



الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أحمد

Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي

Institut de Maintenance et de Sécurité Industrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour obtention du diplôme de Master

Filière : Génie Industriel

Spécialité : Maintenance de l'Automatisme et de l'Instrumentation

Thème

Traitement d'image et de la vidéo par calcul sur processeur parallèle RTX en utilisant Matlab/Python

Présenté par :

Khitri Mohammed El Habib Djaouad

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
A.Hassini	Prof	Université d'Oran 2 MAI	Président
Adda Neggaz Samir	MAA	Université d'Oran 2 MAI	Encadreur
N.Nekrouf	MCB	Université d'Oran 2 MAI	Examineur

Année universitaire : 2025 / 2026

Dédicace

Avant tout, je loue Allah, le Tout-Puissant, pour m'avoir guidée dans l'accomplissement de ce travail.

Je dédie ce modeste travail à mes très chers parents, pour leur amour inconditionnel, leurs sacrifices et leur soutien constant tout au long de mon parcours.

À mon frère et à mes sœurs, pour leur encouragement et leur présence à mes côtés.

À mes amis, pour les moments de partage, de soutien et de motivation.

Mohammed El Habib Djaouad

Remerciements

بسم الله الرحمن الرحيم

Au terme de ce travail de recherche, il m'est particulièrement agréable d'exprimer ma profonde reconnaissance à toutes les personnes qui ont contribué, de près ou de loin, à son élaboration.

Je tiens, en premier lieu, à adresser mes remerciements les plus respectueux à mon directeur de mémoire, monsieur Adda Neggaz Samir, pour la qualité de son encadrement, la pertinence de ses orientations, ainsi que pour sa disponibilité et la rigueur scientifique dont il a fait preuve tout au long de la réalisation de ce travail.

J'exprime également ma gratitude aux membres du jury qui ont bien voulu accepter d'évaluer ce mémoire et d'en apprécier la valeur scientifique.

Je ne saurais oublier d'adresser mes vifs remerciements à ma famille pour son soutien indéfectible, sa patience et ses encouragements constants.

Enfin, je tiens à remercier toute personne ayant apporté, de quelque manière que ce soit, une contribution à l'aboutissement de ce travail.

Table de matières

Dédicace	I
Remerciements	II
Table de matières.....	III
Liste des figures.....	VII
Abréviations et Notations	VIII
Introduction	a
Chapitre I:Généralités sur l'image.....	1
1. INTRODUCTION	2
2. L'IMAGE NUMERIQUE	2
2.1. Définition	2
3. LES DIFFERENTS TYPES D'IMAGE	3
3.1. L'imagerie vidéo	3
3.1.1. Définition d'une vidéo	3
3.1.2. Résolution vidéo	4
3.1.3. Résolution temporelle d'une vidéo	4
3.1.4. Définition d'une objet vidéo	5
3.1.5. Séquence d'image	5
3.1.6. Nature de la scène	6
4. LES FORMATS D'IMAGE	6
4.1. Image numérique matricielle (Bitmap)	6
4.2. Image numérique vectorielle	7
5. LES CARACTERISTIQUES DE L'IMAGE	8
5.1. Dimension	8
5.2. Pixel	8
5.3. Résolution et définition	9
5.4. Contraste	10
5.5. Contour	11
5.6. Bruit	11
5.7. Histogramme	11
5.8. Codage	12
5.8.1. Images binaires	12
5.8.2. Images en niveaux de gris	13
5.8.3. Images en couleurs	13
5.8.4. Images en couleur indexées	14

5.8.5. Le système HSV (Hue. Saturation. Value) ou TSL (Teinte. Saturation. Luminosité)	14
6. CONCLUSION	17
Chapitre II:Traitement d'image.....	18
1. INTRODUCTION :	19
2. TRAITEMENT D'IMAGE	19
3. PRETRAITEMENT D'IMAGE	20
3.1. Filtrage	21
3.1.1. Filtres linéaires	21
3.1.2. Filtre moyenné (lissage)	22
3.1.3. Filtre gaussien	23
3.1.4. Filtre rehausseur de contours	23
3.1.5. Filtres non linéaires	23
3.2. Binarisation	23
3.3. Les opérateurs de la morphologie mathématique	24
3.3.1. Erosion	25
3.3.2. Dilatation	25
3.3.3. Utilisation de l'érosion et de la dilatation	26
3.4. Segmentation	26
3.4.1. Définition	26
3.4.2. Type de segmentation	27
3.4.3. Technique de segmentation	28
4. CONCLUSION	29
Chapitre III:Détection et suivi d'objet	30
1. INTRODUCTION	31
2. DETECTION D'OBJET	31
2.1. Définition	31
2.2. Approches-en une seule étape	32
2.2.1. You Only Look Once (YOLO)	32
2.3. Approche en deux étapes (basée sur la région)	33
2.3.1. Region Convolutional Neural Network (R-CNN)	33
2.3.2. Rapid R-CNN (Fast R-CNN)	34
2.3.3. Faster R-CNN	34
2.3.4. Mask R-CNN	35
3. SUIVI D'OBJET	36
3.1. Définition	36

3.2. Domain d'application	37
3.3. Les défis du suivi	38
3.3.1. Occultations	38
3.3.2. Changement d'illumination	38
3.3.3. Changement d'échelle	39
3.3.4. Changement du modèle d'apparence	39
3.3.5. Mouvements de caméra	40
3.3.6. Objet en mouvement rapide	40
3.4. Méthodes de suivi	40
3.4.1. Suivi par point	40
3.4.2. Suivi de silhouettes	41
3.4.3. Suivi de noyaux	41
3.5. Caractéristiques visuelles pour le suivi d'objets	42
3.5.1. Couleur	42
3.5.2. Texture	42
3.5.3. Gradient	43
4. CONCLUSION	43
Chapitre IV: Implémentation et discussion des résultats	44
1. INTRODUCTION	45
2. PRESENTATION DE LA BOITE À OUTILS	45
2.1. Langage de programmation Python	45
2.2. Bibliothèques utilisées	45
2.3. Sources d'acquisition des données	46
2.4. Environnement matériel	46
2.5. Environnement de développement	47
3. L'APPLICATION IP WEBCAM	47
4. LES ORGANIGRAMMES D'IMPLEMENTATIONS	49
4.1. Algorithme principal du système de traitement d'images et vidéos	50
4.2. Organigramme de Détection de Contours	51
4.3. Organigramme de Segmentation de Couleurs	52
4.4. Organigramme d'Ajustement de la Luminosité	53
4.5. Organigramme des opérations morphologiques (érosion et dilatation)	54
4.6. Organigramme général de l'application	55
5. RESULTATS ET DISCUSSION	56
5.1. Détection des contours	56
5.1.1. Sobel	56

5.1.2. Prewit	57
5.1.3. Canny	58
5.2. Traitement morphologique	59
5.2.1. Érosion	59
5.2.2. Dilatation	60
5.3. HSV	61
5.4. Segmentation	64
5.5. Surprissions de l'arrière-plan en temps réelle	66
5.6. YOLO V8 en temps réelle	68
5.7. YOLO V8 (Détection des objets)	71
6. DES EXEMPLES D'APPLICATIONS SOUS MATLAB	73
6.1. Détection des contours	74
6.1.1. Sobel	74
6.1.2. Prewit	75
6.1.3. Canny	76
6.2. Traitement morphologique	77
6.2.1. Érosion	77
6.2.2. Dilatation	78
6.3. HSV	79
6.4. Surprissions de l'arrière-plan en temps réelle	80
6.5. Détection d'objet	81
7. Un commentaire comparatif entre MATLAB et PYTHON	82
8. CONCLUSION	82
Conclusion.....	84
Bibliographie	87
Résumé	93

Liste des figures

Figure 1: Résolution vidéo : différence entre HD et 4K.	4
Figure 2: Introduction à l'image numérique matricielle (Bitmap).	7
Figure 3: Définition et exemples de l'image vectorielle.	7
Figure 4: Groupe de pixels représentant la lettre A.	8
Figure 5: Comparaison de l'image selon différentes résolutions.	9
Figure 6: Représentation de la structure en pixels d'une image numérique.	10
Figure 7: Exemple d'histogramme.	12
Figure 8: Exemple d'image en niveaux de gris.	13
Figure 9: Palette des couleurs (indexées).	14
Figure 10: HSV (Teinte, Saturation, Valeur).	15
Figure 11: Représentation du cône TSL et de l'ensemble des couleurs de l'espace TSL....	17
Figure 12: Résultats du traitement d'image.	20
Figure 13: Résultat prétraitement.	21
Figure 14: Application du filtre moyenné (lissage) pour la réduction du bruit dans une image.	22
Figure 15: Comparaison avant et après lissage.	22
Figure 16: Un document numérique et sa Binarisation par BIT-Alpha.	24
Figure 17: Illustration de l'opération d'érosion en morphologie mathématique.	25
Figure 18: Segmentation par seuillage."	29
Figure 19: Détection des chaises.	32
Figure 20: L'architecture de R-CNN.	33
Figure 21: Architecture et comparaison des modèles de détection d'objet : Fast R-CNN et Faster R-CNN.	34
Figure 22: Architecture Faster R-CNN.	35
Figure 23: Le Framework Mask R-CNN pour la segmentation d'instances.	36
Figure 24: Quelques applications de suivi d'objets.	37
Figure 25: Détection et suivi d'objet : Les occultations (occlusions).	38
Figure 26: Changement d'illumination.	39
Figure 27: Suivi des oiseaux par des point.	41
Figure 28: Smartphone exécutant l'application IP Webcam.	48
Figure 29: Interface de l'application affichant l'adresse IP et les paramètres de connexion.	49
Figure 30: Organigramme de Détection de Contours.	51
Figure 31: Organigramme de Segmentation de Couleurs.	52
Figure 32: Organigramme d'Ajustement de la Luminosité.	53
Figure 33: Organigramme des opérations morphologiques (érosion et dilatation).	54

Abréviations et Notations

Abréviations :

API : Application Programming Interface.

BIT : Binary Image Technology.

BMP : Bitmap Image File.

CNN : Convolutional Neural Network.

CUDA: Compute Unified Device Architecture.

DPI: Dots Per Inch.

GPU: Graphics Processing Unit.

HD: High Definition.

HOG: Histogram of Oriented Gradients.

HSV: Hue, Saturation, Value.

ICR: Intelligent Character Recognition.

IDE: Integrated Development Environment.

IP: Internet Protocol.

JPG: Joint Photographic Experts Group.

OCR : Optical Character Recognition.

PAO : Publication Assistée par Ordinateur.

PIL: Python Imaging Library.

PNG: Portable Network Graphics.

PPI: Pixels Per Inch.

QR: Quick Response.

R-CNN: Region-based Convolutional Neural Network.

RGB: Red, Green, Blue.

RTSP: Real Time Streaming Protocol.

RVB : Rouge, Vert, Bleu.

SIFT : Scale-Invariant Feature Transform.

TSL : Teinte, Saturation, Luminosité.

YOLO: You Only Look Once.

Notations :

X, Y : coordonnées cartésiennes du point sur l'image.

F (x, y) : degré d'intensité.

Introduction :

L'avancement rapide des technologies numériques et de l'intelligence artificielle a donné lieu à de nombreuses applications basées sur le traitement d'images et de vidéos. Aujourd'hui, ces technologies jouent un rôle crucial dans plusieurs domaines tels que la surveillance intelligente, la médecine, l'industrie, les transports et la robotique. Le traitement d'image permet non seulement d'améliorer la qualité visuelle des données recueillies, mais également d'extraire des informations importantes et de détecter automatiquement des objets à partir d'images ou de flux vidéo.

Cependant, la complexité croissante des algorithmes de vision par ordinateur et l'augmentation du volume des données à traiter engendrent des besoins importants en ressources de calcul, particulièrement lorsque les applications exigent un fonctionnement en temps réel. Face à ces contraintes, les solutions basées sur le calcul parallèle et l'utilisation des processeurs graphiques se sont imposées comme une alternative performante aux architectures de calcul traditionnelles. En effet, grâce à leur capacité à exécuter simultanément un grand nombre d'opérations, les processeurs graphiques permettent d'accélérer considérablement les traitements liés à l'analyse d'images et à la détection d'objets.

La carte graphique NVIDIA Quadro M1200 représente une solution susceptible d'améliorer les performances des applications de vision par ordinateur développées sous Python. Il est donc légitime de se demander si cette technologie est efficace dans le contexte du traitement d'images et de vidéos. Cela conduit à se poser la question suivante :

L'utilisation de la carte graphique NVIDIA Quadro M1200 permet-elle d'améliorer significativement la rapidité et les performances des opérations de traitement d'images, de vidéos et de détection d'objets en temps réel par rapport à une exécution classique ?

Pour répondre à cette question, deux hypothèses ont été retenues :

La première hypothèse suppose que l'utilisation de la carte Quadro M1200 permet une accélération significative des traitements grâce aux capacités du calcul parallèle.

La seconde hypothèse suppose que l'intégration de cette carte n'apporte pas d'amélioration notable des performances et que les temps d'exécution demeurent relativement similaires à ceux obtenus avec une architecture classique.

L'objectif principal de cette étude est d'évaluer l'apport du calcul parallèle de la carte NVIDIA Quadro M1200 dans l'amélioration des performances d'une application de traitement d'images, de vidéos et de détection d'objets développée sous Python, en mettant particulièrement l'accent sur la rapidité d'exécution et le traitement en temps réel.

Cette étude est structurée en quatre chapitres. Le premier chapitre présente les généralités relatives aux images numériques et à leurs principales caractéristiques. Le deuxième chapitre est consacré aux techniques fondamentales du traitement d'images telles que le prétraitement, le filtrage, la binarisation, la morphologie mathématique et la segmentation. Le troisième chapitre traite de la détection et du suivi d'objets ainsi que des

principales méthodes utilisées dans le domaine de la vision par ordinateur. Enfin, le quatrième chapitre est dédié à la réalisation pratique de l'application développée sous Python, à l'exploitation de la carte graphique Quadro M1200 ainsi qu'à l'analyse des résultats obtenus afin de vérifier les hypothèses formulées.

En fin de compte, cette étude vise à mettre en évidence l'intérêt du calcul parallèle dans les applications modernes de vision par ordinateur et à évaluer l'efficacité de la carte Quadro M1200 pour répondre aux exigences de rapidité et de performance des traitements en temps réel.

Chapitre I

Généralités sur l'image

1. INTRODUCTION :

Dans un monde marqué par une révolution numérique sans précédent, l'image s'impose comme un élément fondamental dans la communication, la représentation et l'analyse de l'information. Elle constitue un support riche et expressif permettant de transmettre des données complexes de manière claire et intuitive. Son utilisation s'étend aujourd'hui à de nombreux domaines, tels que la médecine, la télédétection, l'industrie, la sécurité, ainsi que les systèmes intelligents.

L'évolution des technologies informatiques a conduit à l'émergence des images numériques, offrant de nouvelles possibilités en matière de stockage, de traitement et d'exploitation de l'information visuelle. Contrairement aux images analogiques, les images numériques peuvent être manipulées à l'aide d'outils informatiques, ce qui permet d'améliorer leur qualité, de corriger leurs imperfections et d'en extraire des caractéristiques pertinentes.

Dans ce contexte, le traitement d'image apparaît comme un domaine scientifique et technique en plein essor. Il regroupe un ensemble de méthodes et de techniques visant à transformer une image afin d'en faciliter l'interprétation, que ce soit par l'être humain ou par des systèmes automatisés. Ces traitements peuvent concerner l'amélioration de la qualité visuelle, la compression, la segmentation ou encore l'analyse des contenus.

L'étude des images numériques nécessite une compréhension approfondie de leurs propriétés fondamentales, telles que leur structure, leurs formats, leurs caractéristiques physiques et leur mode de codage. Ces éléments constituent la base indispensable pour toute application liée au traitement et à l'analyse d'images.

Et dans le cadre des avancées technologiques actuelles, l'utilisation des unités de traitement graphique (GPU), notamment les cartes NVIDIA Quadro M1200, joue un rôle essentiel dans l'accélération des algorithmes de traitement d'images et de vidéos, en permettant un traitement parallèle rapide et efficace des données volumineuses.

Ainsi, ce travail s'inscrit dans une démarche visant à explorer les fondements théoriques des images numériques ainsi que les principes essentiels du traitement d'image. Il a pour objectif de mettre en évidence l'importance de ces concepts dans le développement des technologies modernes et leur rôle dans diverses applications scientifiques et techniques.

2. L'IMAGE NUMERIQUE :

2.1. Définition :

Selon (Zerairia A, Drici A 2020/2021) Une image numérique est une représentation visuelle où la surface est fractionnée en unités de dimension constante (connues sous le nom d'unités ou pixels), chaque unité possédant un indice distinct de gris ou de couleur.

La conversion d'une image analogique en format numérique est ce que l'on appelle la numérisation. Il est illustré par une matrice à deux dimensions de valeurs $f(x, y)$, où :

X, Y : coordonnées cartésiennes du point sur l'image.

F (x, y) : degré d'intensité.

Chaque point illustre une évaluation de l'intensité de la lumière détectée par le capteur.[1]

Pour mieux comprendre, on peut comparer une image numérique à une mosaïque ou à un tableau composé de petits carrés. Chaque carré correspond à un pixel, et chacun possède une couleur ou un niveau de gris. Plus il y a de pixels, plus l'image est précise et détaillée.

Par exemple, une image en noir et blanc est composée uniquement de niveaux de gris allant du noir au blanc. En revanche, une image en couleur associe plusieurs valeurs (rouge, vert, bleu) à chaque pixel pour former différentes couleurs.

De plus, lors de la numérisation, une image réelle (comme une photo papier) est transformée en une grille de points. Chaque point reçoit une valeur numérique qui représente la lumière captée à cet endroit. Ainsi, une image peut être représentée sous forme d'un tableau de nombres, où chaque valeur correspond à l'intensité lumineuse d'un pixel.

Par exemple, une image de taille 100×100 contient 10 000 pixels, chacun ayant une valeur qui définit sa couleur ou son niveau de gris.

3. LES DIFFERENTS TYPES D'IMAGE :

Les images existent sous différentes formes selon leur nature et leur utilisation. Chaque type d'image possède des caractéristiques particulières qui influencent sa qualité et son usage. Comprendre ces différents types permet de mieux choisir et utiliser les images selon les besoins.

3.1. L'imagerie vidéo :

3.1.1. Définition d'une vidéo :

D'après (Dictionnaire de l'Académie française) L'ensemble des méthodes électroniques facilitant l'enregistrement, le traitement, la diffusion et la représentation d'images en mouvement avec son accompagnement ; équipement, dispositif qui exploite ces techniques. Vidéo analogique et numérique. Vidéo en haute résolution. Réaliser un film, un documentaire. L'utilisation de la vidéo dans le processus d'arbitrage d'une rencontre sportive. Une vidéo à faible poids. [2]

Par exemple, regarder un film sur un téléphone, suivre un cours en ligne ou visionner une vidéo sur internet sont des utilisations courantes de la vidéo numérique. Dans le domaine sportif, la vidéo est utilisée pour revoir certaines actions et aider les arbitres à prendre des décisions plus justes.

On distingue également la vidéo analogique, utilisée autrefois avec les cassettes, et la vidéo numérique, qui est aujourd'hui la plus répandue grâce à sa meilleure qualité et sa facilité de stockage et de partage. Une vidéo en haute résolution offre une image plus nette,

tandis qu'une vidéo à faible poids occupe moins d'espace de stockage, mais peut être de qualité réduite.

3.1.2. Résolution vidéo :

Selon le site (Blog eavs) La résolution correspond à une donnée de densité. Elle représente le nombre de pixels sur une distance d'un pouce (2,54 cm). La résolution est exprimée en ppi (pixels par pouce). Cette information est liée à la fois aux dimensions de l'écran et à sa résolution.

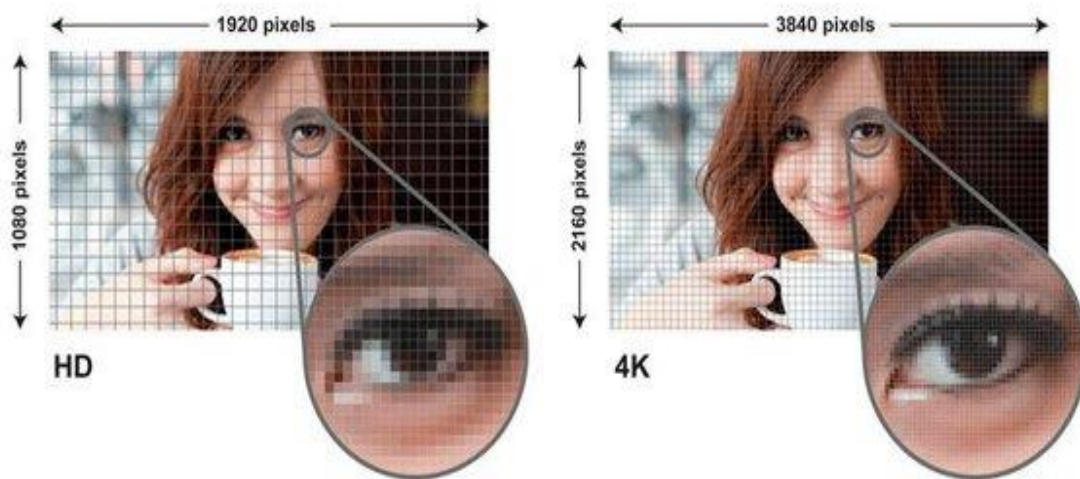


Figure 1: Résolution vidéo : différence entre HD et 4K. [3]

Par exemple, une image en **HD (1920 × 1080 pixels)** contient moins de pixels qu'une image en **4K (3840 × 2160 pixels)**. Ainsi, l'image 4K offre une meilleure qualité visuelle, avec des détails plus précis et des contours plus nets.

De plus, lorsqu'on agrandit l'image, la version HD peut apparaître pixelisée, alors que l'image 4K conserve une meilleure netteté grâce à sa densité de pixels plus élevée.

3.1.3. Résolution temporelle d'une vidéo :

Selon (Ouchar Ch A 2012) La définition de la résolution temporelle d'une vidéo repose sur le nombre d'images qui se succèdent chaque seconde. Pour prévenir les inconvenances liées aux clignotements, tout en tenant compte des particularités de l'œil humain qui conserve une image pendant environ 10 microsecondes sur sa rétine ; il est nécessaire que les images défilent à un rythme déterminé :

- Approximativement 24 images par seconde pour un film de cinéma.
- 25 images par seconde pour la télévision européenne.
- Approximativement 30 images par seconde pour la télévision nord-américaine et japonaise. [4]

La résolution temporelle d'une vidéo traduit la continuité apparente du mouvement en fonction du nombre d'images affichées par seconde. En effet, plus la fréquence d'images est élevée, plus la restitution du mouvement est fluide et fidèle à la réalité.

Par exemple, une vidéo enregistrée à 24 images par seconde, standard du cinéma, offre une perception visuelle satisfaisante du mouvement. En revanche, des fréquences plus élevées, telles que 30 ou 60 images par seconde, permettent d'améliorer la fluidité, notamment dans les scènes rapides comme les événements sportifs.

Ainsi, une faible résolution temporelle entraîne une perception saccadée du mouvement, tandis qu'une résolution temporelle élevée assure une meilleure qualité visuelle.

3.1.4. Définition d'un objet vidéo :

"Définir un objet vidéo est chose difficile. Intuitivement, un objet vidéo est défini par une forme, une texture et un mouvement (rigide ou non rigide). Cependant, la notion d'objet vidéo est beaucoup plus descriptive qu'une simple région. Comme expliqué dans le préambule, et dans la deuxième partie du manuscrit, un objet vidéo peut par exemple être un modèle 3D.

La notion d'objet ne fait pas forcément référence à un objet du monde réel. En effet, dans le domaine de la vidéo, un objet n'est pas nécessairement un objet d'une scène 3D mais plutôt le résultat de l'analyse de la projection d'un monde 3D sur un plan. Ainsi, un objet vidéo est défini comme une région de la vidéo conforme à un modèle. On peut par exemple avoir pour modèle :

- un modèle de mouvement (e.g. affine, quadratique, etc)
- un modèle d'objet physique (e.g. modèle de visage, modèle de corps humain etc)

La notion d'objets est alors subjectivement définie par rapport aux modèles utilisés." [5]

Exemples :

- Une personne qui marche dans une vidéo peut être considérée comme un objet vidéo.
- Une voiture en mouvement dans une scène est un objet vidéo grâce à son déplacement.
- Un visage détecté dans une vidéo (reconnaissance faciale) est un objet basé sur un modèle.
- Une zone qui bouge dans une vidéo de surveillance peut être identifiée comme un objet, même si on ne sait pas exactement ce que c'est.

3.1.5. Séquence d'image :

"Une séquence photographique est une série de clichés pris à intervalles réguliers (fréquence) qui permettent de fixer sur la pellicule un procédé, une situation ou une idée." [6]

Exemples :

- Un sportif en train de sauter : plusieurs photos montrent les différentes étapes du saut (course, impulsion, envol, réception).
- La croissance d'une plante : une photo prise chaque jour montre comment elle grandit progressivement.
- Une personne qui prépare un gâteau : chaque étape (mélange, cuisson, décoration) est capturée en images successives.
- Une animation en stop motion : des objets immobiles sont photographiés plusieurs fois pour créer l'illusion du mouvement.

3.1.6. Nature de la scène :

D'après (Rafael C. Gonzalez, Richard E. Woods 2018) La nature de la scène désigne les caractéristiques physiques et photométriques d'une scène observée, déterminées par deux composantes principales : l'illumination incidente (lumière reçue par les objets) et la réflectance des objets (capacité des surfaces à réfléchir ou absorber cette lumière). [7]

La nature de la scène correspond à l'apparence visuelle d'une image, déterminée par l'éclairage reçu et par la manière dont les objets réfléchissent ou absorbent la lumière.

4. LES FORMATS D'IMAGE :

Les formats d'image sont les différentes façons de sauvegarder une image numérique. Chaque format a ses propres caractéristiques, comme la qualité et la taille du fichier. Le choix du format dépend de l'utilisation de l'image.

4.1. Image numérique matricielle (Bitmap) :

Selon (Fondement de Traitement D'images) Image matricielle (ou image bitmap) : elle est constituée d'une grille multidimensionnelle de pixels, chaque dimension correspondant à une dimension spatiale (telle que la hauteur, la largeur, la profondeur), temporelle (comme la durée) ou autre (par exemple, un certain niveau de résolution). Une image se présente comme une matrice de pixels (composants d'image) organisée en colonnes et en rangées.

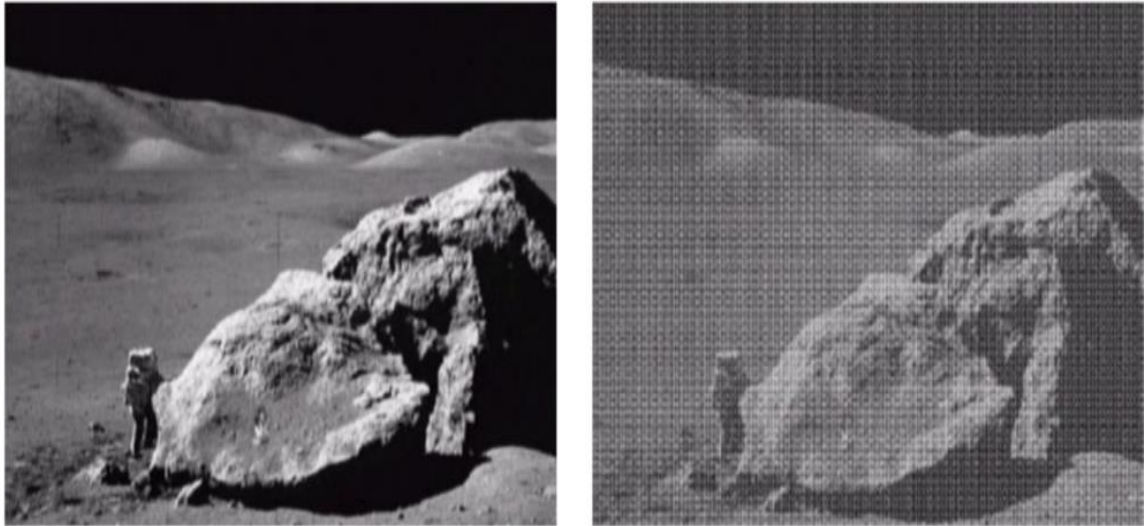


Figure 2: Introduction à l'image numérique matricielle (Bitmap).[8]

- Plus l'image comporte de pixels, plus elle est précise.

Exemples :

- Une photo prise avec un téléphone est une image bitmap : si on zoome beaucoup, on voit apparaître les petits carrés (pixels).
- Une image sur un écran d'ordinateur ou de télévision est composée de pixels alignés en grille.
- Une capture d'écran est aussi une image matricielle.

