiciels d'analyse filmique ont vu le jour par le passé, mais beaucoup ont disparu, soit par manque de notoriété, soit par abandon, les rendant obsolètes. L'un des objectifs de ce projet est donc d'assurer la pérennité du logiciel.

À la suite de plusieurs réunions avec les membres du projet Numalyse, nous avons pu identifier un certain nombre de fonctionnalités essentielles :

- La possibilité de faire des captures d'écran et des enregistrements vidéo.
- La lecture synchronisée de plusieurs vidéos dans une même fenêtre, une fonctionnalité rare et peu proposée par les logiciels actuels, souvent complexe à mettre en place car elle peut nécessiter l'utilisation de scripts spécifiques.
- Un système de segmentation automatique, c'est-à-dire la détection automatisée des plans. Il existe différents niveaux de segmentation, qui seront détaillés ultérieurement. Il est également important de pouvoir ajouter des annotations à chaque plan.
- Enfin, la possibilité d'exporter le travail réalisé, soit sous forme textuelle, soit sous forme de vidéo.

4.2.2 Technique

Le développement de l'application s'appuie sur le langage Python. Pourquoi ce choix ? D'une part, Python facilite grandement le développement grâce à sa simplicité, et d'autre part, il dispose d'un vaste écosystème de bibliothèques, notamment pour le traitement d'images et de vidéos. Malgré cela, il reste possible d'intégrer ou d'appeler du code en C++ lorsque des performances d'exécution supérieures sont requises.

De plus, Python est facilement utilisable sur l'ensemble des systèmes d'exploitation (Windows, Linux, macOS), ce qui facilite grandement la génération d'exécutables à l'aide de PyInstaller. Cela fait partie des contraintes attendues du projet : pouvoir générer un exécutable pour chacun des principaux systèmes d'exploitation.

Pour la gestion de l'interface utilisateur, j'ai opté pour Qt, une bibliothèque très complète qui permet de concevoir des interfaces graphiques flexibles et modernes.

Enfin, pour tout ce qui concerne la lecture vidéo, j'ai utilisé la bibliothèque python-vlc, qui permet d'interfacer Python avec les plugins du lecteur multimédia VLC. Ce choix était le plus logique en raison de la robustesse et de la polyvalence de VLC, largement reconnu pour sa capacité à lire un très large éventail de formats vidéo et audio, y compris dans des conditions où d'autres solutions échouent.

Un autre atout majeur de cette bibliothèque est son excellente compatibilité multiplateforme. Dans le cadre de ce projet, il est nécessaire de développer un logiciel capable de fonctionner sur Windows, macOS et Linux. La combinaison de python-vlc avec Python et Qt répond parfaitement à cette exigence, en permettant une intégration fluide entre l'interface graphique et le moteur de lecture vidéo. De plus, l'utilisation de VLC offre un accès à des fonctionnalités avancées comme la gestion des pistes, des sous-titres, ou des flux réseaux.

4.2.3 Fonctionnalités Implémentées

Lecteur classique

1. Fonctionnalités basiques

- Chargement d'une vidéo
- Lecture, pause, arrêt, activation/désactivation du son, vitesse de lecture.
- Navigation directe vers un point spécifique via le timecode
- Barre de progression permettant de naviguer rapidement dans la vidéo
- Capture d'écran et capture vidéo avec nommage automatique dans le format `titre_-timecode` en sachant que le timecode est représenté de cette manière `H:M:S[Frame]`
- Gestion des sous titres, il suffit que le fichier ".srt" soit dans le même dossier que le fichier vidéo et comporte le même nom.



Figure 2: Aperçu

2. Segmentation automatique

- timeline zoomable qui représente l'ensemble des plans de la vidéo.
- mise en évidence en rouge du plan correspondant au passage de la vidéo
- renommage, suppression, modification du temps de début et de fin, extraction de la séquence, ajout de séquence/bouton
- ajout de notes sur une séquence (modifiable et supprimable)

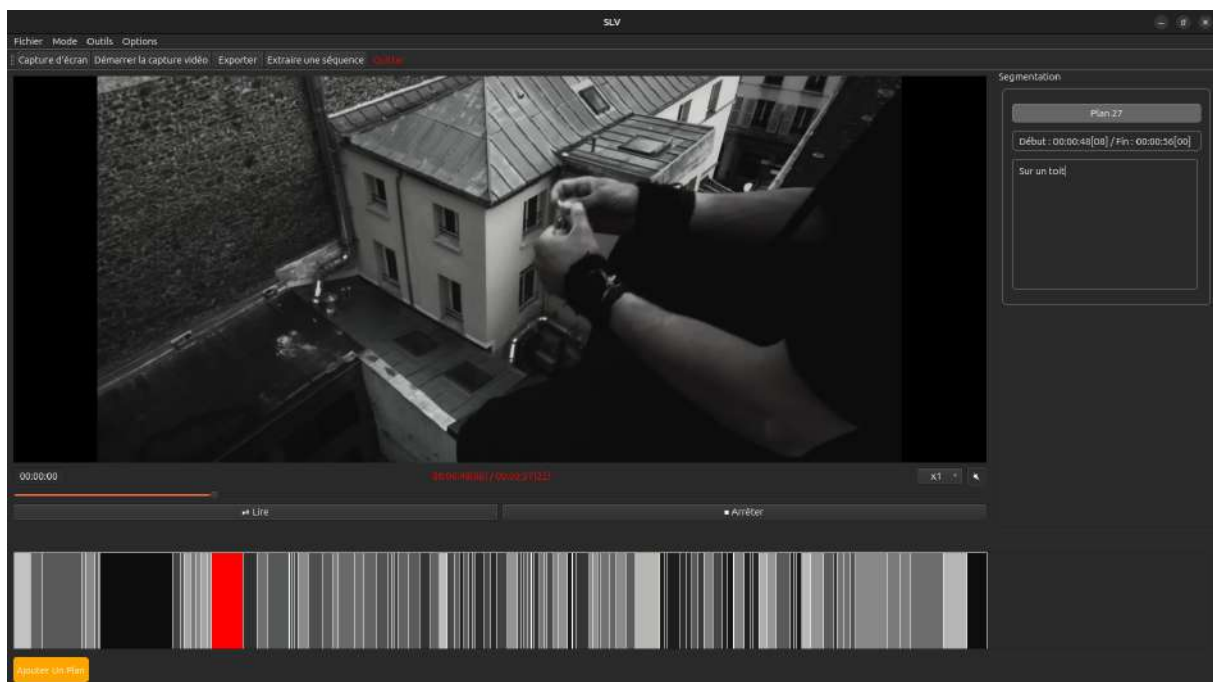


Figure 3: Aperçu du mode de segmentation

3. Menu d'options

- Options pour l'exportation : permet de choisir le format pour l'exportation textuelle, c'est à dire pdf ou docx ou odt.
 - Options pour les captures d'écran :
 - Format : png ou jpeg.
 - Post-traitement : avec ou sans rehaussement de contraste.
4. Capture d'écran avec post-traitement. Si l'option est sélectionné alors : détaillé la formule de l'ajustement gamma
- Lecteur classique : un slider apparaît avec un résultat temps réel qui permet d'ajuster correctement notre contraste.

