الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي والـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT

كـلـيـــــة التكنولوجيا
FACULTE DE TECHNOLOGIE
قسم الإلكترونيك
Département d'électronique

Mémoire de Master

Domaine : SCIENCES ET TECHNOLOGIES
Filière : TELECOMMUNICATIONS
Option :

- **Hadjadj Khaoula :** système et télécommunication
- **Bigaa Naima:** Réseaux et télécommunication

SYSTEMES DES TELECOMMUNICATIONS

Par :
BIGAA NAIMA
HADJADJ KHAOULA

THEME

Technique de Flux Optique Appliqué à La Réalité Augmentée

Dr. ***REGUIEGUE Mourad***	*President*
Dr. ***BIRANE ABdelkader***	
Dr. ***REGGAB Mourad***	*Examinateur*
	Encadrant

Année Universitaire 2022/2023

Remerciements

Le premier merci est Allah, puis nos parents pour tous leurs efforts depuis notre naissance jusqu'à ces moments, vous êtes tout ce que nous aimons en Allah le plus d'amour.

Nous sommes heureux de remercier tous ceux qui ont conseillé, dirigé ou contribué à la préparation de cette recherche en m'envoyant les références et sources requises à n'importe quelle étape de ses étapes, et nous remercions tout particulièrement notre distingué professeur le Dr. REGGAB Mourad Pour notre soutien et nos conseils dans le conseil, la correction et le choix du titre et du sujet, et nos remerciements vont à l'administration du Collège Science et Technologie de l'Université de Laghouat, et au responsable du département électronique Dr. RAMDANI Saadi et pour son soutien aux efforts déployés par nos estimés professeurs de l'université.

Afin de fournir le meilleur environnement d'étude

Dans le meilleur des cas.

Dédicace

Au début, je remercie Dieu, qui m'a donné ces bénédictions.

Je remercie mes chers parents : Mohammed et Fatima, pour mon soutien continu tout au long de ma vie

Je le dédie à mes très chères sœurs : Ibtissam, Ikhlas, Ghania.

Et mon très cher frère : Abdelhafid,

et à tout ma famille sans oublier ma chère tante pour ses conseils

Je remercie ma binôme et ami qu'il connaissait à l'université HADJADJ Khaoula. Et après cela merci est connecté à tous mes enseignants qui ont été éduqués sur leurs mains à toutes les étapes de mes études jusqu'à ce que j'aie été honoré par mon discours devant votre honneur aujourd'hui.

BIG Naima

Dédicace

Au début, je remercie Dieu, qui m'a donné ces bénédictions.

Dédiez ce mémoire
À mon père Yassin
À ma mère Zahra

Pour tous leurs sacrifices consentis, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études. Qu'ils trouvent dans ce mémoire, le témoignage de me vive gratitude et de ma grande reconnaissance, pour l'énergie qu'ils ont su implanter en moi à tous les moments de mes études.

À mes chères sœurs Marwa, djIhan, Razane pour leurs encouragements continus et leur soutien moral,

A mon encadreur Dr. Reggab Mourad pour sa générosité et ses précieux conseils qu'il n'a cessé d'être généreuse avec moi.

À toute ma famille, à tous ceux que j'aime, en particulier Mohammed merci de vous être toujours là pour moi.

À tous mes amis, collègues, en particulier ma chère amie Naima Bigaa, une expression de mes sincères remerciements.

HADJADJ Khaoula

Résume :

Ce travail présente l'utilisation du langage Processing basé sur Java pour calculer le flux optique en tant que technique de réalité augmentée (RA) dans les applications de vision par ordinateur. Cette technique a été améliorée en incorporant certaines techniques de traitement vidéo telles que le suivi des couleurs, la pixelisation, le filtrage temporel passe-bas et la détection des mouvements. Les algorithmes de suivi des couleurs sont utilisés pour viser les objets d'intérêt en fonction de leurs couleurs, ce qui améliore la précision du calcul du flux optique. La pixelisation aide à réduire les calculs du flux optique tandis que le filtrage temporel passe-bas lisse les mouvements sur les images consécutives, réduisant le bruit et améliorant la stabilité vidéo. Les algorithmes de détection de mouvements permettent également de séparer les régions statiques et mobiles lors de l'analyse du flux optique. Le langage Processing basé sur Java fournit une plate-forme puissante et accessible pour développer des applications de vision par ordinateur interactives, ce qui le rend adapté à la création de systèmes AR efficaces.

Mots clés : flux optique, vision par ordinateur, réalité augmentée, suivi des couleurs, pixellisation, filtrage temporel passe-bas, détection de mouvements, Le langage Processing.

خلاصة:

يقدم هذا العمل استعمال لغة Processing المبنية على Java لحساب التدفق البصري كتقنية للواقع المعزز (AR) في تطبيقات رؤية الحاسوب computer vision. تم تحسين هذه التقنية من خلال إدماج بعض تقنيات معالجة الفيديو مثل تتبع الألوان، والتشكيل البكسلي (البكسلة)، والفلترة الزمنية منخفضة الترددات، وتتبع الحركة. تُستخدم خوارزميات تتبع الألوان لتتبع الأشياء ذات الأهمية وفق لونها، مما يحسن دقة حساب التدفق البصري. تساعد تقنية البكسلة على تقليل حسابات التدفق البصري بينما تعمل الفلترة الزمنية منخفضة الترددات على تنعيم الحركة عبر الصور المتتالية، مما يقلل الضوضاء ويحسن استقرار الفيديو. كما تتيح خوارزميات تتبع الحركة فصل المناطق الثابتة والمتحركة أثناء تحليل التدفق البصري. توفر لغة Processing المبنية على Java نظامًا أساسيًا قويًا ومتاحا لتطوير تطبيقات رؤية الحاسوب التفاعلية، مما يجعلها مناسبة لإنشاء أنظمة AR فعالة.

الكلمات المفتاحية: التدفق البصري، رؤية الحاسوب، الواقع المعزز، تتبع اللون، البكسلة، الفلترة الزمنية منخفضة الترددات، تتبع الحركة، لغة Processing.

Abstract:

This work presents a Java-based Processing language approach for implementing optical flow as an augmented reality (AR) technique in computer vision applications. The technique is enhanced through the integration of video processing methods, including color tracking, pixellation, temporal low-pass filtering, and motion detection. Color tracking algorithms are used to track objects of interest, improving the accuracy of optical flow calculations. Pixellation helps to reduce the optical flow calculations. Temporal low-pass filtering smooths motion vectors across frames, reducing noise and improving stability.

Motion detection algorithms enable the separation of static and moving regions during optical flow analysis. The Java-based Processing language provides a powerful and accessible platform for developing interactive computer vision applications, making it suitable for creating efficient AR systems.

Keywords: optical flow, computer vision, augmented reality, color tracking, pixellation, temporal low-pass filtering, motion detection, Processing language.

Sommaire

Liste des Tableaux

Liste des Figures

Introduction Générale

Introduction Générale

La réalité augmentée (RA) est une technologie révolutionnaire qui intègre des informations numériques et des objets virtuels dans l'environnement réel, améliorant la perception et l'interaction de l'utilisateur avec son environnement. Un aspect crucial de la RA est la capacité à enregistrer de manière précise le contenu virtuel avec le monde réel. Le flux optique, une technique de vision par ordinateur, joue un rôle significatif dans cette alignement en fournissant une estimation de mouvement précise et un suivi des objets dans le champ de vision de la caméra.

L'application du flux optique en RA implique l'analyse des variations d'intensité des pixels entre les images consécutives d'une séquence vidéo pour déduire le mouvement des objets dans la scène. En utilisant le concept de flux optique, les systèmes de RA peuvent identifier le déplacement et la vitesse des objets en temps réel, permettant une intégration fluide du contenu virtuel dans l'environnement de l'utilisateur.

L'utilisation d'algorithmes de flux optique en RA présente plusieurs avantages clés. Tout d'abord, cela permet de détecter le mouvement relatif entre la caméra et la scène, facilitant une estimation de position 3D précise et le suivi des objets. Cette capacité est essentielle pour maintenir l'enregistrement entre les objets virtuels et l'environnement réel, assurant une expérience de RA cohérente et immersive pour l'utilisateur.

Deuxièmement, le flux optique permet le suivi en temps réel des objets en mouvement dans le champ de vision de la caméra. En estimant les vecteurs de mouvement des pixels individuels ou des régions, les systèmes de RA peuvent suivre et superposer avec précision le contenu virtuel sur des éléments dynamiques tels que les personnes, les véhicules ou les animaux. Cette capacité de suivi d'objets dynamiques élargit les possibilités des applications de RA, permettant des expériences interactives et améliorant le réalisme des ajouts virtuels.
De plus, les techniques basées sur le flux optique peuvent aider à l'estimation de la profondeur, un autre composant essentiel de la RA. En analysant le mouvement apparent des objets à différentes distances de la caméra, des indices de profondeur peuvent être déduits, permettant la création d'objets virtuels réalistes qui interagissent de manière convaincante avec l'environnement physique.

Dans ce mémoire, nous avons pour objectif d'explorer et d'évaluer l'utilisation du flux optique dans les systèmes de réalité augmentée. Nous examinerons divers algorithmes de flux optique, y compris des méthodes traditionnelles telles que Lucas-Kanade et Horn-Schunck.

Nous analyserons leurs points forts, leurs limites et leurs exigences en termes de calcul dans le contexte des applications de RA.

Dans notre étude nous avons couvert quatre parties, nous mentionnerons l'objectif pour chaque chapitre :

Le premier chapitre on a essayé de donner un aperçu sur la réalité augmentée, en la définissant, et son historique, présentant ses caractéristiques et quelques domaines d'applications (médical, militaire, jeu…). Puis nous résumons tout cela en le comparant à la réalité virtuelle.

Le deuxième chapitre nous présenterons quelques notions de base du domaine de traitement d'image et vidéo tels que : la définition d'mage et la vidéo, les caractéristiques de l'image, le filtrage de l'image et vidéo.

Dans le troisième chapitre, nous définirons le flux optique et la détection de mouvement, ensuite le développement des méthodes décrites précédemment à savoir Horn et Shunck, Lucas et Kanade.

Dans le quatrième chapitre, nous appliquerons quelques programmes grâce auxquels nous pourrons calculer le flux optique.

Chapitre 1 :
Réalité Augmentée

Chapitre 1 : Réalité Augmentée

I. Introduction

La RA a pour but d'enrichir la perception et la connaissance d'un environnement réel par l'ajout d'informations numériques le concernant. Ces informations sont le plus souvent visuelles. Parfois sonores, rarement haptiques. Dans la plupart des applications de RA, l'utilisateur visualise des images de synthèse par l'intermédiaire de lunettes, de casques ou vidéoprojecteurs, voire de tablettes/ Smartphones. [1]

I.1. Définition

La réalité augmentée (AR) est une vue en direct, directe ou indirecte, d'un environnement physique du monde réel dont les éléments sont augmentés par des entrées sensorielles générées par ordinateur telles que le son, la vidéo, les graphiques ou les données GPS. Il est lié à un concept plus général appelé réalité médiatisée, dans lequel une vision de la réalité est modifiée (peut-être même diminuée plutôt qu'augmentée) par un ordinateur. En conséquence, la technologie fonctionne en améliorant la perception actuelle de la réalité. En revanche, la réalité virtuelle remplace le monde réel par un monde simulé.

L'augmentation est classiquement en temps réel et dans un contexte sémantique Avec des éléments environnementaux, tels que les résultats sportifs à la télévision pendant un match.

La recherche explore l'application de l'imagerie générée par ordinateur dans les flux vidéo en direct comme moyen d'améliorer la perception du monde réel. La technologie AR comprend des écrans montés sur la tête et des écrans rétiniens virtuels à des fins de visualisation, ainsi que la construction d'environnements contrôlés contenant des capteurs et des actionneurs. [2]

I.2. L'histoire derrière la Réalité Augmentée

Les débuts de la Réalité Augmentée, telle que nous la définissons, remontent au travail de Sutherland dans les années 1960, qui a utilisé un affichage monté sur la tête (HMD) transparent pour présenter des graphiques 3D. Cependant, ce n'est qu'au cours de la dernière décennie qu'il y a eu suffisamment de travail pour faire référence à la Réalité Augmentée en tant que domaine de recherche. En 1997, Azuma a publié une enquête qui définissait le domaine, décrivait de nombreux problèmes et résumait les développements jusqu'à ce point. Depuis lors, la croissance et les progrès de la Réalité Augmentée ont été remarquables.

À la fin des années 1990, plusieurs conférences sur la Réalité Augmentée ont commencé, notamment l'atelier et le symposium internationaux sur la réalité augmentée, le symposium international sur la réalité mixte et l'atelier sur la conception d'environnements de réalité augmentée. Certaines organisations bien financées se sont formées et se sont concentrées sur la Réalité Augmentée, notamment le Mixed Reality Systems Lab au Japon et le consortium Arvika en Allemagne.

En 2001, MR Lab a terminé ses recherches pilotes et les symposiums ont été réunis dans le Symposium international sur la réalité mixte et augmentée (ISMAR), qui est devenu le principal symposium de l'industrie et de la recherche pour échanger des problèmes et des solutions. [3]

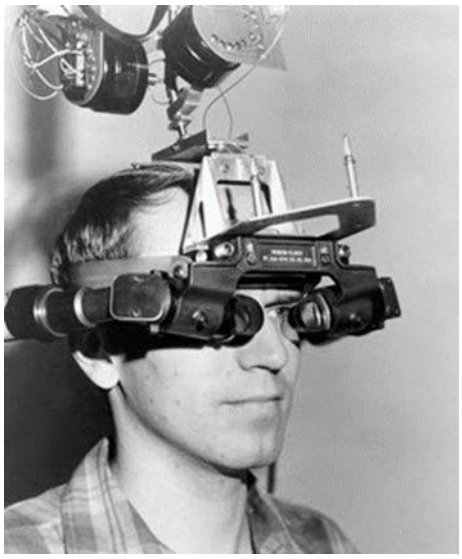


Figure 1 L'épée de Damoclès était le surnom du premier visiocasque au monde, construit en 1968

I.3. Comment fonctionne la vision ?

Le système Réalité Augmentée suit la position et l'orientation de la tête de l'utilisateur afin que le matériau superposé puisse être aligné avec la vision du monde de l'utilisateur. Grâce à ce processus, connu sous le nom d'enregistrement, un logiciel graphique peut placer une image tridimensionnelle d'une tasse de thé, par exemple sur une vraie soucoupe et maintenir la tasse virtuelle fixe dans cette position lorsque l'utilisateur se déplace dans la pièce. Les systèmes Réalité Augmentée utilisent certaines des mêmes technologies matérielles utilisées dans la recherche sur la réalité virtuelle, mais il existe des différences cruciales : alors que la réalité virtuelle vise impétueusement à remplacer le monde réel, la réalité augmentée le complète respectueusement.

Un système Réalité Augmentée

- Combine des objets réels et virtuels dans un environnement réel.
- Enregistre (aligne) les objets réels et virtuels les uns avec les autres.
- S'exécute de manière interactive en trois dimensions et en temps réel.

À l'aide d'une application mobile, l'appareil photo d'un téléphone mobile identifie et interprète un marqueur, souvent une image de code-barres en noir et blanc. Le logiciel analyse le marqueur et crée une superposition d'image virtuelle sur l'écran du téléphone mobile, liée à la position de la caméra. Cela signifie que l'application fonctionne avec l'appareil photo pour interpréter les angles et la distance entre le téléphone portable et le marqueur.

En raison du nombre de calculs qu'un téléphone doit effectuer pour rendre l'image ou le modèle sur le marqueur, souvent seuls les téléphones intelligents sont capables de prendre en charge la réalité augmentée avec succès. [2]

I.4. Affichage

De toutes les modalités de l'entrée sensorielle humaine, la vue, l'ouïe et/ou le toucher sont actuellement les sens que les systèmes Réalité Augmentée appliquent couramment.

I.4.1. Affichage visuel

Il existe des moyens simples de présenter visuellement une réalité augmentée. Le plus proche de la réalité virtuelle est la vidéo transparente, où l'environnement virtuel est remplacé par un flux vidéo de la réalité et la Réalité Augmentée est superposé aux images numérisées.