4.2. Image numérique vectorielle :

"Ensemble d'entités géométriques telles qu'un cercle, un rectangle ou un segment représenté par des formules mathématiques (un rectangle est défini par deux points, un cercle par un centre et un rayon, une courbe par plusieurs points et une équation).



Figure 3: Définition et exemples de l'image vectorielle."[9]

Exemples :

- Un logo d'entreprise utilisé sur une affiche ou une carte de visite sans devenir flou.
- Les dessins techniques et plans d'architecture (bâtiments, schémas).
- Les icônes (comme celles des applications sur un téléphone).
- Les polices de caractères (lettres) affichées sur un ordinateur.

5. LES CARACTERISTIQUES DE L'IMAGE :**5.1. Dimension :**

"C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels).

Le nombre de lignes de cette matrice multipliée par le nombre de colonnes nous donne le nombre total de pixels dans une image." [10]

La dimension d'une image correspond à sa taille en pixels, c'est-à-dire le nombre de points qui la composent en largeur et en hauteur. Plus une image contient de pixels, plus elle est grande et détaillée.

5.2. Pixel :

Selon (Gacemi S, Mimouni M 2016/2017) L'expression anglaise « Picture éléments » désigne les éléments d'image, le pixel étant le plus petit élément de l'image. C'est une entité mesurable à laquelle on peut attribuer une structure et une quantification. Tandis qu'un bit représente la plus petite unité d'information qu'un ordinateur peut gérer, un pixel constitue l'élément le plus minuscule que les matériels et logiciels de visualisation ou d'impression peuvent manipuler. Par exemple, la lettre A peut être représentée comme un ensemble de pixels dans l'image ci-dessous :



Figure 4: Groupe de pixels représentant la lettre A.[11]

5.3. Résolution et définition :

a) Résolution :

Selon (Design graphique et applications multimédias) L'évaluation de la résolution d'une image se fait en fonction du nombre de pixels par unité de longueur de l'objet à scanner (généralement exprimé en dpi : dots per inch, Un pouce équivaut à 2,54 centimètres). Ce paramètre est établi lors de :

- La numérisation d'une image à l'aide d'un scanner.
- La génération d'un nouveau fichier image avec un logiciel approprié (comme Photoshop, par exemple) :



Figure 5: Comparaison de l'image selon différentes résolutions.[12]

La résolution d'une image fait référence à sa clarté, qui est déterminée par le nombre de pixels présents dans une unité de longueur spécifique (généralement le pouce, mesuré en dpi). Plus un espace réduit contient de pixels, plus l'image est claire et détaillée.

Exemples :

- * Une image à 300 dpi présente une netteté et un niveau de détail supérieurs par rapport à une image à 72 dpi, en particulier pour l'impression.
- * Pour éviter le flou sur une impression papier, il est nécessaire d'avoir une résolution élevée pour la photo.
- * Lors de l'utilisation d'un scanner, il est possible d'opter pour une résolution supérieure afin d'obtenir une image extrêmement précise.
- * Pour un chargement plus rapide, une image affichée sur une page web peut avoir une résolution réduite.

b) La définition : la quantité de pixels :

D'après le site (graphiste.com) La définition fait référence à la quantité de pixels (ou d'informations) contenue dans une image. On la détermine en multipliant le nombre de pixels qui composent la largeur de l'image ou du panneau par le nombre de ceux qui figurent sur la hauteur. Cela nous donne donc la définition en pixels.



Figure 6: Représentation de la structure en pixels d'une image numérique.[13]

Exemples :

- Une image de 1920×1080 pixels contient 2 073 600 pixels au total.
- Une image de 800×600 pixels est moins détaillée qu'une image en haute définition.
- Une photo de smartphone en haute qualité contient plusieurs millions de pixels, ce qui permet de zoomer sans trop perdre en netteté.
- Une image de faible définition devient vite floue lorsqu'on l'agrandit.

5.4. Contraste :

"C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images." [14]

Exemples :

- Une photo en niveaux de gris avec un ciel très lumineux et des objets très sombres à un fort contraste.
- Un texte noir sur un fond blanc est un exemple de contraste élevé, donc facile à lire.
- Une image avec des couleurs très proches (gris clair et blanc) a un faible contraste et paraît plus "fade".
- En photographie, augmenter le contraste rend les détails plus visibles, tandis que le diminuer rend l'image plus douce.

5.5. Contour :

"Les contours représentent des changements soudains d'intensité des pixels, qui correspondent généralement aux limites des objets. La détection des contours facilite la reconnaissance des formes et la segmentation." [15]

Exemples :

- Dans une photo, le contour d'une personne permet de la séparer du fond.
- Le bord d'une table ou d'un livre apparaît comme un contour clair sur une image.
- En vision par ordinateur, les contours sont utilisés pour reconnaître des objets comme une voiture ou un visage.
- Dans un dessin en niveaux de gris, les traits qui définissent les formes sont des contours

5.6. Bruit :

"Le bruit (parasites) dans l'image est vu comme un changement soudain de l'intensité des pixels par rapport aux pixels voisins. Il est créé en éclairant les dispositifs optiques et électroniques du capteur." [16]

Exemples :

- Une photo prise dans un endroit sombre peut contenir des points colorés ou granuleux (bruit numérique).
- Une image agrandie peut montrer des petits grains qui dégradent la qualité.
- Les anciennes photos numériques ou les vidéos de mauvaise qualité présentent souvent du bruit visible.
- Sur une image de ciel sombre, on peut voir de petits points lumineux qui ne devraient pas exister.

5.7. Histogramme :

Selon le site (Wikipedia) Un histogramme est une illustration graphique de la répartition des pixels d'une image ou d'une section de celle-ci en fonction de leur intensité. La version traditionnelle distribue la gamme d'intensité sur l'axe horizontal, le noir ou les teintes sombres se trouvant à l'origine du diagramme. Sur la base du nombre de pixels, l'histogramme peut être illustré en termes de proportions ou de probabilités. [17]

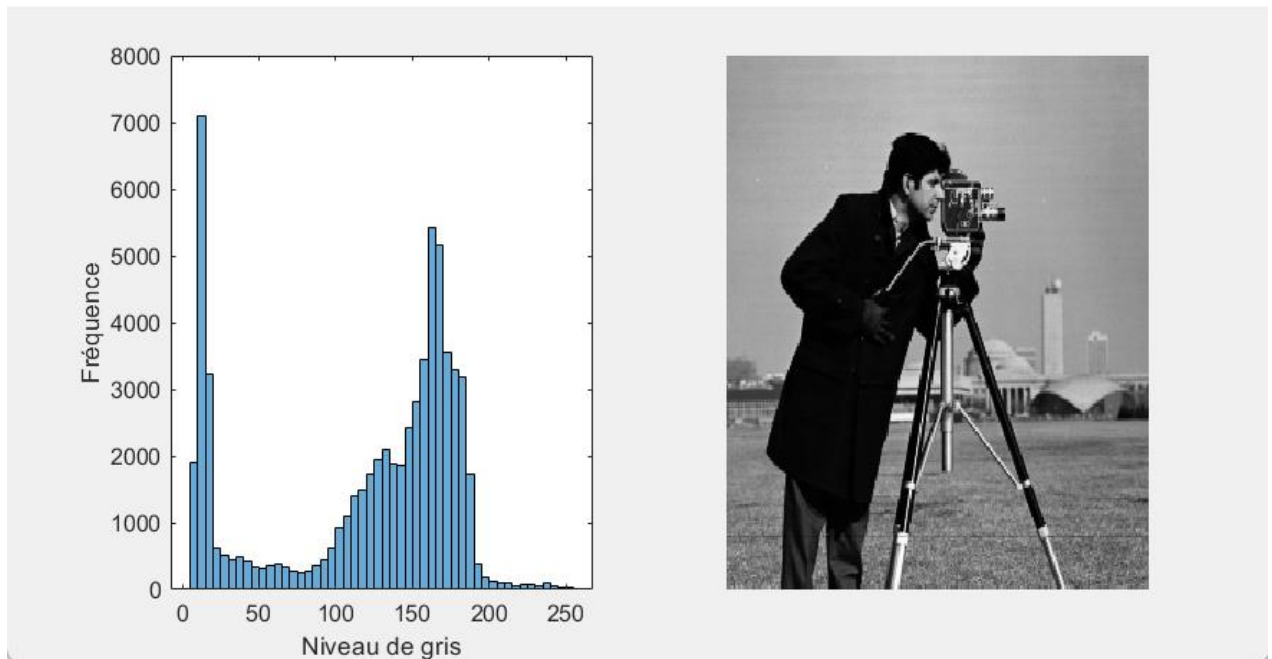


Figure 7: Exemple d'histogramme. [18]

5.8. Codage :

Le codage joue un rôle essentiel dans le traitement des images numériques. Il permet de manipuler, stocker et transmettre les images de manière efficace. Cette étape est indispensable dans le fonctionnement des systèmes informatiques et des technologies modernes.

5.8.1. Images binaires :

"Une image binaire est une image pour laquelle chaque pixel ne peut avoir pour valeur que 0 ou 1. La manipulation de telles images regorge d'outils spécialisés ainsi que de théories mathématiques pour plusieurs raisons :

- Les débuts du traitement des images numériques ne permettaient pas le traitement d'images complexes (problème de temps de calcul, d'espace mémoire disponible et qualité des périphériques de sortie). De plus, les premières applications (reconnaissance de caractères, analyse de traces laissées dans les chambres à bulles par des particules) vers 1950 s'adaptaient bien à ce type d'images.
- Les images binaires sont un contexte simple permettant une formalisation mathématique des problèmes par des outils tels que la topologie. Dans le domaine de la vision industrielle (détection de défauts, contrôle qualité, mesure, ...) on considère souvent l'image binaire comme un passage obligé, suivant en général la phase de segmentation.

- Deux catégories d'outils sont alors nécessaires pour d'une part le codage efficace (et éventuellement la compression) et d'autre part pour le traitement (analyse et description des formes)." [19]

Exemples :

- Un texte scanné en niveaux de gris (lettres noires sur fond blanc).
- Le contour d'un objet après segmentation dans une image.
- Un QR code, composé uniquement de pixels noirs et blancs.
- Une silhouette d'objet utilisée pour la reconnaissance de formes en vision industrielle

5.8.2. Images en niveaux de gris :

"Le codage dit en niveaux de gris permet d'obtenir plus de nuances que le simple noir et blanc.

Il offre des possibilités supplémentaires pour coder le niveau de l'intensité lumineuse.

La couleur est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d'obtenir 256 niveaux de gris (0 pour le noir et 255 pour le blanc). On peut aussi le faire avec 16 niveaux de gris (4 bits)." [20]

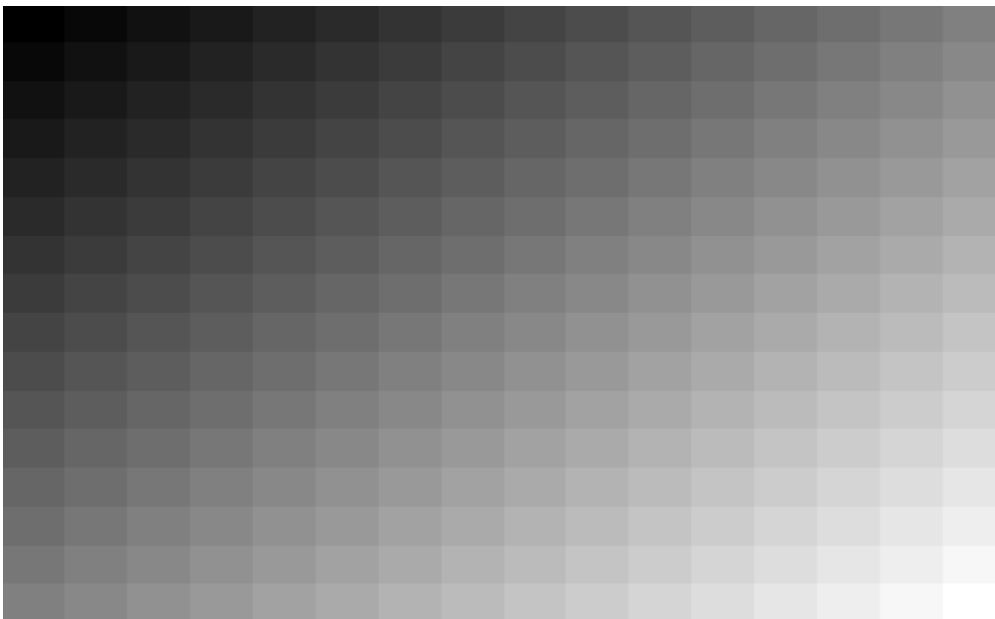


Figure 8: Exemple d'image en niveaux de gris. [21]

5.8.3. Images en couleurs :

"Image en couleur qui résulte de la combinaison d'image monochromes d'une même scène." [22]

5.8.4. Images en couleur indexées :

"La couleur indexée est une manière de reproduire une image en couleur, à l'écran de l'ordinateur aussi bien qu'en impression. En sérigraphie, les images imprimées en couleurs indexées ont un rendu bien supérieur aux images reproduites en quadrichromie.

Les textiles imprimés en couleurs indexées ont un touché doux. Les couleurs ne sont pas superposées mais imprimées en bord à bord. Sur un textile noir, elles sont imprimées mouillées sur mouillées sur un blanc de fond préalablement séché." [23]



Figure 9: Palette des couleurs (indexées). [24]

5.8.5. Le système HSV (Hue. Saturation. Value) ou TSL (Teinte. Saturation. Luminosité) :

1. Le système HSV (Hue. Saturation. Value) :

"Le modèle de couleur HSV (Teinte, Saturation, Valeur) représente les couleurs d'une manière qui correspond étroitement à la perception humaine. Contrairement au modèle RGB, qui se concentre sur le mélange des couleurs primaires, le modèle HSV sépare la couleur en trois composantes intuitives.

Les trois composantes sont les suivantes

1. **Teinte** : Représente le type de couleur (couleur de base).
2. **Saturation** : Représente l'intensité de la couleur.
3. **Valeur** : Représente la luminosité de la couleur.

HSV (Teinte, Saturation et Valeur)

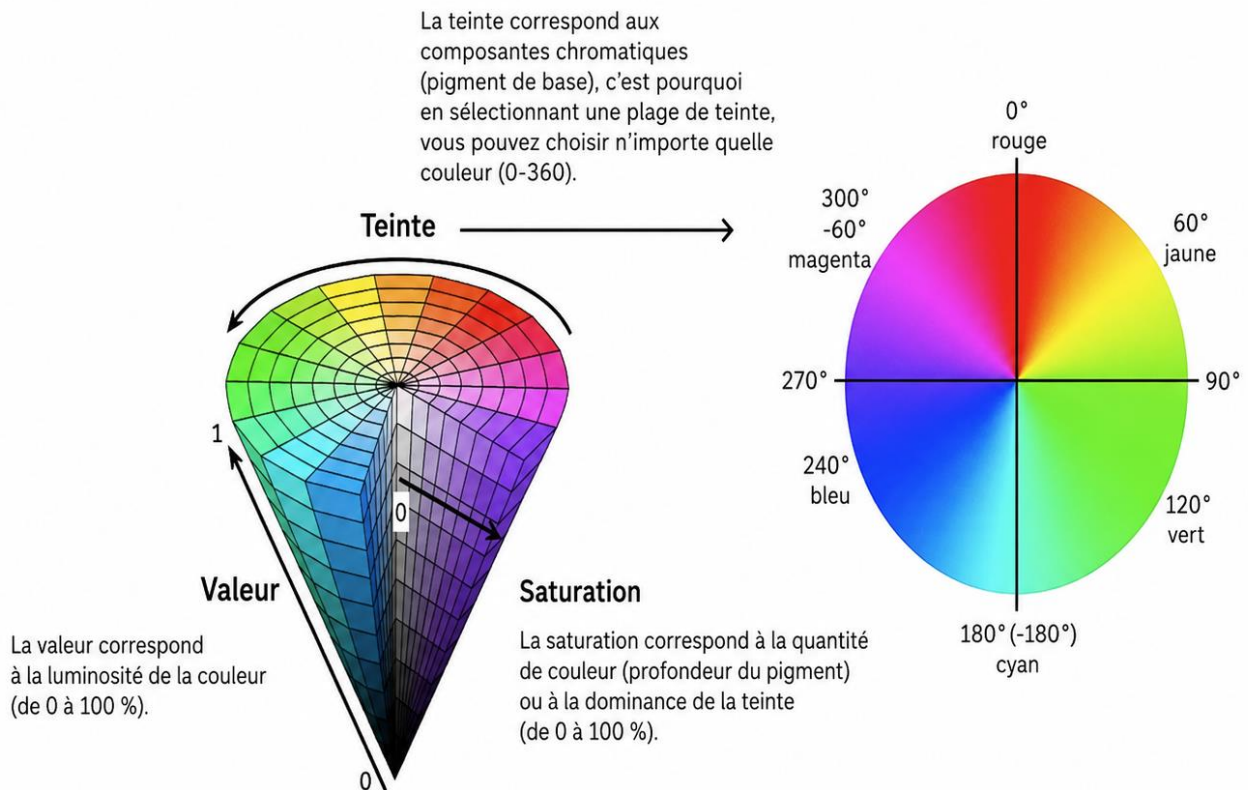


Figure 10: HSV (Teinte, Saturation, Valeur).

Essayons de comprendre cela à l'aide de l'image ci-dessous :

L'image ci-dessus illustre la représentation des couleurs par le modèle HSV. L'axe vertical représente la valeur (V), qui mesure la luminosité, du noir (en bas) à la luminosité maximale (en haut). La base circulaire du cône correspond aux différentes valeurs de teinte (H), chaque couleur étant positionnée sur le cercle selon son angle, de 0° à 360°. En s'éloignant du centre, on observe une augmentation de la saturation (S), qui mesure l'intensité de la couleur, du gris au centre (faible saturation) à la couleur vive sur le bord extérieur (forte saturation).

Teinte : Comme illustré dans le diagramme conique, la teinte représente le type de couleur déterminé par une position sur le cercle chromatique (à droite). Elle s'étend de 0° à 360°, le rouge correspondant à 0°, le vert à 120° et le bleu à 240°. En tournant sur le cercle, on sélectionne différentes couleurs de base.

Saturation : Représentée par l'axe horizontal du cône, la saturation mesure l'intensité ou la pureté d'une couleur. À 0 % de saturation, les couleurs apparaissent comme des nuances de gris (au centre du cône). Plus la saturation augmente (en s'éloignant du centre), plus les couleurs deviennent vives et éclatantes.

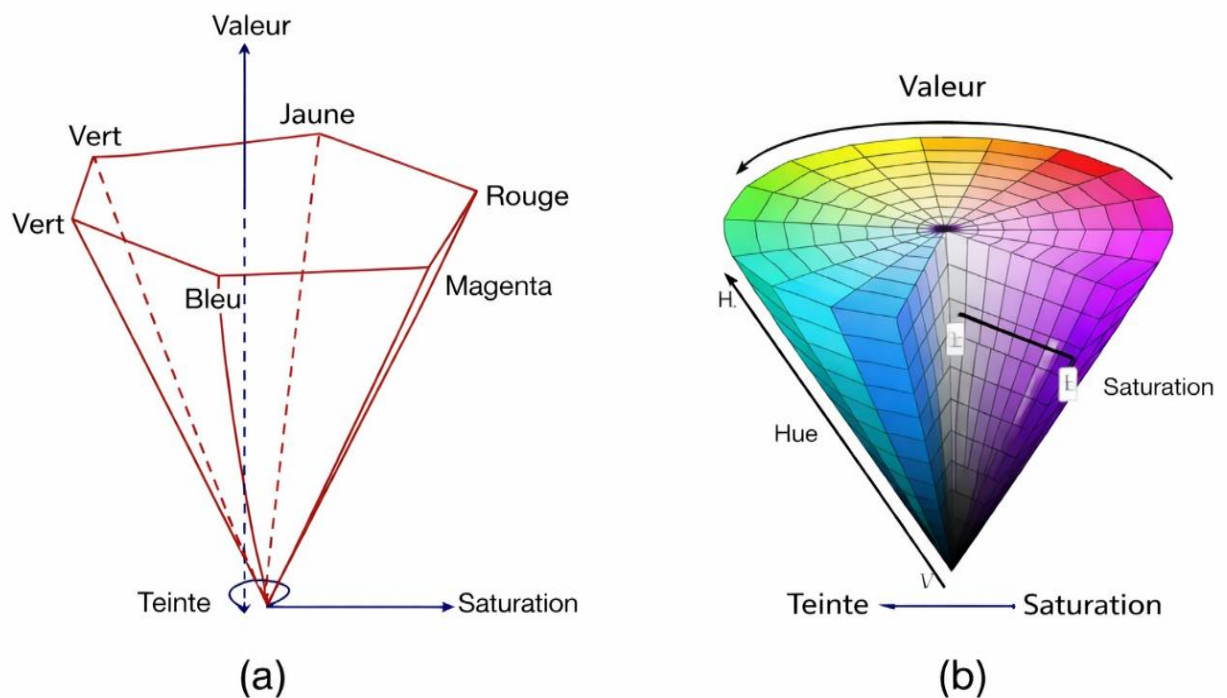
Valeur : Elle est représentée par l'axe vertical du cône. La valeur mesure la luminosité de la couleur, de 0 % (noir) en bas à 100 % (luminosité maximale) en haut. À luminosité maximale, la couleur est parfaitement visible ; les valeurs plus faibles la rendent plus sombre."[25]

2. Le système TSL (Teinte. Saturation. Luminosité) :

"L'espace colorimétrique TSL (Teinte, Saturation, Luminance) a été développé pour offrir une manipulation intuitive des couleurs et permettre une sélection manuelle facile dans les applications interactives de type PAO. Il permet de décomposer une couleur en trois critères physiologiques :

- La teinte qui correspond à la perception de la couleur, $0 \leq T \leq 360$
- La saturation qui correspond à la pureté de la couleur (vif ou terne), $0 \leq S \leq 1$
- La luminance correspondant à la quantité de lumière de la couleur (clair ou sombre), $0 \leq L \leq 1$.

Le modèle de couleurs TSL est utilisé pour la manipulation de la teinte et de la saturation car il permet de modifier directement ces valeurs (contrairement au modèle RVB). Les trois composantes TSL définissent un cône représenté dans la fig. 2.a où l'ensemble des couleurs représentables y est également synthétisé (fig. 11.b). Le principal avantage de cet espace est que chacune de ses composantes est reliée à une grandeur physique facilement interprétable visuellement. Ainsi augmenter la luminosité d'une couleur se fera uniquement en augmentant la composante L. Cette propriété permet de s'affranchir de la corrélation entre la teinte, la luminance et la saturation et offre ainsi un contrôle plus souple dans la manipulation des couleurs. Ce sont les raisons pour lesquelles l'espace de représentation des couleurs TSL a été retenu.



(A) : Cône TSL

(B) : Ensemble des couleurs de l'espace TSL"

Figure 11: Représentation du cône TSL et de l'ensemble des couleurs de l'espace TSL. [26]

6. CONCLUSION :

En conclusion, l'image représente un moyen de communication puissant et incontournable, utilisé dans de nombreux domaines tels que l'éducation, les médias, la publicité et les technologies numériques. À travers ses différentes formes, qu'elle soit fixe ou animée, analogique ou numérique, elle permet de transmettre des informations de manière rapide, claire et souvent universelle.

Par ailleurs, les progrès technologiques ont considérablement transformé la production, le traitement et la diffusion des images, les rendant accessibles à tous et omniprésentes dans notre quotidien. Cette évolution souligne l'importance de bien comprendre les notions de base liées à l'image, notamment sa nature, ses types et ses fonctions.

Ainsi, la maîtrise de ces généralités constitue une étape essentielle pour mieux appréhender les aspects techniques plus avancés.

Chapitre II

Traitement d'image

1. INTRODUCTION :

Ce domaine regroupe un ensemble de techniques permettant de transformer une image numérique afin de la rendre plus lisible, plus nette et plus exploitable. Ces opérations jouent un rôle essentiel dans de nombreux domaines comme la médecine, la surveillance, la robotique ou encore la vision par ordinateur.

Dans ce cadre, plusieurs étapes fondamentales sont mises en œuvre. Tout d'abord, le prétraitement d'image permet d'améliorer la qualité des images en les corrigeant, en les standardisant et en les préparant pour une analyse plus précise.

Ensuite, le filtrage d'image intervient pour réduire le bruit, améliorer les détails et corriger certains défauts à l'aide de filtres linéaires ou non linéaires.

Par la suite, la binarisation permet de simplifier l'image en la réduisant à deux niveaux afin de faciliter la reconnaissance des objets. La morphologie mathématique agit ensuite sur la structure des objets pour améliorer leur forme, éliminer les imperfections et mieux les isoler grâce à des opérations comme l'érosion et la dilatation.

Enfin, la segmentation d'image constitue une étape clé du traitement d'image, car elle permet de découper une image en plusieurs régions afin de distinguer les objets du fond et d'analyser chaque élément séparément. Différentes techniques de segmentation existent, comme la segmentation par régions, par croissance, par contours ou par seuillage, chacune adaptée à des besoins spécifiques.

Ainsi, ce domaine représente un ensemble de méthodes complémentaires permettant de passer d'une image complexe à une information structurée, exploitable et utile pour l'analyse automatique.

2. TRAITEMENT D'IMAGE :

"Le traitement d'images est une branche du traitement de signal dédiée aux images et vidéo.

Le traitement d'images est l'ensemble des opérations effectuées sur l'image, afin d'en améliorer la lisibilité et d'en faciliter l'interprétation. C'est, par exemple, le cas des opérations de rehaussement de contraste, élimination du bruit et correction d'un flou. C'est aussi l'ensemble d'opérations effectuées pour extraire des "informations" de l'image comme la segmentation et l'extraction de contours.



Figure 12: Résultats du traitement d'image." [27]

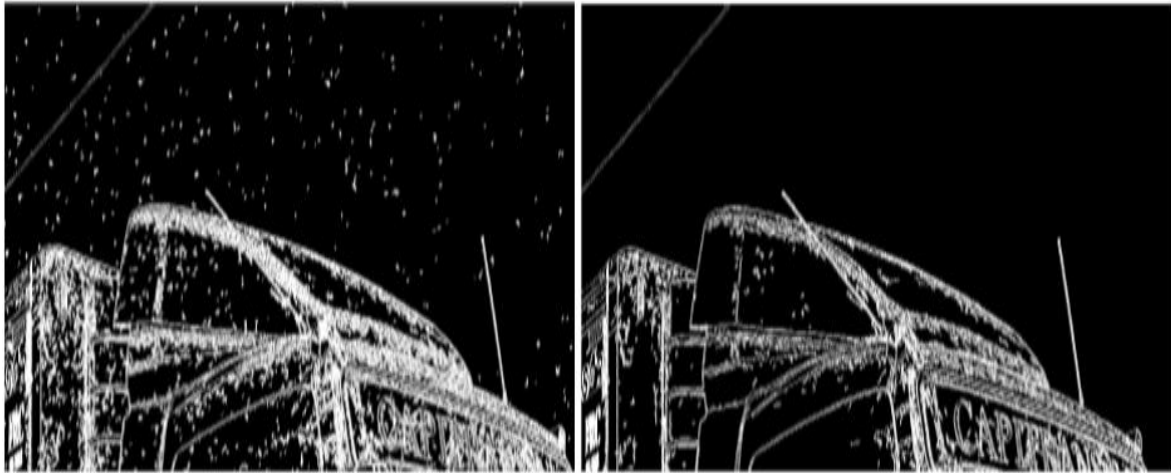
Exemples :

- Améliorer une photo sombre en augmentant la luminosité et le contraste.
- Supprimer le bruit (petits points parasites) pour rendre l'image plus nette.
- Corriger une image floue pour mieux voir les détails.
- Détecter les contours d'un objet pour reconnaître sa forme.
- Segmenter une image pour séparer les objets du fond (par exemple en médecine ou en surveillance).

3. PRETRAITEMENT D'IMAGE :

Selon le site (Microblink) L'amélioration de la qualité et l'extraction d'informations significatives des images numériques avant leur analyse par des algorithmes de vision artificielle ou d'apprentissage automatique sont réalisées grâce à un éventail de techniques connues sous le nom de prétraitement d'images. C'est un ensemble de procédures destinées à rectifier, trier, standardiser ou optimiser les images pour qu'elles soient plus adaptées à une analyse future. [28]

Le prétraitement est une étape préparatoire qui améliore et simplifie les images afin de rendre leur analyse plus précise et efficace.