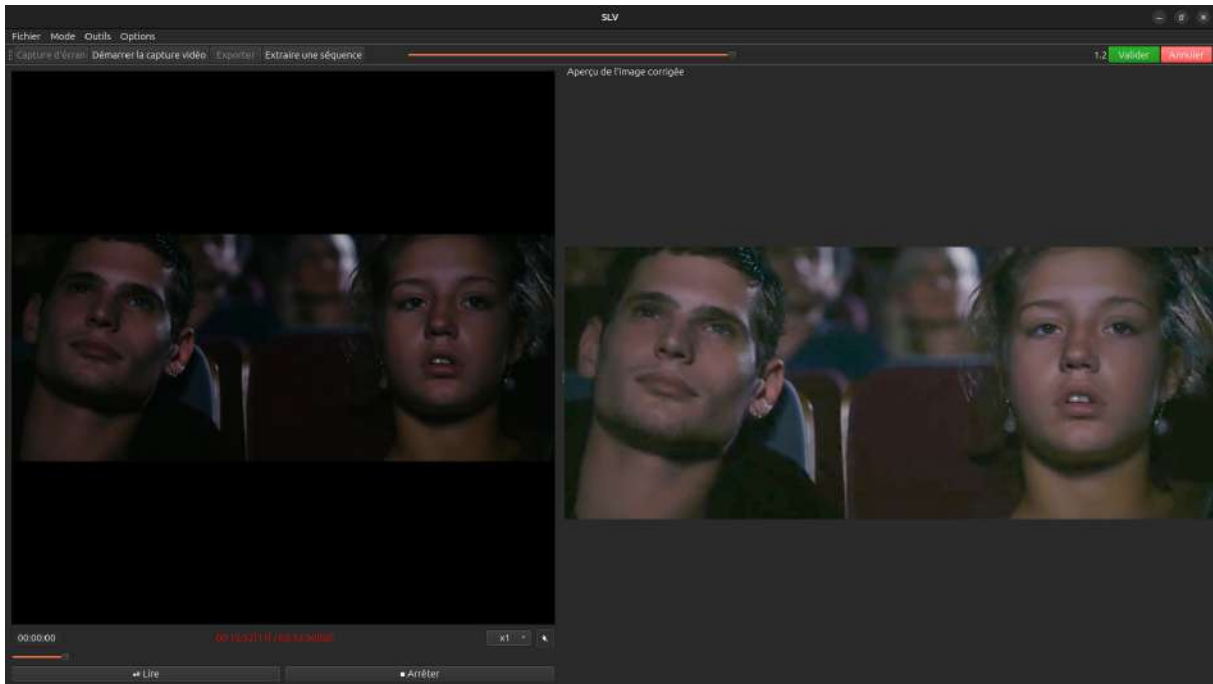


Figure 4: Aperçu rehaussement de contraste temps réel

- Lecteur Synchronisé : fais de façon automatique avec une valeur fixée.
5. Extraction : permet d'extraire un passage de la vidéo.
6. Gestion de projets
- Système de sauvegarde qui enregistre dans un dossier une copie de la vidéo et dans un fichier .json toutes les séquences et annotations. Ce qui permet un partage du travail effectué.
 - Système de chargement d'un projet qui permet de travailler en plusieurs fois.
7. Système d'exportation du travail
- Dans un fichier pdf avec les séquences détecté/ajoutés et leur timecode associé ainsi que les différentes notes. Mais aussi une image représentative de la séquence qui est pour l'instant déterminer de façon naïve.

Étude cinématographique

- Plan 1 -> Début : 00:00:00[00] / Durée : 00:00:04[00]

Début du clip



- Début -> Début : 00:00:04[00] / Durée : 00:00:03[00]

Hugo TSR, de son vrai nom Hugo Jehl[1], né le 18 janvier 1985 dans le 18e arrondissement de Paris, est un rappeur et auteur-compositeur-interprète français d'origine franco-japonaise[2], indépendant qui débuta dans l'underground avant de connaître une notoriété de plus en plus importante.



- Séquence Intro -> Début : 00:00:07[00] / Durée : 00:00:09[15]



Figure 5: résultat d'exportation textuelle

- Dans une super vidéo agrémenté des titres des plans ainsi que leurs timecode.



Figure 6: résultat d'exportation video

8. Ajout de raccourci clavier pour faciliter le travail

- Ctrl+X : fermer
- Ctrl+O : charger une vidéo
- Ctrl+A : charger un projet
- Ctrl+S : sauvegarde de projet
- Barre Espace : pour charger une vidéo ou bien lecture/pause quand une vidéo est chargé dans le lecteur.
- Flèche gauche et droite : reculer/avancer de 5 secondes dans la vidéo.

Lecture synchronisée



Figure 7: Aperçu du mode lecture synchronisée

- Possibilité d'utiliser 2 ou 4 sous-lecteurs
- Chaque sous-lecteur dispose des mêmes fonctionnalités que la page principale

- Super bouton "Lire", "Pause", "Arrêt" permettant de contrôler tous les lecteurs simultanément
- Mode plein écran pour une meilleur expérience de visionnage.
- Capture d'écran combinée pour tous les sous-lecteurs



Figure 8: résultat capture d'écran

- Capture vidéo combinée, la vidéo résultante est sans son donc chaque extrait est enregistrée individuellement avec le son.

4.2.4 Fonctionnalités restantes à implémenter

Cette section a pour objectif de présenter plusieurs pistes d'amélioration, formulées au cours des réunions, mais jugées non réalisables dans le cadre de mon stage. Elles sont donc destinées à être mises en œuvre par un futur ingénieur travaillant sur le développement du logiciel dans le cadre du projet Numalyse.

- La génération d'une image des couleurs dominantes du film, comme dans cet exemple <https://www.youtube.com/watch?v=x3sfopqVyDk>.



Figure 9: Exemple de résultat pour le film Blade Runner 2049

- La possibilité d'intégrer des sous-titres à la vidéo, comme dans le logiciel VLC.
- Une timeline audio.

- Un mode débutant (l'application actuelle, c'est-à-dire très restreint) et un mode expert (permettant d'ajuster de nombreux réglages, tels que le taux de compression JPEG, le nommage des captures, la disposition des lecteurs pour la lecture synchronisée, etc...).
- La génération d'une vidéo comportant plusieurs pistes vidéo et audio.

5 Phase 1 bis : Segmentation Automatique

Une des choses clés et essentiels de l'analyse d'un film est la segmentation, ce qui est clairement une tâche contraignante lorsqu'elle doit être faite manuellement. C'est pour ça qu'il fallait mettre en place un système de segmentation automatisé.

5.1 Etat de l'art

5.1.1 La segmentation

Avant toute chose, il est important de comprendre comment se structure un film. Un film est constitué de séquences, elles-mêmes composées de scènes, qui sont elles-mêmes formées de plusieurs plans, séparés les uns des autres par des transitions.

Lorsqu'on cherche à segmenter un film, plusieurs niveaux de segmentation sont possibles. Le niveau le plus bas consiste à identifier chaque plan. Ensuite, on peut regrouper les plans en scènes, puis les scènes en séquences.

Dans le cadre de mon stage, l'objectif est de se concentrer sur une segmentation de bas niveau, c'est-à-dire l'identification de chaque plan. Pour cela, il est nécessaire de développer ou d'utiliser une méthode capable de détecter les différentes transitions entre plans.

5.1.2 Type de transition

Avant même de chercher à les détecter, il est pertinent de comprendre et d'identifier les principales transitions utilisées dans le montage vidéo, afin de mieux adapter les techniques de détection à leurs caractéristiques.

Cut Transition Un changement instantané d'une image à une autre sans effet intermédiaire. C'est la transition la plus rapide et la plus courante dans le montage vidéo.

Fade out L'image s'assombrit progressivement jusqu'à devenir complètement noire (ou d'une autre couleur). Cette transition est souvent utilisée pour marquer la fin d'une séquence ou d'une scène.

Fade in L'inverse du Fade Out : une image noire devient progressivement visible. Elle est souvent utilisée pour introduire une nouvelle scène ou séquence.

Dissolve Une image s'estompe progressivement pendant qu'une autre apparaît en surimpression. Les deux images se chevauchent pendant une courte période. Cette transition est souvent utilisée pour montrer un passage dans le temps ou un lien narratif entre deux scènes.

Wipe Transition Une image est remplacée par une autre selon un mouvement spécifique (ligne, courbe, motif). Contrairement au Dissolve, les deux images ne se mélangent pas mais sont séparées par une bordure en mouvement. Il existe différentes variantes qui sont les suivantes :

Motion from bottom to top, Motion from right to left, Wipe with oblique motion, The movement is going to the center.

5.2 Algorithme

Donc l'objectif va être d'étudier frame par frame la vidéo à segmenter et de détecter un changement important entre deux frames ce que l'on pourra déterminer comme une transition. Pour faire cela il existe une librairie python que l'on pourrait utiliser : PySceneDetect qui propose 5 méthodes différentes.

- Content : Cette méthode détecte les coupes rapides en analysant les différences visuelles entre images consécutives. Elle convertit chaque image en espace colorimétrique HSV et calcule un score de changement basé sur quatre composantes :

- delta_hue : variation de la teinte.
- delta_sat : variation de la saturation.
- delta_lum : variation de la luminosité.
- delta_edges : variation des contours détectés.

Chaque composante est pondérée selon des poids définis par l'utilisateur (par exemple : `-weights 1.0 0.5 1.0 0.2`). Le score final (`content_val`) est comparé à un seuil (`-threshold`, par exemple 27.0) pour déterminer s'il y a une coupure de scène.

- Adaptive : Cette méthode est une version améliorée de la détection basée sur le contenu. Elle applique une moyenne glissante aux scores de changement pour atténuer les faux positifs causés par des mouvements rapides de la caméra ou des variations d'éclairage. Cela permet une détection plus robuste dans des situations dynamiques.
- Hash : Cette approche utilise des hachages perceptuels pour comparer des images consécutives. Elle calcule la distance de Hamming entre les hachages des images et la normalise par la taille du hachage. Si cette distance dépasse un certain seuil, une coupure de scène est détectée. Cette méthode est efficace pour identifier des changements visuels significatifs tout en étant moins sensible aux petites variations.
- Histogram : Cette méthode convertit chaque image en espace colorimétrique YUV et crée un histogramme basé sur le canal Y (luminance). Elle compare les histogrammes des images consécutives pour évaluer leur similarité. Si la différence dépasse un seuil défini (`-threshold`, par exemple 0.1), une coupure de scène est identifiée. Le nombre de bins de l'histogramme peut être ajusté (`-bins`, par exemple 256) pour affiner la détection.
- Threshold : Cette méthode détecte les transitions progressives, telles que les fondus enchaînés, en analysant la luminosité moyenne des images. Elle calcule la moyenne des valeurs RGB de chaque pixel pour obtenir une valeur d'intensité globale. Lorsque cette valeur franchit un seuil spécifique, une transition de scène est détectée. Cette approche est particulièrement utile pour identifier les fondus au noir ou les fondus en ouverture.

5.3 Évaluation des méthodes

Il a donc été nécessaire d'évaluer ces méthodes afin de déterminer si l'utilisation de cette bibliothèque dans le logiciel était suffisante ou, dans le cas contraire, s'il fallait concevoir et implémenter notre propre méthode.