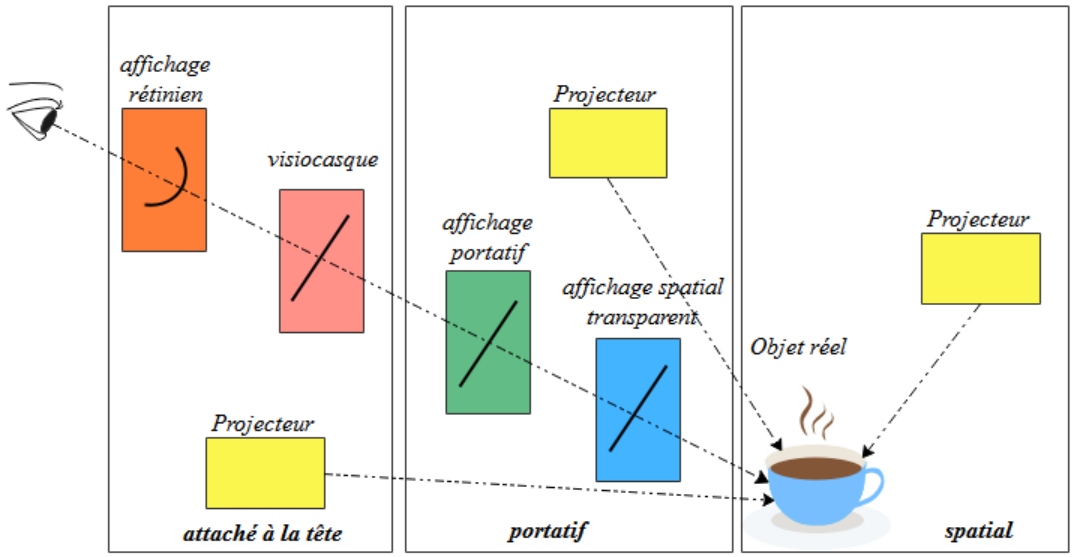


Figure 2 Types d'affichage visuel

Quatre grandes classes de Réalité Augmentée peuvent être distinguées par leur type d'affichage :

(a) Affichage à transparence vidéo,

(b) Affichage à transparence optique,

(c) Affichage du système rétinien virtuels,

(d) Réalité Augmentée basée sur moniteur

(e) Réalité Augmentée basé sur projecteur.

(a) Transparence vidéo à travers le visiocasque

La Réalité Augmentée basée sur la transparence vidéo utilise un visiocasque opaque pour afficher la vidéo fusionnée du VE et la vue des caméras sur le visiocasque. Cette approche est un peu plus complexe que la Réalité Augmentée optique transparente, nécessitant un emplacement approprié des caméras. Cependant, la composition vidéo du monde réel et virtuel est beaucoup plus facile. Il existe une variété de solutions disponibles, y compris l'incrustation en chrominance et la cartographie de profondeur.

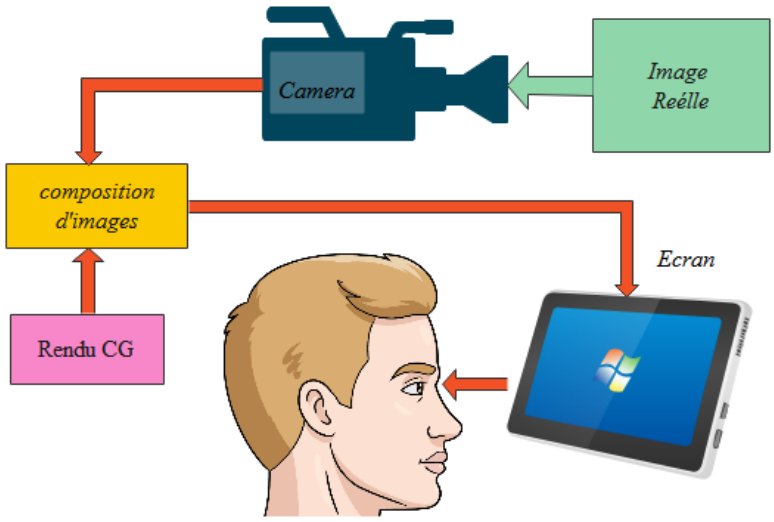


Figure 3 Transparence vidéo

(b) Transparence optique

Non seulement ces écrans laissent la résolution réelle intacte, mais ils ont également l'avantage d'être moins chers, plus sûrs et sans parallaxe (pas de décalage oculaire dû au positionnement de la caméra).

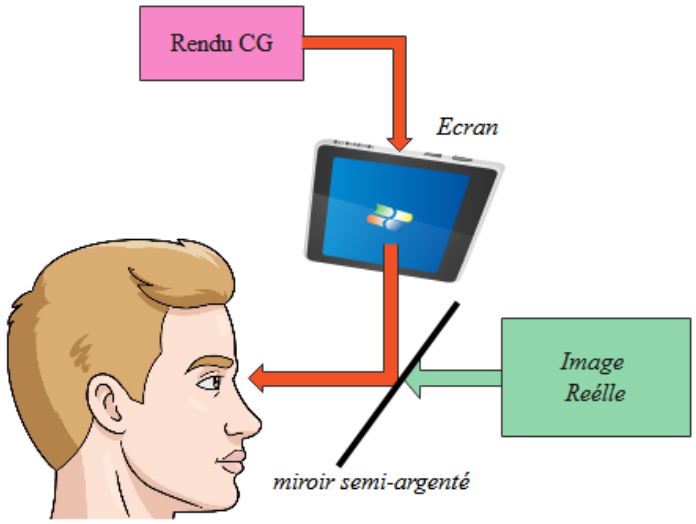


Figure 4 transparence optique

Les techniques optiques sont plus sûres car les utilisateurs peuvent toujours voir en cas de panne de courant, ce qui en fait une technique idéale à des fins militaires et médicales. Cependant, d'autres périphériques d'entrée tels que des caméras sont nécessaires pour l'interaction et l'enregistrement. De plus, la combinaison holographique des objets virtuels à travers des miroirs et des lentilles transparentes crée des inconvénients car elle réduit la luminosité et le contraste des images et de la perception du monde réel, ce qui rend cette technique moins adaptée à une utilisation en extérieur.

(c) Affichage du système rétinien virtuel

Les écrans rétiniens virtuels ou les écrans à balayage rétinien (RSD) résolvent les problèmes de faible luminosité et de faible champ de vision dans les écrans transparents optiques (portés sur la tête).

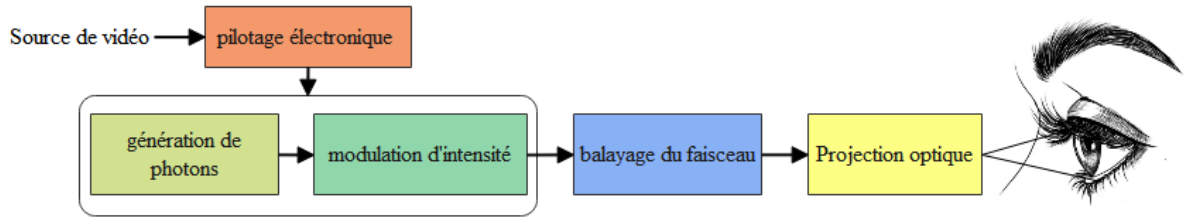


Figure 5 Affichage du système rétinien virtuel

Un laser de faible puissance dessine une image virtuelle directement sur la rétine, ce qui donne une luminosité élevée et un large champ de vision. La qualité RSD n'est pas limitée par la taille des pixels mais uniquement par la diffraction et les aberrations dans la source lumineuse, ce qui permet également des résolutions (très) élevées. Dans son cours de troisième cycle, Fiambolis (1999) fournit des informations supplémentaires sur la technologie

RSD. Avec leur faible consommation d'énergie, ces écrans sont parfaitement adaptés à une utilisation prolongée en extérieur.

(d) Affichage basé sur un moniteur

Le terme basé sur un moniteur (non immersif) ou "fenêtre sur le monde" (WoW), Réalité Augmentée pour désigner les systèmes d'affichage où les images générées par ordinateur sont superposées de manière analogique ou numérique sur des images vidéo en direct ou stockées.

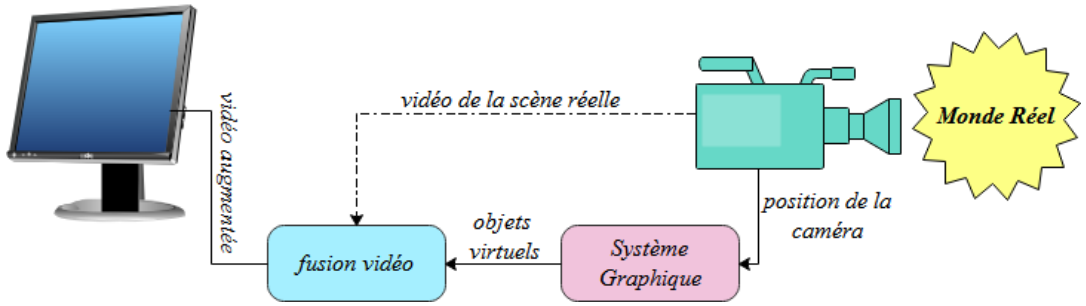


Figure 6 Affichage basé sur un moniteur

Bien que la technologie pour y parvenir soit bien connue depuis un certain temps, notamment au moyen de l'incrustation chroma, un grand nombre d'applications utiles se présentent lorsque ce concept est mis en œuvre de manière stéréoscopique.

(e) Affichage Réalité Augmentée basé sur projecteur

Ces affichages ont l'avantage de ne pas nécessiter de lunettes spéciales, permettant ainsi aux yeux de l'utilisateur de se concentrer, et ils peuvent couvrir de grandes surfaces pour un large champ de vision. Les surfaces de projection peuvent aller de murs plats et de couleur unie à des modèles complexes. Cependant, comme pour les écrans transparents optiques, d'autres périphériques d'entrée sont nécessaires pour l'interaction (indirecte). De plus, les projecteurs doivent être calibrés chaque fois que l'environnement ou la distance à la surface de projection change (crucial dans les configurations mobiles). Heureusement, l'étalonnage peut être automatisé à l'aide de caméras, par exemple un environnement à cave virtuel automatique multiparois (CAVE) avec des surfaces irrégulières.

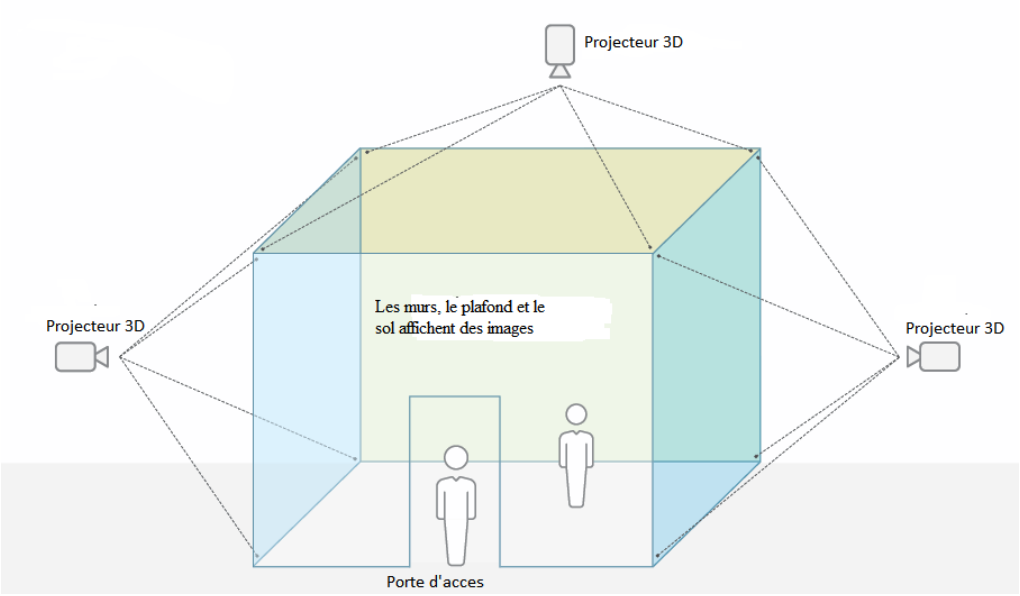


Figure 7 Affichage RA basé sur projecteur

I.4.2. Affichage SONORE

L'application d'affichage sonore en réalité augmentée est limitée aux écouteurs et haut-parleurs mono (0 dimension), stéréo (1 dimension) ou sur round (2 dimensions) explicites. Le véritable affichage sonore 3D se retrouve actuellement dans des simulations plus immersives d'environnements virtuels et de virtualité augmentée ou encore à des stades expérimentaux.

I.4.3. Affichage HAPTIQUE

Les écrans Haptiques font référence aux interfaces fournissant un retour Haptique, généralement en stimulant les récepteurs somatiques pour générer une sensation de toucher. Les affichages Haptiques peuvent être classés en fonction du type de stimuli/sortie qu'ils génèrent et, en conséquence, du type de récepteurs sensoriels qui sont stimulés. L'affichage sensoriel concerne le rendu des forces. Ces derniers sont générés sur la base de modèles informatiques, d'interactions à distance, d'enregistrements ou d'approches basées sur les données. Les affichages de cette catégorie peuvent être subdivisés en fonction du type d'actionnement (c'est-à-dire matériel) utilisé pour la génération de force [3]

Figure 8 Affichage HAPTIQUE

I.5. Réalité augmentée contre réalité virtuelle

Les exigences globales de l'Réalité Augmentée peuvent être résumées en les comparants aux exigences des environnements virtuels, pour les trois sous-systèmes de base dont ils ont besoin sont les suivants :

I.5.1. Générateur de scènes

Le rendu n'est pas actuellement l'un des problèmes majeurs de la réalité augmentée. Les systèmes Réalité Virtuelle ont des exigences beaucoup plus élevées en matière d'images réalistes car ils remplacent complètement le monde réel par l'environnement virtuel. En Réalité Augmentée, les images virtuelles ne font que compléter le monde réel. Par conséquent, moins d'objets virtuels doivent être dessinés et ils ne doivent pas nécessairement être rendus de manière réaliste afin de servir les objectifs de l'application.

I.5.2. Périphériques d'affichage

Les périphériques d'affichage utilisés dans la réalité augmentée peuvent avoir des exigences moins strictes que celles exigées par les systèmes de réalité virtuelle, encore une fois parce que la réalité augmentée ne remplace pas le monde réel. Par exemple, les écrans monochromes peuvent convenir à certaines applications Réalité Augmentée, alors que pratiquement tous les systèmes Réalité Virtuelle utilisent aujourd'hui la couleur. Les HMD transparents optiques avec un petit champ de vision peuvent être satisfaisants car l'utilisateur peut toujours voir le monde réel avec sa vision périphérique ; le HMD transparent ne coupe pas le champ de vision normal de l'utilisateur. De plus, la résolution du moniteur dans un HMD transparent optique peut être inférieure à ce qu'un utilisateur tolérerait dans une

application Réalité Virtuelle, car le HMD transparent optique ne réduit pas la résolution de l'environnement réel.

I.5.3. Suivi et détection

Alors que dans les deux cas précédents, la Réalité Augmentée avait des besoins inférieurs à la Réalité Virtuelle, ce n'est pas le cas pour le suivi et la détection. Dans ce domaine, les exigences pour Réalité Augmentée sont beaucoup plus strictes que celles pour les systèmes Réalité Virtuelle. Une des principales raisons à cela est le problème d'enregistrement. [6]

SOUS-SYSTÈMES DE BASE	REALITE VIRTUELLE	REALITE AUGMENTE
GÉNÉRATEUR DE SCÈNE	PLUS AVANCÉ	MOINS AVANCÉ
DISPOSITIFS D'AFFICHAGE	QUALITÉ HAUTE	QUALITÉ BASSE
SUIVI ET DÉTECTION	MOINS AVANCÉ	PLUS AVANCÉ

Table 1 La comparaison entre la réalité augmentée et la réalité virtuelle

I.6. Comparaison des exigences de la réalité augmentée et de la réalité virtuelle

La réalité virtuelle diffère de la Réalité Augmentée dans quelques domaines clés. Premièrement, la réalité virtuelle cherche non seulement à améliorer la réalité, mais à recréer la réalité dans un environnement immersif. Pour ce faire, les utilisateurs sont souvent séparés du monde réel par des casques (souvent appelés HMD).

Les HMD bloquent complètement l'environnement des utilisateurs, les isolant du monde extérieur. Une telle technologie est certes immersive, mais elle est aussi quelque peu limitée dans ses applications. Certains types d'entraînement pourraient être améliorés avec la réalité virtuelle, et les joueurs salivent à l'idée de pouvoir réellement habiter leurs jeux. Actuellement, les industries du jeu et du divertissement ont rencontré le plus de succès avec ce média qui se développe rapidement.