Sans prétraitement

Après prétraitement

Figure 13: Résultat prétraitement. [29]

3.1. Filtrage :

" Le filtrage d'images est une opération fondamentale qui permet d'éliminer des éléments perturbateurs / non significatifs dans les images numériques afin d'améliorer la perception de certains détails, de réduire le bruit, de compenser certains défauts de capteur." [30]

Exemples :

- Utiliser un filtre pour enlever le bruit (petits points parasites) d'une photo.
- Appliquer un filtre de flou pour adoucir une image.
- Utiliser un filtre de netteté pour rendre les contours plus visibles.
- Corriger les défauts d'une image causés par un capteur (taches, variations de lumière).

3.1.1. Filtres linéaires :

D'après (R Sh Khan) Le traitement d'image utilise une technique appelée filtrage linéaire. Dans cette technique, la valeur du pixel final est calculée à partir d'une combinaison linéaire des valeurs des pixels voisins en entrée. Le filtrage linéaire peut être réalisé en utilisant la convolution. Il existe plusieurs exemples de filtrage linéaire, notamment le filtrage par moyenne et le filtrage gaussien. Le filtrage linéaire est une méthode importante dans le traitement d'image. [31]

Le filtrage linéaire permet de transformer une image de manière prévisible et stable.

Dans ce processus, la valeur de chaque pixel résultant est calculée en fonction d'une combinaison des valeurs des pixels voisins, auxquels sont attribués des poids spécifiques.

Cette approche garantit que le traitement de l'image est uniforme, car chaque pixel est affecté de manière égale, selon les mêmes règles.

Ceci est particulièrement utile pour des opérations telles que le floutage d'images, l'amélioration de la netteté ou encore la détection de contours et détails.

3.1.2. Filtre moyenné (lissage) :

"Le filtre moyenneur est une opération de traitement d'images utilisée pour réduire le bruit dans une image et/ou flouter une image. Par exemple, l'application d'un filtre moyenneur sur l'image de gauche donne l'image de droite :



Figure 14: Application du filtre moyenné (lissage) pour la réduction du bruit dans une image.

En zoomant, on peut voir en détail les effets du filtre ; le bruit clairement visible dans le ciel a bien été réduit mais les détails du visage et de la caméra sont floutés :



Figure 15: Comparaison avant et après lissage." [32]

Donc ; Le filtre moyenneur permet de rendre une image plus homogène et moins bruitée, en atténuant les variations brusques entre les pixels, ce qui améliore l'aspect visuel global mais peut diminuer la netteté des détails.

3.1.3. Filtre gaussien :

D'après (S Hadioui) Le filtre gaussien est une technique utilisée pour améliorer les images. Il aide à réduire le bruit et à créer un flou contrôlé. Pour cela, le filtre gaussien utilise une formule mathématique spéciale appelée fonction gaussienne. Cette fonction attribue de nouvelles valeurs aux pixels de l'image, ce qui permet d'obtenir une image plus lisse. Le filtre gaussien est donc très utile pour améliorer la qualité des images. [33]

Le filtre gaussien peut être compris comme un lissage pondéré où chaque pixel est influencé principalement par ses voisins proches, et beaucoup moins par les pixels éloignés. Cette approche permet de réduire le bruit de manière progressive et naturelle, tout en conservant globalement la structure de l'image, ce qui le rend particulièrement adapté comme étape préalable dans de nombreux traitements d'images (comme la détection de contours).

3.1.4. Filtre rehausseur de contours :

" les filtres rehausseurs de contours sont des passe-bande ou des passe-haut, c'est-à-dire qu'ils laissent les moyennes ou les hautes fréquences presque inchangées et qu'ils atténuent les basses fréquences." [34]

Les filtres rehausseurs de contours mettent en évidence les détails et les transitions rapides d'intensité dans une image, en accentuant les zones où les variations sont fortes (contours), tout en réduisant l'influence des zones uniformes, ce qui améliore la perception des formes et des limites des objets.

3.1.5. Filtres non linéaires :

Selon (R Sh Khan 2020) Un filtrage non linéaire se définit comme un filtrage qui ne peut être effectué par convolution ou multiplication de Fourier. Un exemple simple de filtre non linéaire est le filtre médian en mouvement. [35]

Un filtrage non linéaire traite les pixels en utilisant des règles comme le tri ou la sélection d'une valeur, plutôt que de faire une moyenne, ce qui permet de mieux enlever le bruit tout en gardant les contours de l'image. Il est donc possible de conserver les détails importants de l'image tout en éliminant le bruit indésirable. Le filtrage non linéaire est très utile pour améliorer la qualité des images. Il permet de réduire le bruit et de garder les contours nets. Le filtrage non linéaire utilise des règles comme le tri ou la sélection d'une valeur pour traiter les pixels, et cela donne de meilleurs résultats que la moyenne pour enlever le bruit et conserver les contours.

3.2. Binarisation :

"Le procédé de binarisation constitue une opération informatique de préparation au traitement de reconnaissance et qui n'est pas toujours intégrée dans un logiciel OCR ou ICR.

Elle constitue pourtant une étape majeure dans l'analyse et demande grand soin, car une mauvaise binarisation peut détruire beaucoup d'informations contenues dans le facsimilé scanné de l'original.

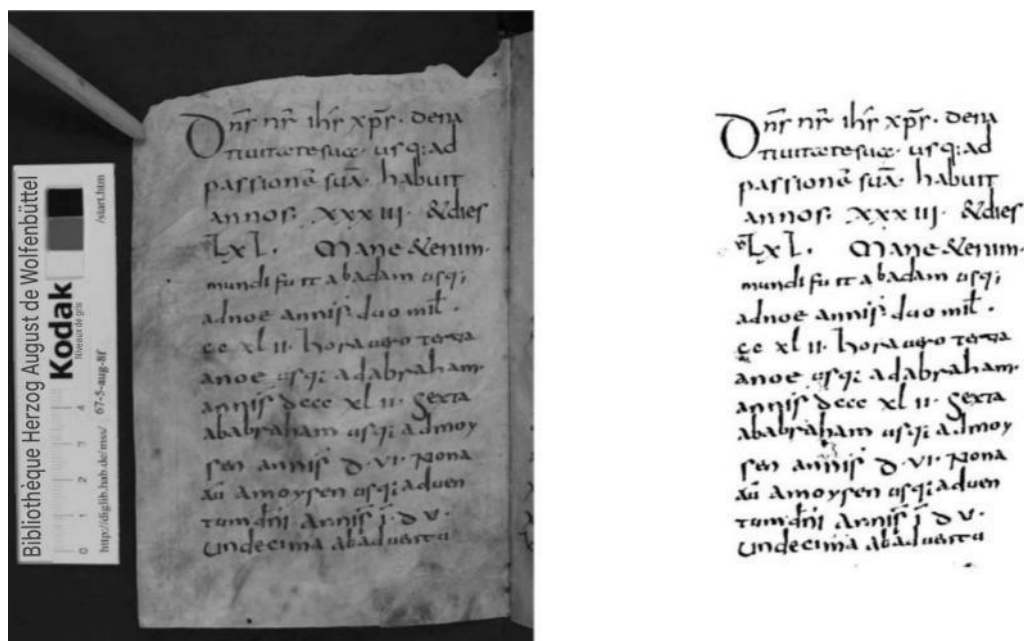


Figure 16: Un document numérique et sa Binarisation par BIT-Alpha." [36]

La binarisation est utilisée pour réduire une image à ses éléments les plus importants, qui sont l'objet et l'arrière-plan. Cela facilite la reconnaissance automatique de l'image. La sélection du seuil est très importante dans ce processus. Si le seuil n'est pas sélectionné correctement, des données essentielles pourraient être perdues ou mal interprétées. La binarisation est donc une étape cruciale pour une reconnaissance automatique précise.

3.3. Les opérateurs de la morphologie mathématique :

"Les opérateurs morphologiques sont couramment employés en segmentation. La morphologie mathématique est une théorie de traitement non linéaire de l'information, qui est apparue en France dans les années 60. Elle est aujourd'hui très largement utilisée en traitement d'images. Contrairement au traitement linéaire des images, la morphologie mathématique ne s'appuie pas sur le traitement du signal mais repose sur la théorie des ensembles, ce qui en fait une discipline relativement "auto-contenue" et formant un tout cohérent. L'objectif de cette fiche est de fournir les bases du traitement morphologique des images." [37]

La morphologie mathématique est principalement utilisée pour modifier la forme des objets dans une image, afin de les rendre plus clairs et plus faciles à analyser. Contrairement aux filtres classiques qui travaillent sur les valeurs de pixels, la morphologie mathématique agit directement sur la structure des objets, en prenant en compte leur taille, leurs contours et leurs trous.

Par exemple, la morphologie mathématique permet d'éliminer les petits points parasites, de combler des espaces vides à l'intérieur d'un objet, ou de séparer deux objets qui

sont collés. La morphologie mathématique est donc très utile en segmentation, car elle aide à mieux isoler les objets et à les préparer avant leur analyse. La morphologie mathématique est un outil précieux pour améliorer la qualité des images et faciliter leur analyse.

3.3.1. Erosion :

D'après le site (Tech Skill Guru) L'érosion est une opération morphologique qui réduit les limites des objets présents dans une image. Elle supprime les pixels situés aux limites des objets, ce qui a pour effet de diminuer leur taille. L'érosion diminue ainsi la taille de l'objet en éliminant les pixels aux bords. L'érosion est utilisée pour éliminer les détails fins et réduire le nombre de pixels des objets dans l'image.

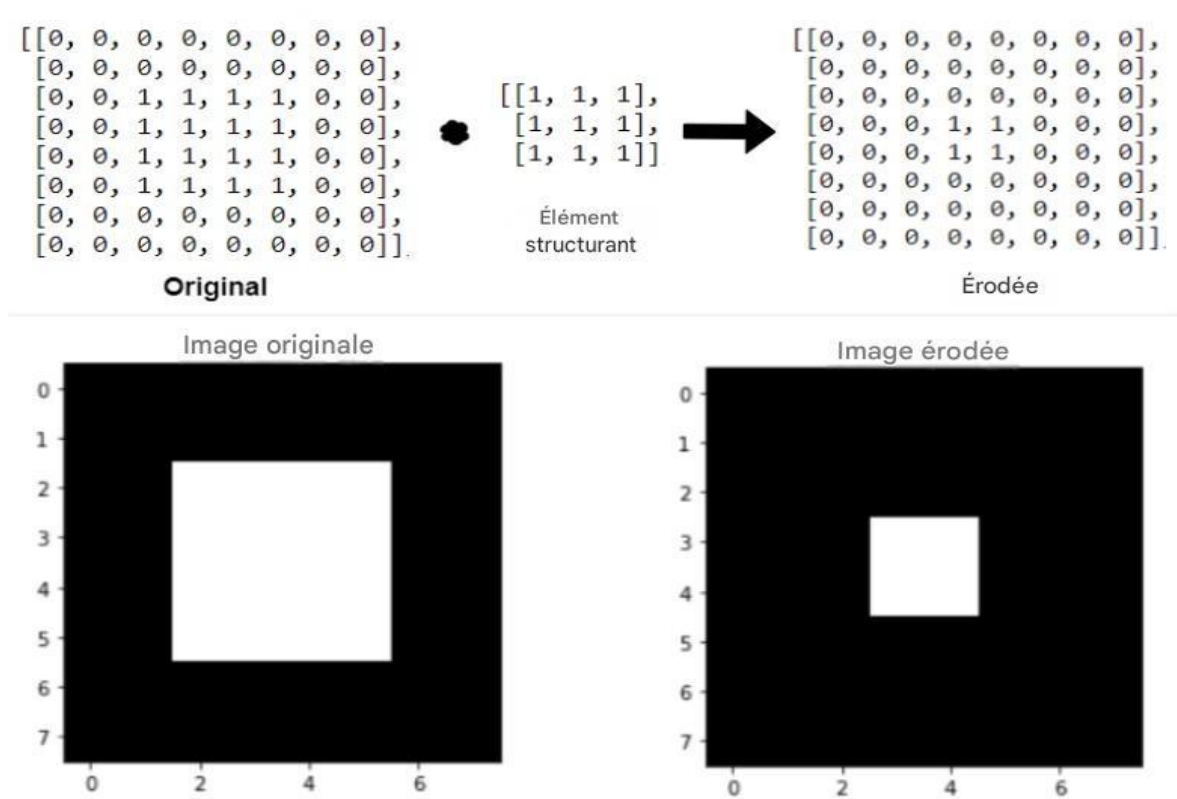


Figure 17: Illustration de l'opération d'érosion en morphologie mathématique.[38]

L'érosion, c'est comme si on rétrécissait un objet dans une image en mangeant ses bords. À chaque fois, on enlève une petite partie du contour, donc l'objet devient plus petit.

Ensuite, les détails fins disparaissent, et l'image devient plus simple.

En résumé : l'érosion enlève les bords des objets pour les rendre plus petits et plus simples.

3.3.2. Dilatation :

"Comme son nom l'indique, la dilatation est un processus d'agrandissement. C'est une technique qui consiste à étendre les contours d'un objet dans une image. Dans ce processus, l'élément structurant détermine la taille et la forme de la dilatation. Il ajoute un nombre de

pixels aux contours de l'objet dans l'image et génère une nouvelle image où les pixels de l'image originale sont agrandis." [39]

La dilatation est en fait le contraire de l'érosion. Elle permet d'agrandir les objets dans une image. Pour ce faire, elle ajoute des pixels tout autour des bords. De ce fait, l'objet devient plus grand et plus épais.

Lors d'une dilatation, les contours des objets s'étendent. De plus, les petits trous ou espaces peuvent même disparaître.

En résumé, la dilatation est une technique qui ajoute des pixels aux bords des objets pour les agrandir et les renforcer.

3.3.3. Utilisation de l'érosion et de la dilatation :

"La dilatation et l'érosion sont souvent utilisées conjointement pour réaliser des opérations de traitement d'images. Par exemple, l'ouverture morphologique d'une image consiste en une érosion suivie d'une dilatation, les deux opérations utilisant le même élément structurant. On peut combiner dilatation et érosion pour supprimer les petits objets d'une image et adoucir les contours des grands objets." [40]

La dilatation et l'érosion sont souvent utilisées ensemble pour améliorer une image.

Cette combinaison permet de supprimer les petits objets ou le bruit, tout en maintenant les objets importants plus propres et plus lisses.

En résumé, on combine l'érosion et la dilatation pour nettoyer l'image et améliorer les formes des objets.

3.4. Segmentation :

La segmentation est une étape très importante dans le traitement d'image. Elle consiste à diviser une image en plusieurs parties pour mieux reconnaître et étudier les objets qu'elle contient.

Cela permet de faire la distinction entre les objets et le fond, et d'identifier les différentes zones de l'image.

3.4.1. Définition :

D'après le site (IBM.com) La segmentation d'image est une technique qui aide les ordinateurs à diviser une image en plusieurs parties plus petites. Cela permet de repérer les objets de manière plus précise et de les analyser plus facilement.

Lorsqu'une image est complexe, la segmentation l'aide à la diviser en parties plus simples. Cela rend le traitement de l'image plus rapide et plus efficace. Les techniques de segmentation d'image peuvent être simples, en se basant sur des caractéristiques comme la couleur ou la luminosité de l'image. Mais elles peuvent aussi être plus avancées, en utilisant le machine learning pour reconnaître automatiquement les objets à partir de données d'apprentissage.

La segmentation d'image est très utile dans plusieurs domaines, comme la médecine, la robotique ou encore l'analyse d'images satellites. Dans ces domaines, la segmentation d'image aide à identifier précisément les éléments importants de l'image. Cela permet de prendre des décisions plus éclairées et de réaliser des tâches de manière plus précise. La segmentation d'image joue donc un rôle clé dans de nombreux domaines qui utilisent les images pour prendre des décisions ou effectuer des analyses. [41]

L'image segmentation consiste à diviser une image en plusieurs segments plus petits. Cela permet d'en améliorer la compréhension.

En effet, au lieu d'analyser l'image dans son ensemble, on la découpe en zones plus simples. De ce fait, la reconnaissance des objets en devient plus facile.

Quand l'image est complexe, cette technique aide à mettre en avant les éléments importants. Elle permet ainsi d'accélérer leur analyse.

La segmentation peut se baser sur des critères simples comme la couleur. Elle peut aussi utiliser des méthodes avancées pour identifier les objets de façon autonome.

C'est ainsi que la segmentation s'avère très utile dans de nombreux domaines. Elle permet d'identifier avec précision les éléments d'une image. Elle aide également à prendre des décisions plus éclairées.

3.4.2. Type de segmentation :

Les types de segmentation sont :

A) Segmentation sémantique :

Selon le site (ultralytics.com) La segmentation sémantique consiste à attribuer une catégorie à chaque pixel d'une image. Cela signifie que tous les pixels d'une même catégorie sont regroupés, sans distinction entre les objets individuels. Par exemple, dans une image, toutes les zones de forêt sont considérées comme une seule région.

B) Segmentation d'instance :

La segmentation d'instance va plus loin. Elle permet de distinguer chaque objet individuellement, même si ces objets appartiennent à la même catégorie. Par exemple, dans une rue, chaque voiture ou chaque personne est identifiée séparément.

C) Segmentation panoptique :

La segmentation panoptique combine les avantages des deux méthodes précédentes. Elle classe tous les pixels d'une image tout en distinguant les objets individuels. Cela permet d'identifier à la fois les éléments de fond, comme le ciel ou la route, et les objets distincts, comme les voitures ou les personnes. [42]

La segmentation d'image peut être réalisée de différentes manières. Elle dépend du niveau de détail que l'on cherche à obtenir. Certaines méthodes servent à regrouper les pixels par catégories. D'autres méthodes servent à identifier chaque objet séparément. Il existe également des méthodes qui combinent ces deux approches. Cela permet d'obtenir une

analyse à la fois globale et précise de l'image. La segmentation d'image est donc très utile pour analyser les images de manière efficace.

3.4.3. Technique de segmentation :

Parmi les techniques de segmentation, on trouve :

A) Segmentation par régions :

"La segmentation par régions est une approche puissante et polyvalente pour segmenter les images en régions homogènes. Cette méthode détecte et regroupe automatiquement les pixels similaires en régions cohérentes, facilitant ainsi la compréhension et l'analyse des données visuelles dans de nombreux domaines d'application." [43]

B) segmentation par croissance des régions :

"Cette méthode consiste à sélectionner un ou plusieurs pixels de départ, appelés « graines », puis à étendre progressivement les régions en ajoutant des pixels voisins présentant des caractéristiques similaires. Le processus se poursuit jusqu'à ce que tous les pixels soient affectés à une région spécifique ou jusqu'à ce que des critères d'arrêt prédéfinis soient atteints. La croissance régionale est sensible aux conditions initiales et peut être influencée par le choix des graines et les critères de croissance." [44]

C) Segmentation par approche contours :

"Ce type s'intéresse aux contours des objets dans l'image. Elle consiste à détecter les points frontières entre 2 zones homogènes de caractéristiques différentes. Elle nous donne accès aux différentes zones de l'image que l'on pourra après coup (par exemple en remplissant ces régions) isoler pour les traiter. La plupart de ces algorithmes sont locaux. Le résultat est en général difficile à exploiter sauf pour des images très contrastées." [45]

D) Segmentation par classification ou seuillage :

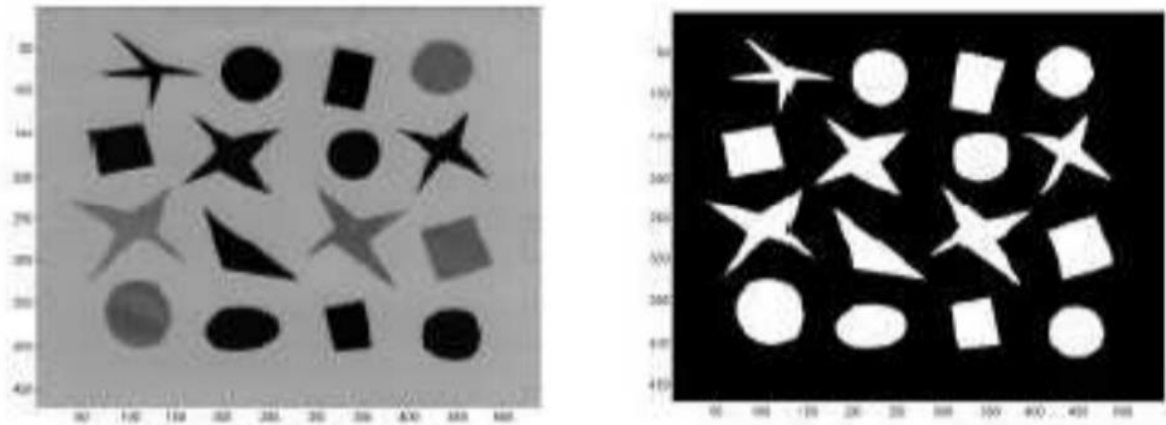
"Le seuillage est une méthode simple et très populaire pour la segmentation d'objets dans les images numériques. Il peut être de nature :

- **Globale** : un seuil pour toute l'image.
- **Locale** : un seuil pour une portion de l'image.
- **Adaptative** : un seuil qui s'ajuste selon les images/parties de l'image.

On part ici d'un rapport qu'entretient chaque pixel individuellement avec des informations calculées sur toute l'image, comme par exemple la moyenne des niveaux de gris de l'ensemble des pixels, ou la médiane, permettant de construire n classes d'intensité. Lorsque les classes sont déterminées par le choix d'un seuil, on parle de seuillage.

Les pixels appartenant à une même classe et étant connexes forment des régions.

Le seuillage constitue une approche simple à la segmentation car plusieurs zones déconnectées ont la même valeur d'étiquette. Le seuillage sert de composante de base à des algorithmes plus complexes de segmentation.



1. Image originale

2. Image seuillée

Figure 18: Segmentation par seuillage." [46]

La segmentation d'image repose sur plusieurs techniques qui permettent de diviser une image en régions ou objets selon différents critères, afin de faciliter son analyse et son interprétation.

4. CONCLUSION :

Le traitement d'images désigne l'ensemble des techniques utilisées pour analyser et améliorer des images numériques afin d'en extraire des informations utiles. Il permet de transformer une image brute en une représentation plus claire, plus précise et plus facile à exploiter.

D'autres techniques, comme la binarisation et la morphologie mathématique, permettent de simplifier l'image et d'améliorer la structure des objets en corrigeant leurs formes et en supprimant les éléments indésirables. Cela rend les images plus claires et plus adaptées à une analyse approfondie.

L'analyse est ensuite renforcée par la segmentation, qui permet de découper l'image en parties significatives afin d'identifier précisément les objets et de mieux comprendre son contenu. Grâce à l'ensemble de ces méthodes, il devient possible de traiter efficacement des images complexes et d'en extraire des informations pertinentes.

Ainsi, le traitement d'images constitue un outil essentiel dans de nombreux domaines, car il permet d'améliorer la qualité des données visuelles et de faciliter la prise de décision à partir des images.

Chapitre III

Détection et suivi d'objet

1. INTRODUCTION :

Le suivi et la détection d'objets sont des domaines clés de la vision par ordinateur. Ils permettent d'analyser et de comprendre automatiquement ce que contiennent les images et les vidéos.

La détection d'objets consiste à identifier et à localiser les objets dans une image à un moment donné. Le suivi d'objets, quant à lui, consiste à observer comment ces objets bougent et changent au fil du temps dans une vidéo.

Ce chapitre commence par présenter les bases du suivi d'objets. Il explique ce que c'est, à quoi ça sert et quels sont les défis à relever pour que ça fonctionne bien dans la vie réelle. Il parle aussi des différentes méthodes de suivi et des caractéristiques visuelles qui aident à suivre les objets de manière fiable.

Ensuite, le chapitre aborde la détection d'objets. Il présente les principes de base et les principales approches utilisées. Il existe des méthodes en une seule étape et des méthodes basées sur les régions. Le chapitre parle également des modèles les plus connus, comme YOLO et les différentes versions de R-CNN. Cela permet de comprendre les avancées récentes dans ce domaine.

2. DETECTION D'OBJET :

La détection d'objets est une étape très importante en vision par ordinateur. Elle consiste à identifier et localiser des objets présents dans une image ou une vidéo de manière automatique. Cela nous permet de distinguer les différents éléments d'une scène. La détection d'objets est souvent la première étape avant de passer à des tâches plus avancées comme le suivi d'objets ou la reconnaissance d'objets.

2.1. Définition :

D'après (Berramdane A) La détection d'objets est un processus qui consiste à identifier la position d'un objet sur une image spécifique. La nature de l'objet n'a pas d'importance dans ce processus. Par exemple, si vous créez un détecteur de chaise, il vous indiquera simplement où se trouve la chaise sur l'image. Il ne vous donnera pas de détails sur la chaise, comme sa couleur ou sa forme. La détection d'objets est une étape importante dans de nombreux systèmes de vision artificielle, car elle permet de localiser les objets dans une scène. Cela signifie que la détection d'objets joue un rôle clé dans diverses applications de vision artificielle. Considérons l'image suivante :

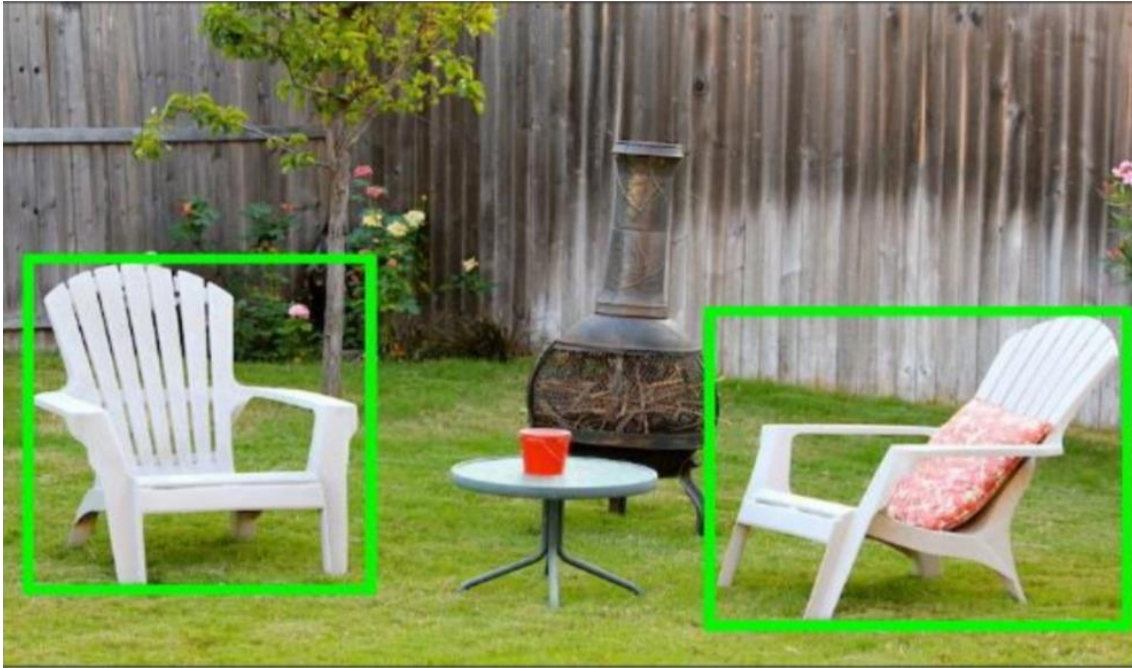


Figure 19: Détection des chaises. [47]

La détection d'objets est une première étape importante pour analyser une image. Elle permet de passer d'une vue d'ensemble à une représentation plus organisée. En repérant les objets et leurs positions, on obtient des informations de base très utiles. Ces informations servent de fondation pour des analyses plus poussées, qui visent à comprendre et à interpréter ce qui se passe dans la scène.

2.2. Approches-en une seule étape :

"Les Approches en une étape ou **"one-shot"**, comme leur nom l'indique, tentent de réaliser l'ensemble de l'analyse en un seul passage. Elles appliquent un unique réseau neuronal convolutif directement à l'image pour effectuer simultanément la détection et la classification des objets." [48]

Les méthodes dites « one-shot » permettent de détecter et de catégoriser les objets dans une image en une seule étape, car elles utilisent un réseau de neurones unique qui examine l'image directement sans avoir besoin de la manipuler plusieurs fois, ce qui rend la tâche à la fois plus rapide et plus facile.

2.2.1. You Only Look Once (YOLO) :

Selon le site (Llamaindex.ai) C'est une méthode de détection d'objets en temps réel. Elle peut repérer et situer plusieurs éléments dans des images lors d'un unique passage par un réseau neuronal. Les techniques de détection classiques ont besoin de plusieurs phases de traitement. Mais YOLO assure une performance en temps réel avec une stratégie intégrée. Elle préservé une précision supérieure. C'est pourquoi YOLO est essentiel pour des usages comme la conduite autonome et l'analyse documentaire. [49]

YOLO est souvent utilisé lorsque l'on a besoin d'analyser une image rapidement, par exemple dans une vidéo en direct. Il traite l'image une seule fois et détecte les objets en même temps, tout en indiquant où ils se trouvent.