Pour cela, nous avons utilisé deux bases de données composées d'extraits vidéo, chacune accompagnée d'un fichier texte contenant la vérité terrain de la segmentation.

La première, intitulée BBC, est composée de 11 vidéos d'une durée moyenne de 45 à 50 minutes, dont le contenu correspond à des documentaires animaliers ou de paysages.

La seconde, nommée AutoShot, contient 164 vidéos d'une durée variant entre 20 et 60 secondes, avec un contenu de style TikTok coréen (vidéos courtes, dynamiques et montées rapidement).

Voici un échantillon de 4 images extraites d'une vidéo de chacune des deux bases de données.



Figure 10: échantillon de 4 images de la BDD AutoShot



Figure 11: échantillon de 4 images de la BDD AutoShot

Pour les tests, nous avons utilisé les algorithmes avec leurs seuils prédéfinis.

5.3.1 BDD 1 : BBC

Algo	Vrai Positif	Faux Positif	Précision	Rappel	F1 Score
Adaptive	3884	487	88.86%	80.00%	84.20%
Content	3609	1013	78.08%	74.34%	76.16%
Hash	3998	1919	67.57%	82.35%	74.23%
Histogram	3710	2332	61.40%	76.42%	68.09%
Threshold	0	428	0.00%	0.00%	0.00%

Table 2: Comparaison des résultats des algorithmes pour la BDD BBC

Nombre total de plans à détecter : 4855.

5.3.2 BDD 2 : AutoShot

Algo	Vrai Positif	Faux Positif	Précision	Rappel	F1 Score
Adaptive	1116	951	53.99%	49.21%	51.49%
Content	1019	871	53.92%	44.93%	49.01%
Hash	856	827	50.86%	37.74%	43.33%
Histogram	905	1789	33.59%	39.90%	36.48%
Threshold	3	41	6.82%	0.13%	0.26%

Table 3: Comparaison des résultats des algorithmes sur la BDD AutoShot

Nombre total de plans à détecter : 2268.

5.3.3 Analyse et Conclusion

Sur la base de données BBC, on observe que la méthode Adaptive obtient un très bon F1 Score. Globalement, les méthodes Content, Hash et Histogram atteignent un nombre de vrais positifs similaire (plus ou moins 200), mais génèrent un nombre beaucoup plus important de faux positifs.

Par exemple, la méthode Hash détecte une centaine de vrais positifs de plus qu'Adaptive, mais produit également 1 500 faux positifs supplémentaires. Ainsi, même si l'on gagne un peu de temps lors de la détection des plans, on en perd beaucoup lors du tri des erreurs de détection.

Sur la base de données AutoShot, les F1 Scores sont globalement plus faibles, notamment en raison de la forte similarité entre les plans au sein de chaque vidéo. Malgré cela, la méthode Adaptive reste la plus performante, ce qui confirme sa robustesse sur les deux bases de données.

D'après ces tests, on peut en conclure que la méthode à privilégier est Adaptive, car elle maximise le nombre de vrais positifs tout en minimisant celui des faux positifs. C'est exactement ce compromis qui permet de gagner du temps lors de la segmentation automatique.

5.3.4 Constatation

En calculant la segmentation sur un film en noir et blanc, j'ai constaté que la méthode considérée comme la plus performante précédemment ne fonctionnait pas du tout (concrètement elle détecte très peu de transition).

Ce qui reste cohérent car la méthode Adaptive se place dans l'espace HSV et sur un film en noir et blanc la teinte ainsi que la saturation sont nul, donc il y a seulement la luminance (luminosité) qui est étudié et ce n'est pas suffisant pour détecter l'ensemble des transitions existantes.

On pourrait penser que les films en noir et blanc sont rares, mais ce n'est pas le cas. Dans le domaine de l'analyse cinématographique, de nombreux films étudiés ont justement été produits en noir et blanc. Ce cas ne doit donc pas être négligé, mais au contraire être considéré avec autant d'attention que les films en couleur.

J'ai donc converti les deux BDD en niveaux de gris pour relancer les tests. Les résultats sont les suivants :

5.3.5 BDD 3 : BBC Gray

Algo	Vrai Positif	Faux Positif	Précision	Rappel	F1 Score
Adaptive	1304	738	63.86%	26.86%	37.81%
Content	57	165	25.68%	1.17%	2.25%
Hash	3996	1915	67.60%	82.31%	74.23%
Histogram	3682	2408	60.46%	75.84%	67.28%
Threshold	1	412	0.24%	0.02%	0.04%

Table 4: Comparaison des résultats des algorithmes pour la BDD BBC (niveau de gris)

Nombre total de plans à détecter : 4855.

5.3.6 BDD 4 : AutoShot Gray

Algo	Vrai Positif	Faux Positif	Précision	Rappel	F1 Score
Adaptive	836	485	63.29%	36.86%	46.59%
Content	247	348	41.51%	10.89%	17.25%
Hash	860	821	51.16%	37.92%	43.56%
Histogram	939	1700	35.58%	41.40%	38.27%
Threshold	2	44	4.35%	0.09%	0.17%

Table 5: Comparaison des résultats des algorithmes pour la BDD Autoshot (niveau de gris)

Nombre total de plans à détecter : 2268.

5.3.7 Analyse et Conclusion

Sur la base de données BBC Gray, la méthode Hash surpasse clairement les autres en atteignant un F1 Score de 74.23%. Elle détecte donc une grande majorité des plans tout en limitant relativement bien les erreurs. La méthode Histogram suit de près, avec un F1 Score de 67.28%, grâce à un bon équilibre entre rappel et précision. À l'inverse, la méthode Adaptive, qui performait bien sur les autres bases, montre ici une baisse notable avec un rappel très faible, ce qui impacte son F1 Score (37.81%). Cela indique que cette méthode passe à côté d'un grand nombre de plans dans le contexte des images en niveau de gris comme déduit précédemment.

Sur la base de données AutoShot Gray, les performances sont globalement plus faibles comme pour les tests en couleur. Ici encore, Adaptive reste la plus efficace avec un F1 Score de 46.59%, le plus élevé parmi les méthodes testées. La méthode Hash, qui était très performante sur BBC Gray, voit son efficacité diminuer ici avec un F1 Score de 43.56%, proche de celui d'Adaptive mais avec davantage de faux positifs.

En résumé, la méthode Hash s'impose sur la base BBC Gray, tandis que la méthode Adaptive reste la plus robuste et constante, surtout sur AutoShot Gray. Cela montre que l'environnement visuel (couleur vs niveau de gris) influence significativement les performances de chaque méthode. Ainsi, pour des contenus en niveau de gris, Hash est à privilégier si l'on cherche un rappel élevé, tandis que Adaptive reste une solution stable et équilibrée, notamment pour des bases de données hétérogènes ou plus complexes comme AutoShot Gray.

5.4 Validation par cas réel

Pour valider les résultats obtenues j'ai calculé la segmentation sur le film La Vie D'Adèle (fournit par Monsieur Le Bihan) et je l'ai transmis à un littéraire (qui est Hugo Raffaitin stagiaire de Monsieur Le Bihan que je tiens à remercier) qui a corrigé cette segmentation et voici ce qu'on obtient :

- Segmentation automatique : 1670 plans détectés.
 - 1654 correctement détectés/vrai positifs (94%)
 - 16 faux positifs
 - 104 plans manquants/faux négatifs
- Segmentation manuelle (valeur de vérité) : 1758 plans détectés.
- Précision : 99.04%
- Rappel : 94.08%
- F1 Score : 96.49%

5.4.1 Bilan sur la segmentation

Grâce à ces tests, nous avons pu déterminer de manière robuste quelles méthodes utiliser pour la détection automatisée des plans dans une vidéo, aussi bien pour les films en couleur que pour ceux en noir et blanc. Nous avons pu valider ces résultats à travers un test sur un cas concret. Bien entendu, pour une validation complète, il aurait fallu appliquer ces méthodes sur une dizaine de films en couleur et une dizaine de films en noir et blanc. Cependant, cela n'était pas envisageable, car il aurait été trop lourd pour des chercheurs en lettres d'évaluer manuellement 20 segmentations.

D'après les tests réalisés et leur validation sur un cas réel, nous concluons que l'utilisation de la bibliothèque Python PySceneDetect est suffisante pour le calcul de la segmentation automatique de bas niveau.

Néanmoins, si l'on souhaite améliorer ces méthodes à partir du code source, plusieurs perspectives d'amélioration peuvent être envisagées :

- Adaptive Detector
 - Utiliser des métriques supplémentaires, comme l'optical flow, pour mieux distinguer un mouvement de caméra rapide d'un véritable cut.
 - Remplacer la moyenne glissante par la variance ou la médiane pour atténuer l'impact des variations dues aux mouvements de caméra
- Hash Detector
 - Ajuster dynamiquement le seuil de Hamming en fonction du contexte local, afin de réduire les faux positifs ou les faux négatifs.
 - Combiner les méthodes de Perceptual Hash et Difference Hash pour mieux gérer les variations de lumière ou les petits mouvements.