Cette distinction entre Réalité Augmentée et Réalité Virtuelle est basée sur l'état actuel des deux technologies. Cependant, l'avenir apportera des affichages montés sur tête capables à la fois de la Réalité Augmentée et de la Réalité Virtuelle. Il est facile d'imaginer un HMD qui permet aux utilisateurs de voir à travers le monde extérieur en mode Réalité Augmentée puis devient opaque et passe en Réalité Virtuelle. La Réalité Augmentée et la Réalité Virtuelle restent des domaines distincts remplissant des fonctions différentes ; La réalité virtuelle cherche à créer un monde distinct de la réalité, tandis que la réalité augmentée cherche à augmenter l'expérience d'un utilisateur dans le monde réel et à améliorer la réalité. [4]

I.7. Avantages et inconvénients de la technologie

I.7.1. Avantages

(a) Immersion multisensorielle

La réalité augmentée conduit à une immersion sensorielle sur des informations ou des connaissances en augmentant les perceptions humaines avec des objets ou des matériaux 3D.

(b) Interface de transition

Réalité Augmentée fournit une interface de transition transparente entre un monde réel et un monde virtuel.

(c) Interface utilisateur tangible

Réalité Augmentée offre une interface utilisateur tangible avec laquelle des objets numériques ou des informations peuvent être touchés dans Réalité Augmentée.

(d) Concordance avec les appareils mobiles

À mesure que les appareils mobiles et leurs applications progressent, les utilisateurs mobiles peuvent profiter de plus de gestes et de toucher.

(e) Faible consommation d'énergie

Seulement six diodes sont nécessaires et quelques watts pour fournir leurs images aux yeux des utilisateurs.

(f) Réduction des coûts

Le coût actuel des systèmes de projection rétinienne est élevé. Néanmoins, il n'y a pas de problèmes de fabrication difficiles à surmonter dans la production de masse et les composants à faible coût, donc peu coûteux deviendront bientôt disponibles.

I.7.2. Inconvénients

(a) Portabilité et utilisation en extérieur

La plupart des systèmes Réalité Augmentée mobiles mentionnés dans cette enquête sont encombrants, nécessitant un sac à dos lourd pour transporter le PC, les capteurs, l'écran, les batteries et tout le reste.

(b) Suivi et (auto) calibrage

Le suivi dans des environnements non préparés reste un défi, mais les approches hybrides deviennent suffisamment petites pour être ajoutées aux téléphones mobiles ou aux assistants numériques personnels (PDA). Le calibrage de ces appareils est encore compliqué et étendu, mais cela peut être résolus grâce à des approches sans calibrage et à calibrage automatique.

(c) Latence (Temps de réponse)

La latence du système peut également être planifiée pour réduire les erreurs grâce à une conception minutieuse du système, et les images pré-rendues peuvent être décalées au dernier instant pour compenser les mouvements de panoramique et d'inclinaison.

(d) Fatigue et fatigue oculaire

Comme le problème de parallaxe, les écrans binoculaires causent beaucoup plus d'inconfort que les écrans monoculaires ou binoculaires, à la fois en termes de fatigue oculaire et de fatigue. Une trop grande dépendance à l'égard des informations numériques peut entraîner une diminution de la mémoire de travail dans le cerveau, ce qui à son tour entrave le développement des fonctions cérébrales.

(e) Acceptation sociale

Amener les gens à utiliser la Réalité Augmentée peut être plus difficile que prévu, et de nombreux facteurs jouent un rôle dans l'acceptation sociale de la Réalité Augmentée, allant de l'apparence discrète à la mode (gants, casques, etc.) aux problèmes de confidentialité. [4]

I.8. Applications de la réalité augmentée

Au fil des ans, les chercheurs et les développeurs trouvent de plus en plus de domaines qui pourraient bénéficier de l'augmentation. Les premiers systèmes se sont concentrés sur les applications militaires, industrielles et médicales, mais les systèmes de réalité augmentée à usage commercial et de divertissement sont apparus peu après.

I.8.1. Système d'informations personnelles

La Réalité Augmentée peut servir d'interface utilisateur avancée, immédiate et plus naturelle pour l'informatique portable et mobile à usage personnel et quotidien. Par exemple, la réalité augmentée pourrait intégrer la communication par téléphone et par e-mail avec des superpositions sensibles au contexte, gérer les informations personnelles liées à des emplacements ou des personnes spécifiques, fournir des conseils de navigation et fournir une interface de contrôle unifiée pour tous les types d'appareils dans et autour de la maison. Une telle plateforme offre également aux agences de marketing direct de nombreuses opportunités d'offrir des coupons aux passants piétons, de placer des panneaux d'affichage virtuels, de montrer des prototypes virtuels, etc.

Avec toutes ces utilisations différentes, les plateformes de réalité augmentée devraient de préférence proposer un filtre pour gérer le contenu qu'elles affichent

I.8.2. La navigation

La navigation dans des environnements préparés a fait ses preuves depuis un certain temps. Une NaviCam a été introduite pour une utilisation en intérieur qui a augmenté un flux vidéo à partir d'une caméra portative à l'aide de marqueurs repères pour le suivi de position. Narzt et al. a discuté des paradigmes de navigation pour les piétons (extérieurs) et les voitures qui superposent les itinéraires, les sorties d'autoroute, les voitures à suivre, les dangers, les prix du carburant, etc... .

I.8.3. Divertissement

Une forme simple de réalité augmentée est utilisée dans le secteur du divertissement et de l'information depuis un certain temps. Chaque fois que vous regardez le bulletin météo du soir, l'orateur reste debout devant des cartes météo changeantes. Dans le studio, le journaliste se tient en fait devant un écran bleu. Cette image réelle est complétée par des cartes générées par ordinateur à l'aide d'une technique appelée chroma-keying. Un autre domaine de divertissement où la Réalité Augmentée est appliquée est le développement de jeux.

I.8.4. Entrainement militaire

L'armée utilise des écrans dans les cockpits qui présentent des informations au pilote sur le pare-brise du cockpit ou la visière du casque de vol. Il s'agit d'une forme d'affichage en réalité augmentée. En équipant le personnel militaire d'écrans de visière montés sur casque ou d'un télémètre à usage spécial, les activités des autres unités participant à l'exercice peuvent être imagées.

I.8.5. Opérations robotiques

Un opérateur télérobotique utilise une image visuelle de l'espace de travail distant pour guider le robot. L'annotation de la vue serait utile car c'est le cas lorsque la scène est devant l'opérateur. En outre, l'ajout de dessins des structures dans la vue peut faciliter la visualisation de la géométrie 3D distante.

I.8.6. Entretien

Lorsque le technicien de maintenance s'approche d'un équipement nouveau ou inconnu au lieu d'ouvrir plusieurs manuels de réparation, il peut afficher un écran de réalité augmentée. Dans cet affichage, l'image de l'équipement serait complétée par des annotations et des informations pertinentes à la réparation. Par exemple, l'emplacement des attaches et du matériel de fixation qui doivent être retirés serait mis en surbrillance.[4]

I.9. Problèmes de la réalité augmentée

Voici une liste de problèmes théoriques liés aux affichages stéréoscopiques en général, avec un intérêt particulier pour les affichages de réalité augmentée et de réalité mixte. Autrement dit, certains des problèmes s'appliquent à tout type d'affichage stéréoscopique, tandis que d'autres ne concernent que les situations de réalité mixte ou virtuelle, ou uniquement les systèmes qui intègrent la dépendance du point de vue via le suivi de la tête.

Aucune tentative n'a été faite ici pour juger les divers problèmes en termes d'importance, de gravité ou de priorité. Les problèmes discutés ont été regroupés en trois catégories : « les erreurs de mise en œuvre », qui peuvent être résolues par une application prudente de la technologie actuellement disponible ; les "limitations technologiques actuelles", qui deviendront vraisemblablement moins importantes à mesure que l'état de l'art s'améliorera ; et les « problèmes difficiles », qui nécessitent de nouveaux développements fondamentaux de la technologie pour être résolus.

• Problèmes de performances (problèmes difficiles)

Le traitement en temps réel des images peut être un défi et peut souvent ralentir les systèmes de réalité augmentée.

• Problèmes d'interaction (Limites technologiques actuelles)

Les utilisateurs dans un environnement mixte à cause de la réalité augmentée ont des difficultés à interagir normalement avec l'environnement.

• Problèmes d'alignement (erreurs de mise en œuvre)

Les personnes travaillant dans une réalité augmentée sont plus sensibles aux erreurs d'alignement.

Un bon calibrage et un alignement avec le cadre de référence du monde sont cruciaux.

I.10. Limites

I.10.1. Limites technologiques

Bien qu'il y ait beaucoup de progrès dans les technologies habilitantes de base, elles empêchent encore principalement le déploiement de nombreuses applications Réalité Augmentée. Les écrans, les trackers et les systèmes Réalité Augmentée en général doivent devenir plus précis, plus légers, moins chers et moins énergivores. Étant donné que l'utilisateur doit porter le PC, les capteurs, l'écran, les piles et tout ce qui est nécessaire, le résultat final est un sac à dos lourd. Les ordinateurs portables d'aujourd'hui n'ont qu'un seul processeur, ce qui limite la quantité de suivi visuel et hybride que nous pouvons effectuer.

I.10.2. Limitation de l'interface utilisateur

Nous avons besoin de mieux comprendre comment afficher les données à un utilisateur et comment l'utilisateur doit interagir avec les données. La Réalité Augmentée introduit de nombreuses tâches de haut niveau, telles que la nécessité d'identifier quelles informations doivent être fournies, quelle est la représentation appropriée pour ces données et

comment l'utilisateur doit effectuer des requêtes et des rapports. Des travaux récents suggèrent que la création et la présentation de performances et de structures narratives peuvent conduire à une expérience de réalité augmentée plus réaliste et plus riche.

I.10.3. Acceptation sociale

Le dernier défi est l'acceptation sociale. Avec un système doté d'un matériel idéal et d'une interface intuitive, comment la réalité augmentée peut devenir une partie intégrante de la vie quotidienne d'un utilisateur, tout comme un téléphone portable ou un assistant numérique personnel. À travers les films et la télévision, de nombreuses personnes sont familiarisées avec les images de réalité augmentée simulées. Cependant, persuader un utilisateur de porter un système implique de résoudre un certain nombre de problèmes. Celles-ci vont de la mode aux problèmes de confidentialité. À ce jour, peu d'attention a été accordée à ces questions fondamentales. Cependant, ceux-ci doivent être résolus avant que la Réalité Augmentée ne soit largement acceptée.

I.11. Améliorations futures

I.11.1. La réalité augmentée extérieure

A beaucoup de potentiel, mais elle n'est tout simplement pas encore pratique. Il ne faudra que quelques années avant que l'informatique mobile et sans fil ne soit suffisamment rapide pour produire des images synthétiques satisfaisantes. Un aspect plus difficile d'une réalité augmentée extérieur est le suivi de l'emplacement et de l'orientation de l'utilisateur. Le système de positionnement global (GPS) a une granularité inférieure à un mètre. Cela conviendra en fait pour certaines applications telles que les indices auditifs, car l'oreille humaine n'est pas si sensible à la direction, et peut-être pour les méta-informations visuelles sur l'environnement. Mais le GPS ne suffira pas pour les applications visuelles où les objets virtuels générés par ordinateur doivent se fondre dans la vue de manière transparente à leur place correcte par rapport à l'environnement.

I.11.2. Les algorithmes et les logiciels

Passeront par plusieurs itérations pour évoluer vers des solutions plus sophistiquées. La récupération de l'environnement, de l'éclairage et de la réflectance à partir d'images réelles sont des défis courants de traitement d'image, qui pourraient être utilisés pour rendre les environnements plus immersifs et naturels. Faire face aux exigences en temps réel des

algorithmes de suivi prédictif et proposer des moyens encore plus imaginatifs d'enregistrer l'emplacement et l'orientation des utilisateurs dans le monde réel connaîtra probablement également des avancées dans un avenir proche.

I.11.3. Les dispositifs d'interaction homme-machine

Qui sont développés pour la réalité virtuelle plus traditionnelle seront également adoptés pour les applications de réalité augmentée. L'immersion des environnements sera de plus en plus profonde avec des sensations de toucher, d'odorat et peut-être même de goût.

I.11.4. La détection et la modélisation des utilisateurs

Sont remises en question avec les applications futures. Cela pourrait être réalisé grâce à des données sensorielles précises et à un modèle comportemental prédéfini de l'utilisateur. Avec suffisamment d'informations, l'ordinateur portable pourrait suivre l'état de l'utilisateur et ajuster son comportement en conséquence. Cela pourrait être utilisé pour prendre des décisions sur le fait de déranger ou non l'utilisateur avec une information, ou pour prédire l'action suivante de l'utilisateur, ou déclarer et mener certaines actions de manière préventive.

Nous pensons cependant que le plus grand changement pour la technologie de réalité augmentée ne réside pas nécessairement dans une innovation haut de gamme spécifique, mais plutôt dans la pénétration des marchés de masse dans diverses formes d'informatique omniprésente. [4]

Conclusion

La réalité augmentée est une vieille idée qui est actuellement sur le point de réussir. Cela est dû au fait que jusqu'à récemment, il n'y avait pas suffisamment de technologies avancées pour créer des applications réalité augmentée réalisables. Ils manquaient de puissance de calcul, de précision de suivi des utilisateurs ou de facilité d'utilisation et de confort, qui sont tous nécessaires pour produire une expérience réalité augmentée satisfaisante.

Nous pouvons conclure que la réalité augmentée qui est l'une des technologies informatiques les plus émergentes et est devenue passionnante pour les générations à venir en tant que domaine de la technologie futuriste. En raison de la capacité d'avoir plusieurs avantages qui sont impliqués dans la fabrication, la conception, le codage sans affichage, cela nécessite beaucoup de connaissances et le processus de développement est toujours en cours d'amélioration. La réalité augmentée promet l'aspect rentable et aussi un avenir meilleur dans

la technologie informatique. Cela a des applications étonnantes qui peuvent très bien nous permettre de vivre nos vies de manière plus productive, plus sûre et plus informative.

Chapitre 2 :
Traitement d'image et vidéo

Chapitre 2 : Traitement d'image et vidéo

II. Introduction

Le traitement d'image et de vidéo numérique couvrent une importance croissante dans de nombreux domaines. Dans le domaine médical, il est utilisé pour l'analyse et la visualisation des images médicales, telles que les radiographies, les scanners et les IRM. Dans le domaine de la vision par ordinateur, il est utilisé pour la reconnaissance d'objets, la détection de mouvement, la surveillance vidéo, la réalité augmentée, etc. Dans l'industrie, il est utilisé pour le contrôle de qualité, l'inspection des produits, la robotique, etc. Enfin, dans le domaine artistique, il est utilisé pour les retouches d'images, la création d'effets spéciaux, la génération d'images de synthèse, etc. [5]

II.1. Partie traitement d'image

II.1.1. Définition du traitement d'images

Le traitement d'image numérique fait référence à l'ensemble des techniques utilisées pour manipuler et analyser des images à l'aide d'outils informatiques. Il s'agit d'une discipline interdisciplinaire qui combine des domaines tels que la vision par ordinateur, le traitement du signal et les mathématiques. Le traitement d'image numérique permet de modifier, d'améliorer, de segmenter, de reconnaître et de comprendre des images afin d'extraire des informations utiles. [5]

II.1.2. Définition d'une image

Une image numérique est une représentation visuelle d'informations qui est stockée sous forme numérique, c'est-à-dire sous forme de données binaires. Elle est généralement composée de pixels qui sont les plus petites unités d'une image, chacun contenant des informations sur la couleur et l'intensité de la lumière à cet emplacement spécifique.

Les pixels sont organisés en une matrice 2D dans une image. Chaque pixel est situé à une position spécifique dans cette matrice et peut être référencé en utilisant ses coordonnées x et y. Par exemple, le pixel en haut à gauche de l'image a les coordonnées (0, 0), tandis que le pixel en bas à droite de l'image a les coordonnées (largeur-1, hauteur-1).

Les images numériques peuvent être créées à l'aide de dispositifs de capture tels que des appareils photo numériques, des scanners ou des capteurs d'images. Elles peuvent également être générées par des logiciels de création graphique, de modélisation 3D ou de traitement d'image. [5]

II.1.3. Les éléments d'une image numérique

Une image numérique est composée de plusieurs éléments qui lui donnent sa structure, sa composition et son apparence visuelle. Les principaux éléments d'une image sont cités a la liste suivante :

(a) Le pixel

Le pixel est l'unité de base d'une image numérique. Le mot "pixel" est dérivé de "picture element" (élément d'image en français). Un pixel est un point ou un élément discret qui compose une image numérique. Il est le plus petit élément indivisible d'une image et contient des informations sur la couleur et l'intensité de la lumière à cet emplacement spécifique de l'image.