Cela permet de rendre le processus beaucoup plus rapide, tout en conservant une bonne précision. YOLO est donc utilisé dans des situations où la vitesse est importante, comme les voitures autonomes ou les systèmes d'analyse automatique.

2.3. Approche en deux étapes (basée sur la région) :

L'approche en deux étapes basées sur la région est une méthode très utilisée en détection d'objets. Cette méthode repose sur une idée simple qui consiste à analyser l'image progressivement. Cela permet de mieux repérer les objets présents dans l'image et d'améliorer la précision des résultats de la détection d'objets.

2.3.1. Region Convolutional Neural Network (R-CNN) :

"Le R-CNN est une architecture de détection d'objet. Le R-CNN commence par extraire des régions intéressantes de l'image, puis il utilise ces régions comme données d'entrée pour un CNN. Cette répartition en régions permet de détecter plusieurs objets de plusieurs classes différentes dans une même image. Cette solution proposée par Girshick et al., 2013 a permis d'améliorer la précision des modèles de détection.

R-CNN : Régions avec des caractéristiques CNN

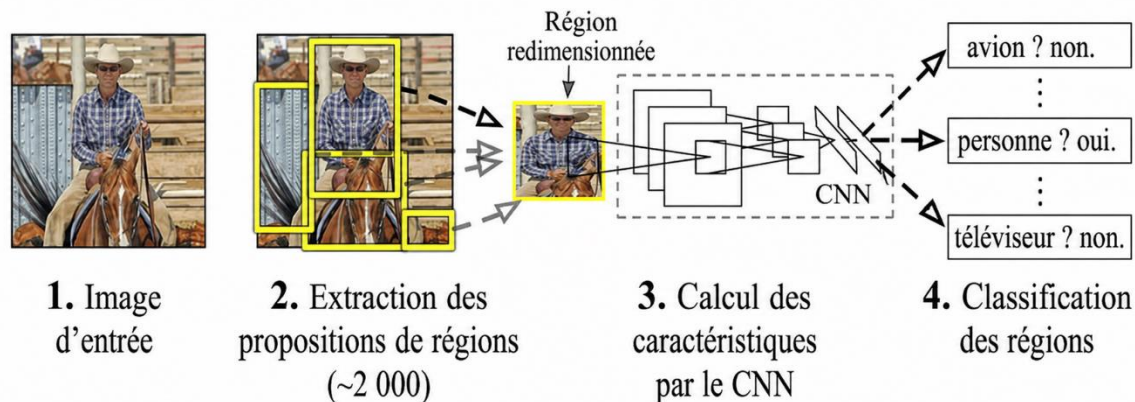


Figure 20: L'architecture de R-CNN.

Les régions sont extraites grâce à une recherche sélective proposée par Uijlings, van de Sande, Gevers et Smeulders, 2013. Celle-ci utilise la structure de l'image et plusieurs techniques de partitionnement pour récupérer toutes les régions intéressantes possibles." [50]

Le R-CNN fonctionne en découpant l'image en plusieurs zones qui pourraient contenir des objets. Il analyse ensuite chaque zone séparément avec un réseau de neurones pour reconnaître les objets et déterminer leurs classes.

Cette approche permet d'obtenir plus de précision. En effet, le modèle se concentre sur des parties spécifiques de l'image plutôt que sur l'image entière.

Les zones à analyser sont sélectionnées automatiquement en fonction de la structure de l'image. L'objectif est de repérer les régions les plus importantes.

2.3.2. Rapid R-CNN (Fast R-CNN) :

Selon le site (Mathworks.com) Le détecteur Fast R-CNN utilise un algorithme similaire à Edge Boxes pour suggérer des zones. Contrairement au détecteur R-CNN, qui modifie et ajuste ces suggestions, le détecteur Fast R-CNN examine l'image entière. Alors qu'un détecteur R-CNN évalue chaque région individuellement, le détecteur Fast R-CNN associe les caractéristiques du réseau de neurones convolutif à chaque proposition de région. Le détecteur Fast R-CNN est plus efficace que le détecteur R-CNN, car les opérations des régions qui se chevauchent sont partagées.

Le Fast R-CNN améliore le fonctionnement du R-CNN. Il traite l'image une seule fois au lieu d'analyser chaque région séparément. Le Fast R-CNN partage les caractéristiques

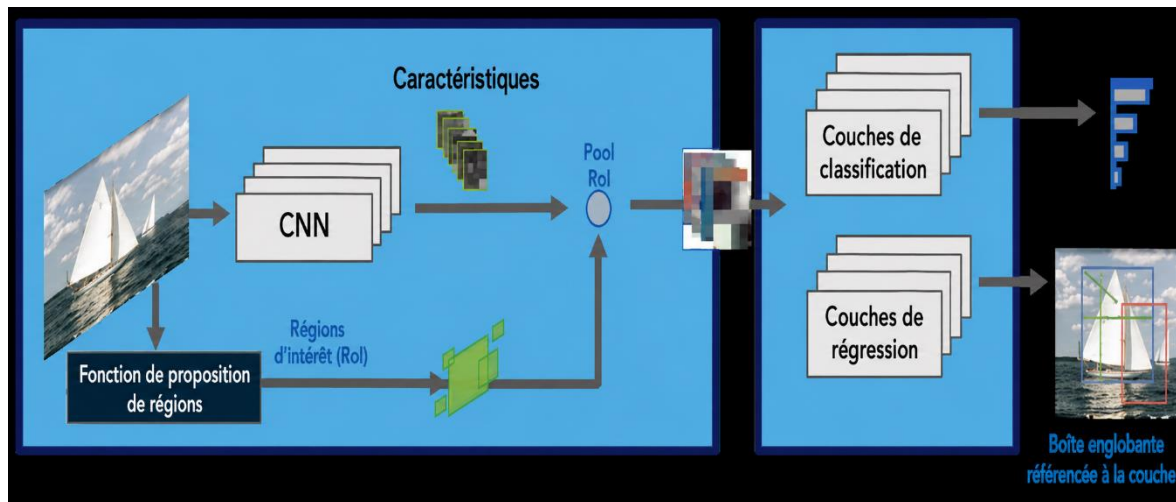


Figure 21: Architecture et comparaison des modèles de détection d'objet : Fast R-CNN et Faster R-CNN. [51]

extraites entre toutes les régions proposées. Cela permet de réduire le temps de calcul et d'augmenter l'efficacité du modèle Fast R-CNN. Le modèle Fast R-CNN conserve de bonnes performances de détection.

2.3.3. Faster R-CNN :

D'après (Chouini M E M 2020) Le Faster R-CNN a été élaboré par Shaoqing Ren et ses collaborateurs en 2016. Il s'agit de la version améliorée du Fast RCNN. Le Faster R-CNN est utilisé pour la détection d'objets en trois dimensions et la segmentation, entre autres. Le Faster R-CNN est constitué de trois réseaux neuronaux. Il y a le Réseau de Fonctionnalités, également appelé Feature Network. Il y a également le Réseau de Proposition Régionale, également appelé Regional Proposal Network. Et enfin, il y a le Réseau de Détection, également appelé Detection Network. Ces trois réseaux neuronaux

travaillent ensemble pour améliorer la détection d'objets. Le Faster R-CNN est très utile pour la détection d'objets en trois dimensions et la segmentation.

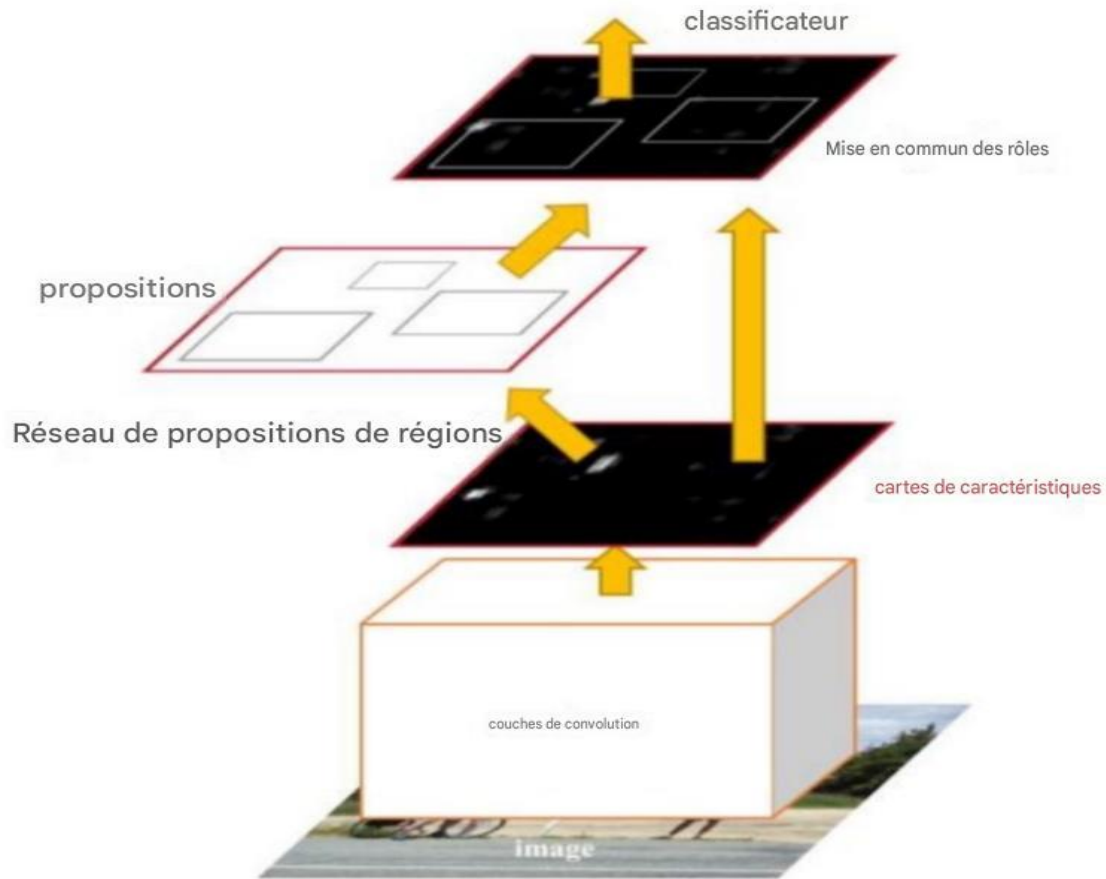


Figure 22: Architecture Faster R-CNN.[52]

Le Faster R-CNN est un système qui fonctionne en plusieurs étapes pour détecter efficacement les objets. Tout d'abord, il analyse l'image pour en extraire les caractéristiques clés. Ensuite, il identifie les régions qui pourraient contenir des objets. Enfin, il classe ces régions et affine leur position. Grâce à cette approche organisée, le Faster R-CNN permet d'améliorer à la fois la vitesse et la précision de la détection d'objets.

2.3.4. Mask R-CNN :

Selon (A Akarid, S El Adib, N Raissouni) Mask R-CNN est une version améliorée de Faster R-CNN. Mask R-CNN ajoute une étape pour créer des masques d'objets. Mask R-CNN est composé de trois parties principales : un réseau pour extraire les caractéristiques, un réseau appelé RPN pour suggérer des régions possibles, et un module final pour détecter les objets et les segmenter. [53]

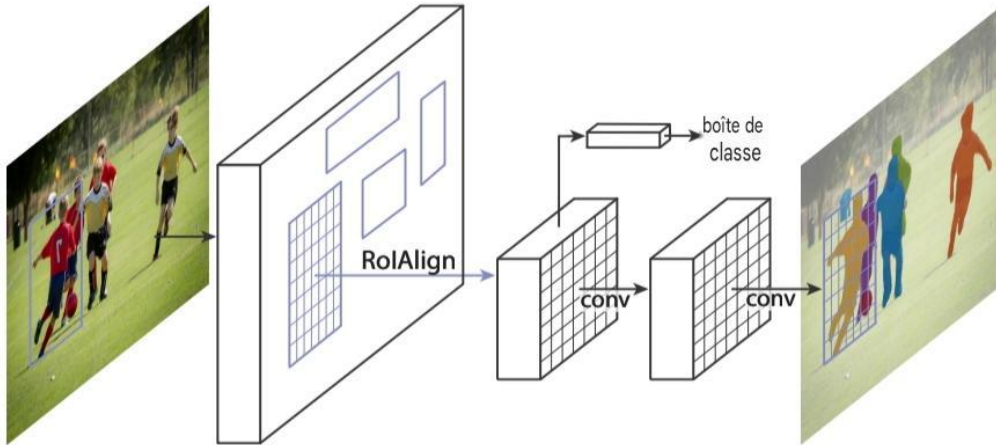


Figure 23: Le Framework Mask R-CNN pour la segmentation d'instances.[54]

Mask R-CNN va au-delà de la simple détection de la position des objets dans une image. Il est également capable de déterminer avec précision la forme de ces objets. Mask R-CNN ne se limite pas à tracer un rectangle autour de l'objet comme le fait Faster R-CNN. Il identifie les pixels exacts qui appartiennent à cet objet. Cela permet une analyse plus détaillée et plus précise de la scène. Mask R-CNN offre ainsi une compréhension plus approfondie de l'image.

3. SUIVI D'OBJET :

Le suivi d'objet est très utile dans le traitement d'images et la vision par ordinateur. Il s'agit de détecter un objet et de suivre son déplacement dans une série d'images ou une vidéo. Cela nous aide à comprendre comment l'objet se déplace, à prédire où il va aller et à obtenir des informations sur son comportement. Le suivi d'objet est utilisé dans de nombreux domaines, notamment la surveillance, la reconnaissance d'activités, la conduite autonome et les systèmes de sécurité. Il nous permet de mieux comprendre ce que fait l'objet observé et de prendre des décisions éclairées. Le suivi d'objet est une technique clé qui nous aide à analyser et à comprendre le mouvement dans les images et les vidéos.

3.1. Définition :

"Le suivi d'objet peut être défini comme le problème de l'estimation de la trajectoire d'un objet dans le plan image lorsqu'il se déplace dans une scène." [55]

Cela signifie concrètement que le système analyse une succession d'images (vidéo) pour déterminer la position de l'objet à chaque instant, puis relie ces positions entre elles afin de reconstituer son mouvement de manière continue.

3.2. Domain d'application :

"Le suivi d'objets dans des séquences vidéo a suscité un grand intérêt dans la dernière décennie en raison de la variété de ses domaines d'applications, tels que :

- La vidéosurveillance (détection, suivi, reconnaissance du comportement de personnes, d'intrus)
- Le militaire (suivi des cibles ou guidage de missiles)
- La vidéoconférence (suivi des interlocuteurs)
- La gestion et l'analyse du trafic (le suivi d'une voiture ou des bords d'une route depuis une caméra embarquée sur un véhicule)
- La robotique : suivie d'obstacles pendant une phase d'évitement
- Suivi d'indices visuels dans une tâche asservie par vision
- Suivi d'un opérateur (corps, visage, main...) pour définir des modes d'interaction évolués entre l'Homme et la Machine...etc.
- Imagerie médicale



Figure 24: Quelques applications de suivi d'objets." [56]

Le suivi d'objets est crucial. Cela permet d'analyser automatiquement les mouvements dans une vidéo. C'est utile dans de nombreux domaines, comme la sécurité, la robotique, la médecine et la gestion du trafic.

3.3. Les défis du suivi :

Les défis du suivi d'objet sont les difficultés rencontrées pour suivre correctement un objet dans une vidéo, car son mouvement et son apparence peuvent changer au cours du temps.

3.3.1. Occultations :

"Occlusions : il s'agit de zones qui ne sont visibles que dans une image et qui sont donc occultées dans l'autre. Cela pose le problème de la détermination des correspondants des pixels occultés.



Figure 25: Détection et suivi d'objet : Les occultations (occlusions)." [57]

Cela signifie que lorsque l'on suit un objet, certaines parties peuvent temporairement disparaître de l'image. Cela se produit lorsqu'elles sont cachées par un autre objet ou lorsqu'elles sortent du champ de la caméra. Dans ce cas, le système a du mal à retrouver ces parties lorsqu'elles réapparaissent. Cela complique le suivi correct de l'objet.

3.3.2. Changement d'illumination :

D'après (Mme M. Chibani) Les images capturées par une caméra sont influencées tant par les éléments physiques de la scène que par les conditions d'éclairage.

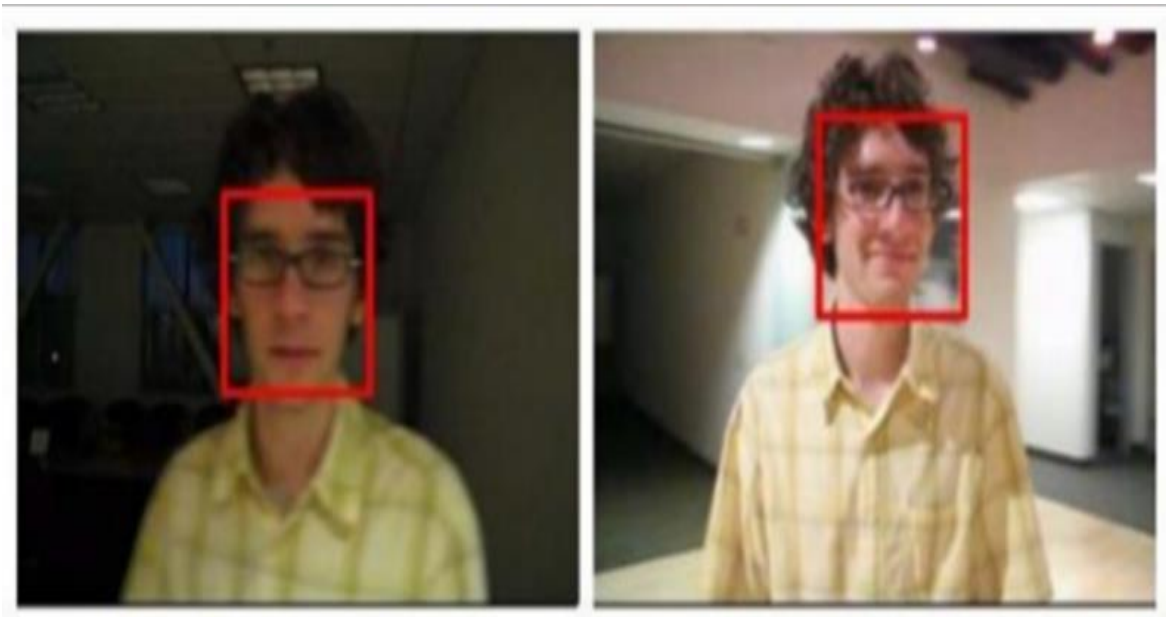


Figure 26: Changement d'illumination. [58]

Autrement dit, l'image obtenue par une caméra ne dépend pas uniquement des objets présents dans la scène. Elle dépend aussi de la manière dont la lumière interagit avec eux. La qualité et l'apparence de l'image peuvent ainsi varier selon l'éclairage. Pourtant, les éléments de la scène restent les mêmes.

3.3.3. Changement d'échelle :

"En infographie et en imagerie numérique, la mise à l'échelle d'une image consiste à redimensionner une image numérique. En vidéo, l'agrandissement d'un contenu numérique est appelé suréchantillonnage ou amélioration de la résolution. " [59]

Cela signifie que la mise à l'échelle permet de modifier la taille d'une image selon le besoin. On peut l'agrandir ou la réduire, tout en essayant de conserver sa qualité visuelle. La qualité visuelle de l'image est donc conservée, que l'image soit agrandie ou réduite.

3.3.4. Changement du modèle d'apparence :

Selon (Feddag I) L'apparence d'un objet en mouvement est essentielle pour le différencier des autres objets en mouvement d'un côté et de son arrière-plan de l'autre. Par exemple, la teinte de l'objet est un détail très utile lorsqu'on utilise l'histogramme de l'image. Toutefois, l'approche des contours rend la bordure de l'objet plus captivante. Il est important de mentionner que la forme et l'apparence peuvent être associées pour identifier l'objet, dans le cadre du suivi d'objets.

Les techniques basées sur des modèles d'apparence visent à identifier les emplacements successifs des objets en traquant diverses mesures de similitude basées sur des aspects tels que la couleur, la forme et/ou la texture. [60]

Pour suivre un objet, on ne se base pas uniquement sur l'endroit où il se trouve. Ce qui le rend unique visuellement est tout aussi important. Sa couleur, par exemple, peut aider à le

distinguer des autres choses. Les contours, eux, donnent une idée de sa forme. En combinant ces détails (couleur, forme, texture), le système peut vérifier d'une image à l'autre si c'est toujours le même objet.

Les modèles d'apparence fonctionnent en comparant l'objet qu'on voit dans l'image actuelle avec ce qu'on a vu dans les images précédentes. Cela permet de savoir où il se trouve au fil du temps avec plus de précision.

3.3.5. Mouvements de caméra :

"Les plus petits mouvements de caméra sont appelés recadrages. Parmi les mouvements de caméra plus développés, on distingue le panoramique et le travelling. Lors d'un panoramique, la caméra pivote sur son pied, lequel reste fixe. Lors d'un travelling, la caméra se déplace au sol sur un pied lui-même monté sur un rail, et/ou dans les airs à l'aide d'un bras articulé. Le zoom permet de simuler un rapprochement ou un éloignement de caméra sans déplacer physiquement la caméra en faisant seulement varier la focale." [61]

Ces mouvements de caméra nous permettent de modifier la façon dont nous filmons une scène. Nous pouvons orienter la caméra ou la déplacer pour suivre l'action plus facilement ou pour mettre en valeur un élément particulier.

3.3.6. Objet en mouvement rapide :

"Un objet en mouvement est une région fermée de l'image ayant une couleur et un mouvement quasi constants." [62]

Cela signifie qu'un objet en mouvement dans une vidéo est une partie précise de l'image. Cette partie a des caractéristiques qui restent similaires, comme sa couleur et la façon dont il bouge. Cela permet de le reconnaître facilement et de le suivre tout au long de la vidéo.

3.4. Méthodes de suivi :

Les méthodes de suivi d'objet sont utilisées pour trouver et suivre un objet dans une vidéo. Elles analysent les caractéristiques de l'objet et son mouvement pour le suivre avec précision tout au long de la vidéo. Ces méthodes de suivi d'objet reposent sur l'analyse des caractéristiques de l'objet pour assurer un suivi continu. Les méthodes de suivi d'objet permettent donc de suivre un objet dans une séquence vidéo en analysant ses caractéristiques et son mouvement.

3.4.1. Suivi par point :

"Les objets détectés dans des images consécutives sont représentés par des points (figure 27). La correspondance entre les points est faite par la connaissance de leurs états dans les images précédentes de la scène (que ce soit sa position ou son mouvement). Ces approches nécessitent un mécanisme de détection dans chaque image de la séquence.

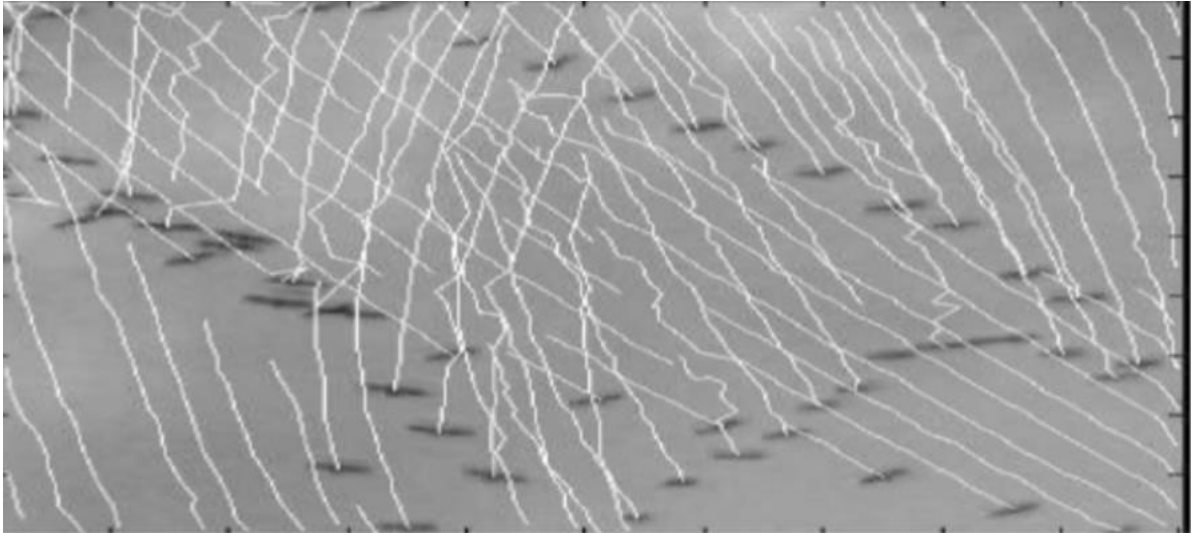


Figure 27: Suivi des oiseaux par des point." [63]

Cela signifie que le suivi d'objet repose sur l'identification de l'objet dans chaque image sous forme de points. Ensuite, le système compare ces points d'une image à une autre. Le système utilise les informations précédentes comme la position ou le mouvement de l'objet. Il détermine quel point correspond au même objet dans les images suivantes. Cela permet de suivre le déplacement de l'objet de manière continue. Le système utilise ces informations pour suivre le déplacement de l'objet dans les images.

3.4.2. Suivi de silhouettes :

D'après (Mme M. Chibani) Lorsqu'un objet présente une structure très complexe, comme une main, un corps humain, un animal ou un arbre, l'utilisation d'une forme simple comme un rectangle ou une ellipse pour le représenter n'est pas appropriée.

L'utilisation d'une silhouette pour représenter un objet permet de considérer avec précision sa forme. Le but des techniques de suivi basées sur la silhouette est d'évaluer la silhouette des objets pertinents pour chaque image de la vidéo.

Il existe deux approches pour le suivi des silhouettes : la correspondance de formes et le traçage de contours. [64]

Ce passage montre que les modèles géométriques simples ne sont pas suffisants pour représenter des objets complexes. Pour obtenir une représentation plus précise de leur forme, il est préférable d'utiliser la silhouette. Cette approche permet de décrire avec précision les contours de l'objet. Le suivi de l'objet repose alors sur l'extraction et la mise à jour de sa silhouette dans chaque image d'une séquence vidéo. Pour cela, on utilise des méthodes spécifiques comme la correspondance de formes ou le traçage de contours.

3.4.3. Suivi de noyaux :

Selon (A Yilmaz, O Javed, M Shah) Le noyau fait référence à la structure et à l'apparence de l'objet. Par exemple, cela peut être un modèle rectangulaire ou une forme elliptique liée à un histogramme. L'identification des objets se fait en analysant le déplacement du noyau à travers diverses images successives. En général, ce mouvement se

manifeste sous la forme d'une transformation paramétrique, comme une translation, une rotation ou une transformation affine. [65]

Le suivi du noyau consiste à représenter un objet par une forme simple qui capture les caractéristiques principales de l'objet, puis à suivre le déplacement de cette forme d'une image à une autre. Ainsi, le système se concentre sur ce modèle simplifié pour estimer le mouvement de l'objet dans la séquence vidéo, au lieu d'analyser tous les détails de l'objet.

3.5. Caractéristiques visuelles pour le suivi d'objets :

Les caractéristiques visuelles sont des détails importants que l'on peut extraire d'une image ou d'une vidéo. En vision par ordinateur, ces détails aident à identifier et à représenter des objets. Ils montrent ce que l'on peut voir sur un objet, comme sa couleur, sa texture, sa forme ou des points spécifiques. Cela permet de détecter l'objet et de le suivre dans le temps.

Quand on parle de suivi d'objets, ces caractéristiques visuelles sont très utiles. Elles aident à reconnaître un objet d'une image à l'autre. Même si l'environnement de l'objet change ou si des choses bougent autour de lui, on peut encore le distinguer. C'est très pratique pour comprendre ce qui se passe dans une scène filmée ou photographiée.

3.5.1. Couleur :

Selon le site (fr.scribd.com) Les couleurs sont utilisées très souvent pour représenter des objets et elles donnent une description naturelle de l'apparence d'un objet. C'est parce que les couleurs sont le moyen le plus évident pour que l'œil humain perçoive les choses. Généralement, les couleurs sont représentées dans l'espace colorimétrique RGB. Cependant, les informations basées uniquement sur la couleur ne sont pas suffisantes, car elles ne sont pas robustes face aux changements de luminosité. Les couleurs peuvent varier considérablement selon la luminosité, ce qui signifie que les informations basées uniquement sur la couleur peuvent ne pas être fiables dans toutes les situations. Les couleurs sont donc un attribut important, mais elles doivent être utilisées en combinaison avec d'autres attributs pour obtenir une représentation complète et robuste des objets. [66]

Les couleurs sont très utiles pour décrire la façon dont les objets apparaissent, parce que les yeux humains peuvent facilement les voir. Mais parfois, les couleurs ne sont pas suffisantes pour donner une idée exacte de l'objet, car elles peuvent varier en fonction de la quantité de lumière qui les entoure. C'est pour cela qu'il est utile de combiner les couleurs avec d'autres caractéristiques de l'objet, comme la forme ou la taille, pour obtenir une description plus exacte et plus fiable.