Enfin, prendre l'intersection des résultats de différents algorithmes pourrait être une piste intéressante. Cela permettrait probablement de réduire considérablement le nombre de faux positifs, mais au prix d'une légère diminution du nombre de vrais positifs et d'une augmentation significative du temps de calcul. Cette amélioration potentielle pourrait être envisagée en

fonction du compromis souhaité par l'utilisateur moyen du logiciel.

Dans la suite du projet, l'un des objectifs exprimés par les membres de l'équipe Numalyse est de mettre en place l'automatisation pour la segmentation de second niveau, consistant à détecter automatiquement les scènes, c'est-à-dire des ensembles de plans cohérents du point de vue de l'action, du décor ou des personnages.

Cette tâche est nettement plus complexe que la simple détection de cuts ou de fondus, car elle repose sur des critères moins formels et plus subjectifs. Plusieurs approches peuvent être envisagées, notamment des méthodes basées sur la continuité visuelle (couleurs dominantes, histogrammes), la similarité audio (musique, ambiance sonore constante). L'utilisation d'algorithmes de clustering tel k-means appliqués à des vecteurs de caractéristiques extraits de chaque plan représente une piste intéressante pour regrouper ces plans en scènes.

Cependant, un enjeu majeur réside dans la nécessité de disposer d'un jeu de données suffisamment riche et annoté, ce qui représente une contrainte importante. De plus, la définition même de ce qu'est une scène varie selon les disciplines (cinéma, littérature ...), ce qui complique l'établissement de critères objectifs pour l'automatisation.

Si une telle méthode venait à être développée dans le cadre du projet Numalyse, elle permettrait d'offrir aux chercheurs en lettres une vision structurée du film à plusieurs niveaux, facilitant ainsi une analyse plus fine et plus approfondie de la construction filmique.

6 Phase 2 : Extraction automatisée d'image clé

Dans le contexte de l'analyse de films et de leur segmentation, il existe un besoin réel d'identifier visuellement chaque plan de manière intuitive. Or, une simple numérotation chronologique des plans n'est pas toujours très efficace.

Dans le cas du logiciel SLV, pour la fonctionnalité d'exportation textuelle du travail d'analyse, une méthode simple et peu coûteuse en temps de calcul a été utilisée provisoirement. Il s'agissait tout simplement de sélectionner la dixième frame de chaque plan comme image clé. Pourquoi pas la première ? Car en cas de transition de type fondu, la première frame peut appartenir à la transition et ne contenir aucune information significative. Malgré cela, même la dixième frame peut parfois être peu représentative du plan. Il est donc nécessaire d'explorer des méthodes plus pertinentes en se basant sur la littérature scientifique existante.

Malheureusement, de nombreux articles de recherche sur ce sujet sont centrés sur des cas d'usage spécifiques, comme la vidéosurveillance, où l'objectif est très clair (par exemple, détecter la présence d'humain). Or, notre objectif est de trouver une méthode capable d'extraire une seule image clé représentative pour n'importe quel type de plan cinématographique.

La première étape de ce travail a donc été d'élaborer un état de l'art, regroupant une dizaine d'articles utilisant des approches variées, tant sur le plan des concepts utilisés que sur le type de résultats produits et les domaines d'application. Cette état de l'art va nous permettre d'avoir une vue d'ensemble des approches existantes pour envisager une méthode d'extraction d'image clé adaptée à notre problématique.

6.1 État de l'art

6.1.1 Approche qui s'appuie sur la segmentation de plan

Cette approche est la plus courante pour l'extraction de frames clés. Dans l'article [1], la première et la dernière frame de chaque plan détecté sont systématiquement sélectionnées comme frames clés. Pour les vidéos éditées (c'est-à-dire montées), l'article [2] propose une sélection par plan basée soit sur la frame de meilleure qualité, soit sur un nombre fixe d'images clés réparties à intervalles réguliers. Enfin, l'article [10] sélectionne les frames en s'appuyant sur un modèle d'expression faciale (pour les plans centrés sur des visages) et un réseau de neurones (CNN) évaluant l'esthétique de l'image (pour les autres plans).

6.1.2 Approche qui utilise la caractéristique du mouvement

La caractéristique du mouvement est largement exploitée pour détecter des moments clés dans les vidéos, en particulier dans les vidéos non-éditées (vidéo de type amateur que l'on peut caractériser de plan unique). Dans l'article [2], l'analyse repose sur des événements post-mouvement combinés à la détection d'objets, de visages et d'éléments audio (voix, rires). L'article [6] identifie les frames atypiques via le flux optique et un classifieur One-Class SVM. Enfin, l'article [7] fusionne plusieurs canaux de mouvement et applique une analyse fréquentielle pour détecter des pics de variation comme indicateurs de frames clés.

6.1.3 Approche par clustering de caractéristiques

Cette approche repose sur le regroupement de frames similaires à l'aide d'algorithmes de clustering appliqués à des descripteurs visuels, audio ou sémantiques. Dans l'article [2], un clustering K-means ou adaptatif est appliqué sur des candidates, puis la frame la plus représentative est sélectionnée selon l'importance et la qualité. L'article [3] utilise une projection aléatoire suivie

d'un codage parcimonieux pour générer une matrice de similarité, sur laquelle un clustering est appliqué pour sélectionner des frames médianes. L'article [5] combine scores visuels (SIFT, objets détectés) et un filtrage final par un classifieur de type AlexNet. Enfin, l'article [9] applique un clustering par densité (TSDPC) sur des segments temporels et retient les points de haute densité comme key-frames.

6.1.4 Approche qui utilise des représentations nouvelles ou particulières

Certains travaux introduisent des représentations alternatives pour réduire les coûts de traitement ou proposer des méthodes originales de sélection. Dans l'article [3], un codage parcimonieux sur des projections aléatoires permet une sélection efficace avec un coût réduit. L'article [4] propose une représentation de type "epitome" construite de façon probabiliste, sur laquelle une sélection min-max est effectuée via la divergence de Kullback–Leibler pour couvrir toute la séquence.

6.1.5 Approche qui produit des images de type fabriquées

Ces approches ne se limitent pas à extraire des frames existantes, mais génèrent une image composite ou enrichie. L'article [10] crée une image porteuse en regroupant des régions saillantes détectées après clustering, selon une heuristique d'optimisation de l'espace. L'article [11], quant à lui, sélectionne une image principale sur la base de la pertinence thématique, puis y intègre images secondaires et textes (phrases clés) en optimisant la lisibilité et la cohérence avec la requête utilisateur.

6.1.6 Tableau Récapitulatif

	1	2	3	4	5	6	7	8	9	10	11
Segmentation	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓
Clustering	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓
CNN (extract caractéristiques)	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗
Classifieur	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗
Audio	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
Sous-titres	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Mouvement	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗
Luminosité	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓
Détection d'objet	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗
Détection d'humain/visage	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗
Sortie	GI	GI	GI	GI	GI	GI	GI	Style	GI	IF	IF
Domaine d'utilisation/tests	G	G	G	G	VS	G	VS	C	G	G	N

Table 6: Tableau récapitulatif des méthodes utilisées selon les articles. (✓: utilisé, ✗: non utilisé, GI: groupe d'image, IF: image fabriqué, G: général, VS: vidéo surveillance, C: cinéma, N: news)

6.2 Méthodes sélectionnées

Grâce à ce tableau, on peut mieux visualiser l'ensemble des méthodes.

- La méthode 1 n'est pas pertinente, car elle correspond à ce qui est déjà réalisé actuellement.

- La méthode 2 aurait pu être intéressante, mais elle n'est pas entièrement automatisée puisqu'il faut lui fournir un ou plusieurs mots-clés. De plus, ses résultats dépendent entièrement de ceux obtenus via la recherche d'images.
- Les méthodes 3 et 4 utilisent des concepts innovants, mais ne prennent pas en compte les métriques essentielles à la sélection de frames clés.
- La méthode 8 est très prometteuse sur le papier, car elle repose sur des concepts variés et a été développée spécifiquement pour le domaine du cinéma. Malheureusement, sa sortie consiste en un style de film et non une image clé.
- Les méthodes 10 et 11 produisent en sortie une image générée. Or, il a été décidé de ne conserver que des images extraites directement des vidéos, sans modifications. Cela dit, dans le cadre de futurs travaux où il faudrait attribuer un tag ou une image représentative à une séquence ou une scène (par exemple lors de dialogues entre personnages), ces méthodes pourraient être plus appropriées.

En conclusion, nous avons décidé de sélectionner les méthodes 5, 6 et 7, car elles exploitent des concepts variés et complémentaires. Cela nous laisse la possibilité d'explorer différentes directions en fonction des résultats obtenus et des attentes des experts en analyse filmique.

6.3 Détail des trois méthodes sélectionnées

6.3.1 Méthode de l'article 5

- Première étape : Détection d'objets et attribution d'un score de qualité.
 - Un détecteur d'objets (R-CNN) est appliqué à chaque image pour identifier des éléments d'intérêt (par exemple, piétons, véhicules).
 - Chaque image se voit attribuer un score de qualité en fonction de la présence et de la pertinence des objets détectés.
 - L'utilisation de ce score permet de prioriser les images contenant des informations significatives et d'éliminer une grande partie des images redondantes.
- La deuxième étape consiste à combiner, pour chaque image, le score calculé précédemment avec le score SIFT, défini comme la somme des points d'intérêt détectés dans l'image.
- Lors de la dernière étape, les n images ayant les scores les plus élevés sont sélectionnées comme candidates. Enfin, à l'aide du classifieur AlexNet, chaque image est acceptée si le label cible figure parmi les résultats, dans le cas contraire elle est rejetée.