(b) La Résolution

La résolution d'une image fait référence au nombre de pixels qu'elle contient, généralement exprimé en largeur x hauteur. Par exemple, une résolution de 1920x1080 indique qu'une image a une largeur de 1920 pixels et une hauteur de 1080 pixels. Une résolution plus élevée signifie plus de détails et une meilleure qualité d'image.

(c) La Profondeur de couleur

La profondeur de couleur, également appelée profondeur de bits, fait référence au nombre de bits utilisés pour représenter la couleur de chaque pixel de l'image. Une profondeur de couleur de 8 bits permet de représenter 256 niveaux de couleurs différents pour chaque composante (rouge, vert, bleu), ce qui donne un total de 16,7 millions de couleurs possibles. Les profondeurs de couleur courantes sont 8 bits (256 niveaux), 16 bits (65 536 niveaux) et 24 bits (16,7 millions de niveaux).

(d) L'espace colorimétrique

L'espace colorimétrique d'une image définit la gamme de couleurs qui peuvent être représentées. Le plus couramment utilisé est le RGB (Rouge, Vert, Bleu), dans lequel chaque pixel est représenté par des valeurs de ces trois valeurs qui représentent les niveaux de rouge, de vert et de bleu. Par exemple, un pixel avec des valeurs RGB de (255, 0, 0) représente la couleur rouge pure, tandis qu'un pixel avec des valeurs RGB de (0, 255, 0) représente la couleur verte pure. Il existe d'autres espaces colorimétriques tels que CMYK (Cyan, Magenta,

Jaune, Noir) utilisé pour l'impression, et HSV (Teinte, Saturation, Valeur) utilisé pour représenter les couleurs en termes de teinte, de saturation et de luminosité.

(e) La taille du fichier

La taille du fichier correspond à l'espace de stockage occupé par une image numérique. Elle dépend de la résolution, de la profondeur de couleur et du niveau de compression appliqué à l'image. Les images de haute résolution avec une profondeur de couleur élevée ont généralement une taille de fichier plus grande.

(f) La compression

La compression d'image est le processus de réduction de la taille du fichier de l'image en utilisant des techniques de compression, telles que la compression sans perte (qui préserve toutes les informations de l'image) ou la compression avec perte (qui supprime certaines informations de l'image pour obtenir une plus petite taille de fichier). Les méthodes de compression courantes comprennent JPEG, PNG et GIF.

(g) Les métadonnées

Les métadonnées d'une image sont des informations supplémentaires qui peuvent être associées à une image numérique, telles que l'appareil photo utilisé, la date de création, les informations de géolocalisation, les paramètres d'exposition, etc. Elles peuvent être stockées dans le fichier image sous forme de balises. [5]

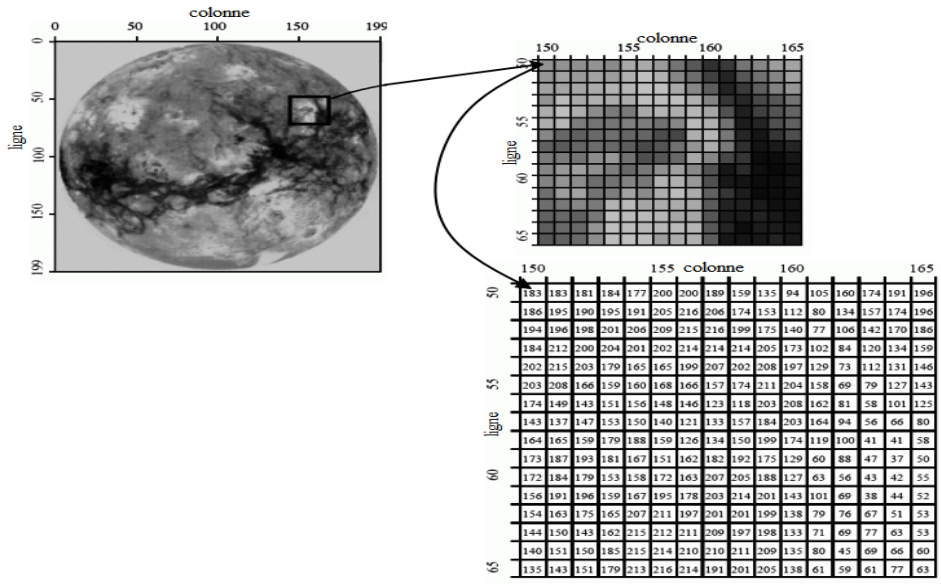


Figure 9 Exemples d'une image numérique

II.1.4. Les Types d'images numériques

Il y a trois principaux types d'image :

(a) Images binaires

Les images binaires sont composées de pixels ne pouvant prendre que deux valeurs distinctes, généralement représentées par 0 (noir) et 1 (blanc). Elles sont utilisées pour représenter des images en noir et blanc ou en deux tons, où l'information visuelle se résume à la présence ou à l'absence de pixels.

(b) Images en niveaux de gris

Les images en niveaux de gris sont des images où chaque pixel peut prendre une valeur de niveau de gris dans une plage allant du noir au blanc. Elles sont souvent utilisées pour représenter des images en nuances de gris, où la valeur du pixel indique l'intensité lumineuse à cet endroit.

(c) Images couleur

Les images couleur sont composées de pixels qui contiennent des informations sur les couleurs primaires, généralement rouge, vert et bleu (RGB). Chaque pixel est représenté par trois composantes, où la combinaison des intensités de ces trois couleurs donne la couleur finale du pixel. Les images couleur permettent une représentation plus réaliste des scènes visuelles. [5]

Image binaire Image en niveaux de gris Image couleur

Figure 10 les types d'image

II.1.5. L'acquisition d'images numériques

L'acquisition d'images numériques est le processus de capture d'images à l'aide de dispositifs tels que des appareils photo numériques, des scanners, des caméras de surveillance, etc. Ce processus comprend plusieurs étapes, telles que la conversion de la lumière en signal électrique, l'échantillonnage de l'image et la quantification du signal.

(a) Formats d'images numériques

Les images numériques sont généralement stockées dans des formats spécifiques qui définissent la structure des données et les méthodes de compression utilisées. Les formats d'images les plus courants sont JPEG, PNG, TIFF et BMP. Chaque format a ses propres avantages et inconvénients en termes de qualité d'image, de taille de fichier et de capacité de compression.

(b) Prétraitement des images acquises

Avant de pouvoir appliquer des techniques de traitement d'image, il est souvent nécessaire de prétraiter les images acquises. Le prétraitement peut inclure des opérations telles que l'égalisation d'histogramme pour améliorer le contraste, le filtrage pour réduire le bruit, la correction de la balance des blancs pour ajuster les couleurs, etc. Ces étapes de prétraitement visent à améliorer la qualité de l'image et à rendre les informations visuelles plus exploitables.[5]

II.1.6. Les éléments d'analyse et d'amélioration d'image

(a) La luminance

La luminance d'une image est une mesure de l'intensité lumineuse perçue dans une scène ou une photographie. Elle représente la quantité de lumière émise ou réfléchie par chaque pixel de l'image.

La luminance est généralement exprimée en niveaux de gris, où une valeur de 0 représente un pixel entièrement noir, et une valeur maximale (généralement 255 pour une image en 8 bits) représente un pixel entièrement blanc. Les valeurs intermédiaires de luminance représentent les différentes nuances de gris présentes dans l'image.

La mesure de la luminance est utilisée dans divers domaines de traitement d'image, notamment pour l'ajustement de la luminosité et du contraste, la correction de l'exposition, la détection de contours, la compression d'image, la vision par ordinateur, etc. [5]

(b) Contours et textures

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes. [5]

Figure 11 Contour d'une image

(c) L'Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image. Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans les cas d'une image trop claire ou d'une image trop foncée. La figure 12 montre une image avec son histogramme.[5]

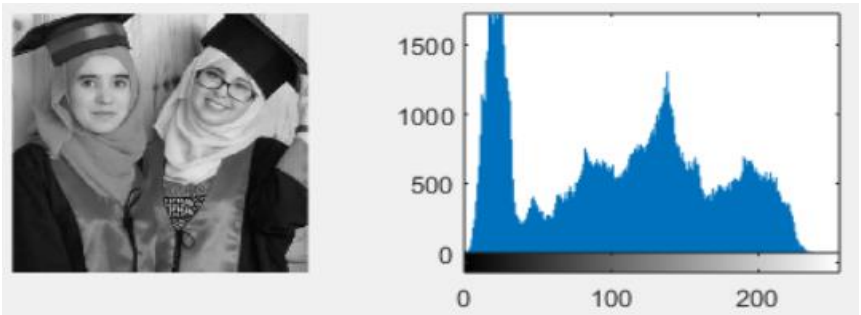


Figure 12 image avec histogramme

(d) 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. Si $\mathbf{L_1}$ et $\mathbf{L_2}$ sont les degrés de luminosité respectivement de deux zones voisines $\mathbf{A_1}$ et $\mathbf{A_2}$ d'une image, le contraste **C** est défini par le rapport :[5]

$$C = \frac{\mathbf{L_1} - \mathbf{L_2}}{\mathbf{L_1} + \mathbf{L_2}} \qquad \text{(II.1)}$$

(e) Le Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur.[5]

Figure 13 image avec bruit et sans bruit

II.1.7. Le filtrage dans le traitement d'images

Le filtrage des images est un processus qui consiste à appliquer un filtre ou une transformation sur une image numérique afin de modifier ou d'améliorer ses caractéristiques visuelles. Il s'agit d'une technique fondamentale dans le domaine du traitement d'images numériques qui permet d'obtenir des résultats visuels souhaités en atténuant le bruit, en améliorant la netteté, en extrayant des informations spécifiques ou en appliquant des effets esthétiques.

Le filtrage des images repose sur le principe de la convolution, où une fenêtre (aussi appelée masque ou noyau de filtrage) est glissée sur l'ensemble de l'image, et une opération mathématique est appliquée à chaque pixel de l'image en fonction des valeurs de la fenêtre.

Cette opération peut impliquer des calculs de moyenne, de pondération, de soustraction, de multiplication, ou d'autres opérations spécifiques, en fonction du type de filtre utilisé.[5]

(a) Principe de la convolution

Considérons une image en niveaux de gris représentée comme une matrice 2D *I*,

Où $I(x, y)$ représente la valeur d'intensité du pixel aux coordonnées (x, y).

- Masque de filtrage : Un masque de filtrage ou un noyau de taille (n x n) est représenté par h, avec des coefficients $h(i, j)$. Le masque de filtrage peut être défini comme suit :

$$h = \begin{bmatrix} h(0,0) & h(0,1) & \cdots & h(0,n-1) \\ h(1,0) & h(1,1) & \cdots & h(1,n-1) \\ \vdots & \vdots & \ddots & \vdots \\ h(n-1,0) & h(n-1,1) & \cdots & h(n-1,n-1) \end{bmatrix} \qquad \text{(II.2)}$$

- Opération de convolution : L'opération de convolution pour chaque pixel (x, y) peut être représentée par :

$$I'(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} h(i,j) \cdot I(x+i, y+j) \qquad \text{(II.1)}$$

- I'(x, y) représente la valeur de pixel filtré après la convolution.
- Pour les images en couleur, la même opération est généralement effectuée sur chaque canal de couleur séparément.

Ces équations représentent les principes mathématiques fondamentaux de l'opération de convolution utilisée dans le filtrage des images. En appliquant différents masques de filtrage avec des coefficients appropriés, il est possible d'effectuer différentes améliorations d'image, réductions de bruit, détections de contours et autres opérations de traitement d'image.[5]

Exemple de masque 3x3 :**[5]**

On a pour un pixel *I (x, y)* et ses 8 voisins :

$$I'(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) \cdot I(x+i, y+j) \qquad \text{(II.4)}$$

$$I = \begin{bmatrix} I(x-1,y-1) & I(x-1,y) & I(x-1,y+1) \\ I(x,y-1) & I(x,y) & I(x,y+1) \\ I(x+1,y-1) & I(x+1,y) & I(x+1,y+1) \end{bmatrix}$$

$$h = \begin{bmatrix} h(-1,-1) & h(-1,0) & h(-1,1) \\ h(0,-1) & h(0,0) & h(0,1) \\ h(1,-1) & h(1,0) & h(1,1) \end{bmatrix}$$

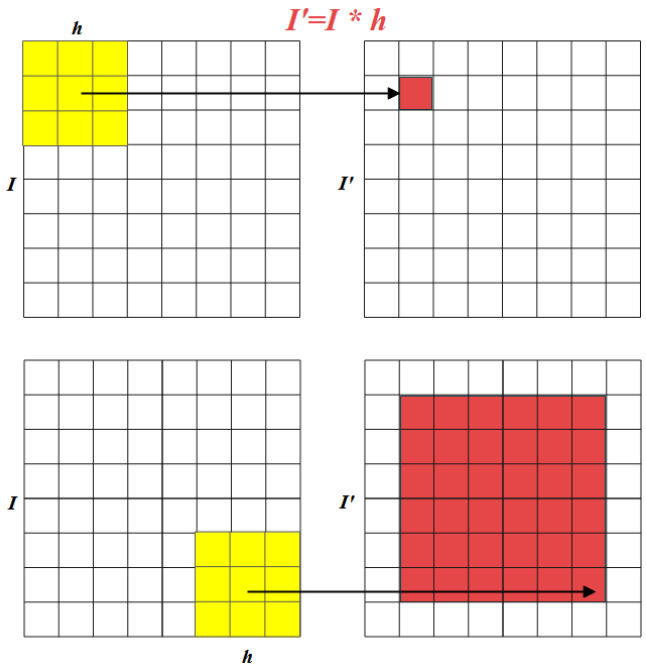


Figure 14 Convolution numérique

(b) Types des Filtrage des Images

Le filtrage des images est utilisé pour améliorer la qualité de l'image en réduisant le bruit et en améliorant la netteté. Les filtres peuvent être linéaires, tels que les filtres de lissage (filtres moyenneurs, filtres de convolution), ou non linéaires, tels que les filtres médians ou les filtres adaptatifs. Le choix du filtre dépend des caractéristiques du bruit à éliminer et des objectifs spécifiques de l'amélioration de l'image.

a) Les Filtres linéaires

Chaque filtre a une taille N x N avec N impair. Les filtres linéaires les plus connus sont les filtres passe-bas, passe-haut.[5]

i. Filtre passe-bas (lissage)

Ce filtre n'affecte pas les composantes de basse fréquence dans les données d'une image, mais doit atténuer les composantes de haute fréquence. L'opération de lissage est souvent utilisée pour atténuer le bruit et les irrégularités de l'image. Elle peut être répétée plusieurs fois, ce qui crée un effet de flou. En pratique, il faut choisir un compromis entre l'atténuation du bruit et la conservation des détails et contours significatifs. Le masque d'un filtre passe-bas et définit comme suit :[5]

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{(II.5)}$$

Figure 15 Application du filtre passe-bas, (a) image originale (b) image filtrée

ii. Filtre Passe-haut (Accentuation)

Le renforcement des contours et leur extraction s'obtiennent dans le domaine fréquentiel par l'application d'un filtre passe-haut. Le filtre digital passe-haut a les caractéristiques inverses du filtre passe-bas. Ce filtre n'affecte pas les composantes de haute fréquence d'un signal, mais doit atténuer les composantes de basse fréquence. Le masque d'un filtre passe-bas et définit comme suit :[5]

$$h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \text{(II.6)}$$

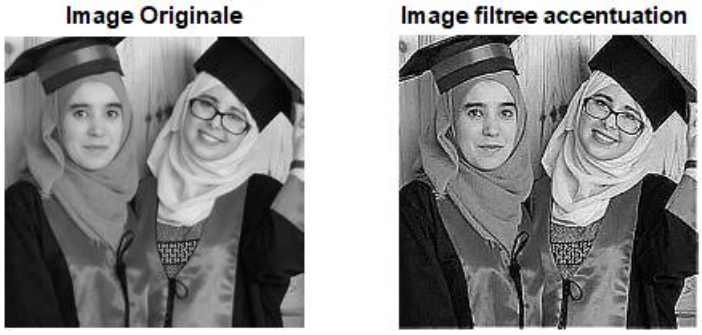


Figure 16 Application du filtre passe-haut, (a) image originale (b) image filtrée

b) Filtres non linéaires

Ils sont conçus pour régler les problèmes des filtres linéaires, Leur principe est le même que celui des filtres linéaires, il s'agit toujours de remplacer la valeur de chaque pixel par la valeur d'une fonction calculée dans son voisinage. La différence majeure, est que cette fonction n'est plus linéaire mais une fonction quelconque (elle peut inclure des opérateurs de comparaisons ou de classification). Les filtres non-linéaires les plus connus sont :[5]

i. Filtre médian

Ce filtre est très utilisé pour éliminer le bruit sur une image qui peut être de différentes origines (poussières, petits nuages, baisse momentanée de l'intensité électrique sur les capteurs, ...).