3.5.2. Texture :

D'après le site (fr.scribd.com) La texture est un élément clé pour représenter les objets de manière réaliste. Elle décrit les variations de luminosité sur une surface, ce qui aide à caractériser la régularité ou la rugosité d'un objet. Cependant, intégrer la texture dans les applications qui fonctionnent en temps réel peut être compliqué. Par ailleurs, elle est moins affectée par les changements d'éclairage que la couleur, ce qui la rend utile dans différentes conditions de lumière. [67]

La texture nous aide à mieux comprendre l'apparence d'un objet. Elle nous donne des informations sur la surface de l'objet, ce qui nous permet de l'identifier et de l'analyser plus facilement. La texture est donc très importante en vision par ordinateur, car elle ne change pas beaucoup même si la lumière autour de l'objet change. Cependant, utiliser la texture en temps réel peut être difficile.

3.5.3. Gradient :

Selon (Medouakh S) Dans le domaine de la vision par ordinateur, les caractéristiques de gradient sont très utilisées, en particulier pour détecter et suivre des objets comme les personnes. Elles fonctionnent en analysant les changements d'intensité dans une image pour en dégager les formes et les contours. C'est ce qui les rend moins sensibles aux changements de luminosité que les caractéristiques de couleur.

Il existe principalement deux méthodes. La première consiste à utiliser les contours et les formes pour définir les objets, ce qui permet de les suivre plus facilement avec des techniques comme les courbes actives et les filtres par particules. La seconde méthode se base sur les statistiques de gradients pour créer des descripteurs solides. Des outils comme SIFT, pour identifier les objets, et HOG, pour détecter les piétons, sont couramment utilisés. Ces techniques sont souvent employées dans l'apprentissage automatique et la détection d'objets. [68]

Les caractéristiques de gradient sont utiles pour comprendre comment l'intensité de la lumière change dans une image. Cela aide à identifier les contours et les formes des objets. En effet, ces caractéristiques ne dépendent pas uniquement de la couleur.

Elles sont très utiles, car elles aident à reconnaître les objets même quand l'éclairage change. C'est pour cela qu'on les utilise souvent pour améliorer la détection ou le suivi des objets.

4. CONCLUSION :

En fin de compte, la détection et le suivi d'objets sont très importants en vision par ordinateur. Ils nous permettent de reconnaître les objets dans une scène et de voir comment ils changent avec le temps. Ces deux processus sont liés et nécessaires pour de nombreuses applications dans la vie réelle.

Il y a eu beaucoup de progrès dans ce domaine, mais il reste encore des défis. Par exemple, les objets peuvent changer d'apparence, de taille ou se déplacer dans des scènes qui évoluent constamment. Pour résoudre ces problèmes, des méthodes de suivi et des caractéristiques visuelles différentes ont été créées pour améliorer la précision et la fiabilité des systèmes.

De plus, les nouvelles approches basées sur l'apprentissage profond, comme YOLO et les modèles de la famille R-CNN, ont nettement amélioré la détection d'objets. En combinant des techniques efficaces de détection et de suivi, nous pouvons créer des systèmes intelligents de plus en plus performants et capables de fonctionner dans des environnements complexes.

Chapitre IV

Implémentation et discussion des résultats

1. INTRODUCTION :

Dans les chapitres précédents, nous avons présenté les bases du traitement d'images numériques et diverses techniques de traitement. Maintenant, nous allons voir comment appliquer ces connaissances de manière pratique, ce qui est une étape très importante. Elle nous permet de mettre en pratique les concepts et algorithmes que nous avons étudiés et d'évaluer leur efficacité sur des images et vidéos réelles.

Le traitement d'images consiste à améliorer la qualité d'une image, à réduire le bruit et les imperfections, et à faciliter l'extraction d'informations importantes. On utilise des techniques comme le prétraitement, le filtrage, les opérations morphologiques, la segmentation et la détection de contours pour analyser les images et interpréter leur contenu.

Dans ce chapitre, nous allons appliquer différentes techniques et algorithmes développés en Python avec la bibliothèque OpenCV à des images et vidéos capturées par des webcams et des caméras IP. Nous allons également voir comment chaque étape de traitement affecte la qualité des résultats et comment elle contribue à améliorer l'extraction d'informations visuelles.

Cette partie pratique vise à montrer l'importance des techniques de traitement d'images et de vision par ordinateur à travers des expériences concrètes. Nous allons faire de la détection de contours, de la segmentation couleur, des opérations morphologiques, de la suppression de l'arrière-plan, ainsi que de la détection et du suivi d'objets en temps réel à l'aide de l'algorithme YOLO.

À la fin de ce chapitre, nous allons analyser et discuter les résultats de ces expériences, en évaluant les performances des méthodes utilisées, en identifiant leurs avantages et leurs limites, et en reliant les aspects théoriques étudiés à leurs applications pratiques dans le domaine du traitement d'images et de la vision par ordinateur.

2. PRESENTATION DE LA BOITE À OUTILS :

Pour développer cette application de traitement d'images et de détection d'objets, plusieurs outils logiciels et matériels ont été utilisés.

2.1. Langage de programmation Python :

Python a été choisi comme langage principal grâce à sa simplicité, sa flexibilité et la disponibilité de nombreuses bibliothèques dédiées à la vision par ordinateur et à l'intelligence artificielle.

2.2. Bibliothèques utilisées :

1. OpenCV : Bibliothèque de vision par ordinateur permettant l'acquisition, le traitement et l'analyse des images et des vidéos en temps réel. Elle représente l'outil central de ce travail grâce à son rôle essentiel dans l'implémentation des différentes fonctionnalités de traitement d'images.

2. **NumPy** : Bibliothèque spécialisée dans le calcul numérique et la manipulation efficace des tableaux multidimensionnels.
3. **SciPy** : Ensemble d'outils scientifiques offrant des fonctions avancées pour l'optimisation, le traitement du signal et les calculs mathématiques.
4. **Pandas** : Bibliothèque dédiée à la manipulation, l'analyse et l'organisation des données sous forme de tableaux structurés.
5. **Pillow (PIL)** : Bibliothèque utilisée pour l'ouverture, la modification et la sauvegarde des images dans différents formats.
6. **Imutils** : Ensemble de fonctions simplifiant les opérations courantes de traitement d'images avec OpenCV.
7. **PyTorch (Torch, TorchVision)** : Framework d'apprentissage profond permettant la création, l'entraînement et le déploiement de modèles d'intelligence artificielle.
8. **Requests** : Bibliothèque facilitant l'envoi de requêtes HTTP et la communication avec les services web.
9. **Python-dotenv** : Outil permettant de charger et gérer les variables d'environnement à partir de fichiers de configuration.
10. **CUDA** : Technologie NVIDIA permettant l'accélération des calculs parallèles sur processeur graphique (GPU).
11. **YOLOv8** : Modèle de détection d'objets en temps réel basé sur l'intelligence artificielle, reconnu pour sa rapidité et sa précision.
12. **Dush** : Outil utilisé pour l'échange et le transfert de données entre différentes applications ou services du système.
13. **Ultralytics** : Framework fournissant les outils nécessaires à l'entraînement, l'évaluation et l'utilisation des modèles YOLO.

2.3. Sources d'acquisition des données :

L'application permet de traiter différents types de sources :

- **Images statiques** : chargement et traitement des fichiers image (JPG, PNG, BMP, etc.).
- **Vidéos enregistrées** : analyse des séquences vidéo stockées sur l'ordinateur.
- **Webcam** : acquisition vidéo en temps réel à partir de la caméra intégrée ou externe.
- **IP Camera** : réception et traitement d'un flux vidéo provenant d'une caméra réseau via une adresse URL.

2.4. Environnement matériel :

- Ordinateur équipé d'un processeur multi-cœurs.
- Carte graphique NVIDIA compatible CUDA.
- Webcam.
- Smartphone utilisé comme caméra IP (si applicable)

2.5. Environnement de développement :

- **PyScripter** : est un environnement de développement intégré (IDE) utilisé pour l'écriture, l'exécution et le débogage du code Python. Il offre une interface conviviale permettant de développer, tester et exécuter les programmes de manière efficace, ce qui facilite la mise en œuvre des différentes fonctionnalités de traitement d'images et de vision par ordinateur développées dans ce projet.

3. L'APPLICATION IP WEBCAM :

Dans ce projet, une caméra IP a été utilisée comme source d'acquisition vidéo en temps réel. Cette technologie permet de diffuser le flux vidéo à travers le réseau local et de l'envoyer directement à l'application de traitement développée en Python.

Pour cela, l'application IP Webcam a été installée sur un smartphone afin de le transformer en caméra IP. Cette application génère un flux vidéo accessible depuis l'ordinateur via une adresse URL ou le protocole RTSP, ce qui permet le traitement des images et des vidéos en temps réel à l'aide de la bibliothèque OpenCV.

Création de l'application sous Python :

```
import cv2
import os

# Optimisation FFMPEG pour l'acquisition du flux réseau de l'IP Webcam
os.environ["OPENCV_FFMPEG_CAPTURE_OPTIONS"] = "rtsp_transport;udp"

# URL réseau générée par l'application IP Webcam
url_stream = "rtsp://10.18.154.218:8080/h264.sdp"

# Initialisation de la capture vidéo réseau distant
cap = cv2.VideoCapture(url_stream, cv2.CAP_FFMPEG)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Affichage du flux vidéo de l'IP Webcam à l'écran
    cv2.imshow("Flux IP Webcam - Temps Reel", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

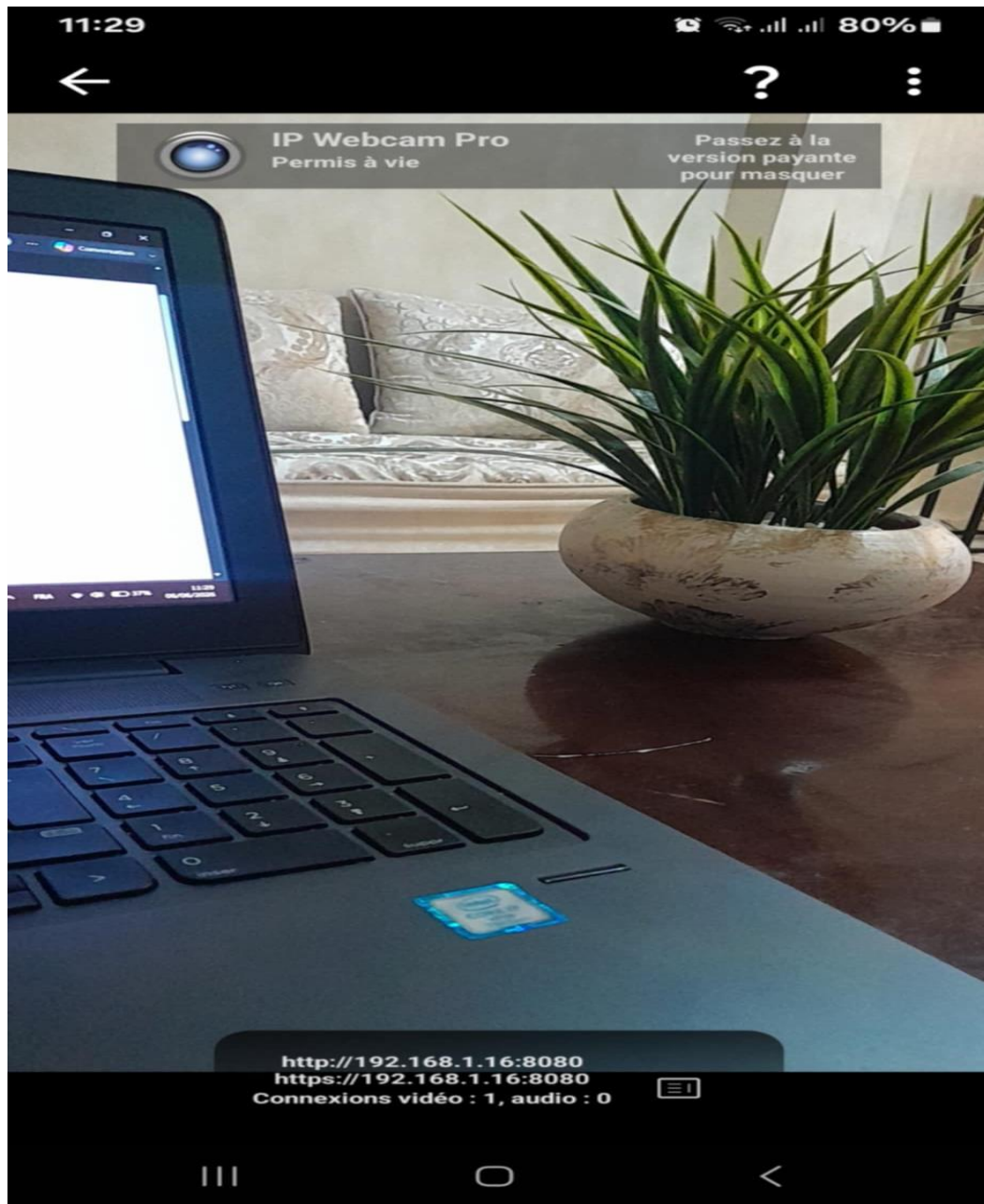
Résultat :

Figure 28: Smartphone exécutant l'application IP Webcam.

Remarque :

Le smartphone et l'ordinateur doivent être connectés au même réseau local (Wi-Fi) afin d'assurer la réception correcte du flux vidéo par l'application.

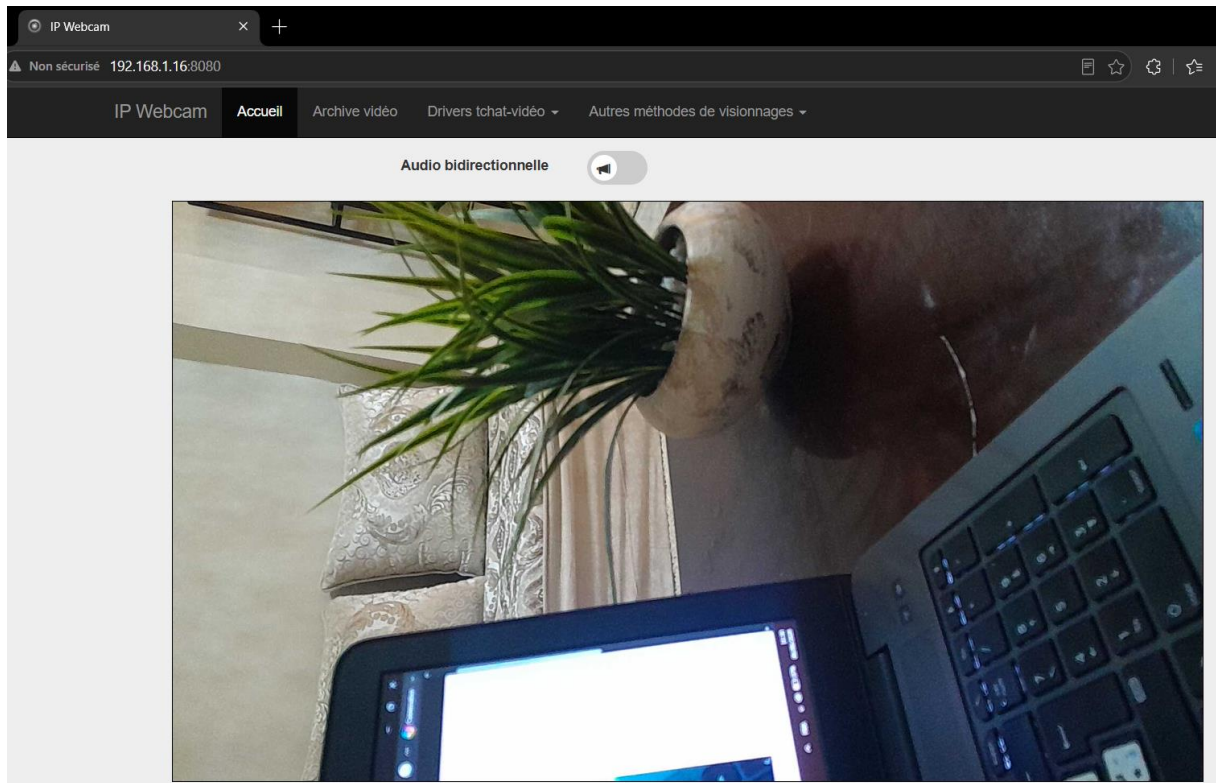


Figure 29: Interface de l'application affichant l'adresse IP et les paramètres de connexion.

Remarque :

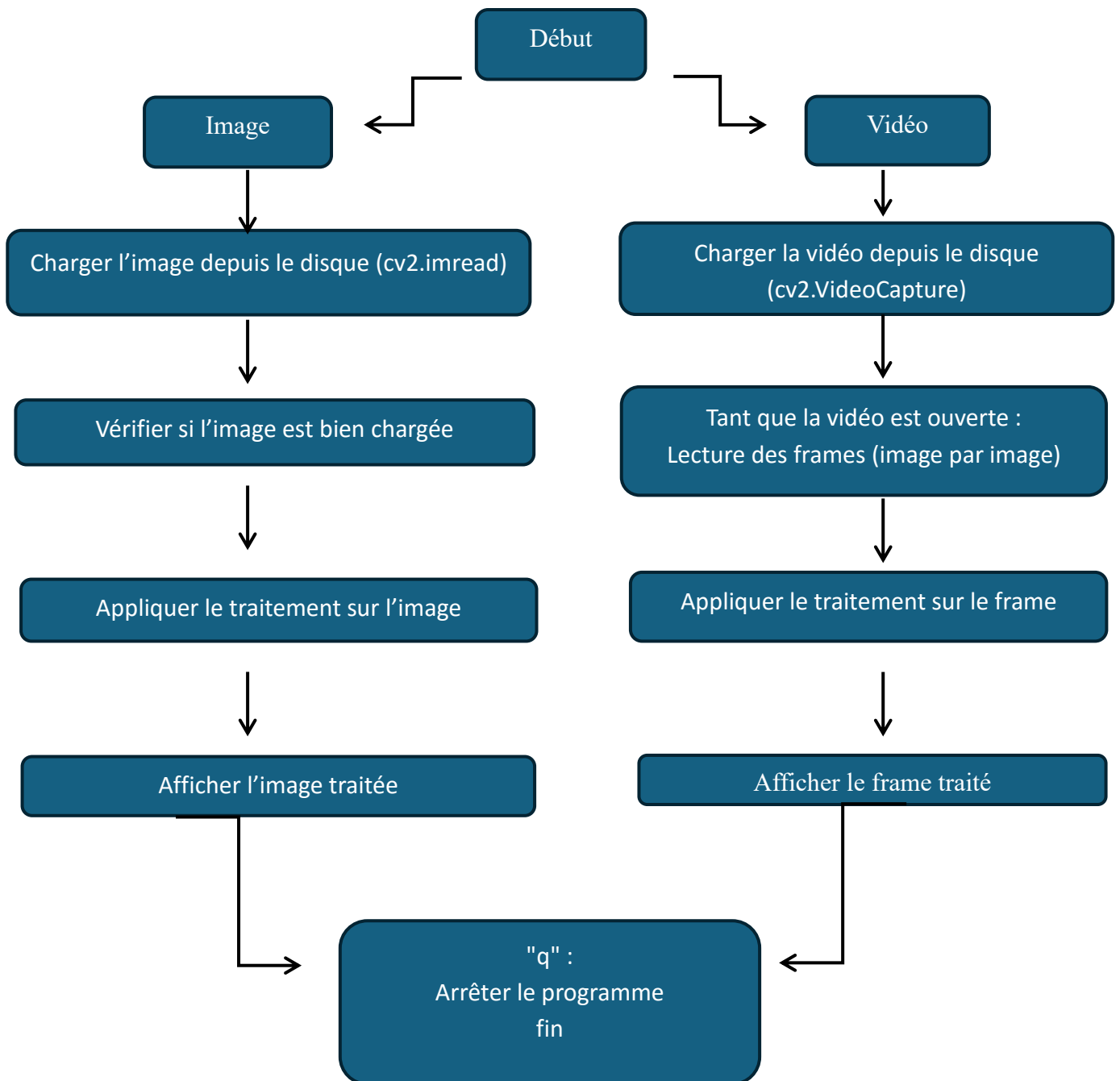
L'utilisation d'une caméra IP offre une plus grande flexibilité qu'une webcam classique, car elle peut être placée à n'importe quel endroit couvert par le réseau sans nécessiter une connexion physique directe avec l'ordinateur.

4. LES ORGANIGRAMMES D'IMPLEMENTATIONS :

Pour comprendre comment fonctionne l'application, nous allons présenter les différents schémas qui montrent comment les algorithmes utilisés sont mis en œuvre. Ces schémas expliquent de manière claire les étapes que le système suit, depuis qu'il reçoit des images ou des vidéos jusqu'à ce qu'il affiche les résultats.

Chaque fonction de l'application a son propre schéma qui décrit comment les opérations sont liées et quelles décisions sont prises pendant le traitement. Cela aide à comprendre la manière dont l'application a été programmée et comment les différents éléments de l'application travaillent ensemble.

Les sections qui suivent présentent d'abord l'algorithme principal de l'application, puis les schémas pour les différentes opérations de traitement d'images, comme détecter les contours, séparer les couleurs, ajuster la luminosité et effectuer des opérations morphologiques. Finalement, un schéma général résume comment l'application fonctionne dans son ensemble.

4.1. Algorithme principal du système de traitement d'images et vidéos :

4.2. Organigramme de Détection de Contours :

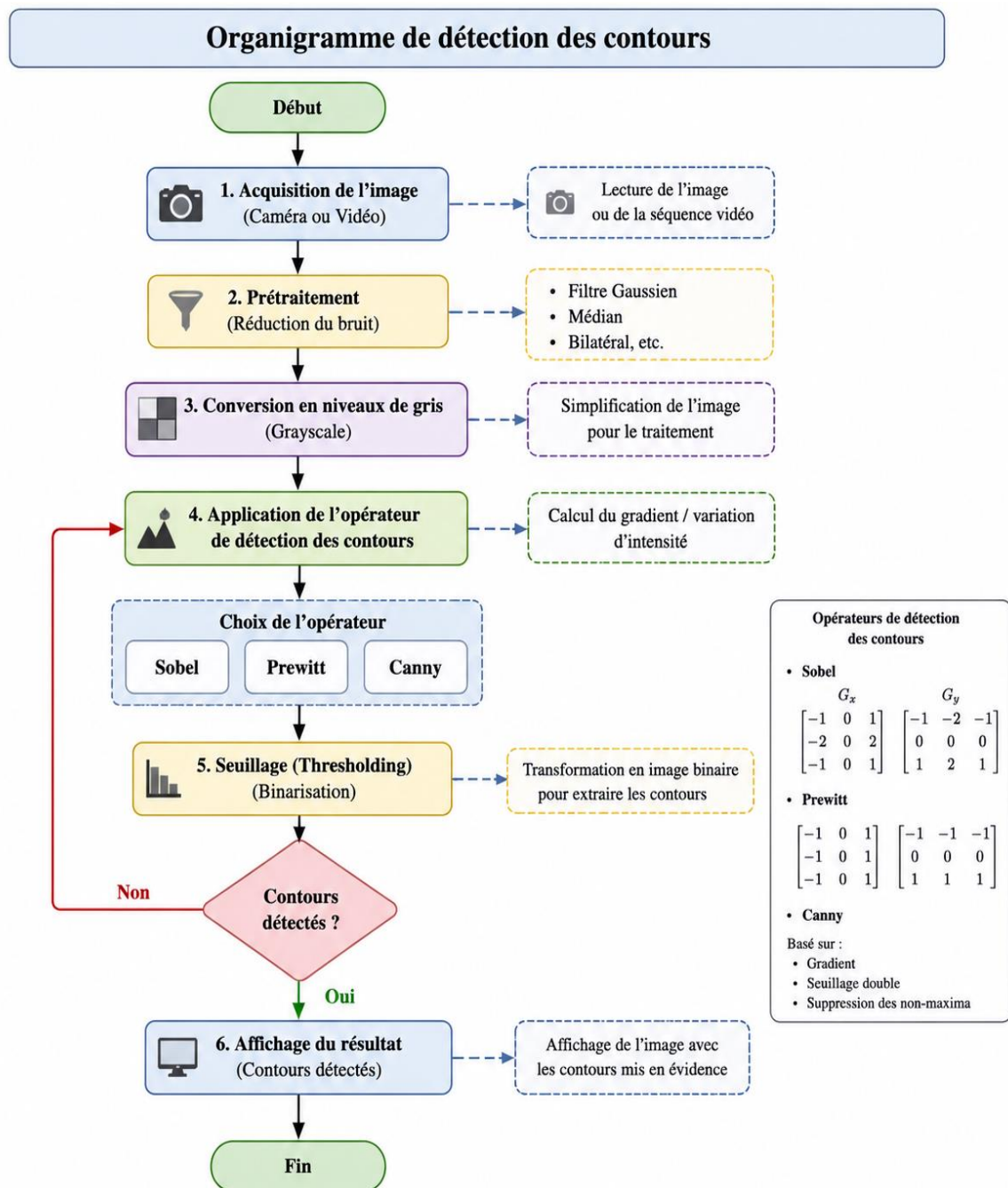


Figure 30: Organigramme de Détection de Contours.

4.3. Organigramme de Segmentation de Couleurs :

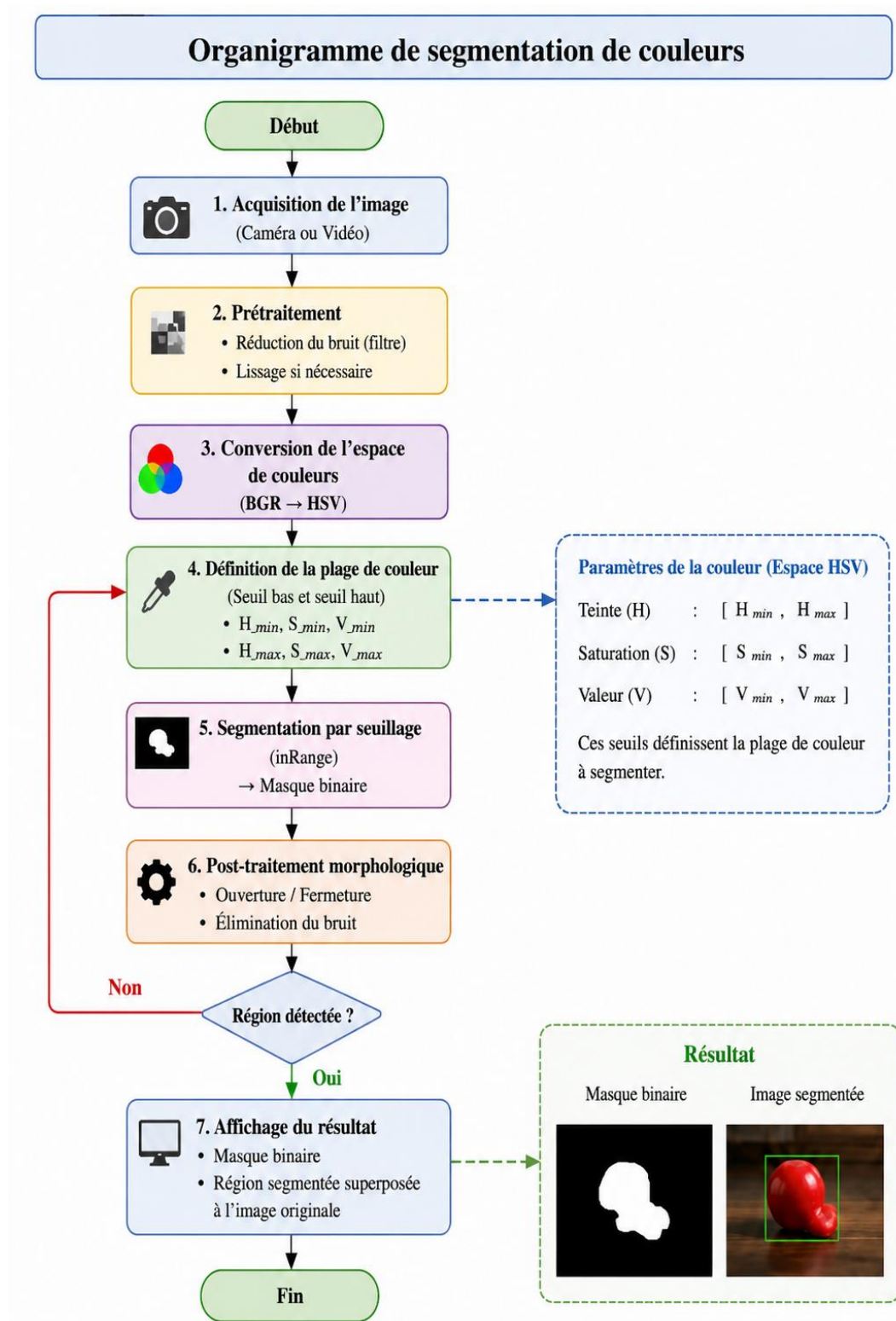


Figure 31: Organigramme de Segmentation de Couleurs.

4.4. Organigramme d'Ajustement de la Luminosité :

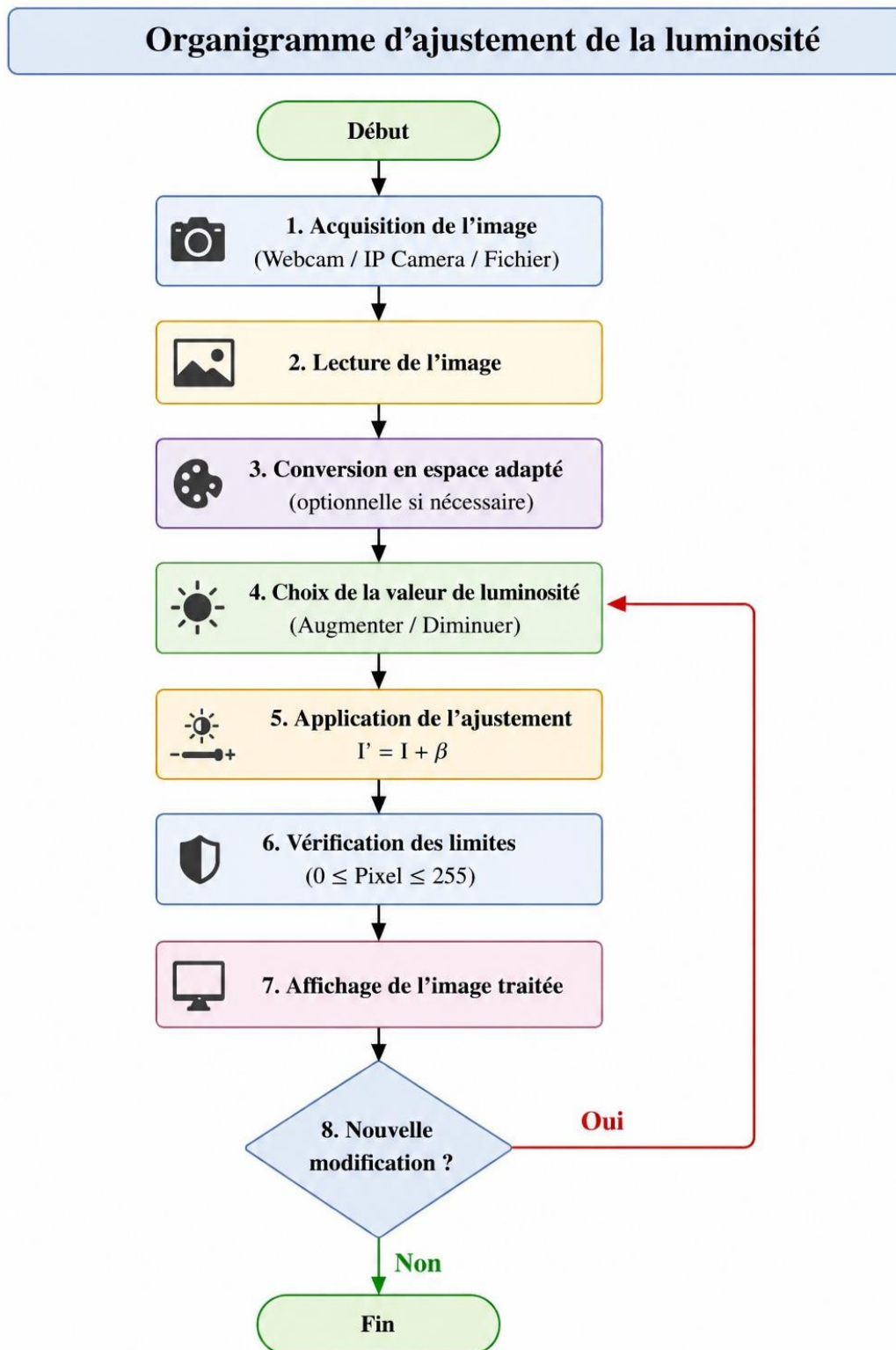


Figure 32: Organigramme d'Ajustement de la Luminosité.

4.5. Organigramme des opérations morphologiques (érosion et dilatation) :

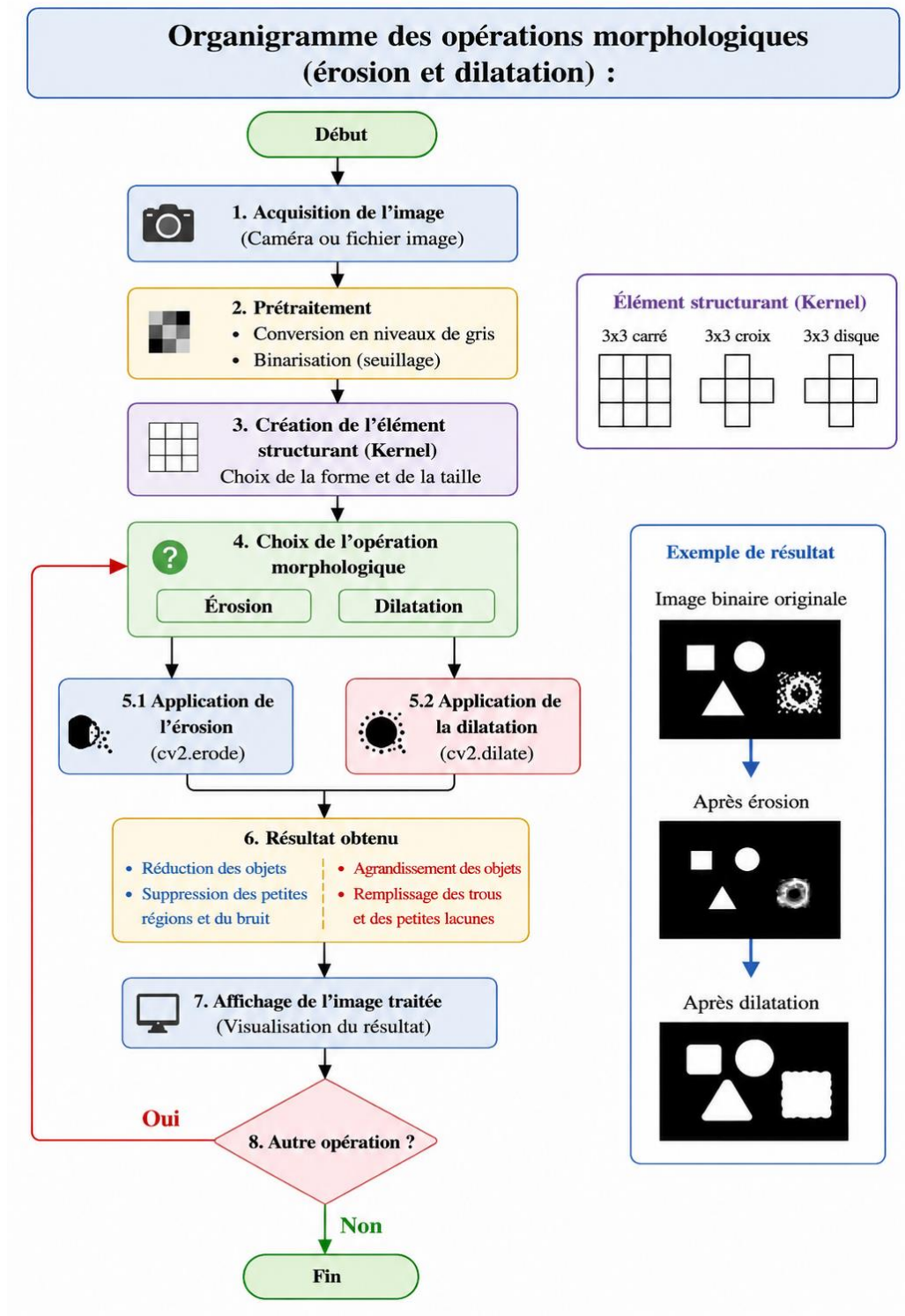
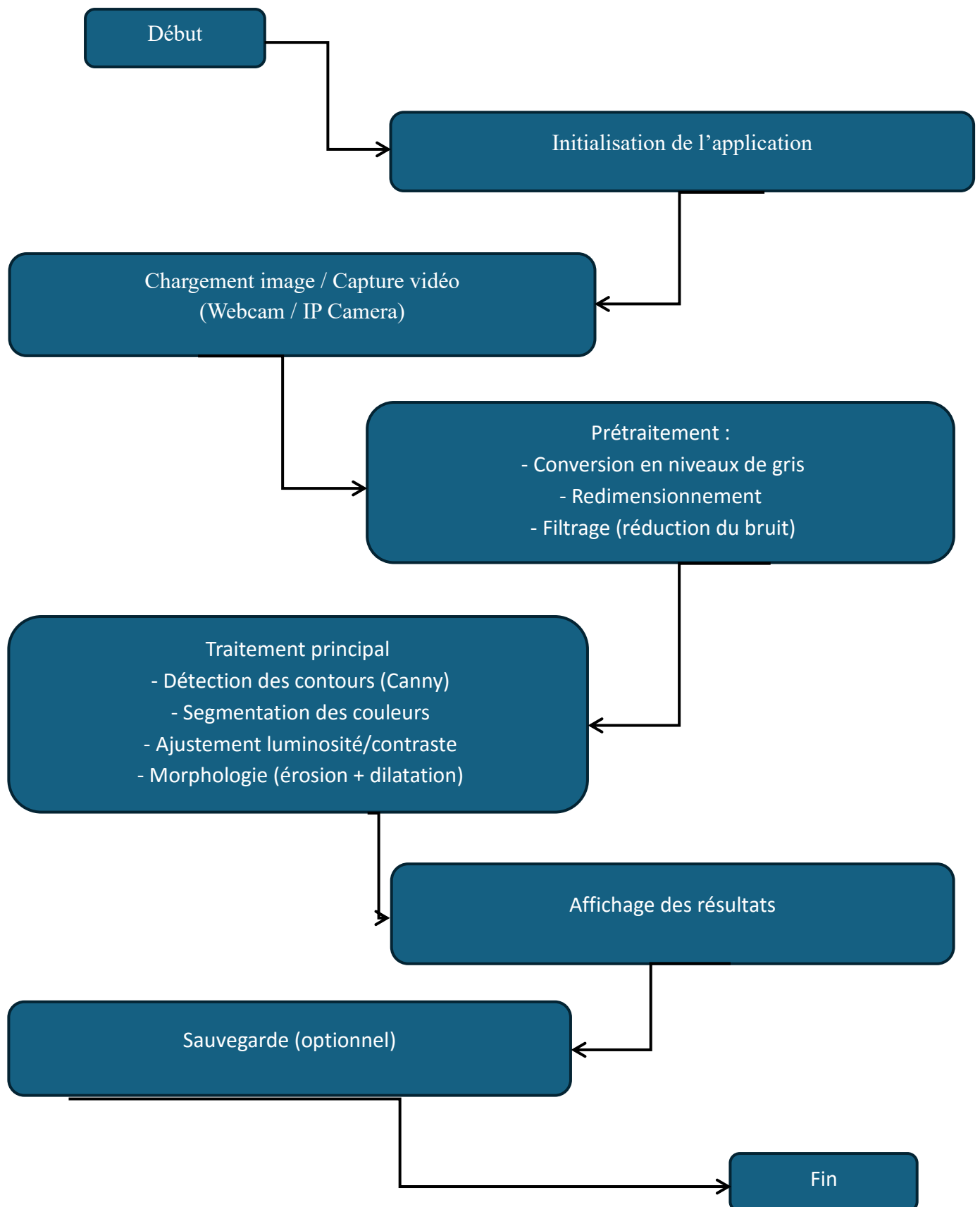


Figure 33: Organigramme des opérations morphologiques (érosion et dilatation).

4.6. Organigramme général de l'application :



5. RESULTATS ET DISCUSSION :

5.1. Détection des contours :

5.1.1. Sobel :



L'image originale



Résultat

Le code python utilisé :