6.3.2 Méthode de l'article 6

Chaque frame est transformée en un vecteur de caractéristiques basé sur la couleur (HSV) et le mouvement (flux optique). À l'aide d'une fenêtre glissante, on extrait des statistiques (changement de texture et intensité du mouvement) pour former des vecteurs $X[i]$.

Un One-Class SVM avec noyau RBF est ensuite entraîné pour détecter les frames atypiques, avec une optimisation des hyperparamètres (ν , γ) via Differential Evolution (DE). Enfin, les frames ayant les scores les plus négatifs sont sélectionnées comme *tag image*.

6.3.3 Méthode de l'article 7

- Pour chaque image, on extrait quatre canaux de caractéristiques : FM (motion) : différence absolue entre l'image courante et la précédente (si elle existe), luminance, FRG (red/-green) et FBY (blue/yellow).
- Fusion par transformée de Fourier quaternion : on regroupe ces quatre caractéristiques en deux plans complexes, Plan 1 : $FM + i \cdot FB$ et Plan 2 : $FRG + i \cdot FBY$. On effectue une FFT 2D sur chacun des deux plans puis on calcule une carte de phase fusionnée. On normalise cette phase en image 8 bits (0–255).
- Filtrage spatial : On applique un flou gaussien sur la carte de phase normalisée, afin d'atténuer le bruit (les hautes fréquences).
- Reconstruction (inverse FFT): On traite la carte floutée comme un signal réel et on fait une IFFT 2D. Le module du résultat constitue la carte fusionnée finale pour cette frame, qui met en valeur à la fois le mouvement et les variations de couleur/brillance.
- Détection de la frame la plus saillante: On calcule la MSE (Mean Squared Error) entre chaque carte fusionnée et celle de la frame précédente, pour obtenir une courbe MSE tout au long de la vidéo. L'indice où cette MSE est maximale correspond à la transformation la plus marquée (changement global + local).

6.4 Implémentation

Malheureusement, pour aucun des trois articles scientifiques sélectionnés, nous n'avons pu obtenir une base de code ni une implémentation existante. Il a donc fallu que j'implémente moi-même les trois méthodes, en y apportant parfois quelques ajustements pour qu'elles s'adaptent à notre cas d'usage. Notamment pour la méthode 5, qui reposait initialement sur une deuxième passe de validation à l'aide d'un label cible : j'ai choisi de supprimer cette seconde étape et de renforcer la pondération du label "humain".

L'étape suivante a consisté à tester ces méthodes sur un petit corpus de six plans extraits de films, fournis par Monsieur Le Bihan : 2 plans issus de *Shining*, 2 de *La Vie d'Adèle* et 2 de *Even Cowgirls Get the Blues*.

Pour chaque plan, nous avons pu comparer les résultats obtenus avec quatre valeurs de vérité déterminées par des experts, VV1 : Amandine D'AZEVEDO, VV2 : Jean-Philippe Trias, VV3 : Loig LE BIHAN et VV4 : Frédéric ASTRUC.

Voici la comparaison des résultats de chaque méthode avec ces valeurs de vérité :

6.4.1 Résultats









VV1 adele1 - 549 	VV2 adele1 - 734 
VV3 adele1 - 461 	VV4 adele1 - 874 
10ème frame 	5 - 870 
6 - 321 	7 sigma 0.5 - 713 

Table 7: La Vie d'Adèle Plan 1 (880 frames)


VV1 adele2 - 592 	VV2 adele2 - 591 
VV3 adele2 - 1112 	VV4 adele2 - 788 
10ème frame 	5 - 8 
6 - 1026 	7 sigma 0.5 - 788 

Table 8: Adele 2 (1349 frames)









VV1 4eme1 - 114 	VV2 4eme1 - 106 
VV3 4eme1 - 179 	VV4 4eme1 - 606 
10ème frame 	5 - 158 
6 - 117 	7 sigma 0.5 - 607 

Table 9: ECGTB 1 (757 frames)









VV1 4eme2 - 188 	VV2 4eme2 - 29 
VV3 4eme2 - 31 	VV4 4eme2 - 15 
10ème frame 	5 - 18 
6 - 63 	7 sigma 0.5 - 15 

Table 10: ECGTB 2 (325 frames)


VV1 Shining1 - 565 	VV2 Shining1 - 534 
VV3 Shining1 - 470 	VV4 Shining1 - 101 
10ème frame 	5 - 625 
6 - 290 	7 sigma 0.5 - 591 

Table 11: Shining 1 (714 frames)

VV1 Shining2 - 465 	VV2 Shining2 - 481 
VV3 Shining2 - 340 	VV4 Shining2 - 87 
10ème frame 	5 - 365 
6 - 212 	7 sigma 0.5 - 271 

Table 12: Shining 2 (504 frames)

6.4.2 Analyse

Il est important de souligner qu'il est impossible de comparer de manière totalement automatisée les résultats obtenus avec les valeurs de vérité. Même une évaluation visuelle par un humain reste délicate. C'est pourquoi l'évaluation présentée ici peut être approximative.

Tout d'abord, la méthode de la 10ème frame s'avère globalement inefficace. Bien qu'elle puisse occasionnellement produire un bon résultat, comme pour le plan ECGTB 2, ces cas restent exceptionnels. Cela confirme la nécessité de développer une méthode plus robuste pour notre logiciel.

Concernant la méthode 5, on observe que, pour 5 des 6 plans, le résultat obtenu correspond à au moins 3 des 4 valeurs de vérité, ce qui témoigne d'une bonne cohérence avec les attentes des experts.

À l'inverse, la méthode 6 donne des résultats globalement décevants, avec seulement 2 plans sur lesquels la sortie est jugée correcte.

Enfin, la méthode 7, avec un paramètre $\sigma = 0,5$, obtient un taux de correspondance de 7 sur 7 avec au moins une des valeurs de vérité, ce qui constitue un excellent résultat sur cet échantillon.

6.4.3 Avantages, inconvénients et perspectives d'amélioration

En complément de l'analyse des résultats, un bilan a été réalisé pour identifier les avantages et inconvénients de chaque méthode, ainsi que des pistes d'amélioration envisageables.

6.4.4 Avantages / Inconvénients méthode 5

- **Avantages :**
 - Évite de sélectionner une frame vide ou non informative. Même si la frame choisie n'est pas optimale, elle contient toujours un minimum d'informations, ce qui rend cette méthode plus stable que d'autres.
 - Méthode très modulable et paramétrable par l'utilisateur.
- **Inconvénients :**
 - A tendance à privilégier les frames contenant de nombreux objets.
 - Favorise fortement les frames dans lesquelles des humains sont détectés.

6.4.5 Perspectives d'amélioration méthode 5

1. Appliquer une fonction de pondération temporelle qui favorise les frames situées entre le milieu et la fin du plan.
2. Ajouter un score basé sur la position des objets dans l'image : un objet situé au centre aurait un poids plus important qu'un objet en bord d'image.
3. Mettre en place un critère de qualité d'image pour éviter la sélection de frames floues.
4. Favoriser les moments où la caméra est immobile plutôt que ceux où elle est en mouvement.
5. Étant donné qu'on peut gérer le nombre de *tag image*, il peut être intéressant d'en extraire plusieurs, puis d'en sélectionner une en fonction de critères secondaires.

6.4.6 Avantages / Inconvénients méthode 6

- **Avantages :**

- Sélectionne une frame dans laquelle il y a du mouvement.

- **Inconvénients :**

- Dans certains cas, la frame présentant le plus de mouvement ne correspond pas au moment contenant le plus d'informations utiles.
- Il arrive que la frame sélectionnée soit floue.

6.4.7 Perspectives d'amélioration méthode 6

1. Mettre en place un critère de qualité d'image pour éviter la sélection de frames floues.
2. Plutôt que de chercher la frame la plus atypique, on peut choisir de faire l'inverse en sélectionnant la frame la plus courante.

6.4.8 Avantages / Inconvénients méthode 7

- **Avantages :**

- Beaucoup plus rapide que les deux autres méthodes.

- **Inconvénients :**

- Très dépendante de l'intensité du flou appliqué, qui peut varier d'une vidéo à l'autre.

6.4.9 Perspectives d'amélioration méthode 7

Aucune amélioration réellement envisageable. La méthode repose sur un principe spécifique et complexe (dont l'utilisation n'est pas forcément justifié), et ses résultats sont très variables et globalement peu convaincants.

6.5 Amélioration d'une méthode

D'après les résultats observés sur les six plans, les perspectives d'amélioration envisagées, ainsi que les attentes formulées par les experts, la méthode 5 apparaît comme la plus adaptée pour être améliorée. Même si la méthode 7, avec un paramètre $\sigma = 0,5$, donne de meilleurs résultats sur cet échantillon, elle se révèle très instable : il est souvent nécessaire d'ajuster ce paramètre en fonction du plan pour obtenir de bonnes performances. Ainsi, bien que les résultats soient excellents sur les six plans testés, des essais sur d'autres extraits montrent une grande variabilité.