L'avantage de ce filtre réside dans le fait qu'il conserve les contours alors que les autres types de filtres ont tendance à les adoucir. L'algorithme de filtre médian est le suivant :

1. Trier les valeurs par ordre croissant.
2. Remplacer la valeur du pixel centrale par la valeur située au milieu de la triée.
3. Répéter cette opération pour tous les pixels de l'image.

$$\begin{bmatrix} 19 & 30 & 12 \\ 15 & 18 & 11 \\ 26 & 16 & 14 \end{bmatrix} \Rightarrow \begin{bmatrix} 19 & 30 & 12 \\ 15 & 16 & 11 \\ 26 & 16 & 14 \end{bmatrix} \qquad \text{(II.7)}$$

$$[30 \quad 26 \quad 19 \quad 18 \quad 16 \quad 15 \quad 14 \quad 12 \quad 11]$$

Figure 17 Application du filtre median, (a) image originale (b) image filtrée

ii. Filtre maximum

On applique le même traitement que celui du filtre médian mais la valeur du pixel du centre comme la montre la figure 18, va être changée par le maximum.

$$\begin{bmatrix} 19 & 30 & 12 \\ 15 & 18 & 11 \\ 26 & 16 & 14 \end{bmatrix} \Rightarrow \begin{bmatrix} 19 & 30 & 12 \\ 15 & 30 & 11 \\ 26 & 16 & 14 \end{bmatrix} \qquad \text{(II.8)}$$

$$[30 \quad 26 \quad 19 \quad 18 \quad 16 \quad 15 \quad 14 \quad 12 \quad 11]$$

Figure 18 Application du filtre maximum, (a) image originale (b) image filtrée

iii. Filtre minimum

On applique le même traitement que celui du filtre maximum mais, cette fois, la valeur du pixel du centre comme la montre la figure 19 va être remplacée par le minimum.

$$\begin{bmatrix} 19 & 30 & 12 \\ 15 & 18 & 11 \\ 26 & 16 & 14 \end{bmatrix} \Rightarrow \begin{bmatrix} 19 & 30 & 12 \\ 15 & 11 & 11 \\ 26 & 16 & 14 \end{bmatrix} \qquad \text{(II.9)}$$

$$\begin{bmatrix} 30 & 26 & 19 & 18 & 16 & 15 & 14 & 12 & 11 \end{bmatrix}$$

Figure 19 Application du filtre minimum, (a) image originale (b) image filtrée

II.2. Partie traitement de vidéo

Une vidéo est une séquence temporelle d'images numériques appelées trames (frames) qui sont enregistrées et reproduites à une vitesse spécifique pour créer une perception visuelle continue du mouvement. Chaque trame de la vidéo est constituée d'une matrice bidimensionnelle de pixels, où chaque pixel représente une valeur d'intensité ou de couleur. La vidéo peut être codée dans différents formats, tels que MPEG, AVI, MP4, et utilise des techniques de compression pour réduire la taille des fichiers vidéo sans compromettre de manière significative la qualité visuelle. Elle peut également être accompagnée d'une piste audio synchronisée pour une expérience multimédia complète. Les vidéos peuvent être lues, éditées et diffusées sur divers supports, tels que les écrans d'ordinateurs, les téléviseurs, les smartphones et les plateformes de diffusion en continu.[5]

II.2.1. Le Signal Vidéo

Un signal vidéo est un signal électrique ou numérique qui transporte les informations de luminance (niveau d'intensité lumineuse) et de chrominance (informations de couleur) pour chaque pixel de l'image. Il peut également inclure des informations supplémentaires telles que la synchronisation des trames, les signaux de contrôle, et éventuellement, le son associé.

Le signal vidéo peut être analogique ou numérique, en fonction du format de transmission ou d'enregistrement utilisé. Dans le domaine professionnel, les signaux vidéo peuvent être basés sur des normes telles que NTSC, PAL ou SECAM. Le signal vidéo est

utilisé dans une variété d'applications, allant des télévisions et des moniteurs aux caméras de vidéosurveillance, aux systèmes de diffusion en continu.[6]

II.2.2. Le traitement vidéo

Le traitement vidéo s'appuie sur des concepts et des techniques issus du traitement d'images et de la vision par ordinateur. Cependant, il se distingue par la prise en compte de la dimension temporelle des séquences d'images, ce qui introduit des défis supplémentaires liés à la gestion du mouvement, à l'analyse des changements dans les images successives et à la synchronisation audio-vidéo.

Le traitement vidéo englobe plusieurs tâches telles que le prétraitement des vidéos (filtrage, normalisation), l'analyse du contenu vidéo (reconnaissance d'objets, détection de mouvement), la compression vidéo (pour réduire la taille des fichiers), le traitement spatial (filtrage, amélioration de la qualité d'image), le traitement temporel (interpolation temporelle, estimation de mouvement) et l'analyse des séquences vidéo (suivi d'objets, reconnaissance d'activités).[6]

II.2.3. Filtrage dans les systèmes vidéo

(a) Le Filtrage Spatial

Le traitement spatial des vidéos concerne les opérations et les techniques appliquées à chaque image individuelle d'une séquence vidéo. Ces opérations visent à améliorer la qualité visuelle, à réduire le bruit, à améliorer la netteté et à extraire des caractéristiques visuelles spécifiques. Le filtrage de lissage spatial est l'une des techniques les plus utilisées dont on peut citer :

Les filtres de lissage

Les filtres de lissage vidéo sont des techniques utilisées pour atténuer les variations rapides d'intensité entre les pixels adjacents d'une image vidéo. Ces variations rapides peuvent être causées par des bruits indésirables, des défauts de capture ou des imperfections du capteur. Le lissage vise à rendre l'image plus homogène et à améliorer sa qualité visuelle en réduisant les détails fins et les variations brusques. Les formules suivantes décrivent quelques filtres spatiaux communs :

1. **Filtre de moyenne**

$$F(x,y)=\frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n}I(x+i,y+j) \qquad \text{(II.10)}$$

2. **Filtre gaussien**

$$F(x,y)=\frac{1}{2\pi\sigma^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n}I(x+i,y+j)\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) \qquad \text{(II.11)}$$

3. **Filtre médian**

$$F(x,y)=\text{median}(I(x+i,y+j)) \qquad \text{(II.12)}$$

(b) Le filtrage temporel

Le filtrage temporel des vidéos est une technique utilisée pour traiter les variations temporelles et les mouvements dans une séquence vidéo. Contrairement au filtrage spatial qui agit sur les pixels d'une seule image, le filtrage temporel considère les images successives d'une séquence vidéo pour améliorer la qualité visuelle, réduire le bruit temporel ou appliquer des effets spécifiques.

En peut citer quelques techniques couramment utilisées pour le filtrage temporel des vidéos dans la liste qui suit :

$F(x,y)$ Est la valeur filtrée du pixel à la position (x,y), N est le nombre d'images utilisées pour le calcul, t_0 est le temps initial et $I(x,y,t)$ est la valeur du pixel à la position (x,y) à l'instant t dans la séquence vidéo.

Filtre de moyenne temporelle

Ce filtre calcule la moyenne des pixels correspondants dans plusieurs images successives pour réduire le bruit et atténuer les variations temporaires indésirables.

Le filtre de moyenne temporelle calcule la moyenne des valeurs des pixels correspondants dans plusieurs images successives pour réduire le bruit et atténuer les variations temporaires.

Le filtre de moyenne temporelle est donné comme suit :

$$F(x,y)=\frac{1}{N}\sum_{t=t_0}^{t_0+N-1} I(x,y,t) \qquad \text{(II.13)}$$

1. Filtre de différence temporelle

Ce filtre compare les pixels correspondants entre deux images consécutives pour détecter les changements et les mouvements. Il est couramment utilisé pour la détection de mouvement, la stabilisation d'image ou la suppression des objets en mouvement.

Le filtre de différence temporelle est donné comme suit :

$$F(x,\ y)\ =\ \left|\, I(x,\ y,t_2)\ -\ I(x,\ y,t_1)\,\right| \qquad \text{(II.14)}$$

2. Filtre de lissage temporel

Ce filtre utilise des techniques de lissage pour atténuer les variations brusques entre les images consécutives. Il peut améliorer la fluidité de la vidéo et réduire les artefacts de scintillement.

$$F(x,y)=\frac{1}{N}\sum_{t=t_0}^{t_0+N-1} I(x,y,t) \qquad \text{(II.15)}$$

Le filtre de moyenne temporelle et le filtre de lissage temporel sont deux termes qui sont souvent utilisés de manière interchangeable pour désigner une même technique de filtrage dans le contexte du traitement vidéo. Alors que la formule de base reste la même, les différences entre les filtres résident dans les poids ou les méthodes de calcul appliqués aux valeurs des pixels pour obtenir le lissage temporel souhaité.[7]

Figure 20 Application du filtre de moyenne temporelle, (a) vidéo originale (b) vidéo filtrée

Conclusion

Dans ce chapitre, nous avons étudié deux parties, le traitement d'image et le traitement vidéo, dans la première partie nous avons vu les éléments, les types et les modes d'acquisitions de collection d'images numérique, ainsi que les éléments d'analyse et d'amélioration d'image, et nous avons aussi vu le principe de convolution et enfin le filtrage des images et ses types, la seconde partie contient la définition du traitement du signal et le filtrage dans les systèmes vidéo.

Chapitre 3 :
Le Flux Optique

Chapitre 3 : Le Flux Optique

III. Introduction

Dans le domaine de la vision par ordinateur, la détection de mouvement a une importance pertinente. En utilisant les informations contenues dans une image de stand, nous pouvons obtenir beaucoup d'informations sur ce que nous observons, mais il n'y a aucun moyen de déduire automatiquement ce qui va se passer dans l'immédiat. D'autre part, une séquence d'images fournit des informations sur le mouvement des objets représentés. Il existe de nombreuses techniques pour reconnaître le mouvement dans une séquence, certaines basées sur la reconnaissance des caractéristiques et des formes, d'autres basées uniquement sur les pixels, quelle que soit leur signification pour un être humain. Certains d'entre eux sont l'analyse d'appariement de blocs et l'estimation de flux optique ; ce dernier sera l'objet d'étude dans les sections suivantes.

III.1. Mouvement et champ de mouvement

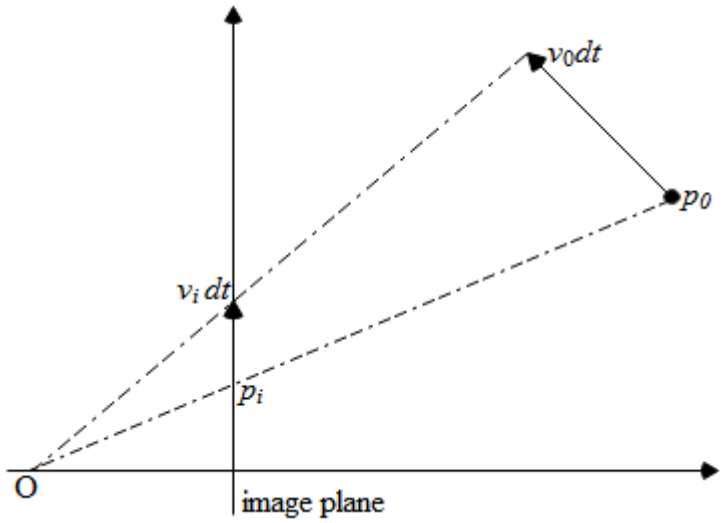


Figure 21 Projection de la vitesse sur la surface de l'image

Lorsqu'un objet se déplace dans l'espace devant une caméra, il y a un mouvement correspondant sur la surface de l'image. Etant donné le point P0 et sa projection Pi, en connaissant sa vitesse V0, on peut connaître le vecteur Vi représentant son déplacement dans l'image (figure 21). Étant donné un corps rigide en mouvement, nous pouvons construire un champ de mouvement en calculant tous les vecteurs de mouvement Vi (img. 2 Figure 22). Le champ de mouvement est un moyen de cartographier le mouvement dans un espace 3D, sur

une image 2D prise par un appareil photo : pour cette raison, puisque nous perdons une dimension, nous ne pouvons pas calculer exactement le champ de mouvement, mais juste l'approximer.

De plus, lorsque l'on considère le mouvement d'un objet, nous devons également considérer le mouvement de la caméra, appelé ego motion : l'estimation la plus fiable du mouvement est de construire à la fois ce mouvement, celui de l'objet et celui de la caméra. Ici, nous ne considérerons pas le mouvement de l'ego, en supposant que nous avons une caméra fixe, et nous n'allons pas non plus envisager de modifier l'éclairage et l'ombrage.

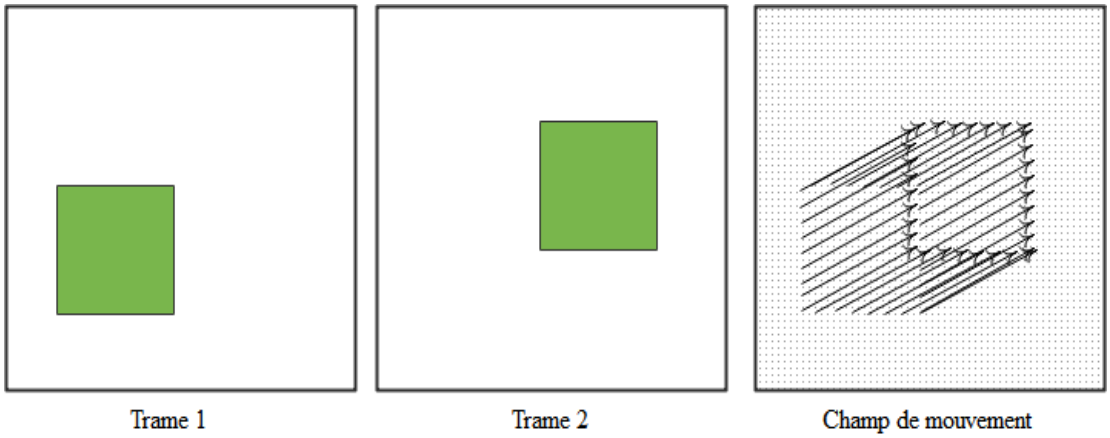


Figure 22 Exemple du champ de mouvement

III.2. Le Flux optique

Le flux optique est un phénomène auquel nous sommes quotidiennement confrontés. C'est le mouvement visuel apparent d'objets immobiles lorsque nous nous déplaçons dans le monde. Lorsque nous nous déplaçons, tout le monde qui nous entoure semble se déplacer dans le sens opposé : plus nous nous déplaçons vite, plus il le fait vite. Ce mouvement peut également nous indiquer à quel point nous sommes proches des différents objets que nous voyons. Plus ils sont proches, plus leur mouvement apparaîtra rapide. Il existe également une relation entre l'amplitude de leur mouvement et l'angle entre la direction de notre mouvement et leur position relative par rapport à nous : si nous nous dirigeons vers un objet, il semblera immobile, mais il deviendra plus grand à mesure que nous nous en approcherons (ce phénomène est aussi appelé FOE Focus Of Expansion, focale d'expansion) ; par ailleurs, si l'objet que nous regardons est à côté de nous, il semblera se déplacer plus rapidement.

En vision par ordinateur, il existe de nombreuses techniques d'estimation du flux optique appliquées dans des domaines tels que la reconnaissance de comportement ou la vidéosurveillance ; bien qu'elles soient "plus directes" que les méthodes basées sur la reconnaissance de formes, il existe des implémentations assez rapides qui nous permettent de créer des applications logicielles en temps réel.

Formalisation mathématique

Les méthodes de flux optique tentent de calculer des flux optiques en utilisant deux images prises aux instants t et $t + \Delta t$; puisque ces méthodes utilisent les séries de Taylor, elles sont appelées méthodes différentielles, car elles fonctionnent avec les dérivées spatiales et temporelles.

Étant donné une image X, on exprime l'intensité de chacun de ses pixels comme suit :

$$I(x, y, t) \qquad \text{(III.1)}$$

Nous pouvons maintenant supposer que l'intensité de l'image de chaque point de scène visible ne change pas dans le temps, nous pouvons donc énoncer l'équation dite de contrainte d'image :

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \qquad \text{(III.2)}$$

En supposant que le mouvement soit petit, l'équation 1.2 peut être développée en série de Taylor comme suit :

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + O\left(\delta x^2, \delta y^2, \delta t^2\right) \qquad \text{(III.3)}$$

Où $O\left(\delta x^2, \delta y^2, \delta t^2\right)$ signifie des termes d'ordre supérieur, qui peuvent être ignorés.