```
# 2. Traitement des contours par Sobel
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (3, 3), 0)

sobel_x = cv2.Sobel(blurred, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(blurred, cv2.CV_64F, 0, 1, ksize=3)

sobel_combined = cv2.magnitude(sobel_x, sobel_y)
sobel_final = np.uint8(np.clip(sobel_combined, 0, 255))
```

Remarque :

La méthode Sobel a permis de détecter les principaux contours de l'image. Les résultats obtenus sont satisfaisants, mais certaines zones présentent du bruit et des contours moins précis.

5.1.2. Prewit :



Le code python utilisé :

```
# 3. Définition des noyaux (Kernels) mathématiques de Prewitt
# Noyau pour les contours verticaux (Prewitt X)
kernelx = np.array([[-1, 0, 1],
                    [-1, 0, 1],
                    [-1, 0, 1]], dtype=np.float32)

# Noyau pour les contours horizontaux (Prewitt Y)
kernely = np.array([[-1, -1, -1],
                    [ 0,  0,  0],
                    [ 1,  1,  1]], dtype=np.float32)

# 4. Application des filtres sur l'image en 64-bit pour la précision
prewitt_x = cv2.filter2D(blurred, cv2.CV_64F, kernelx)
prewitt_y = cv2.filter2D(blurred, cv2.CV_64F, kernely)
```

Remarque :

La méthode Prewitt a été utilisée pour extraire les principaux contours de l'image. Les résultats sont globalement satisfaisants, mais certaines zones présentent du bruit et des contours moins précis.

5.1.3. Canny :



L'image originale

Le code python utilisé :



Résultat

```
# 3. Récupération des valeurs des curseurs en direct
seuil_bas = cv2.getTrackbarPos('Seuil Bas', 'Controleurs Canny')
seuil_haut = cv2.getTrackbarPos('Seuil Haut', 'Controleurs Canny')

# 4. Application de l'algorithme de Canny
canny_final = cv2.Canny(blurred, seuil_bas, seuil_haut)

# 5. Affichage des fenêtres
cv2.imshow('1. Image Originale', frame)
cv2.imshow('2. Contours - Algorithme Canny', canny_final)
```

Remarque :

La méthode de Canny est considérée comme l'un des algorithmes les plus performants pour la détection des contours. Elle repose sur plusieurs étapes successives, notamment la réduction du bruit par filtrage gaussien, le calcul du gradient, la suppression des non-maxima et l'utilisation d'un double seuillage avec hystérésis. Grâce à ce processus, elle permet d'obtenir des contours fins, précis et bien définis, tout en réduisant fortement l'influence du bruit. Ainsi, la méthode de Canny est la plus fiable et la plus utilisée dans les applications de traitement d'images nécessitant une grande précision.

Remarque comparative :

On constate que les trois méthodes de détection de contours (Prewitt, Sobel et Canny) permettent toutes de mettre en évidence les contours d'une image, mais avec des performances différentes. La méthode Prewitt fournit des résultats simples mais sensibles au bruit et peu précis. La méthode Sobel améliore la qualité des contours en offrant une meilleure robustesse et une détection plus nette. Enfin, la méthode Canny donne les résultats

les plus précis et les plus propres grâce à ses étapes de filtrage et de seuillage, ce qui la rend la plus performante pour la détection de contours dans les applications de traitement d'image.

5.2. Traitement morphologique :

5.2.1. Érosion :



Le code python utilisé :

```
# 4. Récupération des valeurs des curseurs en direct
taille_noyau = cv2.getTrackbarPos('Taille Noyau', 'Controles Erosion')
iterations_erosion = cv2.getTrackbarPos('Iterations', 'Controles Erosion')

# Sécurité : taille minimale de 1
taille_noyau = max(1, taille_noyau)

# 5. Création de la matrice carrée d'érosion
kernel = np.ones((taille_noyau, taille_noyau), np.uint8)

# 6. Application de l'Érosion Morphologique
erosion_final = cv2.erode(binary, kernel, iterations=iterations_erosion)

# 7. Affichage des fenêtres
cv2.imshow('1. Image Binaire Originale', binary)
cv2.imshow('2. Resultat - Erosion Morphologique', erosion_final)
```

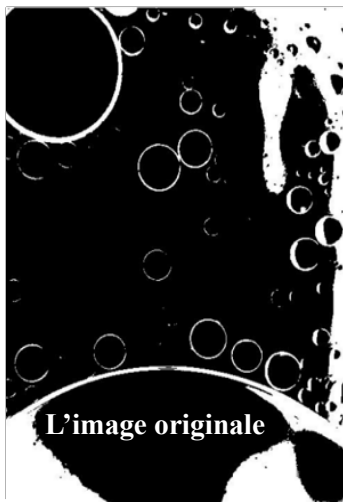
Remarque :

L'érosion (cv2.erode) réduit uniquement la couleur blanche et agrandit la couleur noire.

- **Si votre objet est BLANC** : Il va rétrécir et devenir plus fin.
- **Si votre objet est NOIR** : Il va grossir et s'élargir (car le filtre ronge le fond blanc).

La Règle : Pour nettoyer ou rétrécir correctement un objet, il doit toujours être en BLANC.

5.2.2. Dilatation :



L'image originale

Le code python utilisé :



Résultat

```
# 4. Récupération des valeurs des curseurs en direct
taille_noyau = cv2.getTrackbarPos('Taille Noyau', 'Controles Dilatation')
iterations_dilatation = cv2.getTrackbarPos('Iterations', 'Controles Dilatation')

# Sécurité : taille minimale de 1
taille_noyau = max(1, taille_noyau)

# 5. Création de la matrice carrée (Élément structurant)
kernel = np.ones((taille_noyau, taille_noyau), np.uint8)

# 6. Application de la Dilatation Morphologique (Élargit le blanc)
dilatation_final = cv2.dilate(binary, kernel, iterations=iterations_dilatation)

# 7. Affichage des fenêtres
cv2.imshow('1. Image Binaire Originale', binary)
cv2.imshow('2. Resultat - Dilatation Morphologique', dilatation_final)
```

Remarque :

La dilatation élargit uniquement la couleur blanche.

- Si votre objet est BLANC : Il va grossir et s'épaissir.
- Si votre objet est NOIR : Il va rétrécir (car le blanc autour prend toute la place).

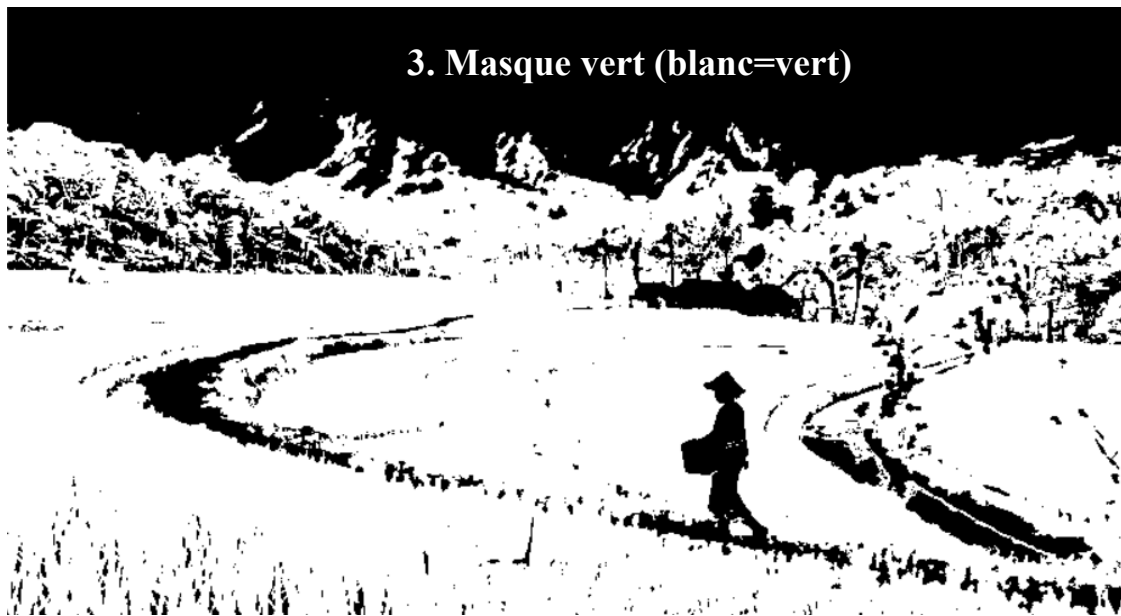
5.3. HSV :

1. Photo originale



2. Vert isolée





Le code python utilisé : (filtrage vert HSV) :

```
# 2. Création de la fenêtre et configuration INITIALE pour le VERT
cv2.namedWindow('Reglages HSV')

# Initialisation des curseurs sur la plage du VERT standard et robuste (36 à 89)
cv2.createTrackbar('H Min', 'Reglages HSV', 36, 179, nothing)
cv2.createTrackbar('H Max', 'Reglages HSV', 89, 179, nothing)
cv2.createTrackbar('S Min', 'Reglages HSV', 40, 255, nothing)
cv2.createTrackbar('S Max', 'Reglages HSV', 255, 255, nothing)
cv2.createTrackbar('V Min', 'Reglages HSV', 40, 255, nothing)
cv2.createTrackbar('V Max', 'Reglages HSV', 255, 255, nothing)

print("Filtrage du VERT actif. Appuyez sur Q pour quitter.")

while True:
    # 3. Conversion BGR vers HSV
    hsv_image = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # 4. Récupération des positions des curseurs
    h_min = cv2.getTrackbarPos('H Min', 'Reglages HSV')
    h_max = cv2.getTrackbarPos('H Max', 'Reglages HSV')
    s_min = cv2.getTrackbarPos('S Min', 'Reglages HSV')
    s_max = cv2.getTrackbarPos('S Max', 'Reglages HSV')
    v_min = cv2.getTrackbarPos('V Min', 'Reglages HSV')
    v_max = cv2.getTrackbarPos('V Max', 'Reglages HSV')

    lower_bound = np.array([h_min, s_min, v_min])
    upper_bound = np.array([h_max, s_max, v_max])

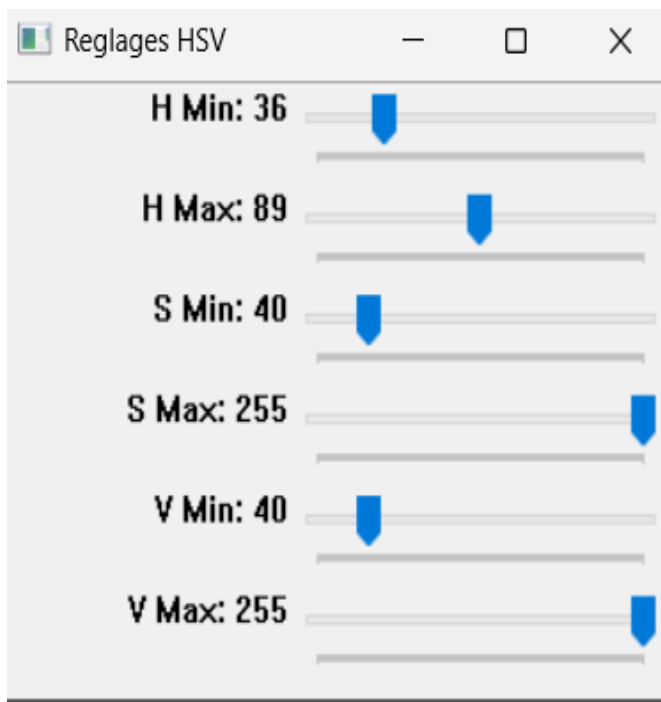
    # 5. Création du masque binaire et application
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)
    result = cv2.bitwise_and(frame, frame, mask=mask)

    # 6. Affichage
    cv2.imshow('1. Image Originale', frame)
    cv2.imshow('2. Masque VERT (Blanc = Vert)', mask)
    cv2.imshow('3. Couleur Verte Isolée', result)
```

Remarque :

Le système HSV sépare la teinte de la couleur de l'éclairage de la pièce pour imiter l'œil humain. Il se compose de :

- **H (Teinte - Hue) :** Détermine le type de couleur (rouge, vert, bleu...). Sa valeur en Python va de 0 à 179.



Filtrage vert HSV

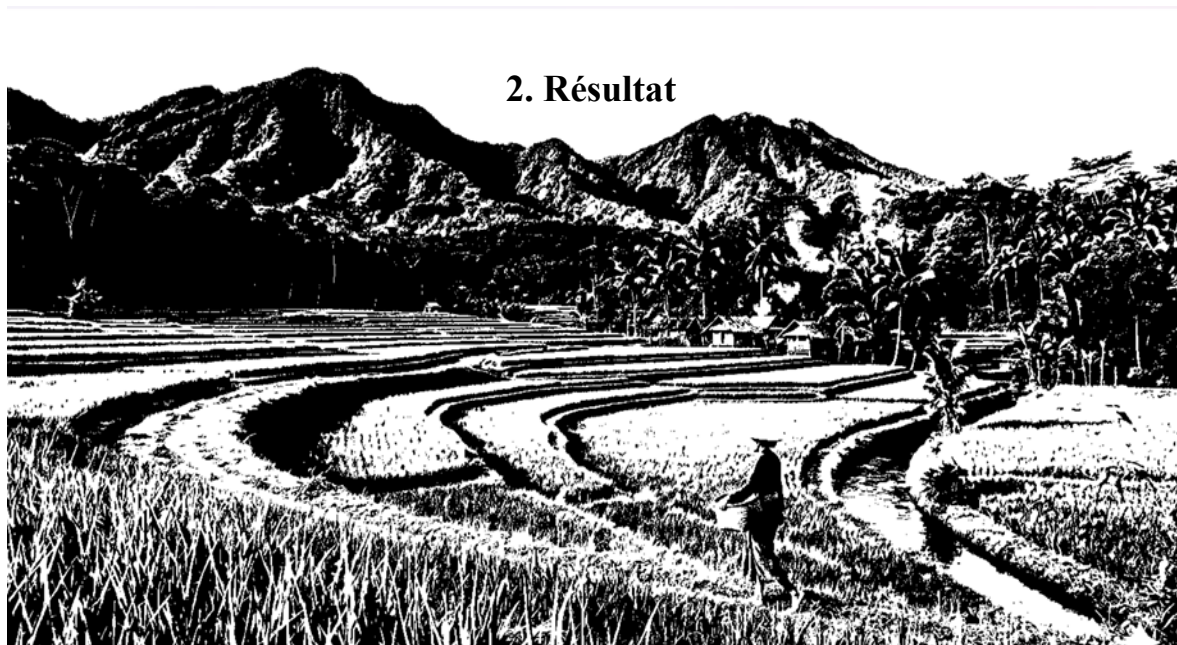
● Pour voir du **VERT** en blanc : H Min = 36 | H Max = 89 | S Min = 50 | V Min = 50
(laissez les Max à 255).

● Pour voir du **BLEU** en blanc : H Min = 90 | H Max = 128 | S Min = 50 | V Min = 50.

● Pour voir du **JAUNE** en blanc : H Min = 20 | H Max = 35 | S Min = 100 | V Min = 100.

- **S (Saturation) :** Détermine l'intensité de la couleur (0 = gris/terne, 255 = très vif).
- **V (Valeur - Value) :** Détermine la quantité de lumière (0 = noir/sombre, 255 = très éclairé).

5.4. Segmentation :



Le code python utilisé :

```
# 2. Convertir l'image en niveaux de gris
image_gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 3. Appliquer un seuillage automatique (Méthode d'Otsu)
_, image_segmentee = cv2.threshold(image_gris, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# 4. Afficher les résultats dans des fenêtres
cv2.imshow('Image Originale', image)
cv2.imshow('Image Segmentee (Masque)', image_segmentee)
```

Remarque :

- **Méthode utilisée :** Seuillage automatique d'Otsu (Otsu's Thresholding) avec la bibliothèque OpenCV.
- **Le principe :** L'image couleur est convertie en niveaux de gris, puis séparée en deux couleurs uniques (**Noir et Blanc**) selon la luminosité de chaque pixel.
- **Analyse du résultat :**
 - **Zones Blanches :** Représentent les éléments très lumineux (le ciel, le cours d'eau, et les rizières qui reflètent la lumière).
 - **Zones Noires :** Représentent les éléments sombres et les ombres (les montagnes, la forêt, les maisons, et les vêtements du fermier).
- **Conclusion**

Le programme a parfaitement réussi à dessiner les contours géométriques des rizières en terrasses. Cependant, comme l'algorithme se base uniquement sur la lumière (et non sur l'intelligence artificielle), il fusionne le fermier et les maisons avec les montagnes dans la même couleur noire.

5.5. Surprissions de l'arrière-plan en temps réelle :

1. Image originale



2. Résultat



Le code python utilisé :