Voici un exemple concret qui illustre parfaitement les propos précédents. Dans le film La Vie d'Adèle, deux plans très similaires en termes de contenu apparaissent à des moments différents. La méthode 5 parvient à produire des résultats très proches pour ces deux plans, tandis que la méthode 7, avec un paramètre $\sigma = 0,5$, sélectionne deux images nettement différentes.



Table 13: Comparaison Méthode 5 et 7

Dans un objectif de stabilité et de robustesse, la méthode 5 est donc préférée, car elle assure des résultats satisfaisants de manière plus constante. Dans les sections suivantes, différentes améliorations seront détaillées : elles ont été implémentées, testées, puis conservées ou écartées en fonction des résultats obtenus.

6.5.1 Pondération basée sur la taille de l'objet détecté

Cette amélioration vise à favoriser les frames où les objets détectés occupent une place importante dans l'image. À chaque détection, il est possible de récupérer les dimensions de la boîte englobante de l'objet. En les comparant à celles de l'image, on peut calculer un ratio de taille. Ce ratio est ensuite utilisé pour pondérer le score de l'objet (rappel : le score est calculé comme poids \times score de confiance). Cela permet donc de renforcer l'importance des objets de grande taille.

Cependant, cette amélioration s'est révélée contre-productive dans notre contexte. Bien qu'elle produise l'effet attendu sur le plan technique, elle n'est pas pertinente selon les experts. En effet, dans le cadre de vidéos issues du cinéma, la taille d'un objet n'est pas toujours un critère déterminant pour représenter la signification d'un plan.

Cela dit, cette approche à l'air efficace dans le contexte de vidéos amateurs, où les objets principaux sont souvent centrés et en gros plan.

J'ai tout de même tenté de limiter l'impact négatif de cette pondération en ignorant les objets trop grands (par exemple, ceux occupant plus de 60% de l'image). Mais même avec cette contrainte, les résultats ne se sont pas améliorés de manière significative. Il aurait été possible de poursuivre les expérimentations en ajustant les tailles minimales et maximales acceptées ou en modulant l'importance de cette pondération, mais étant donné que ce critère n'est pas jugé pertinent dans notre cas d'usage, j'ai préféré consacrer ce temps à l'implémentation d'améliorations plus prometteuses.

6.5.2 Pondération spatiale basée sur la position de l'objet détecté

L'objectif de cette amélioration est de favoriser les objets situés près du centre de l'image, en diminuant l'importance de ceux qui apparaissent dans les coins. Pour cela, on calcule la distance entre le centre de la boîte englobante de l'objet et le centre de l'image. Une pondération est

ensuite appliquée en fonction de cette distance, selon la formule suivante :

$$w_{\text{pos}}(x, y) = \exp \left(-\frac{1}{2} \left(\frac{\sqrt{\left(\frac{x-c_x}{c_x} \right)^2 + \left(\frac{y-c_y}{c_y} \right)^2}}{\sigma} \right)^2 \right)$$

avec $c_x = \frac{W}{2}$, $c_y = \frac{H}{2}$, et $\sigma = 0.5$

Pour mieux comprendre le fonctionnement de cette pondération, une illustration est présentée ci-dessous. Elle permet de visualiser plus clairement le principe appliqué.

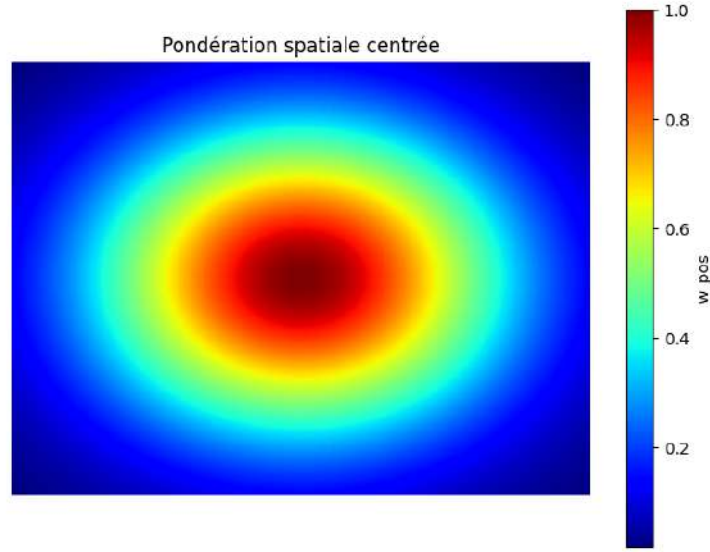


Figure 12: Illustration de la formule utilisé pour la pondération spatial

6.5.3 Pondération temporelle

Dès le début du projet, mes encadrants et moi-même avions l'intuition que, dans un plan cinématographique, les moments importants ne sont généralement pas situés aux extrémités (tout début ou toute fin), mais plutôt vers le centre du plan. Cette intuition s'appuie sur le fait que les plans de films sont conçus et structurés de manière réfléchie. Cette hypothèse a été confirmée lors d'une réunion avec des spécialistes en études cinématographiques, qui nous ont indiqué que l'élément clé d'un plan est souvent positionné autour du milieu, voire juste avant la fin.

Pour exploiter cette information, nous avons donc mis en place une pondération temporelle, destinée à favoriser les frames situées au centre du plan, tout en accordant un second pic de pondération un peu avant la fin. Nous avons choisi d'utiliser une double gaussienne : la première centrée sur le milieu du plan, la seconde légèrement décalée vers la fin. La formule exacte utilisée est donnée ci-dessous.

$$PT(x) = \exp \left(-\frac{(x - \frac{t}{2})^2}{2\sigma^2} \right) + \exp \left(-\frac{(x - (t - \epsilon))^2}{2\sigma^2} \right)$$

avec t le nombre total de frames, $\epsilon = 0.1t$ et $\sigma = 0.15t$

Un graphique permet de visualiser la courbe de cette fonction de pondération.

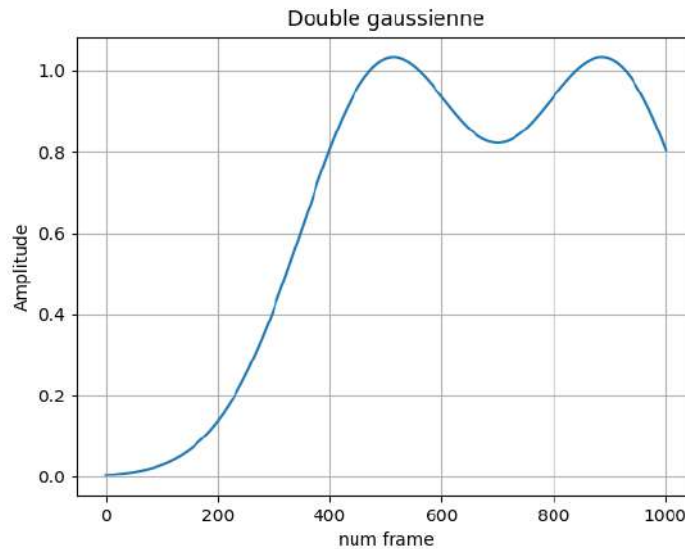


Figure 13: Affichage de la double Gaussienne utilisé pour la pondération temporelle

Deux approches ont été envisagées pour intégrer cette pondération dans le calcul final :

1. Intégration directe dans le score, en ajoutant la pondération temporelle au score global avec un certain poids.
2. Application en deux passes :
 - (a) Une première passe sélectionne les n meilleures frames selon le score brut (par exemple $n = 3\%$ du total de frames du plan).
 - (b) Une deuxième passe applique la pondération temporelle uniquement à ces frames sélectionnées, et la frame avec le score final le plus élevé est retenue.

Après analyse des deux approches, j'ai décidé de conserver la première, car elle intègre plus naturellement la pondération dans le score global, tout en respectant l'ensemble des critères recherchés pour l'évaluation d'une tag image. À l'inverse, la seconde approche tend à surévaluer cette pondération, qui finit par dominer les autres métriques de sélection, ce qui dégrade la qualité des résultats obtenus par rapport à la méthode sans cette pondération.

6.5.4 Pondération basée sur la qualité de l'image

La dernière amélioration ajoutée concerne la qualité visuelle des images sélectionnées. En effet, pour tout utilisateur expert ou non, une image nette sera toujours préférable à une image floue. Pour cela, une métrique simple mais efficace a été mise en place pour mesurer la netteté d'une image : la variance du Laplacien.

Pour chaque frame, une estimation de la netteté est obtenue en suivant les étapes suivantes :

1. Conversion en niveaux de gris de l'image (pour simplifier l'analyse des détails)
2. Application du filtre de Laplacien, qui met en évidence les variations rapides d'intensité (bords, textures)
3. Calcul de la variance de l'image résultante, qui reflète la quantité de détails présents

Cette mesure est ensuite normalisée sur l'ensemble des frames pour obtenir un facteur de pondération.