Des équations III.3 et III.2, il résulte que :

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \qquad \text{(III.4)}$$

C'est-à-dire, en divisant par δt, ce qui suit :

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \qquad \text{(III.5)}$$

Qui se traduit par :

$$\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}=0 \qquad \text{(III.6)}$$

Où u et v sont les composantes de vitesse du flux optique associé au pixel considéré, portant respectivement sur l'axe X et sur l'axe Y, et $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ et $\frac{\partial I}{\partial t}$ sont les dérivées de l'image en (x,y,t). Ces trois mesures peuvent aussi s'écrire I_x, I_y et I_t, donc l'équation 1.4 deviendra :

$$I_x u + I_y v = -I_t \qquad \text{(III.7)}$$

Ou, de manière équivalente :

$$\nabla I^T \cdot \vec{\boldsymbol{v}} = -I_t \qquad \text{(III.8)}$$

Nous avons une équation mais deux variables, cela signifie que nous avons besoin d'autres contraintes. Pour cette raison, le flux optique ne correspond parfois pas au champ de mouvement. C'est ce que l'on appelle le problème d'ouverture et nous pouvons mieux le comprendre en observant la figure 23 : le cylindre étant en rotation, si l'on ne considère que les barres noires, il serait impossible de déterminer si elles se déplacent horizontalement (comme elles le font), ou verticalement, comme détecté par le flux optique. Il est impossible de déterminer le mouvement réel à moins que l'extrémité des barres ne devienne visible.

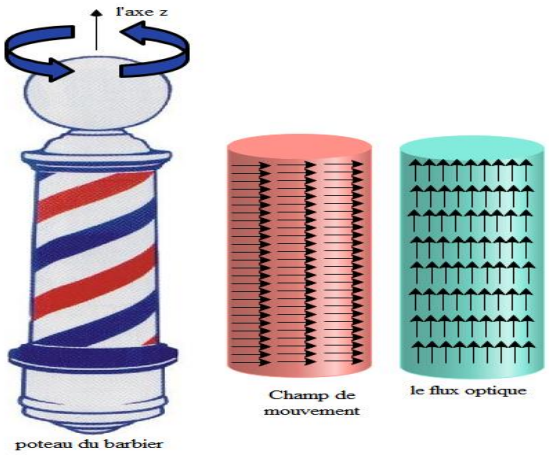


Figure 23 Un exemple lorsque le flux optique est différent du champ de mouvement

III.2.1. Les Méthodes de flux optique

Puisque nous devons déterminer le système, nous pouvons utiliser différentes méthodes qui ajoutent une certaine contrainte au problème, afin d'estimer le flux optique. Certains d'entre eux sont :

1. Les Méthodes basées sur les blocs - minimisation de la somme des différences au carré ou de la somme des différences absolues, ou maximisation de la corrélation croisée normalisée.
2. Les Méthodes différentielles d'estimation de flux optique, basées sur des dérivées spatiales et temporelles partielles du signal image, telles que :

a. {La Méthode de Horn-Schunck - optimisation d'une fonctionnelle basée sur les résidus de la contrainte de constance de luminosité, et un terme de régularisation particulier exprimant la régularité attendue du champ d'écoulement.

b. {La Méthode Lucas-Kanade - division de l'image en patchs et calcul d'un flux optique unique sur chacun d'eux.

(a) La méthode Horn-Schunck

La méthode Horn-Schunck d'estimation du flux optique est une méthode globale qui introduit une contrainte globale de lissage pour résoudre le problème d'ouverture.

Détails mathématiques :

L'algorithme de Horn-Schunck suppose une fluidité du flux sur toute l'image. Ainsi, il essaie de minimiser les distorsions de flux et préfère les solutions qui montrent plus de douceur.

Le flux est formulé comme une fonctionnelle énergétique globale que l'on cherche ensuite à minimiser. Cette fonction est donnée pour des flux d'images bidimensionnels comme :

$$E = \iint \left[\left(I_x u + I_y v + I_t \right)^2 + \alpha^2 \left(\|\nabla u\|^2 + \|\nabla v\|^2 \right) \right] dxdy \qquad \text{(III.9)}$$

Où I_x, I_y et I_t sont les dérivées des valeurs d'intensité de l'image le long des dimensions x, y et temporelle respectivement, $\vec{v} = \left[u(x,y), v(x,y) \right]^T$ est le vecteur de flux optique (qu'on doit résoudre) et le paramètre α est une constante de régularisation. Des valeurs plus grandes de α conduisent à un flux plus fluide (lisse). Cette fonction peut être

minimisée en résolvant les équations d'Euler-Lagrange multidimensionnelles associées. Ceux-ci sont.

- **La technique de Horn et Schunck**

La technique de flux optique de Horn et Schunck est une méthode largement utilisée pour estimer le mouvement apparent des objets dans une séquence d'images. Elle suppose que les intensités des pixels d'un objet restent constantes entre les images consécutives et cherche à calculer le champ de flux optique qui décrit le déplacement des pixels.

Commençons par le concept de base. Étant donné deux images consécutives, nous voulons estimer les vitesses horizontales et verticales (également appelées flux optique) de chaque pixel de la première image. Nous désignons le flux optique au pixel (x, y) par u (x, y), v (x, y)), où u (x, y) représente le déplacement horizontal et v (x, y) représente le déplacement vertical.

La méthode de Horn et Schunck formule le problème comme une tâche de minimisation d'énergie. Elle suppose que le champ de flux optique doit être régulier, ce qui signifie que les pixels voisins doivent avoir un mouvement similaire. De plus, elle suppose que la luminosité de chaque pixel reste constante au fil du temps. La formulation implique la minimisation de la fonctionnelle d'énergie suivante :

$$E(u, v) = \iint [\nabla I(x, y) \cdot (u(x, y), v(x, y))]^2 + \alpha^2 * \|\nabla u(x, y)\|^2 + \alpha^2 * \|\nabla v(x, y)\|^2 \, dxdy \quad \text{(III.10)}$$

Dans l'équation ci-dessus :

- I (x, y) est l'intensité du pixel à la position (x, y) dans la première image.
- ∇I (x, y) = (Ix (x, y), Iy (x, y)) représente le gradient de l'image à la position (x, y), où Ix et Iy sont les dérivées partielles de l'intensité de l'image par rapport à x et y, respectivement.
- ∇u (x, y) et ∇v (x, y) sont les gradients spatiaux de u (x, y) et v (x, y), respectivement.
- α est un paramètre de régularisation qui équilibre le terme de régularité avec le terme de données. Il contrôle le compromis entre la régularité et la précision.

Pour trouver le champ de flux optique, nous devons minimiser la fonctionnelle d'énergie en annulant ses dérivées par rapport à u (x, y) et v (x, y). Cela conduit au système d'équations suivant :

$$\nabla\cdot[\nabla I(x, y) \cdot (u(x, y), v(x, y))] + \alpha^2 * \nabla\cdot\nabla u(x, y) = 0 \qquad \text{(III.11)}$$

$$\nabla\cdot[\nabla I(x, y) \cdot (u(x, y), v(x, y))] + \alpha^2 * \nabla\cdot\nabla v(x, y) = 0 \qquad \text{(III.12)}$$

Ces équations peuvent être résolues à l'aide de méthodes numériques telles que des solveurs itératifs ou des algorithmes d'optimisation. En résolvant ces équations, nous obtenons le champ de flux optique (u (x, y), v (x, y)).

Une fois le champ de flux optique calculé, nous pouvons le visualiser en représentant les vecteurs de mouvement (u (x, y), v (x, y)) sous la forme de flèches partant de chaque emplacement de pixel. La direction et la longueur des flèches représentent respectivement la direction et l'amplitude du mouvement.

Il convient de noter que la méthode de Horn et Schunck suppose un champ de mouvement régulier et peut être sensible aux déplacements importants ou aux occlusions. Diverses extensions et améliorations ont été proposées pour relever ces défis, telles que l'utilisation de pyramides ou l'incorporation de contraintes supplémentaires.

(b) La Méthode de Lucas–Kanade

Dans le domaine de la vision par ordinateur, la méthode Lucas–Kanade est une méthode différentielle utilisée pour l'estimation du flux optique. Cette méthode a été développée par Bruce D. Lucas et Takeo Kanade. Elle suppose que le flot est essentiellement constant dans un voisinage local du pixel considéré, et résout l'équation du flot optique pour tous les pixels dans ce voisinage par la méthode des moindres carrés.

En combinant les informations des pixels proches environnant, la méthode de Lucas-Kanade peut souvent résoudre l'ambiguïté inhérente de l'équation du flot optique : le problème de l'ouverture. Cependant, comme c'est une méthode purement locale, elle ne peut pas fournir d'information sur le flux à l'intérieur d'une région uniforme de l'image.

- **Le Principe de la Methode Lucas-Kanade**

La méthode de Lucas-Kanade suppose que le déplacement d'un point de l'image entre deux instants consécutifs est petit et approximativement constant dans un voisinage du point p. L'équation du flux optique peut être supposée vraie pour tous les pixels dans une fenêtre centrée au point p. Le vecteur vitesse local (u,v) doit satisfaire :

$$\begin{aligned} &I_x(q_1)\cdot u + I_y(q_1)\cdot v = -I_t(q_1) \\ &I_x(q_2)\cdot u + I_y(q_2)\cdot v = -I_t(q_2) \\ &\vdots \\ &I_x(q_n)\cdot u + I_y(q_n)\cdot v = -I_t(q_n) \end{aligned} \qquad \text{(III.13)}$$

Où $q_1, q_2, ...q_n$ sont les pixels à l'intérieur de la fenêtre, et $I_x(q_i), I_y(q_i)$ et $I_t(q_i)$ sont les dérivées partielles de l'image I selon les variables d'espace x, y et de temps t, évaluée au point q_i et au temps courant t.

Ces équations peuvent être écrites sous la forme matricielle suivante : $\mathbf{A}\cdot\mathbf{v} = \mathbf{b}$, où

$$\mathbf{A} = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \\ I_x(q_n) & I_y(q_n) \end{bmatrix} ; \ \mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix} \text{ et } \mathbf{b} = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \qquad \text{(III.14)}$$

Le système a plus d'équations que d'inconnues et est donc surdéterminé. La méthode de Lucas-Kanade apporte une solution par la méthode des moindres carrés. Elle résout un système d'équations normales :

$$\mathbf{A}^T\mathbf{A}\cdot\mathbf{v} = \mathbf{A}^T\mathbf{b} \qquad \text{(III.15)}$$

Où

$$\mathbf{v} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

Où $\mathbf{A}^T$ est la matrice transposée de la matrice $\mathbf{A}$. Alors, on calcule

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_x(q_i)I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix} \qquad \text{(III.16)}$$

Avec les sommes allant de $i = 1 \quad à \quad n$.

La matrice $\mathbf{A}^T\mathbf{A}$ est appelée le tenseur de structure de l'image au point *p*.

Autrement dit, étant donné une fenêtre W, on peut réécrire l'équation 1.5 comme suit :

$$f(u,v) = \iint_W \left(I_x u + I_y v + I_t \right)^2 dxdy \qquad \text{(III.17)}$$

Où on intègre par rapport à la dimension de W. Puisque l'on suppose que la vitesse est la même partout dans W, c'est-à-dire que u et v ne dépendent pas de dx ni de dy, on peut écrire le système suivant :

$$\begin{cases} u\iint I_x^2 dxdy + v\iint I_x I_y dxdy + \iint I_x I_t dxdy = 0 \\ u\iint I_x I_y dxdy + v\iint I_y^2 dxdy + \iint I_y I_t dxdy = 0 \end{cases} \qquad \text{(III.18)}$$

Conclusion

Dans ce chapitre, nous avons étudié le flux optique, nous avons vu la définition du flux optique, la technique de détection de mouvement, le champ de mouvement et les méthodes de calcul du flux optique. Nous avons examiné les principes deux méthodes, la première s'appelle Horn_Schunk et le second est Lucas_ Kanade, ces méthodes ont joué un rôle majeur dans la facilitation et la simplification du calcul du flux optique.

Chapitre 4 :
Résultats et Interprétations

Chapitre 4 : Résultats et Interprétations

IV. Introduction

L'intégration de la vision par ordinateur dans nos applications et projets interactives offre des possibilités fascinantes pour interagir avec les images et les vidéos de manière intelligente. Dans ce chapitre, nous allons explorer plusieurs programmes réalisés avec le langage Processing, qui se concentrent sur des domaines clés de la vision par ordinateur.

Tout d'abord, nous allons commencer par le suivi de couleur, une technique qui permet de détecter et de suivre un objet spécifique en utilisant sa couleur distincte. Le programme que nous allons explorer nous permettra d'identifier des objets ou des zones d'intérêt en temps réel, et de les suivre à mesure qu'ils se déplacent dans le flux vidéo.

Ensuite, nous allons aborder la détection de mouvement, une technique qui nous permettra d'identifier les changements dans un flux vidéo et de détecter les mouvements des objets en mouvement. Nous allons explorer le programme qui utilise l'algorithme de différence de couleurs, de seuillage pour détecter et suivre les mouvements.

La pixellisation est un autre aspect intéressant de la vision par ordinateur que nous allons explorer. Nous allons découvrir un programme qui divisent les images en blocs de pixels et les manipulent pour créer des effets visuels uniques et stylisés. Nous pourrons ajuster la taille des blocs, pour réduire la résolution du vidéo et gagner la vitesse de calcule qui est nécessaire en suite.

Le filtrage vidéo temporel passe-bas pour les mouvements lents est une technique de traitement d'images et de vidéos visant à atténuer les variations rapides et à ne conserver que les mouvements plus lents. Le principe de base consiste à prendre en compte plusieurs images successives et à calculer une moyenne des valeurs de pixels pour chaque pixel de l'image actuelle.

Enfin, nous allons explorer le flux optique, une technique qui permet d'estimer le mouvement des objets dans une séquence d'images. Nous allons découvrir un programme qui utilise des algorithmes de flux optique pour calculer la direction et la vitesse du mouvement des objets, ce qui peut être utilisé pour des applications telles que la stabilisation d'image ou l'analyse du mouvement.

Chaque programme sera présenté avec une explication détaillée du code, des fonctionnalités et des algorithmes utilisés. On a pu ainsi apprendre les bases de la vision par ordinateur en utilisant le langage Processing et les appliquer à nos propres projets et domaines

d'intérêt. On a aussi découvert comment ces programmes on pu ouvrir de nouvelles perspectives et possibilités créatives dans le domaine de l'analyse d'images et de vidéos.

IV.1. Le language de programmation Processing

Le langage de programmation Processing est un langage de programmation open-source basé sur Java. Il a été spécialement conçu pour les artistes, les concepteurs et les débutants en programmation, afin de faciliter la création d'œuvres interactives, de visualisations graphiques, d'animations et d'applications multimédias tout en intégrant la programmation orientée objet (OOP).

Processing offre une syntaxe simplifiée et une interface de programmation graphique qui permet aux utilisateurs de se concentrer sur la création visuelle plutôt que sur les détails techniques de la programmation. Il fournit également une bibliothèque graphique riche qui facilite la création de formes, de dessins, de graphiques et de animations.

Le Langage Processing est accompagné d'un environnement de développement intégré (IDE) appelé Processing IDE, qui fournit des outils pour écrire, exécuter et déboguer le code Processing. Il existe également des variantes de Processing adaptées à d'autres langages de programmation tels que p5.js pour JavaScript et Processing.py pour Python, qui permettent d'utiliser les mêmes concepts et les mêmes principes de base dans ces langages.

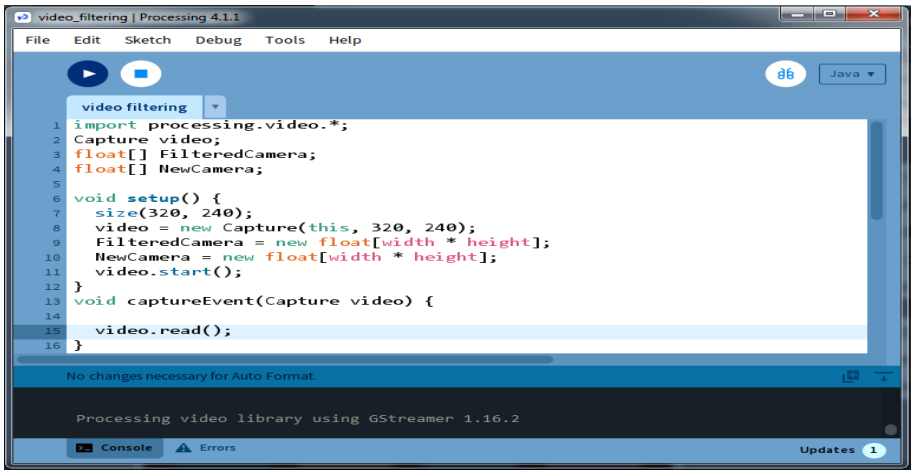
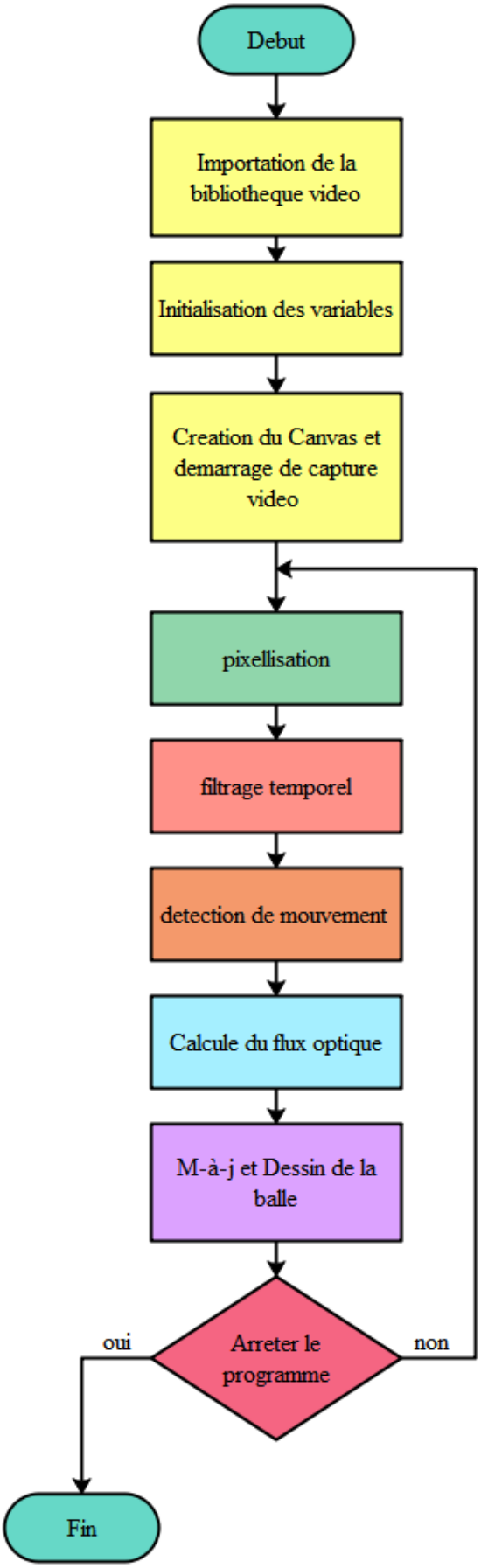


Figure 24 Environnement de développement Processing

Organigramme



IV.2 Le code de suivi des colleurs

```
import processing.video.*;
Capture video;
color trackColor;
void setup() {
  size(320, 240);
  video = new Capture(this, width, height);
  video.start();
  trackColor = color(255, 0, 0); // Démarrer le suivi pour le rouge
}
void captureEvent(Capture video) {
   video.read();//Lire l'image de la caméra
}
void draw() {
  video.loadPixels();
  image(video, 0, 0);
  float worldRecord = 500;
    int closestX = 0;
  int closestY = 0; // Coordonnée (x,y) de la couleur la plus proche
  // Commencer la boucle pour parcourir chaque pixel
  for (int x = 0; x < video.width; x++) {
    for (int y = 0; y < video.height; y++) {
      int loc = x + y * video.width;
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);
      float d = dist(r1, g1, b1, r2, g2, b2); // distance euclidienne pour comparer les couleurs
      if (d < worldRecord) {
        worldRecord = d;
        closestX = x;
        closestY = y;}
    }
  }
  if (worldRecord < 10) { // Dessinez un cercle au pixel suivi
    fill(trackColor);
    strokeWeight(4);
    stroke(0);
    ellipse(closestX, closestY, 16, 16);}
}
void mousePressed() {
  // Enregistrer la couleur où la souris est cliquée dans la variable trackColor
  int loc = mouseX + mouseY * video.width;
  trackColor = video.pixels[loc];}
```

Figure 25 le code de suivi de couleurs

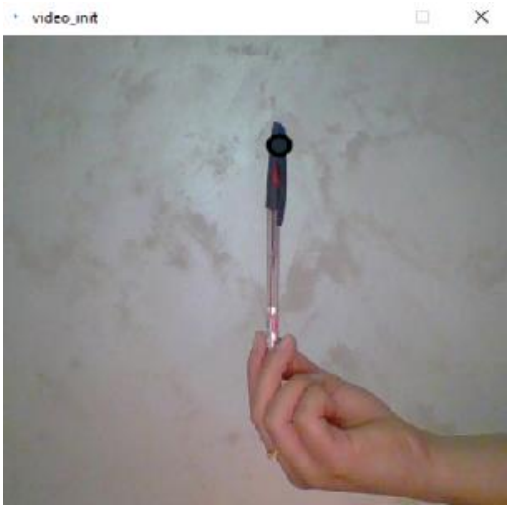


Figure 26 L'exécution du code de suivi de couleurs

IV.2. 1.Description du code de suivi des couleurs

Ce code en langage Processing permet de suivre une couleur spécifique dans une vidéo capturée à partir d'une caméra.

La fonction "**setup ()**" configure la taille de la fenêtre, crée un objet de capture vidéo avec les dimensions de la fenêtre, démarre la capture vidéo et initialise la variable "**trackColor**" avec la valeur correspondant à la couleur rouge (255, 0, 0) pour démarrer le suivi de cette couleur.

La fonction "**captureEvent (Capture video)**" est appelée chaque fois qu'une nouvelle image est capturée depuis la caméra. Elle lit simplement l'image capturée.

La fonction "**draw ()**" est appelée en boucle pour traiter chaque image capturée. Les pixels de la vidéo sont chargés et affichés sur la fenêtre.

Ensuite, une double boucle "**for**" est utilisée pour parcourir chaque pixel de la vidéo. Pour chaque pixel, sa couleur est extraite et les composantes rouge, verte et bleue sont séparées.

Ensuite, la distance euclidienne entre la couleur du pixel actuel et la couleur de suivi (**trackColor**) est calculée à l'aide de la fonction "**dist ()**". La distance est utilisée pour trouver la couleur la plus proche dans la vidéo.

Si la distance est inférieure à un seuil prédéfini (**worldRecord < 10**), cela signifie qu'un pixel correspondant à la couleur de suivi a été trouvé. Dans ce cas, un cercle est dessiné autour de ce pixel.

Enfin, la fonction "**mousePressed ()**" est utilisée pour enregistrer la couleur du pixel sur lequel la souris est cliquée dans la variable "**trackColor**". Cela permet de changer la couleur de suivi en fonction des clics de la souris.

IV.3. Le code de détection de mouvements

```
import processing.video.*;
Capture video; // Variable for capture device
PImage prevFrame; // Previous Frame
float threshold = 50; // How different must a pixel be to be a "motion" pixel
void setup() {
  size(640, 480);
  video = new Capture(this, width, height);
  video.start();
  prevFrame = createImage(video.width, video.height, RGB);  // Create an empty image the same size as
the video
}
void captureEvent(Capture video) {
  prevFrame.copy(video, 0, 0, video.width, video.height, 0, 0, video.width, video.height); // save the
previous frame for comparison!
  prevFrame.updatePixels(); // Read image from the camera
  video.read(); }
void draw() {
  loadPixels();
  video.loadPixels();
  prevFrame.loadPixels();
  // Begin loop to walk through every pixel
  for (int x = 0; x < video.width; x++) {
    for (int y = 0; y < video.height; y++) {
      int loc = x + y * video.width; // pixel location?
      color current = video.pixels[loc]; // current color?
      color previous = prevFrame.pixels[loc]; // previous color?
      // Step 4, compare colors (previous vs. current)
      float r1 = red(current);
      float g1 = green(current);
      float b1 = blue(current);
      float r2 = red(previous);
      float g2 = green(previous);
      float b2 = blue(previous);
      float diff = dist(r1, g1, b1, r2, g2, b2);
      // Step 5, How different are the colors?
      if (diff > threshold) {pixels[loc] = color(0);}// If motion, display black
       else {pixels[loc] = color(255); } // If not, display white
    }
  }
  updatePixels();
}
```

Figure 27 Le code détection de mouvements

Figure 28 L'exécution du code de détection de mouvements

IV.3.1. La description du code de détection de mouvements

Ce code utilise une capture vidéo pour détecter le mouvement entre les images successives et afficher les pixels en mouvement en noir et les pixels statiques en blanc.

La première partie du code concerne la configuration initiale. La taille de la fenêtre est définie et une capture vidéo est créée avec les dimensions de la fenêtre. De plus, une image "**prevFrame**" est créée pour stocker l'image précédent e de la vidéo.

La fonction "**captureEvent (Capture video)**" est appelée chaque fois qu'une nouvelle image est capturée. Elle copie l'image précédente dans "**prevFrame**" pour la comparer avec l'image actuelle. Ensuite, l'image actuelle est lue.

La fonction "**draw ()**" est appelée en boucle pour traiter chaque image capturée. Les tableaux de pixels de la vidéo, de l'image précédente et de l'affichage sont chargés.

Ensuite, une double boucle "**for**" est utilisée pour parcourir chaque pixel de la vidéo. Pour chaque pixel, les couleurs actuelles et précédentes sont extraites et la différence entre ces couleurs est calculée.

Si la différence est supérieure à un seuil prédéfini, le pixel est considéré en mouvement et est affiché en noir. Sinon, le pixel est considéré statique et est affiché en blanc. Enfin, les pixels de l'affichage sont mis à jour pour refléter les modifications apportées.

IV.3.1Le code de pixellisation

```
import processing.video.*;
int videoScale = 16; // Taille de chaque rectangle de la grille
int cols, rows; // Nombre de colonnes et de lignes dans le système
Capture video; // Variable à conserver sur l'objet Capture
void setup() {
  size(400, 400);  // Initialiser les colonnes et les lignes
  background(0);
  cols = width/videoScale;
  rows = height/videoScale;
  video = new Capture(this, cols, rows);
  video.start();
}
void captureEvent(Capture video) {
  video.read(); // Lire l'image de la caméra
}
void draw() {
  video.loadPixels();
  for (int i = 0; i < cols; i++) {  // la boucle pour les colonnes
    for (int j = 0; j < rows; j++) { // la boucle pour les lignes
      int x = i*videoScale;
      int y = j*videoScale;
      color c = video.pixels[i + j*video.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
      float gry = (r+g+b)/3;
      fill(gry);
      stroke(0);
      rect(x, y, videoScale, videoScale);
    }
  }
}
```

Figure 29 le code de Pixellisation

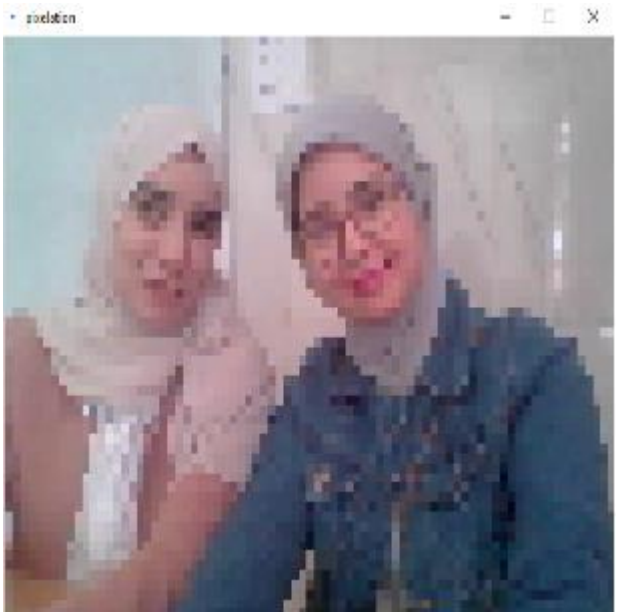


Figure 30 l'exécution du code de pixellisation

IV.3.2. Description du code de pixellisation

Ce code en langage Processing permet d'appliquer une pixellisation à une vidéo provenant de la webcam. Chaque image capturée est divisée en une grille de rectangles de taille fixe.

Dans la fonction "**setup ()**", la taille de la fenêtre est définie et les variables "**cols**" et "**rows**" sont calculées en divisant la largeur et la hauteur par la taille de chaque rectangle de la grille. Ensuite, un nouvel objet "**Capture**" est créé avec les dimensions de la grille et la capture vidéo est démarrée.

La fonction "**captureEvent (Capture video)**" est appelée à chaque fois qu'une image est capturée depuis la webcam. Elle lit simplement l'image capturée.

Dans la fonction "**draw ()**", les pixels de l'image capturée sont chargés. Ensuite, une double boucle "**for**" est utilisée pour parcourir chaque rectangle de la grille. Les coordonnées (x, y) de chaque rectangle sont calculées en multipliant l'indice de la boucle par la taille d'un rectangle.

Pour chaque rectangle, la couleur du pixel correspondant est extraite à partir du tableau de pixels de la vidéo capturée. Les composantes rouge, verte et bleue de cette couleur sont ensuite utilisées pour calculer une valeur moyenne de gris.

Ce niveau de gris est ensuite utilisé pour remplir le rectangle, donnant ainsi l'effet de pixellisation. La couleur de contour est définie comme noire.

IV.4. Le code de filtrage passe-bas temporel

```
import processing.video.*;
Capture video;
float[] FilteredCamera;
float[] NewCamera;
void setup() {
  size(320, 240);
  video = new Capture(this, 320, 240);
  FilteredCamera = new float[width * height];
  NewCamera = new float[width * height];
  video.start();}
void captureEvent(Capture video) {video.read();}
void draw() {
  loadPixels();
  for (int i = 0; i < width; i++) {
    for (int j = 0; j < height; j++) {
      color c = video.pixels[i + j*video.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
      float Luminance = 0.2987 * r + 0.5870 * g + 0.1140 * b;
      NewCamera[j*width + i] = Luminance;
      FilteredCamera[i + j*video.width] += (NewCamera[i + j*video.width] - FilteredCamera[i +
j*video.width]) * 0.01;
      pixels[i + j*video.width] = color(FilteredCamera[i + j*video.width]); }
  }
  updatePixels();
}
```

Figure 31 Le code de filtrage passe-bas temporel

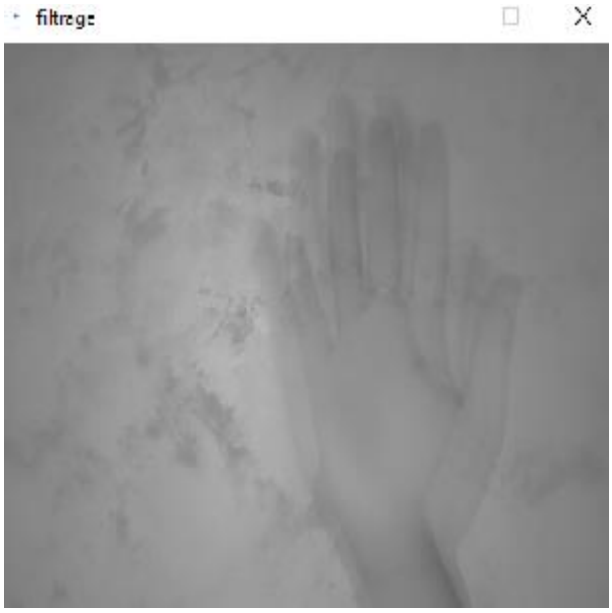


Figure 32 l'exécution du code de filtrage passe-bas temporel

IV.4.1. La description du code de filtrage passe-bas temporel

Ce code effectue un filtrage vidéo temporel passe-bas pour réduire le bruit et améliorer la stabilité de l'image capturée par la webcam.

La configuration initiale est effectuée en définissant la taille de la fenêtre et en créant un objet de capture vidéo avec les dimensions spécifiées. Deux tableaux de type float sont créés pour stocker les valeurs des pixels filtrés et les nouvelles valeurs des pixels respectivement. La capture vidéo est ensuite démarrée.

La fonction "**captureEvent (Capture video)**" est appelée chaque fois qu'une nouvelle image est capturée depuis la webcam. Elle lit simplement l'image capturée.

La fonction "**draw ()**" est appelée en boucle pour traiter chaque image capturée. Les pixels de l'affichage sont chargés.