```
# 2. Préparation des masques pour l'algorithme
mask = np.zeros(img.shape[:2], np.uint8)
bgdModel = np.zeros((1, 65), np.float64)
fgdModel = np.zeros((1, 65), np.float64)

# 3. Rectangle centré sur le fermier
h, w = img.shape[:2]
rect = (int(w * 0.30), int(h * 0.05), int(w * 0.45), int(h * 0.90))

# 4. Exécution de la segmentation GrabCut
cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 7, cv2.GC_INIT_WITH_RECT)

# 5. Création du masque binaire
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

# 6. Application de la transparence (Canal Alpha)
b, g, r = cv2.split(img)
alpha = mask2 * 255
img_final = cv2.merge([b, g, r, alpha])
```

Remarque :

L'algorithme de suppression d'arrière-plan, qui utilise l'espace colorimétrique HSV, a très bien détaché la silhouette principale du fermier, y compris son chapeau de paille, sa chemise et sa salopette. L'arrière-plan, qui comprenait des champs verdoyants et des bâtiments, a été complètement supprimé, laissant un fond blanc pur sans aucun bruit numérique.

En ce qui concerne la disparition de la fourche, et plus précisément le manche en bois, ce résultat est tout à fait logique d'un point de vue analytique. L'algorithme a intentionnellement exclu cet outil à cause de la forte ressemblance de couleur et des mêmes valeurs de saturation entre le bois du manche et le sol en terre battue. Cela a permis d'isoler proprement le sujet humain, sans aucune interférence avec les éléments de son environnement.

5.6. YOLO V8 en temps réelle :

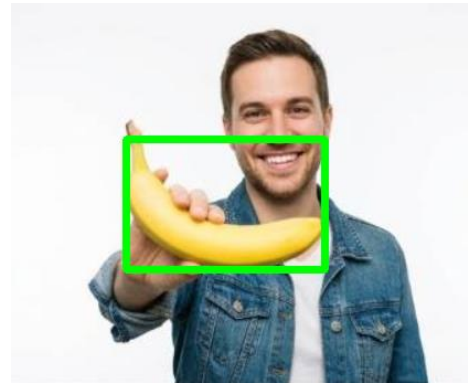


Image originale

Résultat

Remarque technique sur la segmentation du jaune :

L'application du pipeline de traitement sur cette image met en évidence une extraction colorimétrique d'une excellente clarté. Le choix d'une plage HSV adaptée au jaune ($\backslash([20, 100, 100]\backslash)$ à $\backslash([35, 255, 255]\backslash)$) a permis de segmenter intégralement la texture de la banane, en surmontant les variations d'ombrages induites par la courbure naturelle du fruit.

La neutralité de l'arrière-plan (fond blanc pur) élimine tout risque d'interférence chromatique dans le décor. De plus, le fort contraste existant entre la teinte de la peau du sujet et le jaune vif du fruit a permis d'isoler ce dernier sans aucune ambiguïté pixelisée.

L'application du filtre d'ouverture a efficacement corrigé les micro-zones de reflets lumineux, permettant à la fonction géométrique d'ajuster une boîte englobante (Bounding Box) d'une précision chirurgicale, centrée de manière exclusive et stable sur la cible.

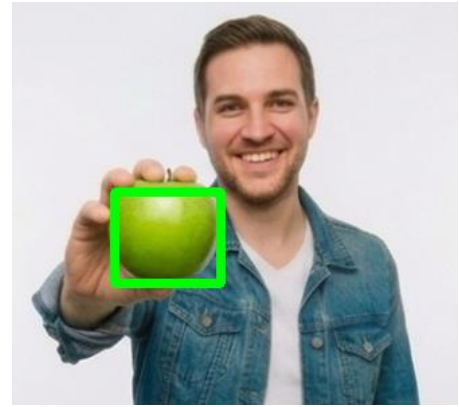


Image originale

Résultat

Remarque technique sur la segmentation du vert :

L'application du script sur cette image démontre une efficacité de détournage remarquable. Le choix d'une plage HSV tolérante ($\backslash[36, 40, 40]\backslash$ à $\backslash[89, 255, 255]\backslash$) a permis d'isoler parfaitement la pomme verte, malgré les variations de luminosité et les ombres propres à sa forme sphérique.

De plus, la neutralité de l'arrière-plan (fond blanc pur) a grandement facilité le processus : n'ayant aucune signature chromatique verte dans le décor, le masque binaire généré est totalement exempt de bruit numérique. L'opération morphologique d'ouverture (`cv2.MORPH_OPEN`) a nettoyé les légers reflets sur la peau du fruit, permettant à la méthode géométrique `getbbox()` de tracer un rectangle vert d'une précision chirurgicale, centré exclusivement sur l'objet ciblé

Le code python utilisé pour la Sélection de la classe jaune :

```
import cv2
import numpy as np
from PIL import Image

# 1. Calcul dynamique des seuils HSV (Gestion du wrap-around)
def get_limits(color):
    c = np.uint8([[color]]) # Conversion format matriciel 1x1 BGR
    hsvC = cv2.cvtColor(c, cv2.COLOR_BGR2HSV)
    hue = hsvC[0][0][0] # Extraction de la teinte pure (Hue)

    if hue >= 165:
        lowerLimit = np.array([hue - 10, 100, 100], dtype=np.uint8)
        upperLimit = np.array([180, 255, 255], dtype=np.uint8)
    elif hue <= 15:
        lowerLimit = np.array([0, 100, 100], dtype=np.uint8)
        upperLimit = np.array([hue + 10, 255, 255], dtype=np.uint8)
    else:
        lowerLimit = np.array([hue - 10, 100, 100], dtype=np.uint8)
        upperLimit = np.array([hue + 10, 255, 255], dtype=np.uint8)
    return lowerLimit, upperLimit

# 2. Prétraitement et conversion de la photo cible
frame = cv2.imread(r"C:\Users\HP\Desktop\dj.png")
hsvImage = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# 3. Extraction du masque binaire pour la couleur JAUNE
yellow_bgr = [0, 255, 255]
lowerLimit, upperLimit = get_limits(color=yellow_bgr)
mask = cv2.inRange(hsvImage, lowerLimit, upperLimit)

# 4. Localisation géométrique automatique et dessin du rectangle vert
mask_pil = Image.fromarray(mask)
bbox = mask_pil.getbbox()

if bbox is not None:
    x1, y1, x2, y2 = bbox
    # Dessin de la Bounding Box (Épaisseur 5 pixels)
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)

# Sauvegarde du résultat directement sur le Bureau
cv2.imwrite(r"C:\Users\HP\Desktop\dj_detection_jaune.png", frame)
```

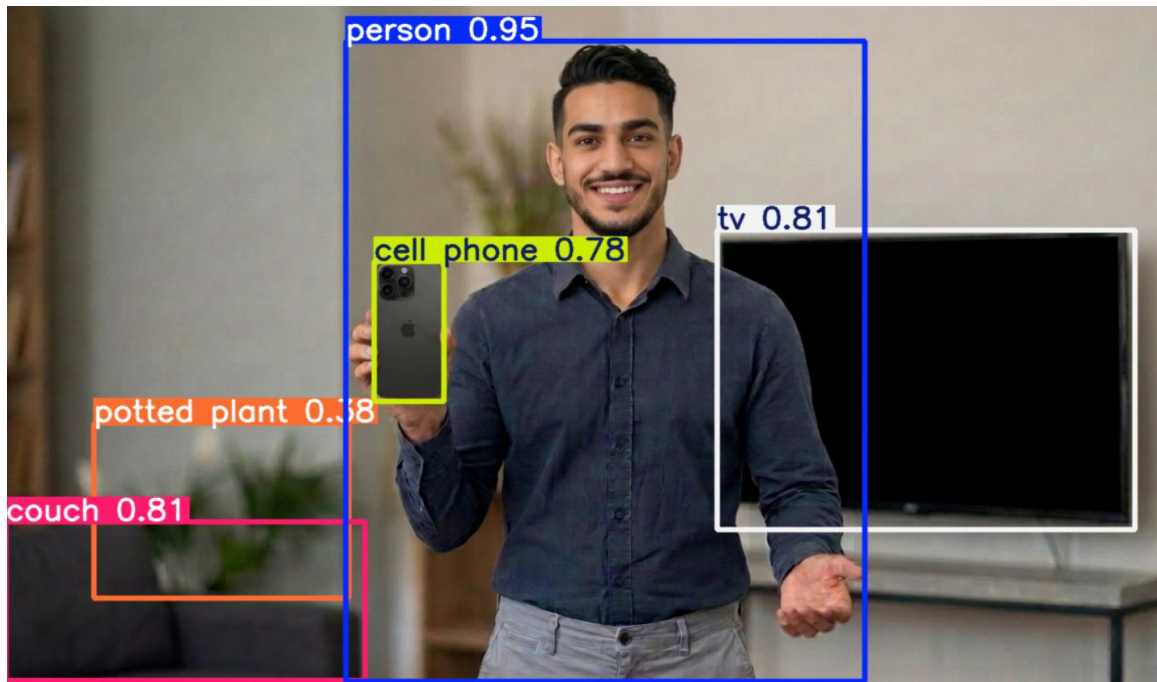
Remarque :

L'intégration du modèle YOLOv8 représente une avancée technologique majeure pour notre système de vision par ordinateur, offrant l'avantage de la « perception contextuelle ».

Contrairement aux méthodes OpenCV traditionnelles (telles que l'isolation HSV) qui analysent les pixels de manière uniforme et discrète, en se basant uniquement sur un éclairage statique ou un fond blanc neutre, YOLOv8 utilise un filtrage sémantique. Grâce à son réseau neuronal convolutif, le modèle reconnaît instantanément la géométrie et la couleur globales des objets.

5.7. YOLO V8 (Détection des objets) :

Image originale



Résultat

Le code python utilisé :

```
import cv2
from ultralytics import YOLO

# 1. Charger le modèle YOLOv8
model = YOLO("yolov8n.pt")

# 2. Charger votre photo (remplacez 'votre_image.jpg' par le chemin de votre photo)
image_path = r"C:\Users\HP\Desktop\djl.png"

frame = cv2.imread(image_path)

if frame is None:
    print(f"Erreur : Impossible de charger l'image à l'emplacement '{image_path}'.")
else:
    # 3. Lancer la détection sur la photo
    results = model(frame)

    # 4. Dessiner les rectangles de détection sur l'image
    for r in results:
        annotated_frame = r.plot()

    print("Appuyez sur n'importe quel touche pour fermer la photo.")

    # 5. Afficher le résultat dans une fenêtre OpenCV
    cv2.imshow("Detection YOLOv8 - Photo", annotated_frame)

    # Attend qu'une touche soit pressée pour fermer (0 = attend indéfiniment)
    cv2.waitKey(0)

    # 6. Fermer proprement la fenêtre
    cv2.destroyAllWindows()
```

Remarque :

Afin d'évaluer les performances du modèle YOLOv8n (version Nano), nous avons testé l'algorithme sur une image complexe contenant plusieurs objets de tailles différentes, soumis à un effet de flou artistique en arrière-plan (effet Bokeh).

Les résultats obtenus révèlent les points clés suivants :

- Précision sur les objets principaux : Le modèle a démontré une excellente robustesse en détectant l'élément humain (Person) avec un indice de confiance très élevé de 95%, ainsi que l'écran de télévision (TV) à 81%, malgré l'occlusion partielle causée par le sujet.
- Détection des petits objets : Le téléphone portable (Cell phone), bien qu'il occupe une surface réduite sur l'image, a été localisé avec succès avec une confiance de 78%. Cela prouve l'efficacité des mécanismes d'attention et des couches de convolution de YOLOv8 pour capturer les caractéristiques des objets de petite taille.
- Sensibilité à l'arrière-plan (Bruit visuel) : Le modèle a réussi à segmenter le canapé (Couch : 81%) et une plante en pot (Potted plant : 38%). Le faible score de la plante s'explique par le flou d'arrière-plan qui atténue les contours de l'objet.

Conclusion de l'expérience :

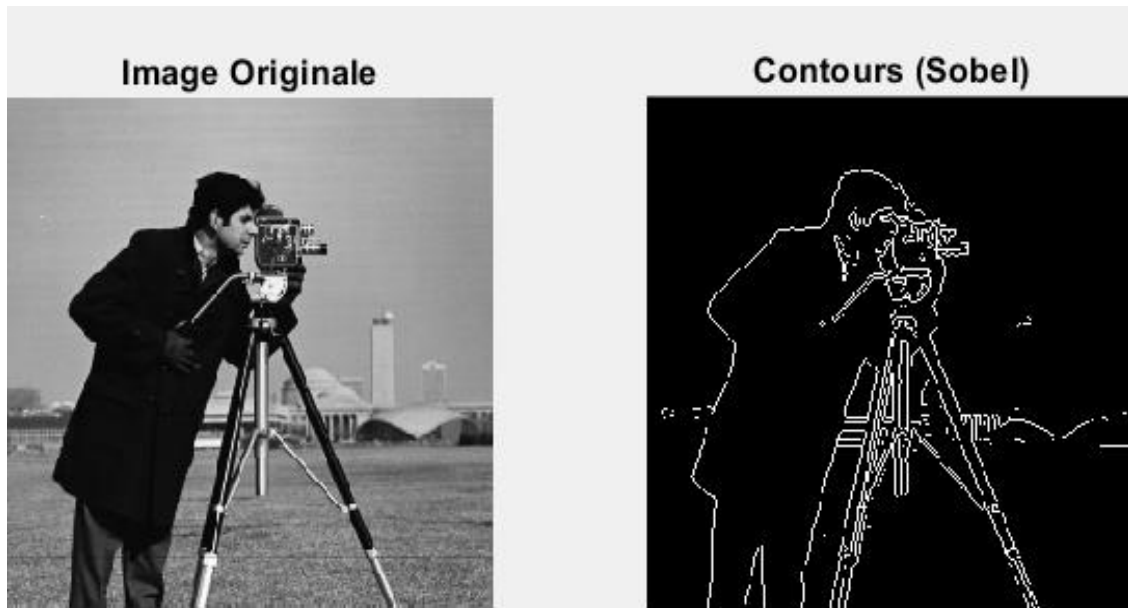
Ces résultats confirment que le modèle YOLOv8n offre un excellent compromis entre la légèreté algorithmique (idéale pour les architectures embarquées ou mobiles) et la précision de détection en temps réel, même dans des conditions d'arrière-plan flouté.

6. DES EXEMPLES D'APPLICATIONS SOUS MATLAB :

Cette section présente quelques exemples d'applications sous MATLAB. Ces exemples permettent de mieux visualiser son fonctionnement ainsi que les résultats obtenus lors de son implémentation.

6.1. Détection des contours :

6.1.1. Sobel :



Le code MATLAB utilisé :

```
%% 2. CONVERSION EN NIVEAUX DE GRIS
if size(I, 3) == 3
    I_gray = rgb2gray(I);
else
    I_gray = I;
end

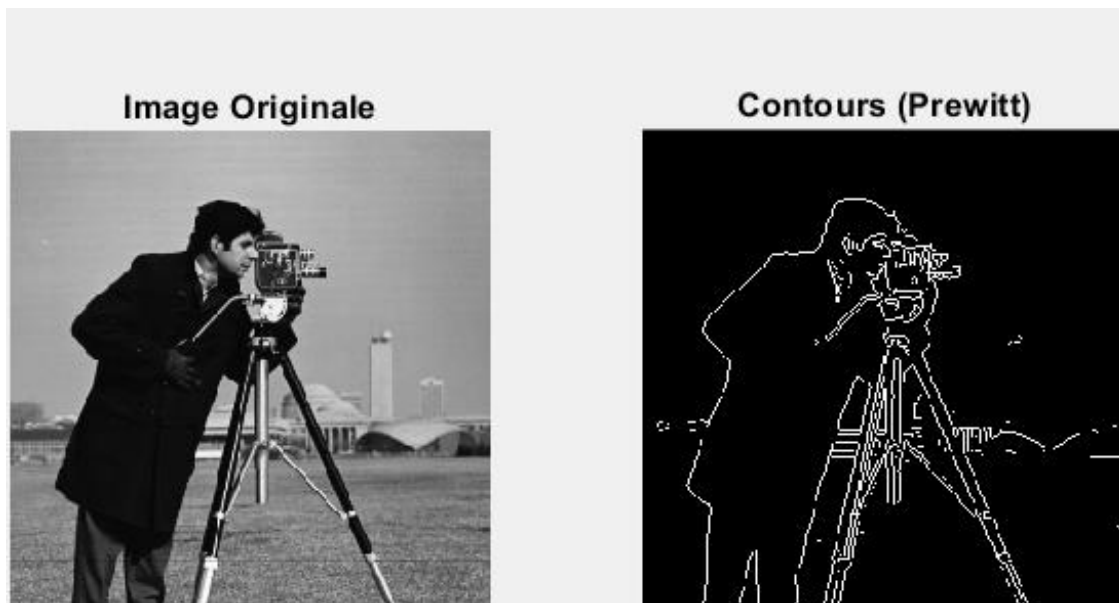
%% 3. APPLICATION DE LA MÉTHODE DE SOBEL
I_edges = edge(I_gray, 'sobel');

%% 4. AFFICHAGE DES RÉSULTATS
figure('Name', 'Détection de contours - Méthode Sobel', 'NumberTitle', 'off');

% Image d'origine
subplot(1, 2, 1);
imshow(I);
title('Image Originale');

% Image avec contours
subplot(1, 2, 2);
imshow(I_edges);
title('Contours (Sobel)');
```

6.1.2. Prewit :



Le code MATLAB utilisé :

```
%% 3. APPLICATION DE LA MÉTHODE DE PREWITT
```

```
I_edges = edge(I_gray, 'prewitt'); % <--- Changement de méthode ici [1]
```

```
%% 4. AFFICHAGE DES RÉSULTATS
```

```
figure('Name', 'Détection de contours - Méthode Prewitt', 'NumberTitle', 'off');
```

```
% Image d'origine
```

```
subplot(1, 2, 1);
```

```
imshow(I);
```

```
title('Image Originale');
```

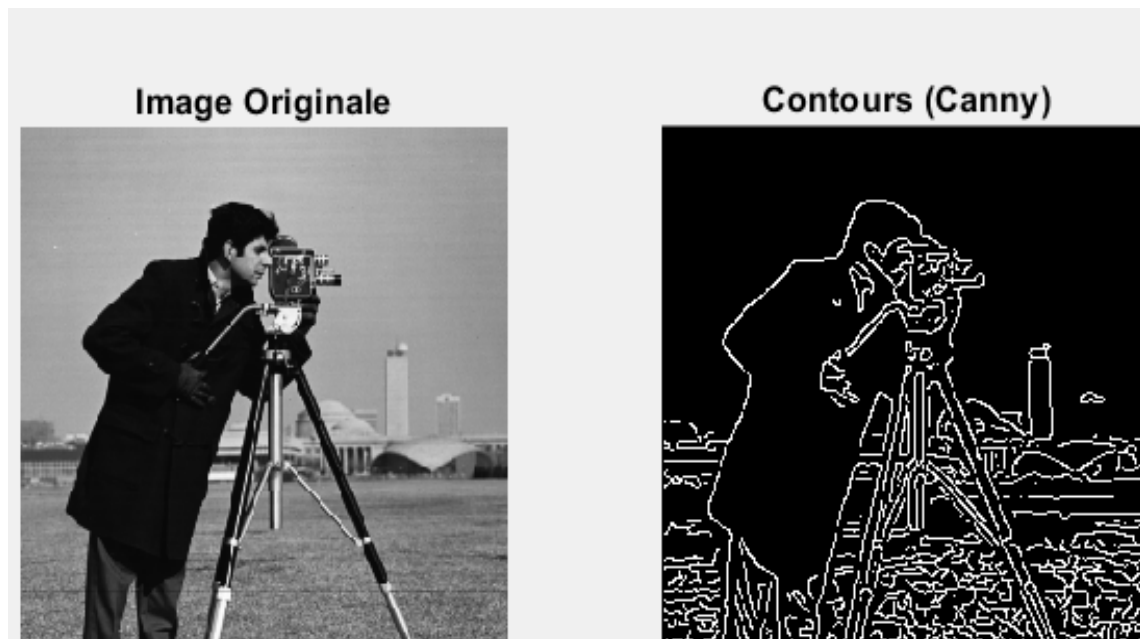
```
% Image avec contours
```

```
subplot(1, 2, 2);
```

```
imshow(I_edges);
```

```
title('Contours (Prewitt)');
```

6.1.3. Canny :



Le code MATLAB utilisé :

```
%% 3. APPLICATION DE LA MÉTHODE DE CANNY
I_edges = edge(I_gray, 'canny'); % <--- Changement de méthode ici

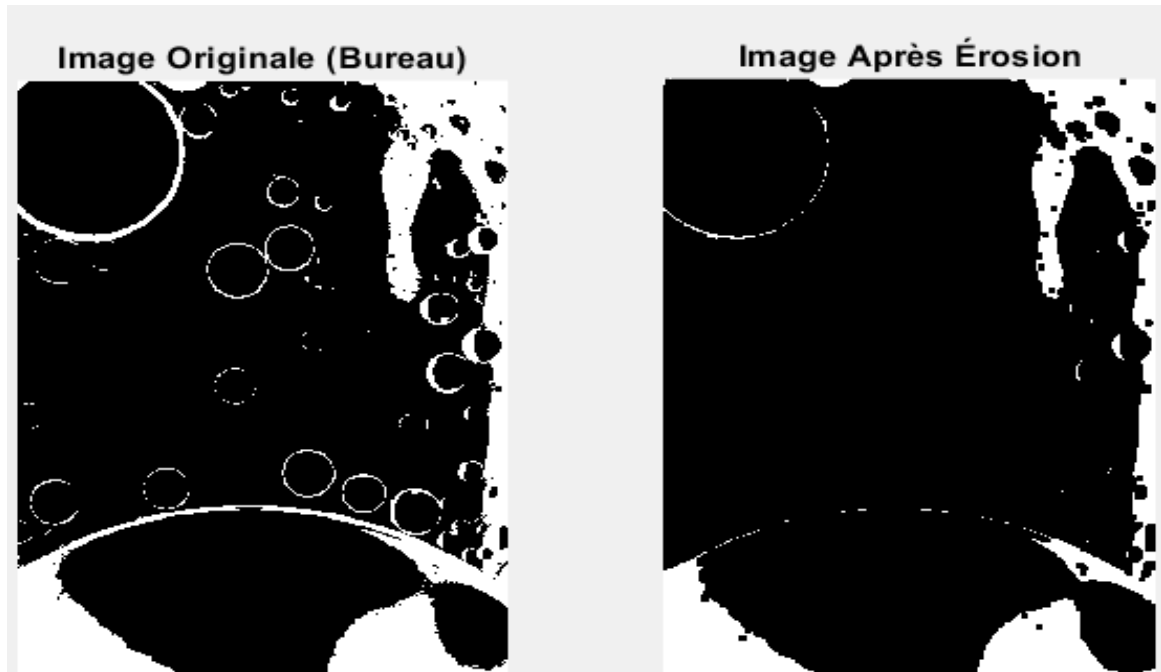
%% 4. AFFICHAGE DES RÉSULTATS
figure('Name', 'Détection de contours - Méthode Canny', 'NumberTitle', 'off');

% Image d'origine
subplot(1, 2, 1);
imshow(I);
title('Image Originale');

% Image avec contours
subplot(1, 2, 2);
imshow(I_edges);
title('Contours (Canny)');
```

6.2. Traitement morphologique :

6.2.1. Érosion :



Le code MATLAB utilisé :

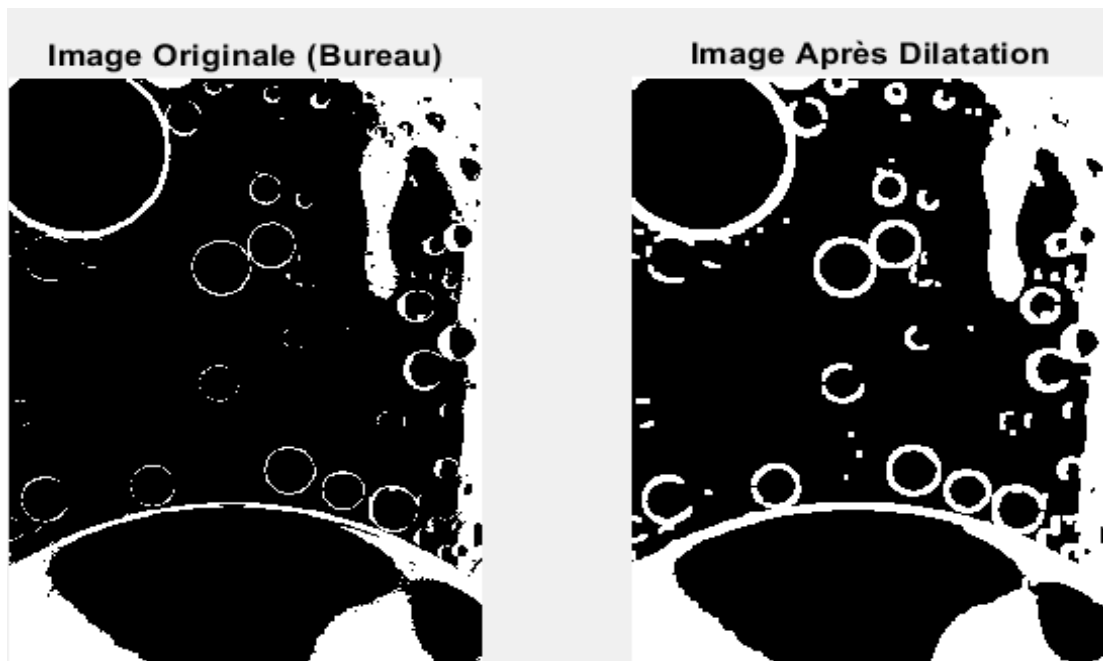
```
%% 4. APPLICATION DE L'ÉROSION
% On crée un disque de rayon 3 pixels. Augmentez ce chiffre pour une érosion plus forte.
se = strel('disk', 3);
I_eroded = imerode(I_binary, se); % <--- Fonction d'érosion ici

%% 5. AFFICHAGE DES RÉSULTATS
figure('Name', 'Morphologie Mathématique - Érosion du Bureau', 'NumberTitle', 'off');

subplot(1, 2, 1);
imshow(I_binary);
title('Image Originale (Bureau)');

subplot(1, 2, 2);
imshow(I_eroded);
title('Image Après Érosion');
```

6.2.2. Dilatation :

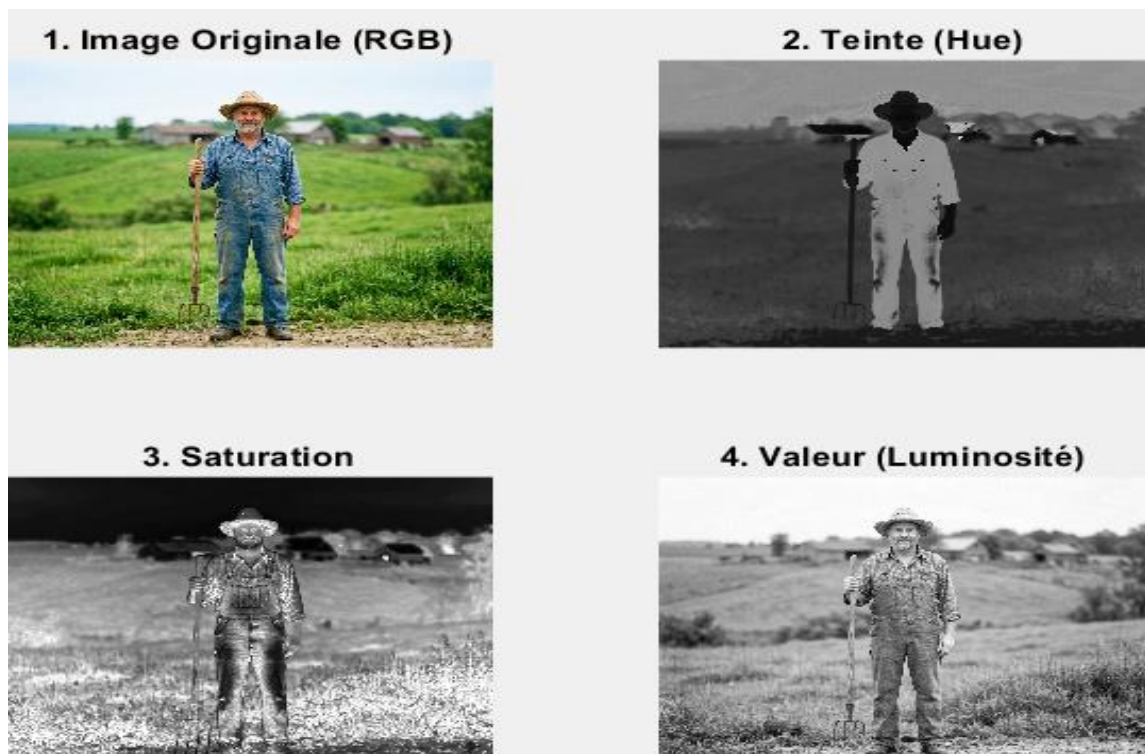


Le code MATLAB utilisé :

```
%% 4. APPLICATION DE LA DILATATION  
se = strel('disk', 3);  
I_dilated = imdilate(I_binary, se);
```

```
%% 5. AFFICHAGE DES RÉSULTATS  
figure('Name', 'Morphologie Mathématique - Image du Bureau', 'NumberTitle', 'off');  
  
subplot(1, 2, 1);  
imshow(I_binary);  
title('Image Originale (Bureau)');  
  
subplot(1, 2, 2);  
imshow(I_dilated);  
title('Image Après Dilatation');
```

6.3. HSV :



Le code MATLAB utilisé :

```

%% 3. CONVERSION EN ESPACE COULEUR HSV
I_hsv = rgb2hsv(I); % <--- Fonction de conversion de MATLAB

% Extraction des 3 canaux individuels
H = I_hsv(:, :, 1); % Teinte (Hue)
S = I_hsv(:, :, 2); % Saturation
V = I_hsv(:, :, 3); % Valeur / Luminosité (Value)

```

```

%% 4. AFFICHAGE DES RÉSULTATS
figure('Name', 'Espace Couleur HSV', 'NumberTitle', 'off');

% Image d'origine
subplot(2, 2, 1); imshow(I);
title('1. Image Originale (RGB)');

% Canal Teinte
subplot(2, 2, 2); imshow(H);
title('2. Teinte (Hue)');

% Canal Saturation
subplot(2, 2, 3); imshow(S);
title('3. Saturation');

% Canal Valeur / Luminosité
subplot(2, 2, 4); imshow(V);
title('4. Valeur (Luminosité)');

```

6.4. Surprissions de l'arrière-plan en temps réelle :



Le code MATLAB utilisé :

```
%% 4. CALCUL DE LA DISTANCE DE COULEUR (Segmentation)
I_double = double(I);

% Calcul de la différence mathématique pour chaque canal
R_diff = I_double(:,:,1) - couleur_fond(1);
G_diff = I_double(:,:,2) - couleur_fond(2);
B_diff = I_double(:,:,3) - couleur_fond(3);

% Distance euclidienne globale de la couleur
distance_couleur = sqrt(R_diff.^2 + G_diff.^2 + B_diff.^2);

%% 5. CRÉATION DU MASQUE ET SUPPRESSION
% Seuil de tolérance : augmentez ce chiffre (ex: 70) si le fond ne s'efface pas assez,
% diminuez-le (ex: 30) si des morceaux de l'objet disparaissent.
seuil_tolerance = 50;

% On garde tout ce qui est ÉLOIGNÉ de la couleur du fond sélectionnée
masque = distance_couleur > seuil_tolerance;

% Nettoyage du masque pour enlever les petits pixels isolés
masque = imopen(masque, strel('disk', 2));
masque = imfill(masque, 'holes');

% Application du masque sur l'image d'origine
I_final = im2double(I) .* double(masque);
```

6.5. Détection d'objet :



Le code MATLAB utilisé :

```
%% 2. DETECTION DU VISAGE (ALGORITHME VIOLA-JONES)
faceDetector = vision.CascadeObjectDetector();

% Paramètres optimisés pour détecter le visage principal
faceDetector.MinSize = [80 80];
faceDetector.MergeThreshold = 6;

% Exécution de la détection
bbox = step(faceDetector, imgOriginale);

%% 3. CRÉATION DE L'IMAGE AVEC LE RECTANGLE
% Dessine un rectangle vert épais (largeur 5) autour du visage détecté
imgDetectee = insertShape(imgOriginale, 'rectangle', bbox, 'LineWidth', 5, 'Color', 'green');

%% 4. AFFICHAGE CÔTÉ À CÔTÉ (ORIGINALE À CÔTÉ DE LA DÉTECTION)
figure('Name', 'Comparaison : Originale vs Détection', 'NumberTitle', 'off');

% Sous-fenêtre 1 (à gauche) : Image originale
subplot(1, 2, 1);
imshow(imgOriginale);
title('1. Image Originale');

% Sous-fenêtre 2 (à droite) : Image avec détection
subplot(1, 2, 2);
imshow(imgDetectee);

% Ajustement dynamique du titre selon le résultat
if isempty(bbox)
    title('2. Aucun visage détecté');
else
    title(sprintf('2. Visage détecté (%d)', size(bbox, 1)));
end
```

7. Un commentaire comparatif entre MATLAB et PYTHON :

Dans le traitement d'images et de vidéos en utilisant le calcul parallèle sur les processeurs graphiques NVIDIA RTX. MATLAB et Python sont deux environnements de développement très performants pour le traitement d'images et de vidéos. MATLAB est souvent utilisé dans les domaines académiques et scientifiques en raison de son environnement intégré, de sa simplicité d'utilisation et de ses nombreuses boîtes à outils spécialisées. Ces outils, tels que la boîte à outils de traitement d'images, la boîte à outils de calcul parallèle et la boîte à outils de vision par ordinateur, permettent de développer rapidement des applications de traitement d'images tout en exploitant la puissance des processeurs graphiques (GPU) via CUDA.

D'un autre côté, Python est devenu une solution très populaire en raison de son caractère open source, de sa flexibilité et de son vaste écosystème de bibliothèques dédiées au calcul scientifique et à l'intelligence artificielle. Des bibliothèques telles que OpenCV, NumPy, CuPy et PyTorch offrent la possibilité d'effectuer des traitements d'images et de vidéos avec une grande efficacité, tout en exploitant pleinement les capacités des cartes graphiques RTX grâce aux technologies CUDA.

En termes de performances, les deux environnements permettent d'obtenir des temps d'exécution très réduits lorsqu'ils utilisent l'accélération par processeur graphique (GPU).

Cependant, MATLAB offre une mise en œuvre plus simple et plus intuitive pour les utilisateurs débutants ou les travaux de recherche. En revanche, Python procure une plus grande liberté de développement, une meilleure extensibilité et un contrôle plus fin des ressources matérielles.

Le choix entre MATLAB et Python dépend donc principalement des objectifs du projet. MATLAB est particulièrement adapté au prototypage rapide, aux expérimentations scientifiques et aux applications pédagogiques. Python, quant à lui, représente une solution plus économique, évolutive et mieux adaptée au développement d'applications industrielles et de systèmes de traitement d'images et de vidéos à grande échelle. Dans cette étude, les deux approches sont présentées afin de mettre en évidence leurs similitudes, leurs différences et leur capacité à exploiter efficacement les performances offertes par les processeurs graphiques NVIDIA RTX dans le domaine du calcul parallèle.

8. CONCLUSION :

Ce quatrième chapitre a été consacré à l'implémentation pratique de l'application développée ainsi qu'à l'analyse et la discussion des résultats obtenus. L'environnement de développement basé sur Python, associé à la bibliothèque OpenCV, a permis de concevoir une application performante et flexible dédiée au traitement d'images et de vidéos en temps réel.

L'utilisation d'une caméra IP comme source vidéo a démontré la capacité du système à acquérir et traiter des flux vidéo à distance de manière efficace. L'architecture logicielle

mise en place, composée de plusieurs modules fonctionnels, a permis d'assurer la détection, le suivi et l'analyse des objets dans différentes conditions d'utilisation.

Les expérimentations réalisées ont permis d'évaluer les performances des différentes techniques de traitement d'images intégrées à l'application. Concernant la détection des contours, une comparaison entre les opérateurs Sobel, Prewitt et Canny a montré que la méthode de Canny fournit les meilleurs résultats grâce à sa capacité à réduire le bruit et à détecter avec précision les contours fins par l'utilisation du filtrage gaussien et du seuil par hystérésis.

Les opérations morphologiques, notamment l'érosion et la dilatation, se sont révélées très efficaces pour améliorer la qualité des images binaires après la segmentation. Elles permettent d'éliminer les petits artefacts, de combler certaines discontinuités et d'obtenir des régions d'intérêt mieux définies.

Les tests effectués dans un environnement contrôlé avec un fond blanc ont confirmé l'efficacité de la segmentation basée sur l'espace colorimétrique HSV. Cette approche offre une détection rapide et précise des objets tout en facilitant l'extraction de leurs caractéristiques géométriques, telles que les contours et les boîtes englobantes.

Par ailleurs, l'intégration du modèle YOLOv8 représente une amélioration majeure de l'application. Contrairement aux méthodes classiques basées uniquement sur les caractéristiques des pixels, YOLOv8 réalise une détection sémantique capable d'identifier et de classer plusieurs catégories d'objets en temps réel. Les résultats obtenus ont démontré sa robustesse face aux variations d'éclairage, aux changements d'arrière-plan ainsi qu'à la présence simultanée de plusieurs objets dans la scène.

En conclusion, l'association des techniques classiques de traitement d'images (HSV, détection de contours et opérations morphologiques) avec les méthodes modernes d'apprentissage profond basées sur YOLOv8 a permis de développer une application performante, robuste et adaptée à diverses situations réelles de détection et de suivi d'objets.

Conclusion

Au cours de la réalisation de ce projet, le langage Python s'est révélé être un outil puissant et facile à utiliser pour le traitement d'images et le développement d'applications. Grâce à sa simplicité, à la richesse de ses bibliothèques telles que OpenCV et NumPy, ainsi qu'à son caractère open source, Python constitue une alternative très avantageuse à MATLAB pour ce type de projets. Il offre une grande flexibilité de développement tout en réduisant les coûts liés aux licences logicielles. Par ailleurs, son utilisation a permis une intégration simple des différentes techniques de traitement d'images et de vidéos au sein d'une même application, tout en facilitant les phases de développement, de test et d'amélioration du système.

L'intégration du traitement d'images et de vidéos dans les systèmes informatiques modernes constitue aujourd'hui un élément essentiel dans de nombreux domaines scientifiques, industriels et technologiques. Grâce aux progrès réalisés dans le domaine de la vision par ordinateur et du calcul parallèle, il est désormais possible d'effectuer des traitements complexes en un temps réduit tout en garantissant une meilleure précision dans l'analyse des données visuelles.

Les résultats obtenus au cours de cette étude ont montré que les techniques de traitement d'images, telles que la détection de contours, la segmentation, les opérations morphologiques et la détection d'objets, jouent un rôle fondamental dans l'extraction et l'interprétation des informations contenues dans les images et les vidéos. Leur mise en œuvre à travers le langage Python et ses différentes bibliothèques a permis de développer une application capable de réaliser plusieurs traitements en temps réel de manière efficace.

Par ailleurs, l'utilisation de la carte graphique NVIDIA Quadro M1200 a démontré l'intérêt du calcul parallèle dans l'accélération des opérations de traitement. En exploitant les capacités de calcul du processeur graphique (GPU), il a été possible d'améliorer considérablement les performances de l'application et de répondre plus efficacement aux contraintes liées au traitement des flux vidéo en temps réel. Contrairement au processeur central (CPU), qui est principalement conçu pour l'exécution séquentielle des tâches, le GPU dispose d'un grand nombre de cœurs capables d'effectuer simultanément des milliers d'opérations. Cette architecture parallèle lui confère des performances supérieures dans les applications de traitement d'images, de vidéos et de vision par ordinateur, où de nombreux calculs doivent être exécutés en parallèle.

Toutefois, malgré les performances obtenues, certaines limites demeurent liées à la complexité croissante des algorithmes de vision par ordinateur ainsi qu'aux ressources matérielles disponibles. L'amélioration continue des modèles de détection et l'utilisation de matériels plus performants peuvent contribuer à augmenter davantage la rapidité et la précision des traitements.

En définitive, cette étude a permis de mettre en évidence l'importance du traitement d'images et de vidéos associé au calcul parallèle comme solution efficace pour répondre aux exigences actuelles des applications intelligentes. Elle confirme également que l'association de Python avec les technologies GPU représente une approche prometteuse pour le

développement de systèmes de vision par ordinateur toujours plus rapides, précis et adaptés aux besoins futurs de l'industrie, de la sécurité et de la recherche scientifique.

Bibliographie

- [1] Zerairia Abdessalem, Drici Aymen, Lissage non linéaire et manipulations multi-échelles, Mémoire de Master, Département : Electronique, Faculté : Sciences de L'ingéniorat, Annaba – mokhtar Badji Université, 2020/2021, p02.
- [2] Dictionnaire de l'Académie française.
- [3] Résolution vidéo ou définition vidéo ? https://blog.eavs-groupe.com/wp-content/uploads/2025/01/eavs_2020-1536x372.jpg , 13/04/2026, 15 :05.
- [4] Ouchar Cherif Ali, Vers la reconnaissance de mouvement d'objet dans une image : détection d'objet en mouvement, mémoire de master, informatique et télécommunication, institut supérieur d'informatique et de gestion ISIG High Tech, Université Aube Nouvelle, 2012, p13.
- [5] Marc Chaumont, Représentation en objets vidéo pour un codage progressif et concurrentiel des séquences d'images, HAL Open Science, 13 Jan 2004, p59.
- [6] <https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/17044010/sequence-photographique> , 13/04/2026, 16 :33.
- [7] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Pearson, New York, 2018, Quatrième édition, pp62-63.
- [8] Fondement de Traitement D'images, Master 1 : Intelligence Artificiel et Multimédia (IAM), département d'Informatique, Université Batna 2, 2020/2021, p04.
- [9] Design graphique et applications multimédias, cours : Chapitre 02 : L'image numérique, elearning.univ-mila.dz, p27.
- [10] Gacemi Salim, Mimouni Mohammed, Conception et réalisation d'un système de Sécurité industrielle par caméra couleur, MÉMOIRE DE Master, Génie Industriel, Faculté de Technologie, Université Abou Bekr Belkaid–Tlemcen ,25 juin 2017, 2016-2017, p35.
- [11] Gacemi Salim, Mimouni Mohammed, Conception et réalisation d'un système de Sécurité industrielle par caméra couleur, MÉMOIRE DE Master, Génie Industriel, Faculté de Technologie, Université Abou Bekr Belkaid–Tlemcen ,25 juin 2017, 2016-2017, p35.
- [12] Design graphique et applications multimédias, cours : Chapitre 02 : L'image numérique, elearning.univ-mila.dz, p19.

[13] Quelle est la différence entre la résolution et la définition ?
<https://graphiste.com/blog/difference-resolution-definition/> , 13/04/2026, 22 :22.

[14] Katia Djaileb, Wahiba Oualli, Détection d'objet basé sur l'estimation de la saillance, Mémoire de master, département d'informatique, faculté du génie Electrique et d'informatique, Université Mouloud Mammeri De TIZI-OUZOU, 2017/2018, p09.

[15] Cours : Détection de Contours, <https://codefinity.com/fr/courses/v2/ef049f7b-ce21-45be-a9f2-5103360b0655/92b3adc5-534f-4d56-9624-4b18045f7be0/9ff87010-c46f-4143-b96e-8d829e2fefcb> , 14/04/2026, 15 :40.

[16] ZERAIRIA Abdessalem, DRICI Aymen, Lissage non linéaire et manipulations multi-échelles, Mémoire de Master, Département : Electronique, Faculté : Sciences de L'ingéniorat, Annaba – mokhtar Badji Université, 2020/2021, p04.

[17] Histogramme (imagerie numérique),
[https://fr.wikipedia.org/wiki/Histogramme_\(imagerie_num%C3%A9rique\)](https://fr.wikipedia.org/wiki/Histogramme_(imagerie_num%C3%A9rique)) , 14/04/2026, 11 :25.

[18] Création MATLAB

[19] Chapitre VI : Les images binaires,
http://bnazarian.free.fr/MyUploads/IN_GBM_06_BINAIRES.PDF , p01, 14/04/2026, 11 :47.

[20] kada Fouzia, zahaf Ahlem, Implémentation des algorithmes de traitement Des images et vidéos en utilisant la bibliothèque OpenCV, Mémoire de Master, Département d'Informatique, Faculté des Sciences, Université Abou Bakr Belkaid Tlemcen, 05 octobre 2016, 2015-2016, pp26-27.

[21] Création MATLAB

[22] <https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/26535345/image-en-couleur-composee> , 14/04/2026, 16 :26.

[23] Cours complet de sérigraphie, La couleur indexée, 14/04/2026, 14 :19, p01.

[24] Palette des couleurs (indexées), <https://docs.gimp.org/2.8/fr/gimp-indexed-palette-dialog.html> , 15/04/2026, 16 :29.

[25] Dijin Dominic , Guide du débutant pour comprendre le modèle de couleur HSV, <https://medium.com/@dijdomv01/a-beginners-guide-to-understand-the-color-models-rgb-and-hsv-244226e4b3e3> , 14/04/2026, 15 :18.

[26] M. Petremand, M. Louys, Ch. Collet, F. Flitti, Représentation couleur d'images multispectrales astronomiques Color display for multiwavelength astronomical images, traitement du signal _volume 21 _numéro spécial L'image numérique couleur, 15 juin 2004, p501.

[27] Mohammed Khamadja, Said Benierbah, Traitement d'images, Département d'électronique, Faculté des sciences de la technologie, Université des frères Mentouri (Constantine 1), p05.

[28] Prétraitement des images, <https://microblink.com/fr/resources/glossary/image-pre-processing/>, 16/04/2026, 18 :30.

[29] Filtrage numérique des images, Chapitre 4, http://elearning.univ-dbkm.dz/pluginfile.php/23660/mod_resource/content/1/Chapitre_4_Filtrage_des_images_M1_1-converti.pdf, 16/04/2026, 18 :51.

[30] Khenfer Ismail, Bettayeb Elhadj Ali, Filtrage Adaptatif 2-D pour la Restauration d'images perturbées par du Bruit Impulsionnel, Mémoire de Master, Département d'Electronique et de Communication, Faculté des Nouvelles Technologies de l'Information et de la Communication, Université Kasdi Merbah Ouargla, 2019/2020, p15.

[31] Ramzan Shahid Khan, Quelle est la différence entre un filtre linéaire et un filtre non linéaire en traitement d'images ?,11 février 2020, <https://www.mathworks.com/matlabcentral/answers/380674-difference-between-a-linear-and-nonlinear-filter-in-image-processing> - :~:text=Linear%20filtering%20is%20the%20filtering,with%20convolution%20or%20Fourier%20multiplication.18/04/2026, 18 :24.

[32] Traitement et analyse d'images, <https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html>, 18/04/2026, 14 :18.

[33] Soukayna Hadioui, Traitement d'images : Application du filtre gaussien, Institut Agronomique Vétérinaire Hasan II, <https://fr.scribd.com/document/840835133/Copy-of-Tp-contrôle-de-stabilité>, p02, 18/04/2026, 14 :44.

[34] Laurent Oudre, Gabriel Dauphin, Azeddine Beghdadi, Traitement des images numériques TP 4 : Filtrage, rehaussement de contours et segmentation, Master Ingénierie et Innovations en Images et Réseaux – 1 ère année, Institut Galilée, Université Paris 13, 2017-2018, p02.

[35] Ramzan Shahid Khan, Quelle est la différence entre un filtre linéaire et un filtre non linéaire en traitement d'images ?,11 février 2020, <https://www.mathworks.com/matlabcentral/answers/380674-difference-between-a-linear-and-nonlinear-filter-in-image-processing> -

Linear filtering is the filtering, with convolution or Fourier multiplication. 18/04/2026, 18:24.

[36] Malte Rehbein, Patrick Sahle, Torsten Schaban, *Codicology & Palaeography In The Digital Age*, Norderstedt, 2009, https://www.google.dz/books/edition/Codicology_and_Palaeography_in_the_Digit/IbmnazcfYrYC?hl=en&gbpv=1&dq=Binarization+traitement+d%27image&pg=PA200&printsec=frontcover, 18/04/2026, 19:30.

[37] Patrice Rey, Traitement Des Images Numérique, 2017, https://www.google.dz/books/edition/Traitement_des_images_numeriques_avec_c/eo9DDwAAQBAJ?hl=en&gbpv=1&dq=Les+op%C3%A9rateurs+de+la+morphologie+math%C3%A9matique&pg=PA665&printsec=frontcover, 18/04/2026, 20 :37.

[38] Opérations morphologiques dans OpenCV, <https://techskillguru.com/opencv/morphological-operations-in-opencv>, 19/04/2026, 15 :02.

[39] Sohail Ali, Différence entre dilatation et érosion, <https://www.naukri.com/code360/library/difference-between-dilation-and-erosion>, 19/04/2026, 15 :13.

[40] Centre L'aide MATLAB, <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html>, 19/04/2026, 15 :39.

[41] Qu'est-ce que la segmentation d'images ?, <https://www.ibm.com/fr-fr/think/topics/image-segmentation>
::~text=La%20segmentation%20d'image%20est%20une%20technique%20de,en%20segm
ents%20de%20forme%20sp%C3%A9cifique%2C%20la%20segme, 19/04/2026, 16 :37.

[42] Segmentation d'image, <https://www.ultralytics.com/fr/glossary/image-segmentation-principaux-types-de-segmentation>, 19/04/2026, 19 :35.

[43] Segmentation d'images : clé de l'intelligence artificielle visuelle?, <https://www.innovatiana.com/en/post/image-segmentation-in-ai>, 19/04/2026, 17 :27.

[44] Segmentation d'images : clé de l'intelligence artificielle visuelle?, <https://www.innovatiana.com/en/post/image-segmentation-in-ai>, 19/04/2026, 17 :27.

[45] Kada Fouzia, Zahaf Ahlem, Implémentation des algorithmes de traitement des images et vidéos en utilisant la bibliothèque OpenCV, Mémoire de Master, Département d'Informatique, Faculté des Sciences, Université Abou Bakr Belkaid Tlemcen, 05 octobre 2016, 2015-2016, pp50-51.

[46] Kada Fouzia, Zahaf Ahlem, Implémentation des algorithmes de traitement des images et vidéos en utilisant la bibliothèque OpenCV, Mémoire de Master, Département d'Informatique, Faculté des Sciences, Université Abou Bakr Belkaid Tlemcen, 05 octobre 2016, 2015-2016, pp50-51.

[47] Berramdane Abderraouf, Détection de la fatigue du conducteur avec Raspberry Pi, Mémoire de Master, Département d'Électronique et Télécommunications, Faculté des sciences et de la Technologie, Université 8Mai 1945 – Guelma, Juin 2022, pp19-20.

[48] Introduction à la détection d'objets en Computer Vision [2025], <https://www.innovatiana.com/fr/post/object-detection-our-guide>, 23/04/2026, 14 :53.

[49] <https://www.llamaindex.ai/glossary/what-is-you-only-look-once-yolo>, 23/04/2026, 15 :12.

[50] Abid Fatima, Brahim Boulares Lobna, Détection de la rétinopathie diabétique avec le deep learning, transfer learning, CNN, U-Net, Mémoire de Master, Département d'Électronique, Faculté de Technologie, Université SAAD DAHLAB de BLIDA, 2020/2021, pp23-24.

[51] <https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html>, 24/04/2026, 12 :01.

[52] Chouini Mohammed El Mounsif, Détection d'objets basé Faster R-CNN, Mémoire de Master, Département d'Informatique, Faculté des Mathématiques, d'Informatique et des Sciences de la matière, Université de 8 Mai 1945 – Guelma -, Octobre 2020, p43.

[53] Abderrahim Akarid, Samir El Adib, Naoufal Raissouni, Harmful Insects Detection Using Convolutional Neural Networks (Faster R-CNN), HAL open science, 27 Sep 2023, p04.

[54] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, Mask R-CNN, Computer vision (ICCV), IEEE, p2961.

[55] Alper Yilmaz, Omar Javed, Mubarak Shah, Object Tracking: A Survey, ACM Computing Surveys, Vol. 38, Article 13, December 2006, p02.

[56] Medouakh Saadia, Détection et suivi d'objets, Thèse de Doctorat, Département de Génie Electrique, Faculté des Sciences et de la Technologie, Université Mohamed Khider – Biskra, 23 Juin 2019, pp08-09.

[57] Mme M. Chibani, Chapitre 05 : Suivi de cible, Matière : Imagerie industrielle, Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie, Université Larbi Ben Mhidi-Oum El Bouaghi, p25.

[58] Mme M. Chibani, Chapitre 05 : Suivi de cible, Matière : Imagerie industrielle, Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie, Université Larbi Ben Mhidi-Oum El Bouaghi, p25.

[59] Mise à l'échelle de l'image, https://en.wikipedia.org/wiki/Image_scaling_-_text=In%20computer%20graphics%20and%20digital,as%20upscaling%20or%20resolution%20enhancement., 24/04/2026, 16 :46.

[60] Feddag Idir, Étude de méthodes de suivi d'objets : application en temps réel, Mémoire de Master, Département D'automatique, Faculté de Génie Electrique et de L'informatique, Université MOULOUD MAMMERI, TIZI-OUZOU, 22/10/2012, 2011/2012, p15.

[61] Mouvements de caméra, Caméra portée, Plans subjectifs, <https://upopi.ciclic.fr/vocabulaire/fr/definition/sceance-5>, 24/04/2026, 18 :37.

[62] Aurélie Bugeau, Détection et suivi d'objets en mouvement dans des scenes complexes, application a la surveillance des conducteurs, HAL open science, 4 Jan 2011, p32.

[63] Feddag Idir, Étude de méthodes de suivi d'objets : application en temps réel, Mémoire de Master, Département D'automatique, Faculté de Génie Electrique et de L'informatique, Université MOULOUD MAMMERI, TIZI-OUZOU, 22/10/2012, 2011/2012, p15.

[64] Mme M. Chibani, Chapitre 05 : Suivi de cible, Matière : Imagerie industrielle, Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie, Université Larbi Ben Mhidi-Oum El Bouaghi, p25.

[65] Alper Yilmaz, Omar Javed, Mubarak Shah, Object Tracking: A Survey, ACM Computing Surveys, Vol. 38, Article 13, December 2006, p02.

[66] Caractéristiques visuelles pour le suivi d'objets, fr.scribd.com, p01, 23/04/2026, 11 :23.

[67] Caractéristiques visuelles pour le suivi d'objets, fr.scribd.com, p01, 23/04/2026, 11 :23.

[68] Medouakh Saadia, Détection et suivi d'objets, Thèse de Doctorat, Département de Génie Electrique, Faculté des Sciences et de la Technologie, Université Mohamed Khider – Biskra, 23 Juin 2019, pp13-14

[L'élaboration du plan de ce mémoire s'est appuyée sur la consultation de travaux antérieurs réalisés dans la même spécialité et portant sur une thématique similaire. Le plan retenu a ensuite été adapté et réaménagé afin de répondre aux objectifs et aux exigences spécifiques de la présente recherche.]

Résumé :

Ce travail présente le développement d'une application de vision par ordinateur en temps réel sous Python, interfaçant le flux d'une IP Camera. Le système combine le filtrage colorimétrique traditionnel (HSV, Canny, morphologie) sur fond contrôlé et l'intelligence artificielle (YOLOv8) pour assurer un filtrage sémantique robuste face aux ombres. L'ensemble de l'architecture logicielle est accéléré matériellement par la carte graphique NVIDIA Quadro M1200 via l'API CUDA, garantissant un traitement fluide à 30 images par seconde.

Mots-clés : Python, IP camera, CUDA, NVIDIA Quadro M1200, YOLO V8.

Abstract:

This work presents the development of a real-time computer vision application in Python, interfacing with the stream from an IP camera. The system combines traditional color filtering (HSV, Canny, morphology) on a controlled background with artificial intelligence (YOLOv8) to ensure robust semantic filtering in the face of shadows. The entire software architecture is hardware-accelerated by the NVIDIA Quadro M1200 graphics card via the CUDA API, guaranteeing smooth processing at 30 frames per second.

Keywords: Computer Vision, YOLOv8, CUDA, NVIDIA Quadro M1200.

ملخص:

يُقدّم هذا العمل تطوير تطبيق رؤية حاسوبية آنية مكتوب بلغة بايثون، يتفاعل مع بث مباشر من كاميرا IP . يجمع النظام بين الترشيح التقليدي القائم على الألوان (HSV، كاني، مورفولوجيا) على خلفية مُتحكّم بها، والذكاء الاصطناعي (YOLOv8) لضمان ترشيح دلالي دقيق حتى في وجود الظلال. يتم تسريع بنية البرنامج بالكامل بواسطة بطاقة الرسومات NVIDIA Quadro M1200 عبر واجهة برمجة تطبيقات CUDA، مما يضمن معالجة سلسلة بمعدل 30 إطارًا في الثانية.

الكلمات المفتاحية: الرؤية الحاسوبية، الذكاء الاصطناعي، YOLOv8، CUDA.