6.6 Formule Final

Voici donc la formule final qui permet de calculer le score de chaque frame après les différentes amélioration qui ont été ajouté :

$$5F = \alpha \cdot S + \beta \cdot D + \gamma \cdot PT + (1 - \alpha - \beta - \gamma) \cdot Q_i$$

- $S = \sum \text{points_intérêt_SIFT}$
- $D = \sum \text{Confiance} \cdot \text{Poids} \cdot \text{Pondération Spatiale}$, avec Poids = 10 si humain, sinon 1
- avec $\alpha = 0.2$, $\beta = 0.6$ et $\gamma = 0.1$

6.7 Résultats




<p>VV1 adele1 - 549</p> 	<p>VV2 adele1 - 734</p> 
<p>VV3 adele1 - 461</p> 	<p>VV4 adele1 - 874</p> 
<p>5 - 870</p> 	<p>5+pos - 606</p> 
<p>5+pos+gauss - 606</p> 	<p>5+pos+gauss+net - 606</p> 

Table 14: La Vie d'Adèle Plan 1 (880 frames)

VV1 adele2 - 592 	VV2 adele2 - 591 
VV3 adele2 - 1112 	VV4 adele2 - 788 
5 - 12 	5+pos - 1043 
5+pos+gauss - 1043 	5+pos+gauss+net - 1043 

Table 15: Adele 2 (1349 frames)








VV1 4eme1 - 114 	VV2 4eme1 - 106 
VV3 4eme1 - 179 	VV4 4eme1 - 606 
5 - 158 	5+pos - 108 
5+pos+gauss - 108 	5+pos+gauss+net - 108 

Table 16: ECGTB 1 (757 frames)









VV1 4eme2 - 188 	VV2 4eme2 - 29 
VV3 4eme2 - 31 	VV4 4eme2 - 15 
5 - 18 	5+pos - 0 
5+pos+gauss - 5 	5+pos+gauss+net - 18 

Table 17: ECGTB 2 (325 frames)

VV1 Shining1 - 565 	VV2 Shining1 - 534 
VV3 Shining1 - 470 	VV4 Shining1 - 101 
5 - 625 	5+pos - 452 
5+pos+gauss - 353 	5+pos+gauss+net - 353 

Table 18: Shining 1 (714 frames)

VV1 Shining2 - 465 	VV2 Shining2 - 481 
VV3 Shining2 - 340 	VV4 Shining2 - 87 
5 - 365 	5+pos - 372 
5+pos+gauss - 283 	5+pos+gauss+net - 423 

Table 19: Shining 2 (504 frames)

Malgré toutes les améliorations apportées, les résultats obtenus sur les six plans de référence ne montrent aucun gain significatif. Les frames sélectionnées restent généralement proches des valeurs de vérité (à l'exception notable du plan Adèle 2). Il est donc possible que ces ajustements apportent une forme de stabilité ou, au contraire, qu'ils aient introduit une forme de "sur-apprentissage" : autrement dit, la méthode fonctionnerait particulièrement bien sur ces six exemples, mais s'avérerait inefficace, voire contre-productive, lorsqu'on l'applique à d'autres plans.

La meilleure façon d'évaluer cette hypothèse serait de mettre en place un questionnaire, destiné à comparer différentes méthodes à travers une évaluation qualitative par des experts, ce qui permettrait d'obtenir un score de précision pour chacune.

Avant d'en arriver là, mes encadrants ont proposé une nouvelle variante inspirée de la méthode 5, intégrant potentiellement les mêmes types d'améliorations, mais conçue de manière plus cohérente avec l'objectif final : sélectionner l'image la plus représentative du plan, celle qui permettrait le mieux de l'identifier visuellement.

6.8 Création d'une variante

6.8.1 Fonctionnement

Pour chaque frame i , on calcule un histogramme des objets détectés pondéré spatialement :

$$H_i(c) = \sum_{d \in \mathcal{D}_i^c} s_d \cdot \exp \left(-\frac{1}{2} \left(\frac{\sqrt{\left(\frac{x_d - c_x}{c_x} \right)^2 + \left(\frac{y_d - c_y}{c_y} \right)^2}}{\sigma_{\text{pos}}} \right)^2 \right)$$

On multiplie ensuite par la netteté normalisée Q_i :

$$\tilde{H}_i(c) = Q_i \cdot H_i(c)$$

La pondération temporelle est définie par :

$$PT(i) = \exp\left(-\frac{(i - \frac{t}{2})^2}{2\sigma^2}\right) + \exp\left(-\frac{(i - (t - \epsilon))^2}{2\sigma^2}\right)$$

avec $\epsilon = 0.1t$, $\sigma = 0.15t$.

On calcule l'histogramme moyen pondéré temporellement :

$$\bar{H}(c) = \frac{1}{\sum_i PT(i)} \sum_i PT(i) \cdot \tilde{H}_i(c)$$

Et le score final est la distance entre \tilde{H}_i et \bar{H} :

$$\text{Score}(i) = d(\tilde{H}_i, \bar{H})$$

où d est une distance euclidienne.

6.8.2 Résultats







VV1 adele1 - 549 	VV2 adele1 - 734 
VV3 adele1 - 461 	VV4 adele1 - 874 
hist - 399 	hist+gauss - 474 
hist+gauss+pos - 774 	hist+gauss+pos+net - 566 

Table 20: La Vie d'Adèle Plan 1 (880 frames)









VV1 adele2 - 592	VV2 adele2 - 591
	
VV3 adele2 - 1112	VV4 adele2 - 788
	
hist - 474	hist+gauss - 1116
	
hist+gauss+pos - 471	hist+gauss+pos+net - 1090
	

Table 21: Adele 2 (1349 frames)

VV1 4eme1 - 114	VV2 4eme1 - 106
	
VV3 4eme1 - 179	VV4 4eme1 - 606
	
hist - 333	hist+gauss - 443
	
hist+gauss+pos - 438	hist+gauss+pos+net - 209
	

Table 22: ECGTB 1 (757 frames)









VV1 4eme2 - 188 	VV2 4eme2 - 29 
VV3 4eme2 - 31 	VV4 4eme2 - 15 
hist - 262 	hist+gauss - 137 
hist+gauss+pos - 137 	hist+gauss+pos+net - 110 

Table 23: ECGTB 2 (325 frames)

VV1 Shining1 - 565 	VV2 Shining1 - 534 
VV3 Shining1 - 470 	VV4 Shining1 - 101 
hist - 683 	hist+gauss - 529 
hist+gauss+pos - 563 	hist+gauss+pos+net - 415 

Table 24: Shining 1 (714 frames)

VV1 Shining2 - 465 	VV2 Shining2 - 481 
VV3 Shining2 - 340 	VV4 Shining2 - 87 
hist - 271 	hist+gauss - 474 
hist+gauss+pos - 479 	hist+gauss+pos+net - 472 

Table 25: Shining 2 (504 frames)

Contrairement à la méthode 5, on constate que la version basique de la méthode par histogramme donne des résultats mitigés, tandis que l'ajout des pondérations améliore significativement les performances.

Cependant, le même problème d'évaluation se pose : les résultats obtenus sur ces six plans ne suffisent pas à garantir la fiabilité ni la précision de la méthode de manière générale.

6.9 Élaboration d'un questionnaire

Comme mentionné précédemment, disposer de valeurs de vérité pour un plan ne permet pas d'évaluer correctement les différentes méthodes. La solution la plus pertinente consiste donc à concevoir un questionnaire, destiné principalement à un public de professionnels du monde du cinéma. Le format retenu est le suivant :

- Le questionnaire comporte 10 questions, chacune correspondant à un plan extrait d'un film différent.
- Pour chaque question, 6 images sont proposées, chacune générée par l'une des méthodes à évaluer, à savoir : la méthode classique (extraction de la frame centrale du plan), les méthodes décrites dans les articles [5], [6] et [7], la méthode de l'article [5] avec les améliorations apportées, ainsi que la méthode basée sur l'histogramme qui contient aussi les améliorations.
- L'utilisateur doit sélectionner l'image (ou les images) qu'il juge pertinente pour identifier le plan.
- L'ordre des questions et celui des réponses sont aléatoires pour chaque participant.

L'avantage de cette présentation aléatoire est qu'elle limite les biais d'apprentissage. En effet, il est souvent observé qu'un répondant devient plus performant à mesure qu'il progresse dans un questionnaire.

6.10 Bilan

6.10.1 Conclusion

Même si nous ne disposons pas encore de résultats concrets permettant d'évaluer précisément chaque méthode, certaines conclusions peuvent déjà être tirées. Les deux méthodes finales se sont révélées globalement efficaces, avec une légère supériorité en termes de précision pour la méthode basée sur les histogrammes. Leur principal avantage réside dans leur adaptabilité aux vidéos issues du monde du cinéma. En effet, là où de nombreuses méthodes existantes échouent sur des plans cinématographiques (malgré leur efficacité sur d'autres types de vidéos), nos deux approches s'avèrent pertinentes et performantes.

6.10.2 Perspectives d'amélioration

Dans le cadre de la poursuite du projet Numalyse, plusieurs pistes d'amélioration peuvent être envisagées pour affiner les résultats, que ce soit pour la méthode 5 améliorée ou pour celle basée sur les histogrammes.

Tout d'abord, il serait pertinent d'ajuster la double gaussienne utilisée pour la pondération temporelle, en relevant légèrement le début de la fonction afin d'obtenir une croissance plus progressive et moins abrupte.