Ensuite, une double boucle "**for**" est utilisée pour parcourir chaque pixel de l'image capturée. Pour chaque pixel, les composantes rouge, verte et bleue sont extraites pour calculer la luminance selon une formule spécifique. La valeur de luminance est ensuite stockée dans le tableau "**NewCamera**" à la position correspondante.

Ensuite, une opération de filtrage est effectuée sur chaque pixel en utilisant la différence entre la nouvelle valeur du pixel et la valeur précédente du pixel filtré. Cette différence est pondérée et ajoutée à la valeur filtrée pour créer une image filtrée temporellement. La valeur filtrée est ensuite stockée dans le tableau "**FilteredCamera**" à la position correspondante.

Enfin, les pixels de l'affichage sont mis à jour en utilisant les valeurs filtrées pour obtenir une image filtrée visible.

IV.5. Le code de calcule du flux optique

```
import processing.video.*;
int videoScale = 10;
int cols, rows;
Capture video;
// 2D Maps for image processing
float[] OldCamera;    // Previous raw frame from camera
float[] NewCamera;    // Recent raw frame from camera
float[] FilteredCamera;  // low-pass filtered image
float[] OldFilteredCamera;  // low-pass filtered image
float[] OldMotionImage;  // previous motion image
float[] MotionImage;    // recent motion image
float[] FlowX;      // x-component of flow field vector
float[] FlowY;      // y-component of flow field vector
float BallX = 0.0;        // Ball position 2D
float BallY = 0.0;
float BallVX = 0.0;        // Ball Velocity 2D
float BallVY = 0.0;
void setup() {
  size(800, 600);
  cols = width/videoScale;
  rows = height/videoScale;
  background(0);
  video = new Capture(this, cols, rows);
  video.start();
  // Allocate memory for images
  OldCamera = new float[cols * rows];
  NewCamera = new float[cols * rows];
  FilteredCamera = new float[cols * rows];
  OldFilteredCamera = new float[cols * rows];
  FlowX = new float[cols * rows];
  FlowY = new float[cols * rows];
  OldMotionImage = new float[cols * rows];
  MotionImage = new float[cols * rows];
  // Set ball position to middle of frame
  BallX =     (cols / 2.0);
  BallY =     (rows / 2.0);}
void captureEvent(Capture video) {video.read();}
void draw() {
  background(0);
  for (int i = 0; i < cols; i++) {
    for (int j = 0; j < rows; j++) {
      //int x = i*videoScale;
      //int y = j*videoScale;
      color c = video.pixels[i + j*video.width];
      float r = red(c);
      float g = green(c);
      float b = blue(c);
      OldCamera[j*cols + i] = NewCamera[j*cols + i];
      float Luminance = 0.2987 * r + 0.5870 * g + 0.1140 * b;
      NewCamera[j*cols + i] = Luminance;
      OldFilteredCamera[j*cols + i] = FilteredCamera[j*cols + i];
      OldMotionImage[j*cols + i] = MotionImage[j*cols + i];
      FilteredCamera[j*cols + i] += (NewCamera[j*cols + i] - FilteredCamera[j*cols + i]) *
0.8;
      float Diff = abs(getpixel(FilteredCamera, i, j) - getpixel(OldFilteredCamera, i, j));
      if (Diff >= 0.01) { MotionImage[j*cols + i] = Diff; }
      else { MotionImage[j*cols + i] = 0.0;}
    }//j
  }//i
  // === Calculate Optic Flow Vector Map ========================================
  // Brute Force Local Spatial Pattern Matching
  int PatchSize = 9  ;
 int SearchSize = 7  ;
  for (int x = 0; x < cols; x++) {
    for (int y = 0; y < rows; y++) {
```

```
      // Initialise serach variables
      float PatchDifferenceMax = 10000;
      FlowX[y*cols + x] = 0.0;
      FlowY[y*cols + x] = 0.0;
      // Search over a given rectangular area for a "patch" of old image
      // that "resembles" a patch of the new image.
      for (int sx = 0; sx < SearchSize; sx++) {
        for (int sy = 0; sy < SearchSize; sy++) {
          // Search vector is centre of patch test
          int SearchVectorX = (x + (sx - SearchSize / 2));
          int SearchVectorY = (y + (sy - SearchSize / 2));
          float AccumulatedDifference = 0.0;
          // For each pixel in search patch, accumulate difference with base patch
          for (int px = 0; px < PatchSize; px++) {
            for (int py = 0; py < PatchSize; py++) {
              // Work out search patch offset indices
              int PatchPixelX = SearchVectorX + (px - PatchSize / 2);
              int PatchPixelY = SearchVectorY + (py - PatchSize / 2);
              // Work out base patch indices
              int BasePixelX = x + (px - PatchSize / 2);
              int BasePixelY = y + (py - PatchSize / 2);
              // Get adjacent values for each patch
              float PatchPixel = getpixel(NewCamera, PatchPixelX, PatchPixelY);
              float BasePixel = getpixel(OldCamera, BasePixelX, BasePixelY);
              // Accumulate difference
              AccumulatedDifference += abs(PatchPixel - BasePixel);
            }//py
          }//px
          // Record the vector offset for the search patch that is the
          // least different to the base patch
          if (AccumulatedDifference <= PatchDifferenceMax) {
            PatchDifferenceMax = AccumulatedDifference;
            FlowX[y*cols + x] = (SearchVectorX - x);
            FlowY[y*cols + x] = (SearchVectorY - y);
          }//if
        }//sy
      } //sx
    }//y
  }//x
  // Modulate Optic Flow Vector Map with motion map, to remove vectors that
  // errornously indicate large local motion
  for (int i = 0; i < cols*rows; i++) {
    if (MotionImage[i] > 0.05) {  FlowX[i] *= 1;  FlowY[i] *= 1;}
    else {  FlowX[i] *= 0;  FlowY[i] *= 0;    }
  }//for
  // === Update Ball Physics ==========================================================
  // Ball velocity is updated by optic flow vector field
  BallVX += 1 * FlowX[int(BallY) * cols + int(BallX)];
  BallVY += 1 * FlowY[int(BallY) * cols + int(BallX)];
  // Ball position is updated by velocity
  BallX += 1 * BallVX ;
  BallY += 1 * BallVY ;
  // Add "drag" effect to ball velocity
  BallVX *= 0.85;
BallVY *= 0.85;
  // Bounce or Stop ball at screen walls
  if (BallX > cols-5) {   BallVX = -BallVX;    BallX = cols-5;  }
  if (BallY > rows-5) {   BallVY = -BallVY;    BallY = rows-5;  }
  if (BallX < 5+1) {    BallVX = -BallVX;   BallX = 5;   }
  if (BallY < 5+1) {    BallVY = -BallVY;   BallY = 5;   }
  display(NewCamera);
  fill(200, 0, 0, 150);
  stroke(0);
  ellipse(width-(BallX)*videoScale-1, (BallY)*videoScale, 100, 100);
}//draw
void display(float[] vid) {
  for (int i = 0; i < cols; i++) {
    for (int j = 0; j < rows; j++) {
      int x = width-(i+1)*videoScale;
      int y = j*videoScale;
      color c = int(vid[i + j*cols]);
      fill(c);
      noStroke();
      rect(x, y, videoScale, videoScale);
      stroke(0, 0, 200);
```

```
float getpixel(float[] image, int x, int y){
  if (x >= 0 && x < cols && y >= 0 && y < rows)
    return image[y*cols + x];
  else   return 0.0;};
int signum(float f) {  if (f > 0) return 1;  if (f < 0) return -1;  return 0;}
```

Figure 33 Le code de calcule du flux optique

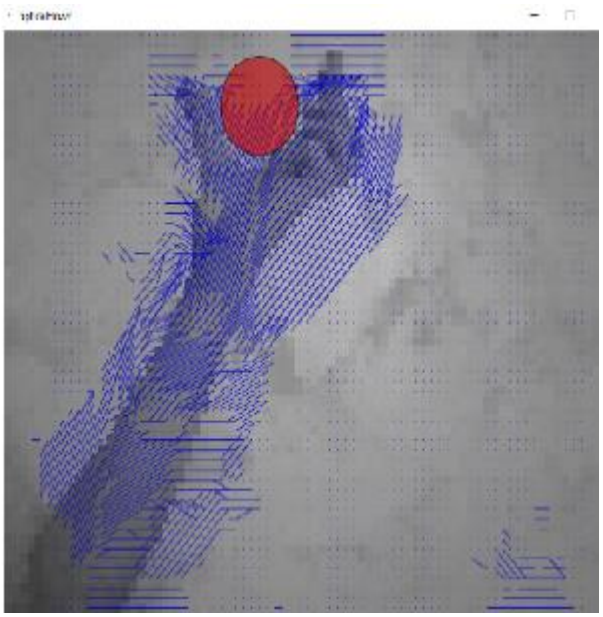


Figure 34 L'exécution du code de calcule du flux optique

IV.5.1. La description du code du Flux optique

Ce code est le code final qui calcule le flux optique dans la scène pour agir sur un objet virtuel. Tout d'abord le code utilise la bibliothèque "processing.video" pour capturer une vidéo à partir d'une caméra et effectue des calculs de flux optique pour suivre les mouvements dans la vidéo. La fonction "setup ()" configure la taille de la fenêtre, crée un objet de capture vidéo avec une résolution réduite (cols x rows), démarre la capture vidéo et initialise différentes variables.

La fonction "captureEvent (Capture video)" est appelée chaque fois qu'une nouvelle image est capturée depuis la caméra. Elle lit simplement l'image capturée.

La fonction "&draw ()"effectue les étapes suivantes :

1. Elle réinitialise les images et les vecteurs utilisés pour le traitement de flux optique.
2. Elle parcourt chaque pixel de l'image capturée et convertit la couleur du pixel en une valeur de luminance en utilisant les composantes rouge, verte et bleue.

3. Elle effectue un filtrage passe-bas pour obtenir une image filtrée à partir de la luminance.
4. Elle calcule l'image de mouvement en comparant les valeurs de pixel entre l'image actuelle et l'image filtrée précédente.
5. Elle effectue le calcul des vecteurs du flux optique pour chaque pixel.
6. Elle modifie les vecteurs du flux optique en fonction de l'image de mouvement pour éliminer les vecteurs erronés indiquant un mouvement local non-important.
7. Elle met à jour les variables de position et de vitesse d'un objet virtuel en utilisant le vecteur de flux optique.
8. Elle affiche l'image filtrée et les vecteurs de flux optique à l'écran.
9. Elle dessine un cercle rouge pour représenter l'objet virtuel à la position calculée.

La fonction "display (float [] vid)" affiche l'image filtrée à l'écran en utilisant des rectangles de taille "videoScale". Elle affiche également les vecteurs de flux optique à partir de chaque pixel en utilisant des lignes.

Les fonctions auxiliaires "getpixel()" et "signum()" sont utilisées pour obtenir la valeur d'un pixel à une position spécifique dans une image et pour calculer le signe d'un nombre respectivement.

Conclusion

L'utilisation du langage Processing pour implémenter des programmes de suivi de couleurs, pixellisation, filtrage vidéo temporel passe-bas, détection du mouvement et flux optique a démontré une grande efficacité dans le traitement et l'analyse d'images en temps réel.

Pour commencer, le suivi de couleurs a permis de détecter et de suivre des objets spécifiques en utilisant un algorithme simple de calcul de la distance entre les couleurs. La pixellisation a permis de modifier l'apparence visuelle des images en divisant les pixels en blocs de tailles variables, créant ainsi un effet graphique intéressant pour réduire les calculs.

Le filtrage vidéo temporel passe-bas a été utilisé pour réduire le bruit et les artefacts dans les vidéos en utilisant des techniques de moyenne temporelle. Cela a abouti à des vidéos qui captent que les mouvements lents.

La détection du mouvement a été réalisée en comparant les images successives et en identifiant les changements significatifs. Cela a permis de détecter les mouvements d'objets dans la scène et d'effectuer des actions en fonction de ces mouvements.

Enfin, le flux optique a été calculé en utilisant des techniques d'analyse de mouvement pour estimer la direction et la vitesse du déplacement des pixels entre les images successives. Cela a permis de comprendre et de suivre les mouvements complexes des objets dans la scène.

Le code source des différents programmes en utilisant le langage Processing ont été listé. Les résultats d'exécution ont démontré la précision et l'efficacité des algorithmes, avec des performances en temps réel sur des vidéos de différentes résolutions.

Conclusion Générale

Conclusion Générale

La réalité augmentée est une vieille idée qui est actuellement sur le point de connaître le succès. Cela est dû au fait que jusqu'à récemment, il n'y avait pas assez de technologies avancées pour créer d'éventuelles applications de réalité augmentée. Ils manquaient de puissance de calcul, de précision dans le suivi des utilisateurs ou de facilité d'utilisation et de commodité, tous nécessaires pour produire une expérience de réalité augmentée satisfaisante.

Nous pouvons conclure que la réalité augmentée qui est l'une des technologies informatiques les plus émergentes et est devenue passionnante pour les générations futures en tant que domaine de la technologie future. En raison de la possibilité d'avoir de nombreux avantages impliqués dans la fabrication. La réalité augmentée promet le côté rentable et aussi un avenir meilleur dans la technologie informatique.

L'utilisation du langage de processing pour implémenter des programmes de suivi des couleurs, de pixellisation, de filtrage vidéo temporel passe-bas, de détection de mouvement et de flux optique a montré une grande efficacité dans le traitement et l'analyse d'images en temps réel. Tout d'abord, le suivi des couleurs a permis de détecter et de suivre certains objets à l'aide d'un algorithme simple de calcul de la distance entre les couleurs. La pixellisation a permis de modifier l'aspect visuel des images en divisant les pixels en blocs de tailles variables, créant ainsi un effet graphique intéressant pour minimiser les calculs. Le filtrage vidéo time-lapse à défilement lent a été utilisé pour réduire le bruit et les artefacts dans les vidéos à l'aide de techniques de moyenne temporelle. Cela a abouti à des vidéos qui ne capturent que des mouvements lents.

Le mouvement est détecté en comparant des images successives et en identifiant des changements significatifs. Enfin, le flux optique a été calculé à l'aide de techniques d'analyse de mouvement pour estimer la direction et la vitesse de déplacement des pixels entre des images successives. Cela a permis de comprendre et de suivre les mouvements complexes des objets dans la scène le code source de divers programmes a été répertorié à l'aide du langage de traitement. Les résultats de la mise en œuvre ont montré la précision et l'efficacité des algorithmes, avec des performances en temps réel sur des vidéos de différentes résolutions.

Les travaux futurs dans ce domaine pourraient se concentrer sur l'amélioration des techniques de traitement vidéo afin d'optimiser les performances des calculs de flux

optique. Cela pourrait impliquer d'explorer des algorithmes avancés de suivi des couleurs et d'affiner les paramètres du filtrage passe-bas temporel. De plus, l'étude de l'intégration d'autres techniques de vision par ordinateur, telles que la correspondance des caractéristiques (feature matching) ou des approches basées sur l'apprentissage profond (deep learning), pourrait encore améliorer les capacités du système de réalité augmentée.

De plus, envisager la mise en œuvre d'algorithmes d'optimisation en temps réel pour améliorer l'efficacité de calcul du système global serait précieux pour les applications pratiques. Dans l'ensemble, les recherches futures devraient viser à faire progresser la précision, la vitesse et la robustesse des systèmes de réalité augmentée basés sur le flux optique pour un large éventail d'applications de vision par ordinateur.

Reference :

[1] https://books.google.dz/books?hl=ar&lr=&id=u6xqDwAAQBAJ&oi=fnd&pg=PP1&dq=introduction+pour+la+réalité+augmenté&ots=I8awlLprwh&sig=AQIO7e0V7Q7n0ZPn1xsiXtDiM9I&redir_esc=y#v=onepage&q=introduction%20pour%20la%20réalité%20augmenté&f=false,vu le 08/05/2023.

[2] R. Azuma, A Survey of Augmented Reality (http:/ / www. cs. unc. edu/ ~azuma/ ARpresence. pdf) Presence: Teleoperators and Virtual, vu le 17/02/2023.

[3] Environments, pp. 355–385, August 1997.

[4] https://pdfcoffee.com/download/augmented-reality-report-pdf-free.html, vu le 22/02/2023.

[5] https://dspace.univuargla.dz/jspui/bitstream/123456789/11567/1/BENFRIHA-HAMEL.pdf, vu le 03/03/2023

[6] Digital Video Image Quality and Perceptual Coding by H.R. Wu and K.R. Rao 2006.

[7] Video Noise Reduction Method Using Adaptive Spatial-Temporal Filtering by Ali Abdullah Yahya, Jieqing Tan, and Lian Li paper published in 2015.