Ensuite, l'adoption d'un détecteur d'objets plus performant (avec davantage de labels), une meilleure précision et un meilleur temps de calcul permettrait d'améliorer significativement la qualité des résultats.

Par ailleurs, l'intégration d'une métrique plus sophistiquée pour évaluer la netteté, telle que BRISQUE, offrirait une meilleure capacité à privilégier les images de haute qualité, au-delà d'une simple mesure de netteté.

Plus spécifiquement :

- Pour la méthode 5, l'optimisation des paramètres α , β et γ semble indispensable pour en améliorer l'efficacité.
- Pour la méthode basée sur les histogrammes, viser une convergence vers l'histogramme médian plutôt que vers l'histogramme moyen pourrait potentiellement produire de meilleurs résultats.

Enfin, dans une perspective d'intégration pratique dans le logiciel, l'enjeu du temps de calcul devient central pour garantir une bonne expérience utilisateur. Une optimisation du code, couplée à une exploitation des accélérations matérielles, permettrait de réduire au maximum les délais de traitement.

6.10.3 Utilisation du Deep Learning

L'utilisation du deep learning représente une alternative tout à fait viable pour notre objectif. Cette approche est d'ailleurs déjà mise en œuvre par des plateformes comme YouTube, qui proposent automatiquement plusieurs miniatures lorsqu'un utilisateur souhaite publier une vidéo.

L'un des principaux avantages du deep learning réside dans sa capacité à extraire des informations visuelles pertinentes à partir d'un grand volume de données, ce qui permet d'identifier les images les plus représentatives.

Cependant, cette méthode implique une contrainte majeure : la nécessité de disposer d'une base de données riche, diversifiée et bien annotée. En effet, la qualité et la variété des données d'entraînement influencent directement les performances du modèle.

7 Conclusion et perspectives

Au cours de ce stage, j'ai pu développer une version bêta d'un logiciel de lecture et de traitement de films, intégrant des fonctionnalités peu courantes, voire inexistantes dans les logiciels les plus avancés actuellement disponibles sur le marché.

Ce projet m'a également amené à me documenter de manière approfondie sur les méthodes existantes d'extraction automatisée d'images clés dans les vidéos. L'une de ces méthodes a été améliorée par mes soins selon plusieurs axes identifiés au fil des réunions du projet Numalyse, dans lequel ce stage s'inscrivait.

Comme évoqué dans ce rapport, le travail réalisé présente de nombreuses pistes d'amélioration, susceptibles d'aboutir à des résultats nettement plus performants. D'autres orientations de développement se dessinent également pour la suite du projet, comme l'élaboration d'une méthode basée sur un réseau de neurones convolutifs (CNN) pour la détection de types de scènes, ou encore l'identification automatique des différents individus apparaissant dans un film, avec la possibilité d'extraire les scènes ou plans où ces personnages sont présents.

D'un point de vue personnel, ce stage a été extrêmement formateur. Il m'a permis d'explorer à la fois le travail de l'ingénieur en développant et consolidant mes compétences en développement logiciel et celui du chercheur, où l'autonomie, l'esprit d'analyse et l'initiative sont essentiels.

Au fil des réunions, j'ai également pu renforcer ma capacité d'écoute, une compétence cruciale pour comprendre au mieux les besoins des membres du projet Numalyse tout en sachant expliquer clairement les contraintes techniques ou matérielles. Cet exercice est d'autant plus délicat lorsque les interlocuteurs viennent de domaines théoriquement très éloignés.

Enfin, le fait de contribuer concrètement à un projet de cette envergure et de pouvoir y apporter ma propre touche a été une expérience à la fois valorisante et motivante, qui me donne envie de poursuivre dans cette voie.

8 Annexes

8.1 Diagramme de Gantt

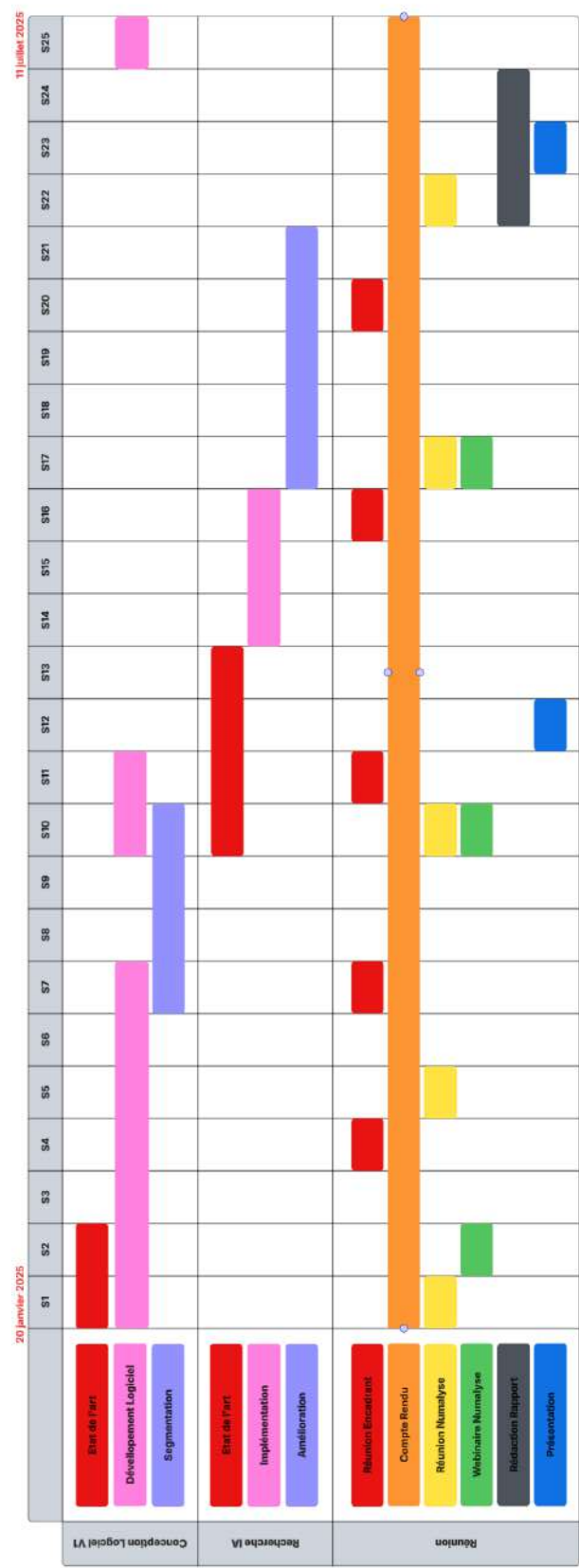


Figure 14: Diagramme de Gantt de mon stage

9 Bibliographie

- 1 - Wisnu Widiarto & Eko Mulyanto Yuniarno & Mochamad Hariadi, "Video Summarization Using a Key Frame Selection Based on Shot Segmentation", 2015 International Conference on Science in Information Technology (ICSITech).
- 2 - Yuli Gao & Tong Zhang & Jun Xiao, "THEMATIC VIDEO THUMBNAIL SELECTION", 2009 16th IEEE International Conference on Image Processing (ICIP).
- 3 - Mrityunjay Kumar & Alexander C. Loui, "KEY FRAME EXTRACTION FROM CONSUMER VIDEOS USING SPARSE REPRESENTATION", 2011 18th IEEE International Conference on Image Processing.
- 4 - C.T.Dang & M.Kumar & H.Radha, "KEY FRAME EXTRACTION FROM CONSUMER VIDEOS USING EPITOME", 2012 19th IEEE International Conference on Image Processing.
- 5 - Mingju Chen & Xiaofeng Han & Hua Zhang & Guojun Lin & M.M. Kamruzzaman, "Quality-guided key frames selection from video stream based on object detection", 2019 J. Vis. Commun. Image R.
- 6 - Xiao-Gen PEI, "The key frame extraction algorithm based on the indigenous disturbance variation difference video", 10th International Conference of Inforamtion and Communication Technology (ICICT-2020).
- 7 - Yunzuo Zhang & Jiayu Zhang & Ruixue Liu & Pengfei Zhu & Yameng Liu, "Key frame extraction based on quaternion Fourier transform with multiple features fusion", Expert Systems With Applications Volume 216 (2023).
- 8 - Jiangnan Sun & Chunfang Li & Ruihan Tang, "Film Analysis from the Perspective of Cinemeitrics Based on OpenCV and Deep Learning", 2022 IEEE/ACIS 22nd International Conference on Computer and Information Science (ICIS).
- 9 - HAO TANG & LEI DING & SONGSONG WU & BIN REN & NICU SEBE & PAOLO ROTA, "Deep Unsupervised Key Frame Extraction for Efficient Video Classification", 12 Nov 2022.
- 10 - Baoquan Zhao & Hanhui Li & Ruomei Wang & Xiaonan Luo, "Automatic Generation of Informative Video Thumbnail", 2020 8th International Conference on Digital Home (ICDH).
- 11 - Jinyu Li1 & Shujin Lin & Fan Zhou & Ruomei Wang, "NewsThumbnail: Automatic Generation of News Video Thumbnail", 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC).