ECOLE PRATIQUE DES HAUTES ETUDES
Laboratoire de Cognition Humaine et Artificielle

Thèse

présentée par **Ba Linh NGUYEN**

pour l'obtention du grade de

Docteur

Spécialité : Informatique

Suivi du regard par la caméra

soutenue publiquement le XX Janvier 2009 devant le jury composé de :

Président	Pr Charles TIJUS	Professeur à l'Université Paris 8
Rapporteurs	Pr Ivan LAVALLÉE	Professeur à l'Université Paris 8
	Pr Thierry BACCINO	Professeur à l'Université de Nice-Sophia Antipolis (PR1)
Examinateurs	Pr	Professeur à
	Pr	Professeur à
Directeur	Pr François JOUEN	Professeur et Directeur d'Études à l'EPHE
	Pr Youssef CHAHIR	Professeur à l'Université de Caen

Laboratoire de Cognition Humaine et Artificielle(CHArt)

À mes parents,

Remerciements

Suivi du regard par la caméra

Résumé :

L'objectif de cette thèse est d'étudier et de développer un système de vision temps réel pour la capture du regard dans le cadre de la communication homme machine. Cet outil doit respecter les contraintes liées à l'interaction. En particulier, il ne doit pas être intrusif ni gêner l'utilisateur dans ses mouvements naturels. D'autre part, le système ne doit pas avoir un coût trop élevé et doit donner des résultats en temps réel.

Pour le suivi du regard, le système, localise d'abord les deux yeux de l'utilisateur. Ensuite, il les suit dans la séquence d'images de la caméra. Puis, analyse des images des yeux pour détecter la position du regard de l'utilisateur.

Pour détecter la position des yeux de l'utilisateur (sur la caméra), nous avons utilisé le système de la détection rapide de l'objet basé sur les descripteurs de Haar. Ensuite, nous avons utilisé l'algorithme de Lucas-Kanade pour suivre les yeux dans la séquence d'images en temps réels Enfin, nous avons utilisé le Processus Gaussien pour faire la détection le point de regard de l'utilisateur sur l'écran de l'ordinateur.

La contribution principale de cette thèse est de proposer une solution non-intrusive et économique en utilisant le Processus Gaussiens pour faire la prédiction du regard. En autre, notre solution autorise l'utilisateur à bouger la tête rendant le dispositif proche des conditions ergonomiques écologiques.

Mots-clés : détection des yeux, suivi des yeux, suivi du regard, Processus Gaussiens.

Eye Gaze Tracking

Abstract :

The objective of this thesis is to study and develop a vision system for eye gaze tracking in real time under the human-machine communication. This tool must meet the constraints of the interaction. In particular, it should not be intrusive or interfere with the user's natural movements. On the other hand, the system should not be too costly and should give results in real time.

To track the eye gaze, first, the system has to locate two eyes of users in the camera. After that, tracking eye's movements in the camera. Then analyzing images of eyes to detect the point of regard of the user.

To detect the position of the user's eyes on the camera, we used the system for rapid detection of the object based on Haar-like features. Then we use the algorithm of Lucas-Kanade to track the eye's movements in real time. Finally, we use the Gaussian process to predict the user's gaze on the screen of computer.

The main contribution of this thesis is to propose a low-cost, non-intrusive solution using the Gaussian process to predict the eye's gaze with a simple camera. In addition, this system allows the user to move his head comfortably while tracking the eye's gaze.

Keywords : eye detecting, eye tracking, eye gaze tracking, Gaussian process.

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte

Le suivi du regard (l'occulométrie cognitive) (ou *eye gaze tracking* en anglais) est le processus de mesure les mouvements des yeux d'un utilisateur, afin de connaître où il regarde (le point de regard) sur l'écran de l'ordinateur. L'utilisation du suivi du regard est variée dans plusieurs domaines de pratique. Par exemples :

- Les psychologues peuvent surveiller ce que les pilotes ou des opérateurs de salle de contrôle regardent lors qu'ils effectuent certaines tâches ou sont placés dans certaines situations.
- Les spécialistes de la lecture peuvent utiliser l'occulométrie pour connaître quand une personne est entraîne de lire et les mots qu'ill/elle a le plus de difficile à lire.
- Les chercheurs en marketing peuvent déterminer quelles caractéristiques de la publicité sur les produits et les emballages permettent d'attirer l'attention de l'acheteur.
- Les personnes handicapées qui ne peuvent utiliser leurs mains pour manipuler un ordinateur peuvent le faire avec leurs yeux, en utilisant des claviers et les contrôleurs de la souris à l'écran.
- Les hôpitaux peuvent fournir un équipement de communication grâce à l'œil, pour les gens qui ont perdu leur capacité à se déplacer et à parler, que ce soit temporairement ou de façon permanente par un accident traumatique.

Dans le cadre du projet *"l'œil et la main"* (une partie du projet ENEIDE) du laboratoire de Cognition Humaine et Artificielle de l'EPHE, le travaux consistent en l'analyse des fixations et des saccades oculaires lors de la présentation de documents divers comme un texte ou une illustration sur l'écran du portable de l'élève. L'enseignant aura à sa disposition une analyse des traces oculomotrices, lui permettant de contrôler que l'élève a effectué correctement la tâche proposée et d'adapter sa pédagogie aux capacités attentionnelles de ses élèves. C'est pourquoi nous avons besoin une application qui permet de suivre le regard des élèves.

État de l'art

Le problème du suivi du regard a été étudié et développé depuis de nombreuses décennies en raison de ses usages potentiels dans de nombreuses applications (listées précédemment). Au début, les méthodes de suivi du regard étaient particulièrement intrusives. Ils impliquaient des contacts physiques avec l'utilisateur. Certains systèmes nécessitaient de porter des capteurs qui handicapent les mouvements spontanés de l'utilisateur : une bobine magnétique intégrée dans une lentille posée sur l'œil [118], des électrodes placées autour de l'œil pour mesurer l'activité musculaire [90], des lunettes solidement attachées à la tête

et équipées d'émetteurs et de capteurs infrarouge [23]. Aujourd'hui le système intrusif le plus répandu comprend un casque contenant deux micro caméras placées devant l'œil de l'utilisateur : une pour capter l'image de l'écran d'ordinateur et une pour capter l'image de l'œil [50]. Ce système fixe l'image de l'écran et celle de l'œil pour résoudre le problème des mouvements de la tête. Il existe également d'autres systèmes plus bruyants, avec des parties en mouvements. Il s'agit d'un semi réfléchissant placé devant l'écran, couplé à un système de rotation permettant de suivre les mouvements de l'utilisateur, de manière à garder une image fixe reflétée par le miroir dans une caméra. Les difficultés de mise en œuvre des méthodes intrusives les réservent à des application de suivi du regard dans les seuls laboratoires de recherche. Elles ne peuvent donc pas convenir à des applications plus grand public.

En raison de l'augmentation de la vitesse des processeurs et de l'amélioration des techniques de vision par ordinateur, la technologie du suivi du regard basé sur l'analyse de vidéo numérique des mouvements oculaires a été largement étudiée. Sans entraver l'utilisateur, la technologie de la vidéo ouvre des perspectives construire un système non intrusif de suivi du regard utilisable par tous. À partir des images de l'œil captées par les caméras vidéo, plusieurs techniques ont été proposées pour faire l'estimation de regard .

Les systèmes non-intrusifs (par exemples [97] [3] [79] [33] [68] [117] [75] [19]) utilisent une caméra et des méthodes traitements d'images avec des réseaux de neurones pour la détection et le suivi des yeux, ainsi que pour l'évaluation de la direction du regard. Le système fonctionne avec une précision d'un degré, mais oblige l'utilisateur à tenir la tête stable lors de la réalisation d'une session de suivi du regard. Si la tête de l'utilisateur s'éloigne de la position au moment de la calibration, la précision du système de suivi du regard baisse de façon spectaculaire.

Pour résoudre le problème du mouvement de la tête, il existe des systèmes (par exemples [123] [73] [74]) qui utilisent deux caméras pour faire la calibration et un infrarouge pour obtenir la position des yeux par réflexion. Ce système permet à l'utilisateur de bouger librement la tête pendant la session. Bien que peu chers, ces équipements restent difficiles à mettre en œuvre. C'est pourquoi, cette méthode est peu utilisée dans la pratique.

Au cours des dernières années, il est apparu des produits de suivi du regard non intrusif sur le marché par exemples *MyTobii*, *Visioboard*, *EyeGaze*,..[60]. Ils permettent des mesures très précises et, autorisent l'utilisateur à bouger librement sa tête. Cependant le prix reste très élevé (environ de 15.000 à 30.000 euros [60]), les rendant inaccessible au grand public.

Objectifs

L'objet de ce travail de thèse est d'étudier et de développer un système de suivi du regard qui pallie aux inconvénients listés précédemment.

- Le dispositif de suivi ne doit pas être intrusif, il n'a pas besoin d'aucun équipement attaché à l'utilisateur.
- Le système doit être simple à mettre en œuvre, il ne dont pas nécessiter deux caméras pour faire la calibration, ni un infrarouge pour obtenir les locations des yeux par réflexion.

– Le système ne doit pas gêner l'utilisateur dans ces mouvements. L'utilisateur doit pouvoir bouger librement sa tête lors de la session.
– Le système ne doit pas avoir un coût trop élevé.

Sur la base de ces contraintes, nous avons mis en place un système de suivi du regard en temps réel avec une caméra simple. Le système peut fonctionner soit avec la caméra numérique intégrée dans la plupart des ordinateurs portables d'aujourd'hui, soit une caméra simple du marché au prix de 15 euros. Notre système n'a donc pas besoins d'équipements spéciaux attachés à l'utilisateur, ni d'équipements spécifiques comme une caméra infrarouge.

Méthodologie

Pour suivre le regard, on doit résoudre trois problèmes principaux, ce sont :
– détecter les yeux,
– suivre les yeux,
– détecter le point de regard,

Pour détecter les yeux, nous utilisons la méthode de détection des objets basée sur descripteurs de Haar [106], [52] pour détecter les yeux sur le visage de l'utilisateur. Cette méthode entraîne en ensembles de classifieurs à partir de quelques milliers d'échantillons d'image objet et ensuite construire une cascade de classificateurs pour détecter rapidement l'objet.

Pour suivre les yeux, nous recherchons des coins sur le visage de l'utilisateur. Nous trouvons les coins sur des yeux, le nez et la bouche de l'utilisateur. Nous utilisons l'algorithme de Lucas Kanade [7] pour suivre les coins, et la méthode de détection Outliers [69] pour détecter les coins perdu et les corriger.

Pour détecter le point de regard sur l'écran, les liens fonctionnels entre l'œil de l'utilisateur et le point sur l'écran où l'utilisateur regarde doivent être trouvés. Nous n'utilisons pas de réseaux de neurones pour former cette fonction, mais estimons la distribution de cette fonction. Nous utilisons le Processus Gaussiens pour calculer la moyenne et la covariance de cette fonction et pour faire des prévisions sur les nouvelles entrées non vues dans les données.

Structure du mémoire

La présentation des recherches menées dans ce mémoire suit le plan suivant :
Dans le premier chapitre, nous nous intéressons aux recherches actuelles et passées concernant le système de suivi du regard, afin de savoir les techniques utilisées, des problèmes rencontrés pour la réalisation d'un système de suivi du regard. Puis nous proposons un nouveau système de suivi du regard qui pallie aux inconvénients des anciens systèmes et les solutions pour le réaliser. Dans les chapitres suivants nous présentons les solutions détaillées pour réaliser ce système. Chaque chapitre décrit une solution pour un problème dans le système de suivi du regard, et des résultats expérimentaux. Nous ne présentons pas les différentes solutions pour résoudre un problème, mais ne présentons que la solution utilisée dans le projet. À partir du chapitre 3, le résultat d'un chapitre est l'entrée du chapitre suivant : le résultat de la détection de l'œil dans le chapitre 3 est l'entrée du chapitre 4, le résultat du suivi de l'œil dans chapitre 4 est l'entrée du chapitre 5 pour faire la prédiction du regard.

- **Le chapitre 1** est un état de l'art des systèmes de suivi du regard, nous présentons les systèmes existants, ses techniques utilisées, ses avantages et désavantages de chacun et notre solution proposée.
- **Le chapitre 2** présente des études sur des composantes faciales. Nous présentons dans ce chapitre un état de l'art sur les analyses des caractéristiques des composantes faciales, de la couleur peau et des méthodes différentes de détection des composantes faciales.
- **Le chapitre 3** permet de faire la détection des yeux basée sur les descripteurs de Haar. Nous allons présenter le détail des descripteurs de Haar, les algorithmes d'apprentissage pour créer un cascade de descripteur de Haar pour détecter des yeux, et les résultats expérimentaux.
- **Le chapitre 4** explicite comment suivre des mouvements des yeux que nous avons détectés dans le chapitre précédant dans la caméra en temps réel. Pour suivre des yeux, nous suivons les coins des deux yeux puis récupérons les yeux dans chaque frame d'image. Nous présentons donc la méthode pour détecter les coins appropriés pour la suivi. Ensuite, nous présentons l'algorithme de Lucas Kanade pour suivre les mouvements des coins, et la méthode de détection Outlier pour détecter des erreurs (des coins perdus dans la processus de suivi) et les récupérer.
- **Le chapitre 5** est le chapitre le plus important. Il décrit de détection du point de regard. Nous présentons la raison pour laquelle nous utilisons le Processus Gaussien pour faire la prédiction le point de regard. Ensuite, nous présentons le détail de cette méthode, comment utiliser cette méthode pour faire la prédiction le regard et les résultats expérimentaux. Nous présentons également la solution pour résoudre le problème de mouvement de la tête d'utilisateur et ses résultats.
- **La conclusion** permet de faire le point sur le travail présenté, et donne les nouvelles directions à suivre dans les recherches futures.

Chapitre 1

État de l'art

1.1 Introduction

Le système de suivi du regard a été étudié et développé depuis près de 100 ans en raison de ses usages potentiels dans de nombreuses applications comme l'interaction homme-machine, la psychologie, le diagnostic des maladies des yeux, le marketing,... Mais à ce jour les progrès faisant bon usage des mouvements oculaires ont été lents. Les travaux de recherche sont prometteurs, mais nous n'avons pas encore vu de large utilisation de ces approches en pratique. Dans ce chapitre, nous allons décrire les systèmes existants, ses techniques utilisées, ses avantages et désavantages de chacun et notre solution proposée pour un nouveau système de suivi du regard.

1.2 Les différents systèmes de suivi

Il existe deux types de système de suivi : le système de suivi des mouvements des yeux (ou *eye traking* en anglais) qui mesure la position de l'œil par rapport à la tête et le système de suivi du regard (ou *eye gaze traking* en anglais) : qui mesure l'orientation de l'œil dans l'espace ou le "point de regard" (Young et Sheena, 1975)[118]. Au début, il n'y a que les systèmes de suivi des mouvements des yeux, on appelle ce système *eye tracking*. Après, il est apparu des systèmes de suivi du regard (*eye gaze tracking*), mais aujourd'hui le mot *eye tracking* est utilisé pour les deux : suivi des mouvements des yeux et suivi du regard. La mesure du point de regard est généralement utilisée pour identifier des éléments dans un scène visuelle, par exemple, dans les applications interactives. Le système le plus largement appliqué pour la mesure du point de regard est le système basé sur la vidéo de réflexion cornée.

Il existe quatre grandes catégories de méthodes de mesure des mouvements oculaires : Electro-Eculographique(EOG), lentilles de contact scléral/ bobine de recherche, Photo-Oculographique (POG) ou Video-Oculographique (VOG), et vidéo-basée combinant la réflexion de pupille/corne.

1.2.1 La technique Electro-Oculographique (EOG)

Son principe est de mesurer des différences de potentiel bioélectrique, résultat du champ bioélectrique rétino-cornéen modulé par les rotations de l'œil dans son orbite. Des électrodes sont situées à proximité du globe oculaire, en contact avec la peau. Au milieu des années 1970, cette technique a été la plus largement appliquée pour mesurer les mouvements des yeux (Young et Sheena, 1975)[118]. Cette technique est l'une des premières techniques utilisées en clinique pour enregistrer la mobilité oculaire et à fait l'objet de plusieurs applications dans le domaine du handicap (Kate et Van der Meer, 1984 [99] ; Lacourse et Hludik, 1990 [49] ; Kaczmarek, 1992 [41]). Une image d'un personne portant l'appareil EOG est montré dans la figure 1.1. Les potentiels enregistrés sont entre 15 et 200 μV, avec des sensibilités nominales de l'ordre de 20 μV/deg de mouvements des yeux.

FIGURE 1.1: Exemple de l'électro-oculographique (EOG) mesurant les mouvements oculaires.

Avantages :
- Peu onéreux malgré les coûts de mise en conformité avec les normes de sécurité des appareils électro-médicaux.
- L'enregistrement peut s'effectuer les yeux fermés ou semi-ouverts.

Inconvénients :
- Le contact des électrodes avec la peau peut provoquer à la longue des irritations et un inconfort.
- Les dérives des potentiels d'électrode et les hétérogénéités du champ électrique empêchent la réalisation de mesures absolues et rendent peu fiable la mesure des mouvements verticaux.
- La tête doit être maintenue immobile pour avoir une indication précise de la direction du regard.
- L'utilisateur a besoin d'une tierce personne pour l'installation.

1.2.2 Lentilles de contact scléral/ bobine de recherche

Une des méthodes de mesure des mouvements des yeux parmi les plus précises consiste à attacher une objet référence mécanique ou optique monté sur une lentille de contact portée directement sur l'œil. Ces premiers systèmes (Young et Sheena,1975)[118] ont utilisé un anneau en plâtre attaché directement à la cornée et à travers des liaisons mécaniques jusqu'aux stylos à l'enregistrement. Cette technique a utilisé une lentille moderne à laquelle une tige est attachée. La lentille de contact est nécessairement large, s'étendant sur la cornée et la sclère (la lentille est soumise à des dérapages si la lentille ne couvre que la cornée). Divers dispositifs mécaniques ou optiques ont été placés sur la tige attachée à la lentille : des phosphores reflex, des fils diagrammes, et des bobines de fil ont été les éléments les plus utilisés dans des configurations magnéto-optiques. La méthode principale utilise une bobine de fil, qui mesure ensuite mesuré le déplacement à travers un champ électromagnétique. Une image de la bobine de recherche intégrée dans une lentille de contact sclérale et le cadre des champs électromagnétiques sont présentés dans la figure 1.2. La technique d'insertion de la lentille de contact est montrée dans la figure 1.3.

Avantages

– C'est la méthode la plus précise de mesurant des mouvements des yeux (Young et Sheena,1975)[118].

Inconvénients

– C'est la méthode la plus intrusive. L'insertion de la lentille nécessite de l'attention et de la pratique. Le port de la lentille provoque un malaise.
– Cette méthode sert aussi à mesurer la position des yeux par rapport à la tête, et n'est généralement pas approprié pour la mesure du point de regard.

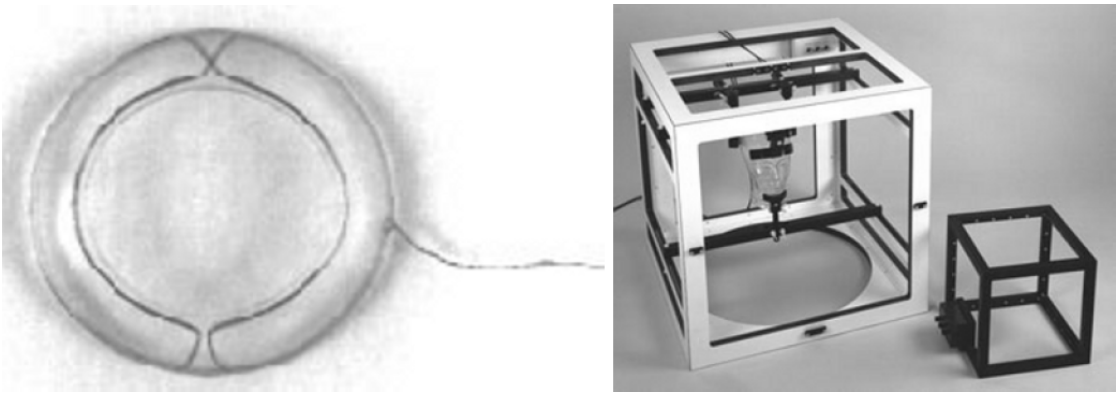


FIGURE 1.2: Exemple de la bobine de recherche intégrée dans une lentille de contact et une frame de champ électromagnétique pour mesurer les mouvements des yeux.

1.2.3 Photo-Oculographie (POG) ou Vidéo-Oculographie (VOG)

Cette catégorie regroupe une grande variété de techniques d'enregistrement des mouvements oculaires associés à la mesure de caractéristiques qui distinguent par rotation, translation, par exemple : la forme apparente de la pupille, la position de la limbe (la

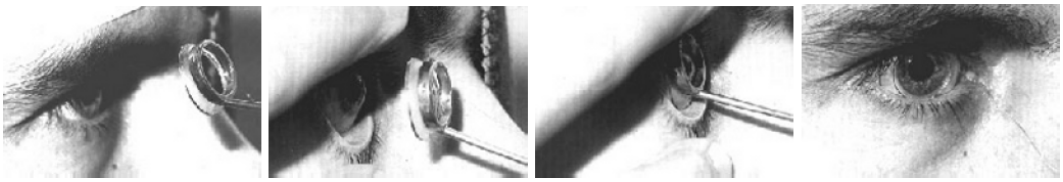
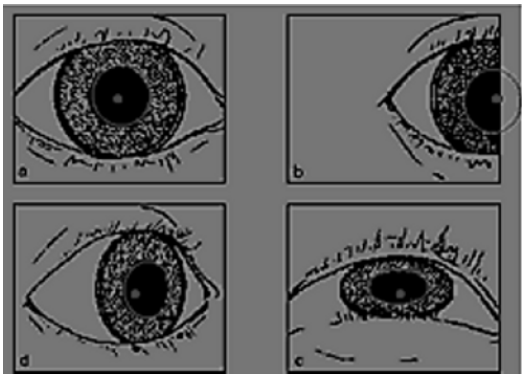
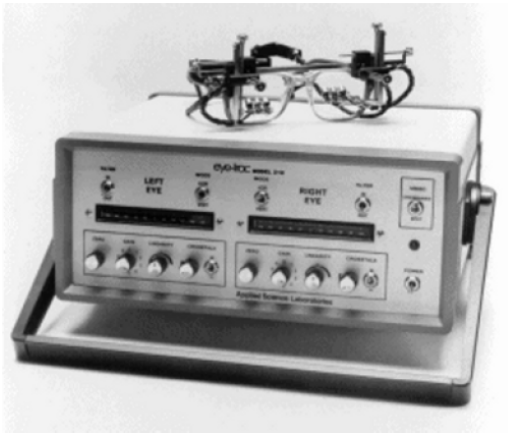


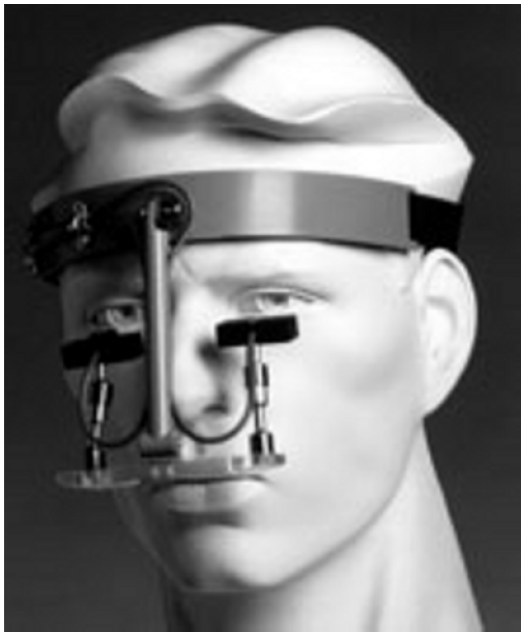
FIGURE 1.3: Exemple d'insertion sclérale par aspiration anneau bobine de recherche pour la mesure des mouvements oculaires.



(a) Exemple de taille de la pupille apparente.



(b) Exemple d'appareil infra rouge de suivi de la limbe.



(c) Un autre exemple d'appareil infra rouge de suivi de la limbe porté par sujet.

(d) Exemple de "pupille brillante" (et reflet cornéen) illuminée par la lumière infra-rouge.

FIGURE 1.4: Exemples d'éléments entrant dans la mesure des mouvements oculaires par la réflexion.

frontière sclérotique de l'iris), et de la cornée par réflexion d'une source de lumière directe située à côté (souvent un infra-rouge). Quelques exemples d'appareils et d'images de l'œil enregistrées utilisées dans la photo-oculographie ou la vidéo-oculographie et le suivi de la limbe sont montrés dans la figure 1.4. Mesure des caractéristiques oculaires fournis par ces techniques de mesure ça peut ou ça peut pas effectuée automatiquement, et peut impliquer l'inspection visuelle des mouvements oculaires enregistrées (en général enregistré sur cassette vidéo). L'évaluation visuelle effectuée manuellement (par exemple, pas à pas à travers un cadre vidéo frame à frame) peut être extrêmement fastidieuse et sujette aux erreurs car limitée au taux d'échantillonnage temporel de l'appareil vidéo. Le suivi automatique de la limbe implique souvent l'utilisation de photodiodes montées sur des montures de lunettes (voir figure 1.4b et 1.4c) et en utilisent presque toujours l'illumination (le plus souvent infra-rouge) (voir figure 1.4d). Plusieurs de ces méthodes nécessitent que la tête soit fixée (Young et Sheena, 1975)[118].

Avantages

- Peu coûteux, une simple source de lumière.
- Cette méthode analyse bien les mouvements des yeux, elle est donc utilisée pour diagnostiquer des troubles du mouvement des yeux des patients.

Inconvénients

- Bien que différents dans l'approche, ces techniques sont regroupées ici parce qu'elles ne fournissent pas la mesure de point de regard.
- Ce sont des méthodes intrusives.
- Plusieurs de ces méthodes nécessitent que la tête soit fixée (Young et Sheena, 1975)[118].

1.2.4 Vidéo-basée combinant la réflexion de pupille/corne

Bien que les techniques ci-dessus soient en général adaptées pour les mesures de mouvement des yeux, ils ne fournissent pas souvent la mesure de point de regard. Pour fournir cette mesure, soit la tête doit être fixée de telle sorte que la position de l'œil par rapport à la tête et le point de regard coïncident, soit plusieurs caractéristiques oculaires doivent être mesurées afin de lever l'ambiguïté des mouvements de la tête et de la rotation de l'œil. Les deux caractéristiques sont le reflet de la cornée (d'un source de lumière, généralement infra-rouge) et le centre de pupille (voir figure 1.4d).

Les systèmes de suivi basés sur la vidéo utilisent des caméras peu coûteuses et le matériel de traitement d'image pour calculer le point de regard en temps réel. Les appareils peuvent être montés sur table, comme le montre dans la figure 1.5, ou portés sur la tête, comme le montre la figure 1.6. L'optique des deux systèmes table-montée ou tête-montée sont quasiment identiques, à l'exception de la taille. Ces appareils, qui sont de plus en plus disponibles, sont les plus appropriés pour une utilisation dans les systèmes interactifs.

Le reflet cornéen de la source lumineuse (généralement infra-rouge) est mesuré par rapport à l'emplacement du centre des pupilles. Les reflets cornéens sont connus comme les *reflets de Purkinje* ou les *images de Purkinje* (Crane, 1994)[17]. En raison de la construction de l'œil, quatre réflexions Purkinje sont formées, comme le montre la figure 1.7. Les systèmes de suivi du regard basé sur la vidéo utilisent généralement la première image de Purkinje. Avec des procédures de calibration appropriées, ces systèmes de suivi sont

FIGURE 1.5: Exemple du système de suivi du regard de table-montée basé sur la vidéo.

FIGURE 1.6: Exemple du système de suivi du regard de tête-montée basé sur la vidéo.

capables de mesurer le Point de Regard (Point Of Regard - POR) de l'utilisateur sur une position de surface adéquate (perpendiculairement planaire) sur laquelle les points de calibration sont affichés.

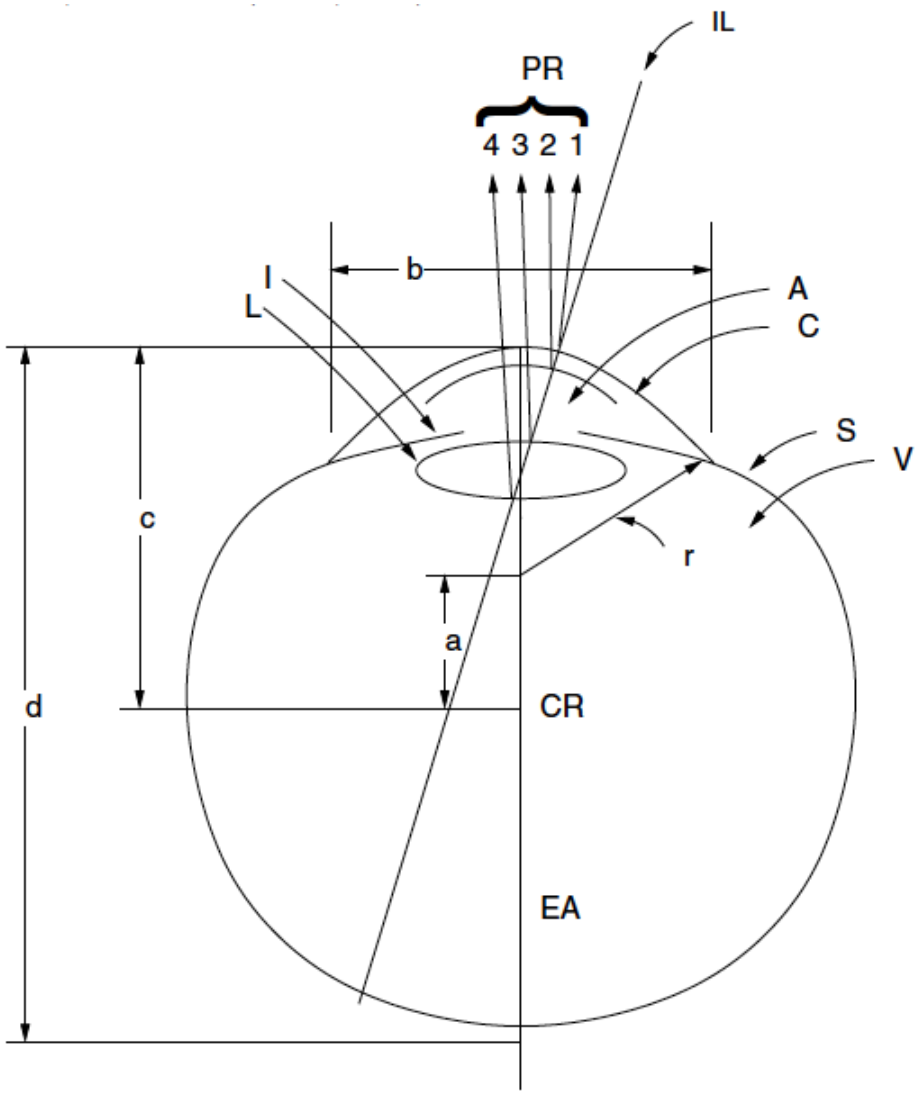


FIGURE 1.7: Reflets de Purkinje.

Deux points de référence sur l'œil sont nécessaires pour séparer les mouvements des yeux et les mouvements de la tête. La différence de position entre le centre de la pupille et le reflet cornéen change avec la rotation pure de l'œil, mais elle est relativement constante avec les petits mouvements de la tête. L'approximation des positions relatives de la pupille et des premières réflexions de Purkinje sont représentées graphiquement dans la figure 1.8, correspondant à l'œil gauche qui tourne pour fixer les neuf points de calibration placés. La réflexion de Purkinje est indiquée par un petit cercle blanc à proximité de la pupille, représentée par un cercle noir. La source de la lumière infra-rouge est généralement placée à une position fixe par rapport à l'œil. Les PR - les reflets de Purkinje sont : 1, la réflexion de la surface avant de la cornée ; 2, la réflexion de la surface arrière de la cornée ; 3, la

réflexion de la surface avant de la lentille ; 4, la réflexion de la surface arrière de la lentille (presque de la même taille et formée dans le même plan que la première image de Purkinje, mais à cause du changement de l'indice de réfraction à l'arrière de la lentille, l'intensité est inférieure de 1% à celle de la première image de Purkinje) ; IL, la lumière entrante ; A, l'humeur aqueuse ; C, la cornée ; S, la sclère ; V, l'humeur vitrée ; I, l'iris ; L, la lentille ; CR, le centre de rotation ; EA, l'axe des yeux ; $a \approx 6$mm ; $b \approx 12.5$mm ; $c \approx 13.5$mm ; $d \approx 24$mm ; $r \approx 7.8$mm (Crane, 1994).

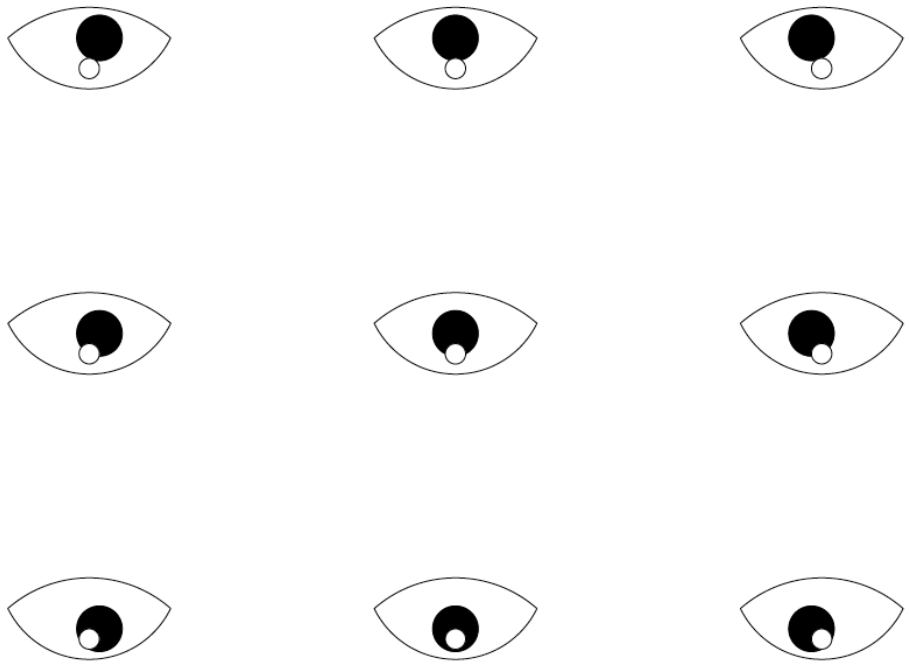


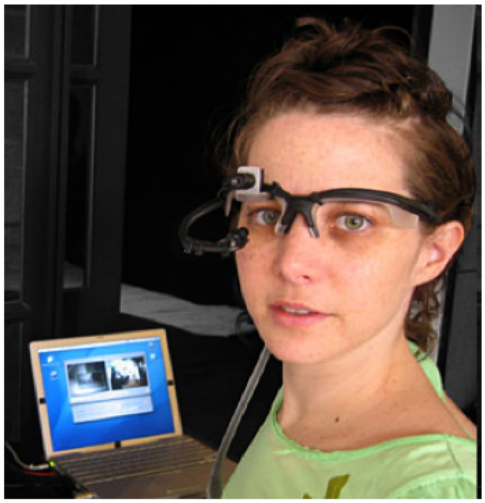
FIGURE 1.8: Positions relatives de la pupille et des premières images de Purkinje vu par la caméra du système de suivi.

Avec l'augmentation de la vitesse des processeurs d'ordinateur et de l'amélioration des techniques de vision par ordinateur, cette méthode est de plus en plus étudiée. Depuis le début du $XXI^{\text{ème}}$ siècle, il est apparu plusieurs approches et techniques basées sur cette méthode, nous allons lister quelques systèmes typiques utilisant cette méthode :

1.2.4.1 Système de suivi avec un casque avec caméra

Le système de suivi du regard avec un casque avec caméra par exemple le système de Dongheng Li [50] utilise un casque attaché à la tête de l'utilisateur. Ce casque contient une caméra placée devant l'œil de l'utilisateur pour capturer l'image de l'œil, et une caméra placée de manière opposée pour capturer la scène visuelle regardée par l'utilisateur, voir la figure 1.9. Deux caméras sont fixées sur la tête de l'utilisateur, donc l'image de l'œil est fixée par rapport à l'image de la scène visuelle. Donc, ce système résout le problème des mouvements de la tête, l'utilisateur peut bouger la tête librement lors de suivi du regard. Il existe plusieurs systèmes utilisant la même technique dans différents laboratoires de recherche ou entreprises comme **Eye tracking labs at the**

MPI (*http ://www.mpi.nl/world/tg/eye-tracking/eye-tracking.html*), le système **Perfech** (*http ://www.pertech.fr/*), etc.



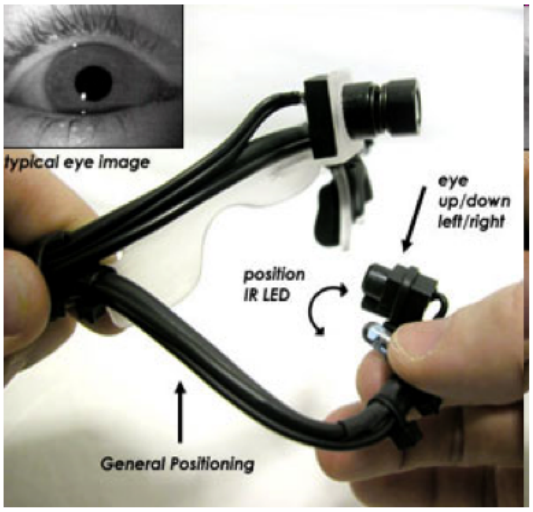


FIGURE 1.9: Système de suivi du regard de Dongheng Li.

Avantages
- peu coûteux.
- La tête n'a pas besoin de rester immobile.
- Le système peut suivre le regard en dehors de l'écran d'ordinateur, parce que les deux caméras attachées à la tête sont indépendantes de l'ordinateur.
- technique assez précise [50].

Inconvénients
- C'est le système intrusif, les utilisateurs doivent porter une casque avec caméra, ce n'est pas pratique.

1.2.4.2 Système non-intrusif avec deux caméras et un infra-rouge

Il existe des systèmes non-intrusifs par exemple [123] [73] [74] permettent à l'utilisateur de bouger la tête librement avec deux caméras et un infra-rouge. La figure 1.10 présente le système de suivi du regard de Takehiko Ohno [73]. Ces systèmes utilisent un infra-rouge pour trouver la réflexion de la cornée et utilisent deux caméras pour la calibration.

Avantages
- Non-intrusif, l'utilisateur n'a pas besoin de porter des équipements sur la tête.
- La tête est libre, elle n'a pas besoin de rester immobile.

Inconvénients
- Bien que peu coûteux, ces équipements gênent l'utilisateur par l'installation de deux caméras et d'un infrarouge. Par cet inconvénient, cette méthode est peu utilisée dans la pratique.

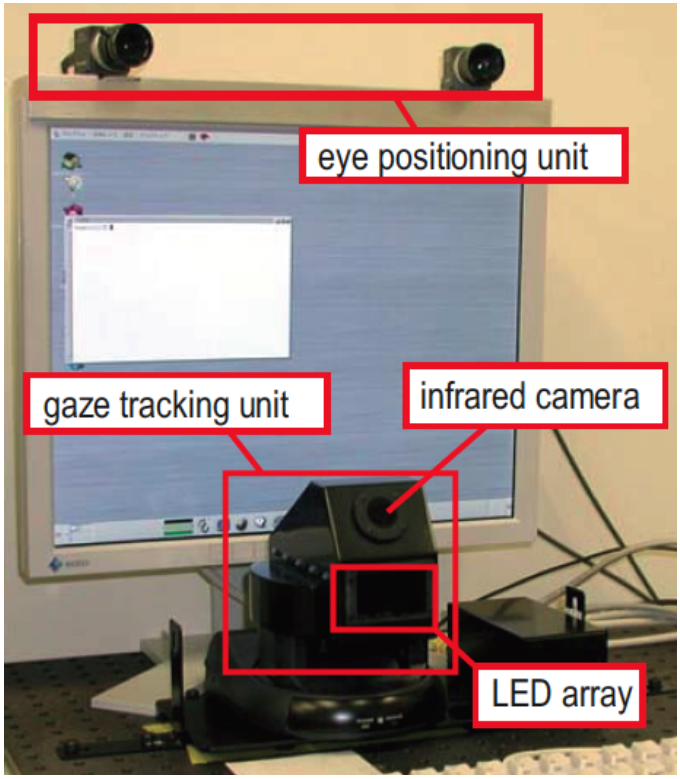


FIGURE 1.10: Système de suivi du regard de Takehiko Ohno.

1.2.4.3 Système non-intrusif avec un caméra et un infra-rouge

Il y a des systèmes de suivi du regard non-intrusifs utilisant seulement une caméra et un infra-rouge pour par exemple détecter le point de regard [79] [3] [97] [33] [68] [117] [75] [19]. En général, ces systèmes utilisent des réseaux neuronaux pour entraîner la fonction d'apprentissage et pour détecter de point de regarde, et ainsi que l'infra-rouge pour détecter la réflexion cornéenne.

Avantages

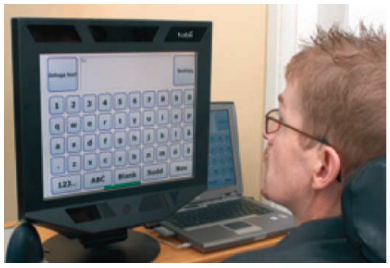
– Non-intrusive, l'utilisateur n'a pas besoin de porter des équipements sur la tête.

Inconvénients

– La tête doit être fixée. Si la tête de l'utilisateur s'éloigne de la position au moment de la calibration, la précision du système de suivi du regard baisse de façon spectaculaire.
– Le système a besoin d'installer un infra-rouge.

1.3 Systèmes sur le marché

Au cours des dernières années, il est apparu des produits de suivi du regard non intrusif sur le marché, par exemple **MyTobii** (*www.tobii.com*), **Visioboard** (*www.delta7.asso.fr*), **Eye Gaze**(*www.eyegaze.com*),etc. Ils permettent des mesures précises et autorisent l'utilisateur à bouger librement sa tête. Ce sont des produits complets, qui ont intégré des



(a) MyTobii.

(b) Visioboard.

(c) Eye Gaze.

FIGURE 1.11: Exemples des systèmes sur le marché.

équipements spécifiques comme un écran, des infra-rouges, des caméras spécifiques, etc. (voir la figure 1.11)C'est la raison pour laquelle leurs prix sont très élevés :

- **MyTobii** le prix est environ de 17 000 €[60].
- **Visioboard**, environ 27 000 €[60].
- **EyeGaze**, environ 21 000 €[60].

Ce prix empêche une large utilisation.

1.4 Système proposé

Nous allons proposer un système de suivi du regard qui pallie aux inconvénients ci-dessus :

- Ce système utilise seulement un caméra numérique simple, il marche avec la caméra numérique intégrée dans la plupart des ordinateurs portables d'aujourd'hui.
- Ce système n'a pas besoin d'équipements spéciaux attachés à l'utilisateur ni d'équipements spécifiques comme une caméra infrarouge.
- Ce système ne doit pas gêner l'utilisateur dans ces mouvements. L'utilisateur doit pouvoir bouger librement sa tête lors de la session.

Pour cela nous proposons un modèle du système de suivi du regard représenté dans la figure 1.12.

Pour détecter les yeux, nous utilisons la méthode de détection des objets basée sur les caractéristiques de Haar [106], [52] pour détecter les yeux sur le visage de l'utilisateur. Cette méthode entraîne des classificateurs avec quelques milliers d'échantillons d'image d'objets et construit une cascade de classificateurs afin de détecter rapidement l'objet.

Pour suivre les yeux, nous recherchons des coins sur le visage de l'utilisateur. Nous trouvons les coins sur les yeux, le nez et la bouche de l'utilisateur et nous utilisons l'algorithme de Lucas Kanade [7] pour suivre les coins. Puis nous utilisons la méthode de détection Outliers [69] pour détecter les coins qui sont perdus de suivi et les corriger.

Pour détecter le point de regard sur l'écran, nous devons trouver les liens fonctionnels entre l'œil de l'utilisateur et le point sur l'écran où l'utilisateur regarde. Nous n'utilisons pas de réseaux de neurones pour former cette fonction, mais nous allons estimer la distribution de cette fonction. Nous utilisons le Processus Gaussien pour calculer la moyenne et la covariance de cette fonction et utiliser cette fonction pour faire des prévisions sur les nouvelles entrées non vues dans les données.

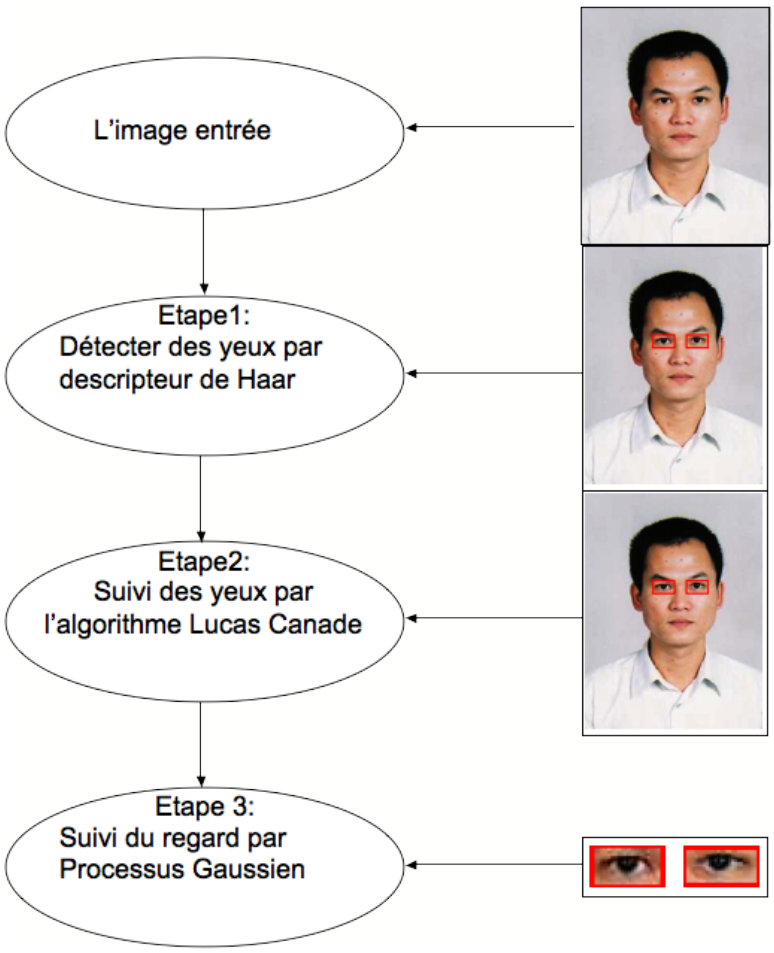


FIGURE 1.12: Modèle proposé de suivi du regard.

Pour résoudre le problème des mouvements de la tête, l'utilisateur doit effectuer une procédure de calibrage pour obtenir la fonction de suivi du regard sous les différences des angles de vue de la tête. Après la calibration, l'utilisateur pourra déplacer librement la tête devant la caméra, améliorant ainsi le confort de l'utilisateur.

1.5 Conclusion

Dans ce chapitre, nous avons décrit les systèmes existants, ses techniques utilisées, ses avantages et désavantages de chacun et notre solution proposée pour un nouveau système de suivi du regard. Nous allons décrire le détail de ces méthodes dans les chapitres suivants.

Chapitre 2

Étude des composantes faciales

2.1 Introduction

Les attributs du visage jouent le rôle le plus important dans le processus de communication. Ils fournissent des informations de l'état émotionnel d'une personne. Selon Albert Mehrabian, psychologue américain [63], dans les situations de communication 55% du message est exprimé par l'expression faciale, 38% par le message verbal prononcé et à 7% par le sens des mots. L'analyse du visage est une discipline dont les premiers travaux ont été proposés d'abord en psychologie. Ensuite, d'autres communautés de recherche s'y sont penchées motivées par le progrès de la vision par ordinateur et par la diversité des applications qui en découlent.

Les applications d'analyse du visage et de ses composantes faciales sont nombreuses notamment en :

- *Vidéo surveillance :* Cette application sert à la surveillance de site sensible. Elle intègre dans son processus l'analyse du visage afin de valider si l'objet détecté est considérer comme un individu.
- *Biométrie :* Les moyens classiques d'identification des personnes facilitent l'usurpation d'identité suite à plusieurs faille dans le système. Un changement de modalité biométrique est nécessaire plusieurs caractéristiques physiques peuvent être utile pour renforcer la sécurité des biens et des informations tels que le visage, et les yeux via un système de reconnaissance par l'iris.
- *Analyse comportementale :* Les différentes postures du visage, de déformations de la bouche, et du positionnement de l'iris, nous permettent par exemple, d'analyse le comportement du client vis à vie d'un produit mis en valeur par un commercial.
- *Réalité virtuelle :* L'utilisation des informations bas niveau issue de l'analyse du visage et des composantes faciales peuvent servir à définir des données sémantique de haut niveau, qui seront utile pour produire des visages animés pour chaque individu. Actuellement, la production de films utilise des contenus virtuels en utilisant l'analyse des comportements des vrais acteurs (mouvement du corps humain, orientation du visage etc...).

Nous présentons dans ce chapitre un état de l'art sur la modélisation des différentes composantes faciales, notamment les modèles issus de règles de datamining.

2.2 Modélisation de la couleur peau

La détection de peau est une étape préliminaire indispensable pour de nombreux problèmes de reconnaissance de formes. Un visage humain a une distribution de couleur qui diffère souvent de celle des objets de fond de l'image. Cette différence a poussé les chercheurs à s'intéresser essentiellement à l'indice couleur, celui-ci étant la primitive la plus simple à calculer et la plus riche. Plusieurs types d'espaces de couleurs ont été utilisés en appliquant certaines règles de décision pour localiser les pixels de peau. Chaque espace de couleur est modélisé par un cube où chacun des axes du référentiel représente une composante de base. La construction d'un détecteur de teinte chair, au niveau du pixel, soulève deux problèmes : le choix de l'espace colorimétrique et la modélisation de la teinte chair. Le lecteur trouvera dans [81] [104] une synthèse sur la modélisation de la peau.

Brand et Mason [9], ont proposé un modèle, de détection de visages appelé (Skin Probability Map), qui se base sur un corpus d'exemple de peau (80 millions) et de non-peau (1.600 millions). Les données sont intégrées dans deux histogrammes l'un contient des pixels de non peau et l'autre de peau. Ensuite, un tableau contenant le rapport de vraisemblance dans l'espace RGB et calculé à partir des deux histogrammes. Sigal et al [93] initialisent le suivi de la teinte chair en modélisant la peau et la non-peau dans l'espace RGB par un histogramme. Mais la forte corrélation entre les canaux R, G et B et le mélange des informations de luminance et de chrominance n'en font pas un espace de prédilection en détection de la couleur peau. Vieux et al [105] ont proposé une modélisation de la peau qui est utilisée à des fins de compression vidéo, qui permet de transmettre une vidéo qui est focalisée sur le visage. Hadid et al [28] applique un seuillage particulier pour détecter des pixels de peau dan le plan $r - b$ pour chaque type de caméra utilisée. Yang et Waibel [111] [112] modélisent la couleur peau par une Gaussienne dans le plan $r - g$. Oliver et al [76] modélisent la peau par un mélange de Gaussiennes, dans $r - g$, appris et mis à jour par l'algorithme E-M.

Dans le cas de la couleur de peau, l'espace HSV permet de rejeter les pixels peu colorés à monochrome, donc de saturation faible, qui sont généralement des pixels de fond. Sigal et al [93] utilisent les trois canaux de HSV dans un système de suivi de la teinte chair, et modélisent la couleur peau et la non-peau par des histogrammes. Zhu et al [122] segmentent la main en modélisant la peau par une Gaussienne et la non-peau par un mélange de Gaussiennes (une pour la peau et quatre pour la non-peau), appris par un algorithme E-M restreint, dans le plan H-S. McKenna et al [61] modélisent la couleur peau par un mélange de Gaussiennes dans le plan$H$-S, appris par l'algorithme E-M. Un modèle statique est utilisé dans la première image, et les régions de forte vraisemblance et de taille suffisante sont regroupées. Saxe et Foulds [87] suivent le visage et les mains par intersection d'histogrammes dans le plan H-S. Un premier histogramme modèle est défini par l'utilisateur, une rétine 5×5 parcourt l'image en calculant l'intersection entre l'histogramme modèle et celui de la rétine. Le visage et les mains sont alors localisées et l'histogramme modèle mis à jour.

Seguier [89] modélise la teinte chair dans une séquence d'image de manière adaptative en analysant les histogrammes C_b et C_r sur plusieurs images, retournées par une détection de mouvement et une détection d'ellipse. Wang et Brandstein [108] définissent la couleur peau par un intervalle donné du canal I de l'espace YIQ. Garcia et Tziritas [27] définissent

la localisation de la teinte chair dans l'espace YC_bC_r par deux régions (une pour les pixels de luminance $Y > 128$, une autre pour $Y < 128$) délimitées par des plans. Hu et al [36] partitionnent les valeurs du canal Y en 6 intervalles, et construisent un histogramme de la chrominance $C_b - C_r$ de la couleur peau pour chaque intervalle. 43 millions d'exemples de peau sont utilisés en apprentissage. Campadelli et al [12] modélisent la couleur peau par un mélange de deux Gaussiennes, appris par E-M, dans le plan P_bP_r. Phung et al [80] déterminent trois clusters Gaussiens dans l'espace YC_bC_r, par k-moyennes, modélisant la peau. Hsu et al [35] compensent l'effet de la luminosité par une transformation de la luma dans un premier temps puis transforment non-linéairement l'espace YC_bC_r et définissent une frontière elliptique dans le plan de chrominance obtenue. Menser et Brunig [64] modélisent la teinte chair par une Gaussienne dans le plan C_bC_r. Collobert et al [14] utilisent une table pré-calculée indexant les pixels de couleur peau dans le plan U-V. Ce modèle est réutilisé par Saber et Tekalp [86] qui modélisent la couleur de peau par une Gaussienne dans le plan E-S de l'espace YES.

Yang et Ahuja [113] modélisent la distribution de la couleur peau dans le plan $u-v$ de l'espace Luv par une Gaussienne. Dans [114] ils utilisent un mélange de deux Gaussiennes, appris par l'algorithme d'Expectation-Maximization, pour modéliser la teinte chair dans le plan $u - v$, en justifiant par des tests statistiques le choix d'un mélange plutôt que d'une Gaussienne seule et le choix du nombre de Gaussiennes. Schumeyer et Barner [88] utilisent l'espace Lab pour encoder les mains et le visage avec une meilleures résolution dans des séquences de langage des signes. Cai et Goshtasby [11] modélisent la couleur peau par un histogramme dans le plan $a - b$. Li et al [51] utilisent le même système pour détecter et poursuivre des visages dans une séquence d'images.

Hsu et al [35] représentent la distribution de la teinte chair dans les espaces YC_bC_r, HSV et Yrg notamment, et illustrent la dépendance non-linéaire de la chrominance à la luminance. Brand et Mason [8] comparent trois méthodes de classification dans des espaces différents : seuillage du rapport R/G, seuillage du canal I de YIQ et rétroprojection du rapport de vraisemblance de la teinte chair dans RGB [40]. Terrillon et al [100] comparent deux modèles paramétriques, une Gaussienne et un mélange de Gaussiennes, de la couleur peau dans neuf plans définissant la chrominance, dont les plans H-S, $rgnormalisé$ et ab. Shin et al [92] étudient l'influence de la représentation des couleurs en comparant la séparabilité des clusters de pixels de peau et de non-peau sur la base de quatre métriques (matrices de diffusion et intersections d'histogrammes) sur neuf espaces colorimétriques différents, dont RGB, $RGBnormalisé$, HSI, $YCbCr$, YIQ, YUV et Lab.

2.3 Analyse du visage

Dans la littérature de nombreuses approches ont été proposées pour l'analyse du visage, qu'on peut classer en 3 classes :

1. **Approches structurelles à base de connaissances :** Cette méthode s'intéresse au visage et ses composantes comme la bouche, le nez et les yeux. Par exemple, la composante faciale comme les yeux sont symétriques entre eux. Ces informations connues sont utilisées dans un processus qui permet de localiser les composantes faciales nez, bouche, yeux, et puis d'analyser les relations spatiales entre elles à

partir de certaines règles prédéfinies. Une phase de vérification est nécessaire pour éliminer les fausses détections. Les inconvénients de cette approche résident dans la difficulté de traduire les connaissances humaines afin de tenir en compte de tous les cas possibles qui permettent de parvenir à une détection correcte du visage. Il est difficile aussi de prolonger cette approche à des situations où le visage a différentes postures car il est difficile d'énumérer tous les cas.

2. **Approches basées sur des caractéristiques visuelles invariables :** L'idée principale de cette approche est de chercher des caractéristiques invariantes pour la détection du visage quelque soit les conditions d'éclairage et de posture du visage dans des séquences d'images tels que la texture, la couleur chair , niveau de gris, etc.... L'information de texture nous permet de localiser le visage et ses composantes faciales puisque chaque zone du corps humain à une texture différente de celle du visage. Par exemple : la peau, les cheveux, et d'autres attributs peuvent être utilisés pour séparer chaque élément du corps humain. La bouche les yeux ont aussi leur propre texture. Les informations de couleur de peau peuvent servir pour réduire l'espace de recherche qui contient des zones du visage. Bien que, la couleur de la peau soit variable selon les personnes. Des études ont montré que la différence principale de la couleur de la peau se trouve en grande partie par la différence d'intensité plutôt que de la chromaticité. Les espaces les plus couramment utilisés sont les espaces HSV et YC_rC_b. D.Saxe et R.Foulds [87] ont proposé un modèle d'identification de peau itérative dans l'espace HSV en utilisant l'intersection d'histogramme. Initialement, l'utilisateur sélectionne Une zone de pixels de couleur de peau qu'on appel graine de commande. Ensuite, l'algorithme itératif commence à comparer l'histogramme de commande de la région sélectionné et l'histogramme des autres régions jusqu'à la localisation des pixels de peau. Dans le chapitre 4 nous présentons en détails un état de l'art des modèles basés sur des règles de décision de la couleur de peau. Cette méthode dépend de certaines contraintes, par exemple des zones du visage exposé à la lumière conduit souvent à des résultats incorrects ou incomplets.

   Dans des images au niveau de gris le visage contient un ensemble de zone tels que l'iris, les lèvres les narines et les sourcils qui sont généralement plus sombres que les régions connexes. De nombreuses méthodes combinées, basés sur des caractéristiques faciales pour localiser le visage, tels que la forme de la peau, la taille, la couleur, et le mouvement. Après la localisation du visage, une phase de vérification est utilisée en s'appuyant sur des caractéristiques locales, tels que les yeux, le nez, les cheveux pour confirmer la détection.

   Une autre méthode proposée par Sobottka et al. [96] proposent une méthode de localisation des visages et d'extraction des composantes faciales, en utilisant la forme et la couleur. La localisation des régions de peau est effectuée en appliquant certaines règles de décision dans l'espace HSV. Les attributs du visage comme les yeux et la bouche sont extraites, grâce aux hypothèses qu'elles sont plus sombres que le reste du visage. Un algorithme de croissance de régions est ensuite appliqué sur l'image segmentée permet de connecter les composantes correspondant aux régions de la

peau. Une ellipse modélisant le visage est détectée par les moments d'inertie des composantes connectées. Dans [116], on peut trouver un état de l'art très exhaustif sur les différentes méthodes de détection des visages.

3. **Approches basées sur les modèles d'apparence :** Elles utilisent des techniques d'apprentissage pour trouver des caractéristiques discriminantes du visage et du non visage. Ces techniques donnent de bons résultats lorsque la personne est de face. Différents types de méthodes par apprentissage ont été développés :

   Approche de mise en correspondance : Elle prend une image quelconque qui contient un visage frontal prédéfini. Le modèle de visage est obtenu par une moyenne de plusieurs visages. Pour renforcer la qualité du visage, des opérations de pré-traitement sont appliquées tel que l'égalisation de l'histogramme, et la suppression de la zone de fond. La comparaison s'effectue selon l'intensité des pixels entre le modèle du visage prédéfini et plusieurs sous-régions de l'image de test que l'on désire analyser. L'inconvénient de cette approche est qu'il ne peut pas tenir en compte des différents modèles du visage tels que la tailles, la posture, ainsi que la variabilité de la luminance de l'image modèle et les images de teste. Yuille et al [119], ont proposé un modèle actif de forme qui utilise des caractéristiques locales de l'image tels que le contour et l'intensité. La courbe initiale dans l'image à tester se déforme jusqu'à ce qu'il trouve les mêmes caractéristiques du modèle prédéfini à priori. Bien que leurs résultats expérimentaux démontrent la bonne exécution en détectant des composantes, l'inconvénient de cette approche c'est d'avoir la courbe initiale à proximité de l'objet d'intérêt. Si la courbe est placée sous les yeux, elle peut être attirée par les sourcils.

   Approche par classification linéaire : Yang et al [115] développent deux approches de la détection de visages sur des vignettes 20 x 20 en niveau de gris : une méthode générative, s'appuyant sur l'analyse factorielle, et une méthode discriminante, l'analyse discriminante linéaire. L'analyse factorielle est similaire à l'analyse en composante principale, à la différence qu'elle cherche à trouver une représentation des données dans une dimension moindre maximisant la corrélation entre les entrées et non la variance.

   Les réseaux de neurones : Sung et Poggio [98] utilisent une approche similaire, combinée à un réseau de neurones discriminant. Les vignettes traitées sont de taille 19×19, les pixels de bords sont masqués : l'espace image est de dimension 283 ($19 \times 19 = 361$). Le prétraitement appliqué, qui sera repris par nombre d'auteurs par la suite, consiste en une correction de la luminosité par soustraction de la surface modélisant le mieux la luminosité (une fonction linéaire est ajustée au niveau de gris de la vignette).

   Les Support Vector Machines (SVM) : Un classificateur S.V.M est un classificateur linéaire dans un espace de grande dimension, où l'hyperplan séparateur est choisi

minimiser l'erreur de classification attendue, et pour maximiser la marge de classification (la marge est l'hyperplan séparateur le plus éloigné de toutes les classes). Cet hyperplan est défini par une combinaison de poids d'un petit sous ensemble des vecteurs d'entraînements, appelés les "Support Vectors". Les S.V.M projettent donc les données (rectangle du visage) dans cet espace de plus grande dimension, formant une surface de décision linéaire (l'hyperplan séparateur) entre la projection des visages et la projection des non-visages.

2.3.1 Analyse labiale

Diverses techniques ont déjà été proposées pour extraire des informations visuelles associées à la zone de bouche, qu'on peut classer en 2 grandes familles :

- Approche pixelique : C'est une méthode de bas niveau qui n'utilise que les informations des pixels de l'image. Elle suppose que les pixels des lèvres possèdent des caractéristiques homogènes et différentes de celles de la peau.
- Approche «forme et/ou apparence» : Cette une méthode basée sur des modèles caractéristiques des lèvres, obtenus de manière heuristique ou statistique. Elle détecte les contours des lèvres en utilisant des modèles plus ou moins complexes de la forme de la bouche. Cette approche peut utiliser des modèles pour décrire à la fois la forme et l'apparence.

Il est à noter que ces approches ne répondent pas forcément aux même objectifs de précision et de finesse et ne sont pas nécessairement mutuellement exclusives, ainsi une méthode de type «pixel» peut par exemple être utilisée comme prétraitement d'une méthode utilisant un modèle de forme de lèvre.

2.3.1.1 Approches pixeliques

Les méthodes de bas niveau pour la segmentation des lèvres s'effectuent plus souvent en deux étapes : la localisation par des opérateurs de morphologie mathématique, et par un seuillage qui permet de ressortir les lèvres. Ensuite, d'autres attributs visuels de ces régions, tels que la hauteur, la largeur, la surface et le périmètre, sont exploités pour l'analyse de lèvres.

Les primitives colorimétriques des lèvres sont homogènes et sont facilement détectables, par un simple seuillage, par rapport à la couleur de la peau. Cependant, ce processus est limité d'une part par les conditions d'illumination qui différent d'une base d'images à une autre et d'autre part par la difficulté d'extraction de contours fins de la bouche.

Ces dernières années de nombreuses études concernant la caractérisation de la couleur des lèvres et de la teinte peau ont montré que la principale différence réside dans le choix du canal de l'espace de représentation de la couleur.

Nous avons présenté dans la section précédente les différents espaces colorimétriques, leur intérêt et inconvénient pour la détection de la couleur de la peau et des composantes faciales. De manière perceptuelle, la couleur visible d'un objet est définie par trois composantes : La luminosité, la teinte et la saturation. La luminosité est la lumière diffusé par l'objet. La teinte est relative à la longueur d'onde et la saturation à la pureté de la couleur.

La répartition de la couleur des lèvres et de la teinte chair a été largement étudié essentiellement dans deux espaces colorimétriques qui exploitent la teinte et la chrominance : HSV et YC_bC_r.

Les techniques les plus répandues de cette famille reviennent à faire des seuillages sur une grandeur colorimétrique jugé pertinente. Ce principe est par exemple utilisé dans [58], où l'on procédait à un double seuillage de l'image de luminance et de la composante Rouge. Pour s'affranchir de la dépendance à l'éclairage des simples niveaux de gris ou du RVB, de nombreux auteurs ont proposé d'utiliser d'autres espaces couleurs, où la séparation entre les lèvres et le reste du visage serait facilité grâce à l'utilisation de composantes chromatiques.

Dans l'espace RVB, les pixels des lèvres et ceux de la peau ont des composantes sensiblement différentes. Dans les deux cas, le rouge est dominant, mais il y a plus de vert que de bleu dans la peau alors que ces couleurs sont en quantité égale dans les lèvres. Ainsi la peau apparaît plus jaune que les lèvres car la différence entre le rouge et le vert est plus importante pour les lèvres que pour la peau.

Chiou et Hwang [13] utilisent le rapport $Q = R/G$ pour caractériser les lèvres, suivi d'un double seuillage haut et bas de Q pour segmenter les lèvres. De même Wark et al [109],. utilisent un seuillage sur le quotient R/G, complété par des opérations morphologiques et par l'ajustement d'un modèle paramétrique sur la zone binaire extraite. Dans [57] la détection de la bouche est effectué par un seuillage adaptatif sur Q. Dans [121], Zhang et Mersereau font remarquer que les composantes colorimétriques des lèvres et de la peau sont relativement uniformes dans les espaces (C_r, C_b) et (r, g, b). Cependant, leurs distributions se chevauchent beaucoup trop souvent, ce qui rend la segmentation difficile. A l'opposé, ils constatent que la teinte (H) des lèvres est relativement constante et bien séparé de celle de la peau. Ils proposent donc de localiser les lèvres par un seuillage sur la teinte, et sur la saturation de manière à ne garder que les pixels fortement colorés.

Dans [35], Hsu et al. utilisent une transformation intéressante pour faire ressortir les lèvres. Ils montrent que $C_r/C_b - C_r^2$ est beaucoup plus important pour les lèvres que pour le reste du visage. Après quelques opérations morphologiques, ils seuillent cette grandeur pour localiser la bouche.

Alan liew et al [55] traite de l'utilisation des techniques d'agrégation floues, où les pixels sont classés à partir d'une carte d'appartenance aux lèvres dans l'espace Luv et Lab. Les champs de Markov aléatoires ont été utilisés dans [53], afin d'obtenir une segmentation hiérarchique incluant des paramètres colorimétriques mais aussi de mouvement, avec la particularité que le voisinage spatiotemporel des pixels est pris en compte lors de la classification. Enfin, diverses techniques nécessitant au préalable un étiquetage manuel d'une base d'apprentissage afin d'obtenir des connaissances colorimétriques de référence ont également été développés.

L'Analyse Discriminante Linéaire a par exemple été utilisé dans [71] afin d'obtenir une combinaison des composantes de l'espace effectuant la meilleure discrimination possible entre les lèvres et le reste du visage. Dans [78], les auteurs se servent des donnés d'apprentissage pour obtenir une estimation gaussiennes des distributions statistiques pour les classes de peau, de lèvre et de non-visage dans l'espace RVB, la classification étant ensuite effectuée par une classification bayesienne.

Ces approches ne prenant pas en compte des critères géométriques de forme. Ce type de méthode ne donne en général que des formes approximatives des lèvres et est particulièrement sensible au bruit.

2.3.1.2 Approches «formes et/ apparence»

Les méthodes de bas niveau décrit dans les sections précédentes sont des processus à forme libre. Elles n'intègrent aucune connaissance a priori sur les formes admissibles. A l'opposé, les méthodes de haut niveau, basées sur les modèles déformables ou modèles actifs d'apparence, sont basés sur des modèles caractéristiques des formes à segmenter, obtenus de manière heuristique ou statistique. Ces modèles génériques sont déformés de manière à être adaptés aux contours de l'objet.

Introduits par Yuille au début des années 90 [119], les modèles déformables analytiques (Deformable Templates) permettent de décrire une forme de manière très compacte à l'aide d'un ensemble de courbes paramétrés. Par exemple, le modèle générique de lèvres proposé par Yuille est constitué de 3 quartiques pour les contours extérieurs de la bouche et de 2 paraboles (confondues ou non, selon que la bouche est ouverte ou fermé) pour les contours internes. La variation des paramètres des courbes permet de faire évoluer le modèle correspondant vers les contours de l'objet à segmenter. Cette convergence est guidée par la minimisation d'une fonction d'énergie qui est la somme pondéré d'une énergie interne et d'une énergie externe. Dans le cas des snakes, l'énergie interne, paramétré par les termes d'élasticité et de courbure, impose une contrainte relativement faible sur la forme finalement obtenue ("le contour doit être lisse et compact"). A l'opposé, l'énergie interne des modèles déformables permet de favoriser ou de pénaliser explicitement certaines déformations de la structure. Les contraintes exprimées par l'énergie interne seront donc fortement heuristiques. Un exemple parmi d'autres d'applications des contours actifs à la zone labiale est donné dans [20].

Récemment, dans [70], il a été proposé d'utiliser des snakes «adaptatifs» utilisant un algorithme d'Estimation/Maximisation (EM, [62]). Les points de forts gradients de l'image, correspondant aux contours, sont séparés en plusieurs segments qui vont être classifiés comme étant valides ou non valides avec un certain degré de confiance qui est réévalué par l'algorithme EM à chaque itération de la convergence du snake. Cela revient à rendre le champ de forces extérieures guidant la convergence du snake dynamique et adaptatif au fur et à mesure de la convergence permettant en quelque sorte de débruiter l'image.

Néanmoins si les résultats se révèlent très satisfaisants si les conditions sont bonnes (éclairage contrôlé, bon contraste entre la couleur de la peau et celle des lèvres), les performances décroissent lorsque ces conditions favorables ne sont plus réunies, la phase d'initialisation du snake manquant de robustesse ce qui nécessite de complexifier la méthode en procédant en plusieurs phases de convergence.

Dans [34], les auteurs ont décidé d'améliorer l'approche par contours actifs en rajoutant une force intérieure baptisée « force template ». A chaque itération de la convergence du snake, la force template ramène le contour actif sur une forme admissible de lèvre. L'utilisation du template fournit en quelque sorte une connaissance a priori des formes possibles et contraint le snake à des déformations encadrées. Une autre approche plus

flexible a été proposée dans [25] qui consiste à utiliser quatre courbes paramétriques cubiques pour décrire le contour extérieur des lèvres en imposant conditions et limites aux dérivés des courbes au niveau des points saillants. Après une initialisation où un snake simplifié (sans force intérieure) donne une première estimation du contour supérieur et la détection d'un point sur le contour inférieur, les paramètres des cubiques sont optimisés en maximisant le flux de vecteurs gradients à travers les différentes courbes.

Cootes a présenté les Modèles Actifs de Forme (ASM, [16]). Les ASMs correspondent en fait à l'application des Modèles de Distribution de Points (Point Distribution Model, PDM, [15]) pour segmenter un objet sur une image. La méthode consiste à déplacer individuellement les points du contour de façon à les rapprocher des zones de forts gradients correspondant aux contours. Dans [120] Zhang et al, dans le cadre d'un PDM appliqué au visage humain entier, il a été proposé une version utilisant des contraintes sur les contours du visage et une description des points de contrôle du PDM (yeux, nez et bouche) dans une base de filtres de Gabor [18]. Les points présents sur les contours du visage doivent minimiser un critère faisant intervenir un modèle local de texture qui sera minimisé si les points sont placés sur des zones de forts gradients. Les points de contrôle du PDM sont, quant à eux, décrits par un jeu de 40 coefficients correspondant aux réponses des filtres de Gabor pour différents angles et échelles. L'utilisation de fonctions de similarités permet de définir à chaque itération des points présentant de meilleures réponses aux filtres. Le PDM intervient enfin pour régulariser à chaque étape de la convergence les déplacements de points sur des formes plausibles. La principale amélioration apportée aux ASMs peut être considérée comme la prise en compte de l'apparence et des valeurs de niveaux de gris.

Dans [54] Liew et al , les auteurs ont proposé d'utiliser une carte de probabilité résultant d'une classification floue des pixels en deux zones 'lèvre' et 'non lèvre' pour optimiser les paramètres d'un modèle déformable à trois paraboles décrivant le contour extérieur des lèvres. Dans [101] Tian et al, une méthode est présenté où un modèle de forme analytique à plusieurs états (ouvert, relativement fermé, étroitement fermé) est utilisé en parallèle à une description de la distribution colorimétrique des lèvres par une mixture de gaussienne. Un algorithme de suivi de mouvement inspiré par la méthode Lucas-Kanade [56] combinant les informations de forme et de couleurs permet alors d'effectuer la segmentation.

2.3.2 Analyse des yeux

De nombreuses méthodes permettent la détection des yeux, qu'on peut classer en deux catégories. Celles qui utilisent un éclairage spécifique afin de faciliter la recherche et celles qui localisent les yeux grâce à des caractéristiques anatomiques. En ce qui concerne la première catégorie l'éclairage produit un reflet très brillant sur l'iris. Celui-ci peut être facilement détecté par un simple seuillage. Les inconvénients de cette méthode sont multiples. En effet, pour estimer le mouvement de l'iris il faut étudier le mouvement d'un point particulier de celui-ci (le plus simple étant le centre de la pupille). Malheureusement le reflet détecté en utilisant un éclairage spécifique ne se trouve pas forcement au centre de la pupille. Il faut donc appliquer un deuxième traitement, mais cette fois-ci localement, pour localiser le centre de celle-ci. Un autre inconvénient de cette méthode est le fait qu'il puisse y avoir plusieurs reflets ; ils sont généralement causés par des réflexions parasites

en particulier chez les personnes portant des lunettes. Quant à la deuxième catégorie, les algorithmes proposés sont souvent lents car ils font appel à des processus itératifs.

2.3.2.1 Les méthodes d'apparence

Les méthodes de segmentation des yeux et de sourcils peuvent être classifiés dans deux approches principales : Approche basée sur l'information pixeliques (la luminance et/ou la chrominance), des approches déformables qui peuvent être libres ou par mise en correspondance.

Dans Kapmann et al [42], l'information de luminance dans une zone locale prédéfini est exploitée pour caractériser les régions de l'iris. Après la détection des centres des yeux,. Les sourcils sont détectés dans les zones au-dessus les plus foncés. D. Maio [59] précisent que les yeux et les sourcils (dans les visages verticaux) peuvent être détectées en utilisant la projection verticale/horizontale du gradient du visage. Dans Tsekeridou [102] les yeux et les sourcils sont recherchés dans la moitié supérieure du visage. La position y des yeux est évaluée par la localisation des minima du gradient de luminance du profil vertical du visage. Ko J-G et al [48] ont exploité d'autres caractéristiques pour la détection des yeux, et ont utilisé un seuillage adaptatif avec des contraintes morphologiques. Dans [95] P. Smith et al la détection des yeux est basée sur l'information de couleur. D'abord, la couleur de peau est détectée en utilisant le modèle de Kjedlsen [44], qui classifie le visage en deux régions peau et non-peau. Puisque les yeux ne sont pas des pixels de peau, elles doivent apparaître comme des trous qui doivent se trouver au-dessus de la région de lèvre et qui doivent satisfaire des critères géométriques par rapport au visage. Pour trouver l'iris les auteurs emploient l'information d'intensité de la région des yeux. Deng X [21] utilise conjointement l'intensité et le gradient de la composante luminance, et détecte les pics sur les projections verticale et horizontale des contours.

Les techniques précédentes n'obtiennent pas une segmentation fine des bords de l'iris et des yeux. Vezhnevets [103] présente un modèle basé contour pour détecter l'iris, qui est exécuté en trois étapes : la détection approximative du centre yeux, l'extraction de la forme de l'iris par une courbe polynomiale cubique et de l'extraction de la paupière.

Haro et al. [29] exploitent les propriétés physiques de l'iris et leur mouvement pour extraire des régions des yeux. Pour cela, ils utilisent un modèle probabiliste et un filtre de kalman pour le suivi des yeux. Singh et al. [94] ont proposé une approche qui sert à détecter la fatigue d'un conducteur. Dans un premier temps, le modèle localise le visage et traite les informations de couleur de peau. Ensuite, il réduit l'espace de recherche par l'analyse de l'intensité horizontal du gradient de la zone du visage, pour trouver la position exacte de l'iris. Les mesures de la fatigue du conducteur peuvent être déterminées par la vitesse de clignotements des yeux c'est –à-dire par le nombre d'image ou les yeux restent fermés. Kashima et autres [43] présentent une méthode de détection d'iris qui est adaptable dans différent orientation du visage. D'abord, la méthode détermine la couleur standard de peau par la différence des couleurs. Ensuite, les composantes faciales de bouche sont extraites à partir de la région de visage par des modèle hybride de mise en correspondance. La région ou se trouve l'iris et séparer de la partie blanche des yeux en appliquant l'opérateur de Prewitt. Pour obtenir le contour fine de l'iris la transformer de hough et utilisé. Weiner et autres [110] présentent un système pour la détection de

la direction du regard dans des images fixe du visage. D'abord, le visage est détecté par l'intermédiaire de la détection de peau. La peau est détecté en utilisant des règles sur les composantes chromatique C_r (rouge chromatique) et C_b (de bleu chromatique) de l'espace de couleur YC_bC_r de l'espace de couleur. Après la détection et l'extraction de visage, la détection des positions des yeux est effectué avec le canal C_r de l'image de couleur. Les contours exacte de l'iris sont déterminé par la transformer de Hough . Puis, détermination fine de la position centrale d'iris et le rayon de l'iris sont détecter en appliquant une autre version de la transformer de Hough dans une zone rectangulaire qui englobe les régions des yeux.

2.3.2.2 Méthodes déformables

Yuille [119] et all ont présenté un modèle déformable qui est guidée par un ensemble de paramètres qui ont une connaissance a priori de la forme des yeux. Le modèle prédéfinis par Yuille se compose d'un cercle pour l'iris et deux paraboles pour les deux paupières. Les modèles déformable ont quelques similitudes avec les snakes, mais, en outre ils ont L'avantage à évoluer explicitement sous les contraintes d'un modèle spécifique, au contraire modèle snake ou la déformation est effectuer sans aucune visibilité. Dans la méthode de Yuille les composants des yeux sont liés à trois forces correspondant à l'énergie interne. Les algorithmes plus rapides sont ceux qui n'utilisent pas les techniques d'optimisation itérative afin de minimiser la fonction d'énergie. Cette approche a été adaptée par Tian et al [101] . Le modèle des sourcils est composé de trois points caractéristiques de raccordement de deux segments. Pour les yeux un multi-modèle est défini. Le modèle des yeux ouvert correspond à deux paraboles pour les paupières et un cercle pour l'iris (neuf paramètres). Les yeux fermé sont modélisés par une ligne de démarcation entre les coins des yeux. La détection est basée sur d'initialisation dans les points caractéristiques des yeux et des sourcils. Ces points sont ensuite suivis dans les autres séquences d'images avec un algorithme de suivie proposé par Lucas-Kanade [56].

2.4 Synthèse et implémentation

2.4.1 Masque du visage

La détection des composantes du visage peut se faire à partir des résultats de l'extraction de la peau. Une des méthodes envisageable consiste à analyser les trous contenus dans la zone de peau, comme par exemple ceux générés par les yeux. La validation peut alors être réalisée en respectant différentes règles de positionnement (des trous entre eux et par rapport au visage) et de taille. La figure 2.1 présente le processus de détection d'un visage dans les espaces de couleur RGB, YUV, HSV.

2.4.2 Détection des lèvres

Pour détecter la bouche, on peut se baser sur la rougeur de la zone des lèvres. Après un lissage intensif des composantes C_r et C_b, elles sont normalisées. La méthode consiste

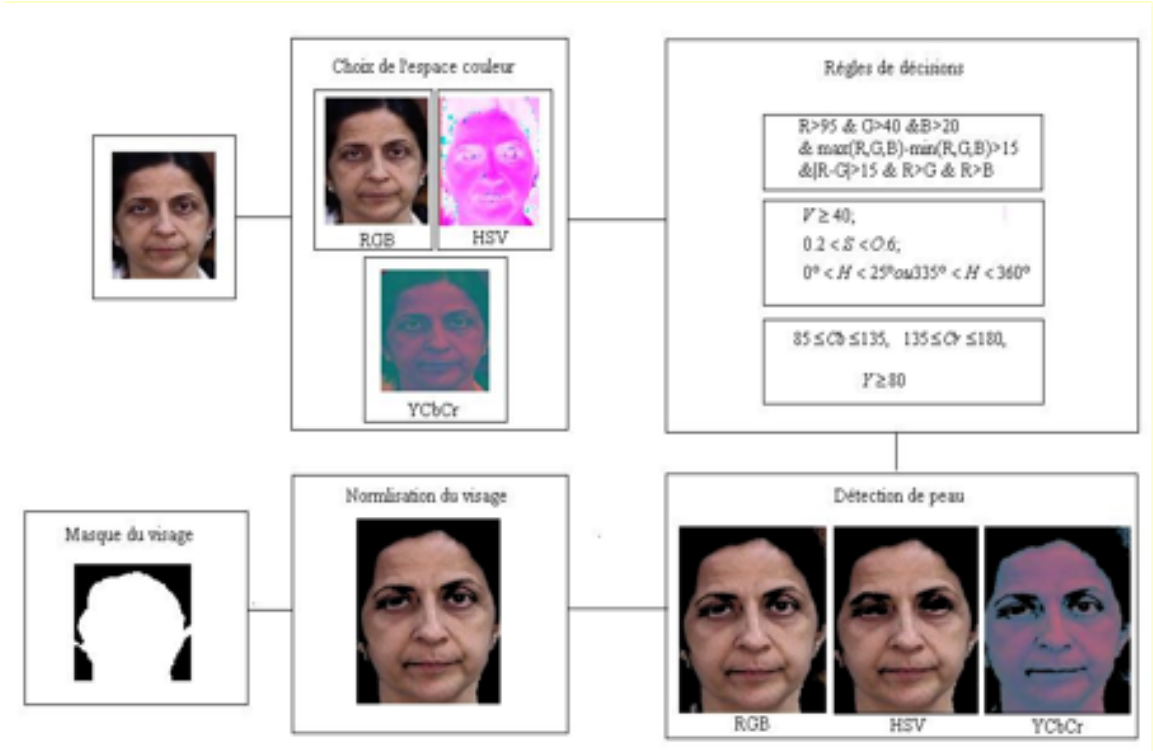


FIGURE 2.1: Processus de détection d'un visage dans les espaces de couleur RGB, YUV et HSV.

donc à évaluer une fonction de la couleur, prenant des fortes valeurs sur la composante rouge et de faible valeurs sur la composante bleue. Selon la fonction ci dessus, on obtient l'image MouthMap.

$$MouthMap = C_r^2 * (C_r^2 - \eta * C_r/C_b)^2$$

$$\text{où } \eta = \frac{\frac{1}{n}\sum C_r^2}{\frac{1}{n}\sum (C_r/C_b)}$$

La structure suivante est utilisée, pour recréer l'effet décrit dans la formule ci-dessus. Le modèle de détection des lèvres est présenté dans la figure 2.2.

Nous procédons ensuite à un seuillage de MouthMap en conservant de 1 à 10% des pixels avec les valeurs les plus élevés. La figure 2.3 montre quelques résultats de la localisation des lèvres.

2.4.3 Détection du nez

La détection des trous de nez est inspirée de la détection des yeux présenté précédemment. Elle exploite des propriétés de luminance de Y d'une image de nez (YC_rC_b), et des propriétés de luminance /color de V de la même image de nez dans l'espace HSV. Ces deux images sont lissées et normalisés. Ainsi NoseMap est calculée selon la formule suivante :

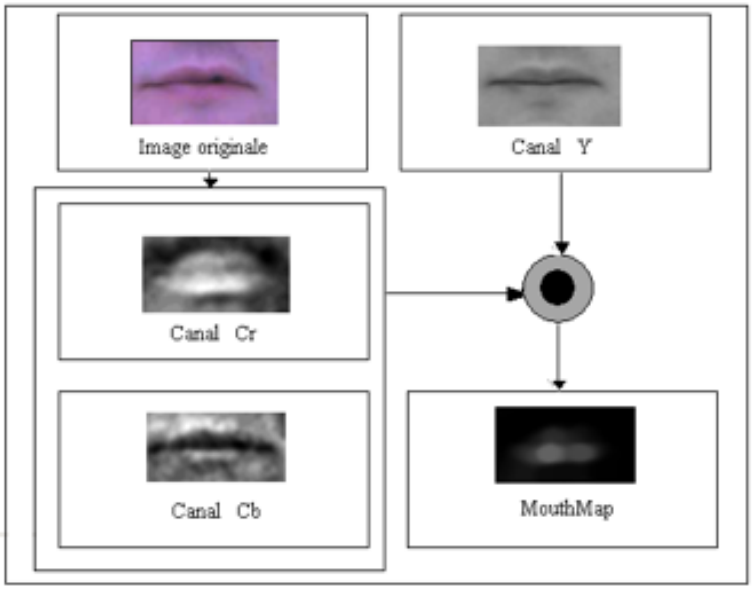


FIGURE 2.2: Modèle de détection des lèvres.

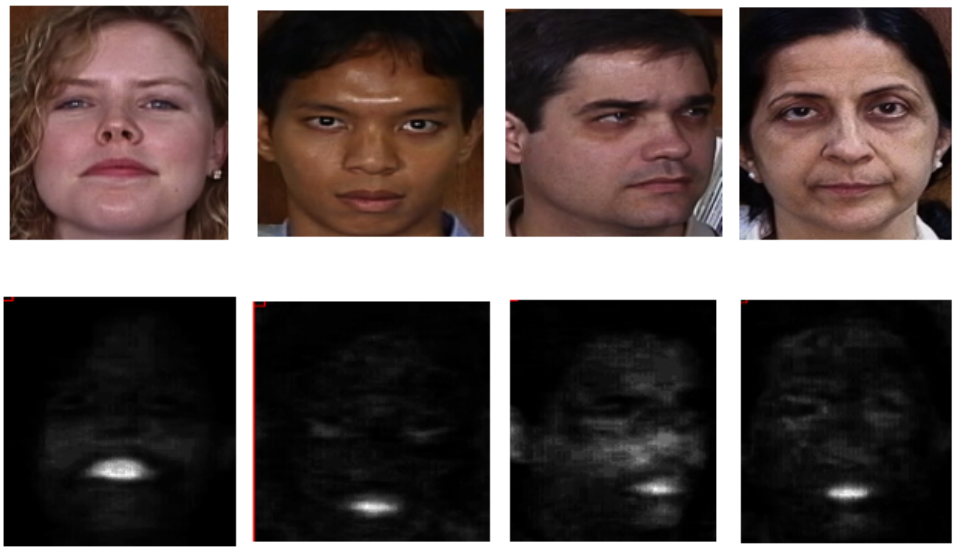


FIGURE 2.3: Les résultats de la localisation des lèvres Avec MouthMap.

$$NoseMap = (255 - V)^3 * (255 - Y)^3 \text{normalisé } [0..255]$$

La structure suivante est utilisée, pour recréer l'effet décrit dans la formule ci-dessus. Le modèle de détection du nez est présenté dans la figure 2.4.

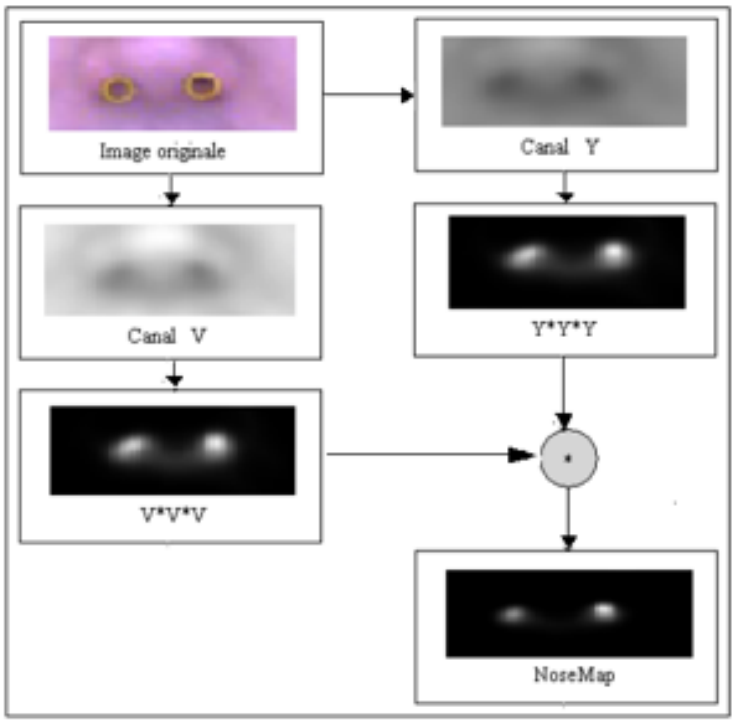


FIGURE 2.4: Modèle de détection du nez.

2.4.4 Détection des yeux

Les régions des yeux et de la bouche sont définis en effectuant des opérations dans l'espace de couleur YC_bC_r. Ces opérations produisent une image au niveau de gris avec des forts contraste dans les régions des traits du visages.

La chrominance de la carte des yeux $EyeMapC$ est calculée aussi après un certain nombre d'étapes de prétraitement. D'abord on lisse le canal de C_r et le canal de C_b tout à fait intensivement, pour éliminer le bruit. Particulièrement le canal bleu (C_b) a beaucoup de bruit dans l'image originale (les sondes de CMOS produisent beaucoup de bruit dans le domaine bleu de couleur). Ensuite, les deux canaux sont normalisés, avec des valeurs séparées.

L'analyse des composantes de chrominance indique, qu'au tour des contours des yeux, on a des fortes valeurs de C_b et de faible valeurs C_r. Afin de détecter les yeux on a utilisé une fonction de la couleur pour chaque pixels de l'image.

$$E_{chrom} = EyeMapC = \frac{1}{3}\{(C_b^2) + (\bar{C}_r)^2 + (C_b/C_r)\}$$

$\bar{C}_r$ est l'inverse de C_r (i.e. $255 - C_r$). Les pixels dont $C_r = 0$ ne sont pas calculés. Le résultat E_{chrom} est normalisé entre $[0, 255]$.

La structure suivante est utilisée, pour recréer l'effet décrit dans la formule ci-dessus. Le modèle de détection des yeux est présenté dans la figure 2.5.

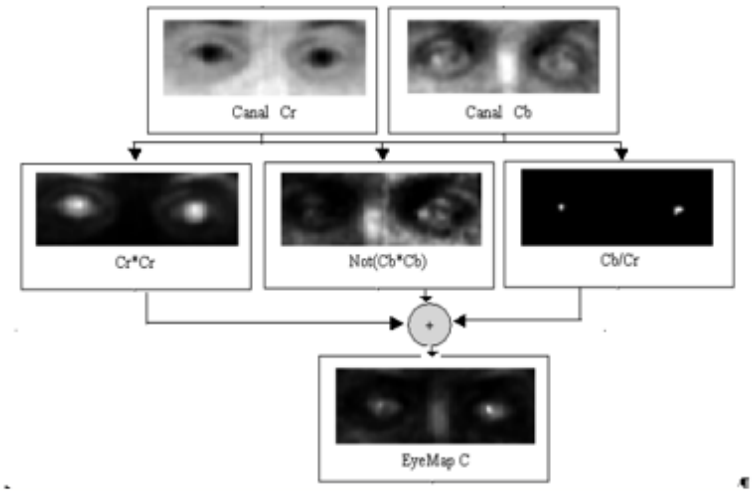


FIGURE 2.5: Modèle de détection des yeux.

L'étape de normalisation n'est pas nécessaire après le calcul de C_b^2 et C_r^2 à cause de la normalisation des valeurs initiales et pendant de l'étape de prétraitement. Le résultat pour C_b/C_r permet d'assurer que toutes les valeurs soient dans une limite légale. La figure 2.6 illustre quelques résultats de EyeMapC.

FIGURE 2.6: Les résultats Avec EyeMap C.

La chrominance (C_r, C_b) et l'information de la luminance (Y) peuvent être exploités

pour localiser les deux régions des yeux. Les résultats des testes montre que la zone autour des yeux a des valeurs colorimétriques spécifiques. Le but de cette étape est d'accentuer les valeurs de pixel les plus lumineux des yeux, en utilisant les canaux C_b de C_r de chrominance et de la luminance (Y).

La Luminance de la carte des yeux (Luminance Eyemap) Avant de calculer $EyeMapL$, il faut au préalable un certain nombre d'étapes de prétraitement sont appliquées. Ces étapes de prétraitement sont nécessaires pour compenser la qualité plus ou moins parfaite des images de webcam. D'abord l'image est lissée pour enlever le bruit. En second lieu l'image est normalisée, pour éviter les phénomènes de reflet et d'ombrage.

Après ces étapes de prétraitement, on met en évidence les zones de luminance des yeux par la formule suivante :

$$E_{lum} = EyeMapL = \frac{Y \oplus g_\sigma}{Y \otimes g_\sigma + 255}$$

où la $\oplus$ et $\otimes$ sont des opérations de dilatation et d'érosion sur une fonction f avec un élément structurant circulaire g_σ . La structure suivante est utilisée, pour recréer l'effet décrit dans la formule ci-dessus. Le modèle de détection des yeux est présenté dans la figure 2.7.

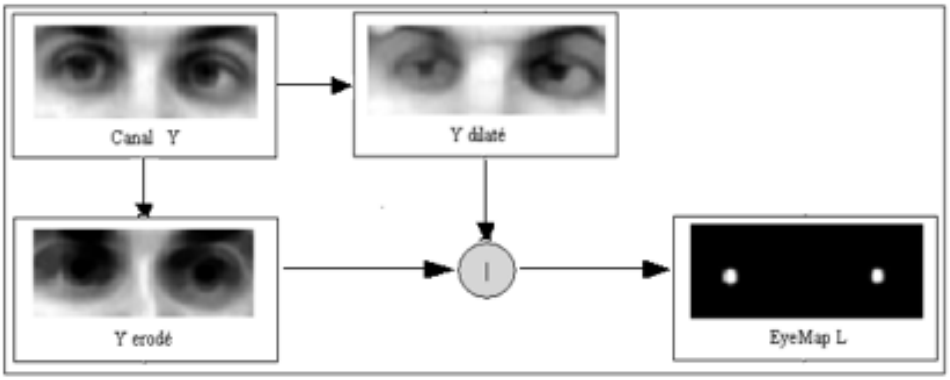


FIGURE 2.7: Modèle de détection des yeux.

La détection des yeux se base sur deux images, l'image de la luminance Y, dans la quelle on a autour des yeux des zones d'ombres et des zones très lumineuses. La deuxième est obtenu à partir des composantes de chrominance.

Un exemple du canal de Y *est dilaté* en utilisant un élément structurant de taille 5. Un deuxième exemple du canal de Y *est érodé*, également en utilisant le même élément structurant . L'image $EyeMapL$ est obtenue par la division des deux images une image de luminance Y *dilaté*, et une image de luminance érodée (figure 2.8).

L'image $EyeMapL$ est multipliée par une autre image qui exploite les informations de chrominance autour des yeux, pour localiser les régions des yeux. En multipliant les deux images $EyeMapC$ et $EyeMapL$, on obtiendra donc une image ou les yeux auront un fort contraste. Une première version de $EyeMap$ peut être créé en multipliant $EyeMapL$ par $EyeMapC$. Enfin quelques étapes de post-traitement sont faites pour augmenter le résultat final.

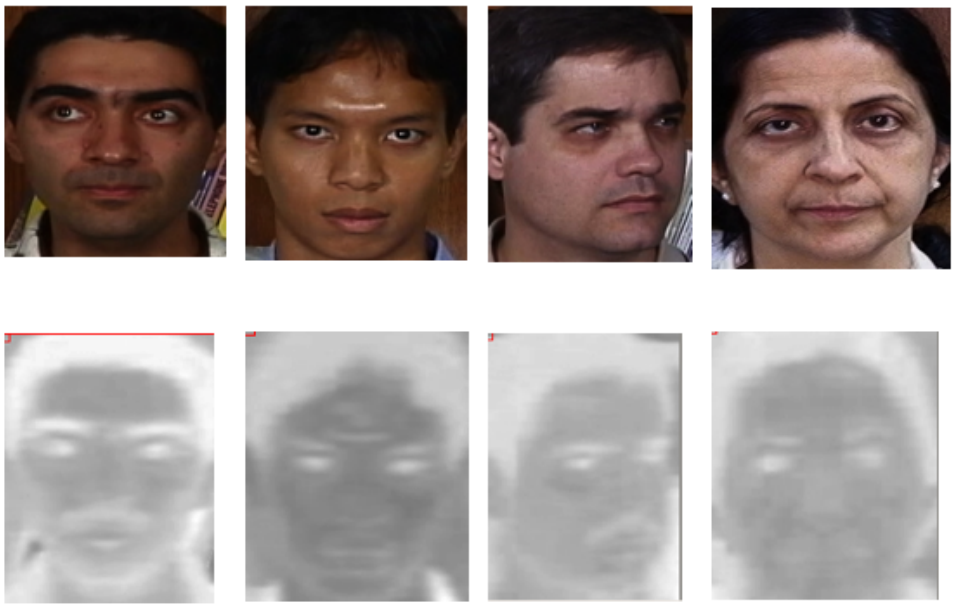


FIGURE 2.8: Les résultats de EyeMap L.

$$E_{map1} = EyeMap_1 = EyeMapL * EyeMapC$$

Le Bord de la carte des yeux (Edge Eyemap) :

Les yeux contient des zones lumineuses (blanches) et foncées (l'iris). Pour séparer la forme ou les bords de ces deux régions on utilise l'espace RGB. La première étape consiste à utiliser le filtre de sobel pour extraire les contours. Le résultat obtenu est ensuite lissé par un filtre moyenneur. Ceci peut être écrit comme suit :

$$E_{edge} = P\big(\mid P(\frac{(R+G+B)}{3}, S_h) \mid + \mid P(\frac{(R+G+B)}{3}, S_v) \mid, A\big)$$

avec $P(X, A)$ qui représente le filtre spatial de X par A. S_h and S_v est le filtre de sobel horizontal resp. vertical et A le filtre moyenneur.

RGB EyeMap :

Couleur de la carte des yeux (RGB Eyemap) :

Une 4ème carte des yeux (Eyemap) est construite avec le constat suivant qui est que les yeux sont toujours des zones sombres dans les trois composantes R, G et B. $RGBEyemap$ est calculé en multipliant les inverses des composantes R, G et B. Ensuite le résultat est filtré avec le même filtre présenté précédemment. Ainsi :

$$E_{RGB} = R\big((\overline{R} \times \overline{G} \times \overline{B}), A\big)$$

Carte finale des yeux (RGB Eyemap) :

Un simple eyemap ne peut pas localiser la position des yeux. D'où la proposition d'étendre la construction de Eyemap aussi à partir de E_{edge} et de E_{RGB} . Il suffit donc de combiner les quatres par la multiplication selon la formule suivantes :

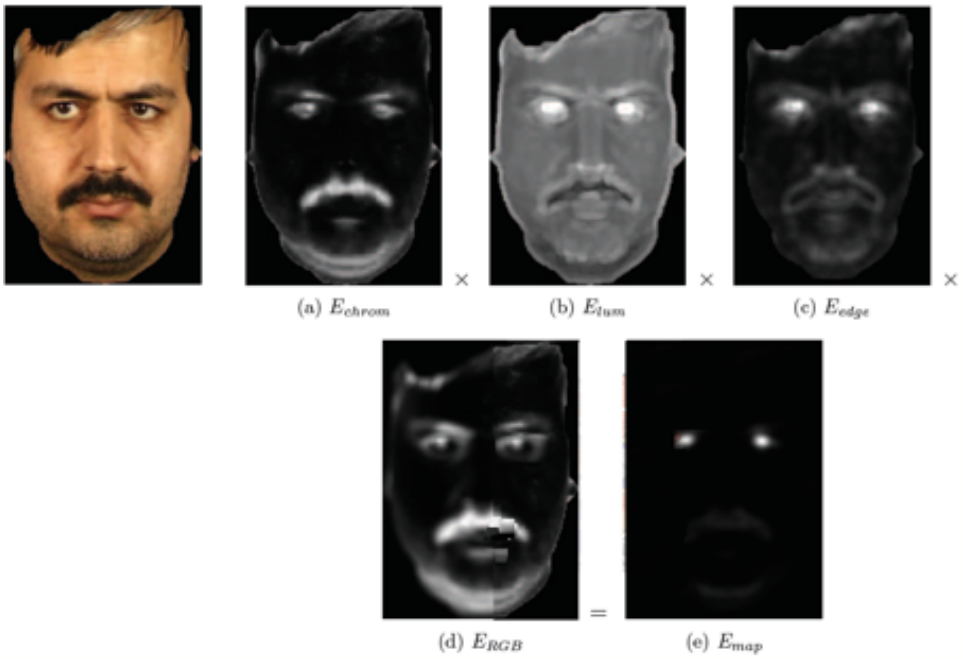


FIGURE 2.9: Combinaison de toutes les cartes de yeux.

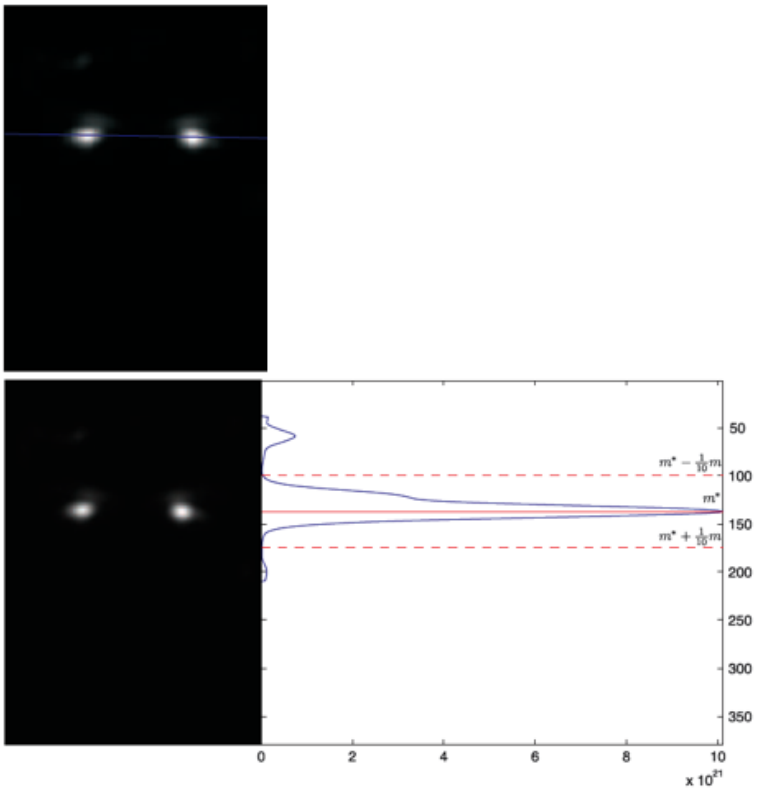


FIGURE 2.10: Position des yeux.

$$E_{map2} = E_{map1} \times E_{edge} \times E_{RGB}$$

Une multiplication est proposée plutôt qu'une addition car elle est plus déterministe et permet de supprimer les faux positifs à partir de eyemap simple E_{map1} .Cependant, la multiplication accentue également les faux négatifs et les expériences montrent qu'elles restent largement moins que les faux positifs. Les différentes cartes et leur produit final sont illustrées dans la figure 2.9.

Une fois le visage, les lèvres et les yeux détectés, les composantes faciales de chaque trame de la vidéo peut alors être projeté dans l'espace de visage obtenu par analyse en composantes principales suivant le principe des eigenface. Cependant, toutes les trames ne contiennent pas automatiquement un visage parfaitement localisé, avec un éclairage optimal, etc. Aussi, une sélection des meilleurs visages (selon un critère que l'on définit par la suite) est effectuée afin de ne conserver que les vecteur de paramètres correspondant pour mener à bien la modélisation et/ou la reconnaissance.

2.4.5 Position des yeux

On considère que les yeux dans l'image E_{map} sont bien alignées verticalement. Sinon, un étape de calcul de l'axe de rotation est nécessaire (figure 2.10).

2.5 Conclusion

La détection du visage est une étape particulièrement importante dans un système d'analyse du visage. Plusieurs techniques de détection ont été présentées au cours de ce chapitre, chacune possédant ses forces et ses faiblesses. Dans le chapitre suivant, nous allons aborder le problème de la détection et le suivi des yeux par une nouvelle approche basée sur les descripteurs de Haar en utilisant le processus Gaussien pour faire la prédiction du regard.

Chapitre 3

Détection des yeux basée sur les descripteurs de Haar

Pour faire la prédiction du regard de l'utilisateur, nous devons tout d'abord détecter la position de l'œil de l'utilisateur dans la vidéo capté par la caméra. Nous utilisons une méthode rapide de détection d'objet basée sur un algorithme en cascade simulé par des descripteurs simples nommés descripteurs de Haar (ou *Haar-like features* en anglais). Cette méthode a été initialement proposée par Paul Viola [106] et améliorée par Rainer Lienhart [52]. L'idée principale de cette méthode est de créer un classifieur qui permet de détecter rapidement des yeux dans une image de caméra. D'abord, les classifieurs sont formés avec des milliers d'images positives et négatives (les images d'objets et les images sans objet) basées sur les descripteurs de Haar. Il existe un grand nombre de descripteurs de Haar dans une sous-fenêtre de l'image 24x24 pixels (117,941 descripteurs) [52], un nombre beaucoup plus grand que le nombre de pixels. L'algorithme de la formation est AdaBoost [106], il sélectionne un petit ensemble de descripteurs qui sont meilleurs sépare les exemples positifs et négatifs et crée des classifieurs basés sur des descripteurs sélectionnés. Après la formation des classifieurs, une cascade de classifieurs est construite pour accroître la performance de la détection et réduire radicalement les temps de calcul. La cascade de classifieurs est construit de manière à rejeter un grand nombre de sous fenêtres négatives et à détecter partout des sous fenêtres positives. Les classifieurs simples sont utilisés pour rejeter la majorité des sous fenêtres avant que des classifieurs plus complexes soient appelés pour réduire le temps de calcul et pour obtenir un faible taux de faux positifs. Nous détaillerons dans ce chapitre cette méthode de détection.

3.1 Les descripteurs de Haar

3.1.1 Les descripteurs de Haar

Notre procédure de détection d'objets classifie des images à partir la valeur des descripteurs simples. Il y a beaucoup de motivations pour utiliser les descripteurs plutôt que les pixels directement. La raison la plus courante est que les descripteurs peuvent agir pour encoder les connaissances de domaine qui est difficile à apprendre en utilisant une qualité

finie de données d'apprentissage [106]. Avec ce système, il y a une deuxième motivation critique pour les descripteurs : le système fondé sur les descripteurs est beaucoup plus rapide qu'un système fondé sur les pixels. Les descripteurs simples utilisés sont inspirés

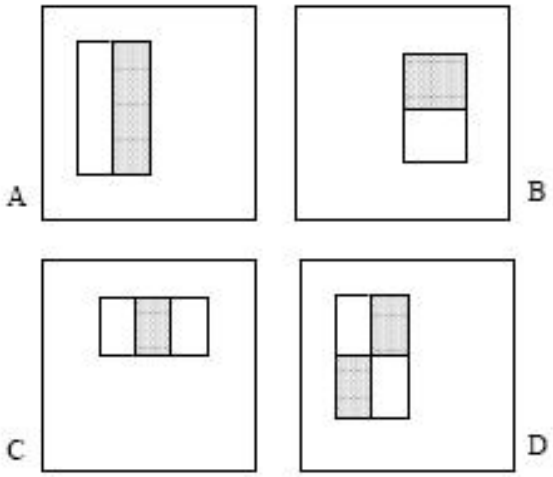


FIGURE 3.1: Exemples de descripteurs rectangles

des fonctions de base de Haar qui ont été employées par Papageorgiou et al. dans [77]. Plus spécifiquement, Viola et al. (2001)emploient trois types de descripteurs (voir figure 3.1) :

- Un descripteur à deux rectangles, la valeur d'un descripteur à deux rectangles est la différence entre la somme des pixels des deux régions rectangulaires. Les régions ont même taille et même forme et sont horizontalement ou verticalement adjacents (voir figure 3.1.A, B).
- Un descripteur à trois rectangles, la valeur d'un descripteur à trois rectangles est la somme des pixels dans les deux rectangles extérieurs soustraits de la somme dans le rectangle central (voir figure 3.1.C).
- Un descripteur à quatre rectangles est la différence entre les paires diagonales de rectangles (voir figure 3.1.D).

Si la taille de la fenêtre de base (image en entrée) est 24×24, cette méthode utilise un nombre de descripteurs rectangles tout à fait grand (plus de 180 000 descripteurs). Ainsi, le nombre de descripteurs est beaucoup plus grand que l'espace de l'image (dans ce cas 576 pixels).

3.1.2 L'extension des descripteurs de Haar

Rainer Lienhart et al (2002) a étoffé l'ensemble de descripteurs de Haar proposés par Viola et al (2001) à 14 descripteurs pour améliorer la capacité de détection d'objet. Supposons que l'unité de base pour vérifier la présence d'un objet est une fenêtre de pixels. Supposons également que nous avons une façon très rapide de calculer le nombre

de pixels de tous les rectangles droits[1] et de tous les rectangles en rotation de 45^0 à l'intérieur de la fenêtre. Un rectangle est défini par le 5-uplet $r = (x, y, w, h, \alpha)$ avec $0 \leq x, x + w \leq W; 0 \leq y, y + h \leq H; x, y \geq 0; w, h > 0$ et $\alpha \in \{0^0, 45^0\}$ et son nombre de pixels est noté $RecSum(r)$. Deux exemples de ces rectangles sont donnés dans la Figure 3.2. L'ensemble des descripteurs crus est l'ensemble de tous les descripteurs possibles de

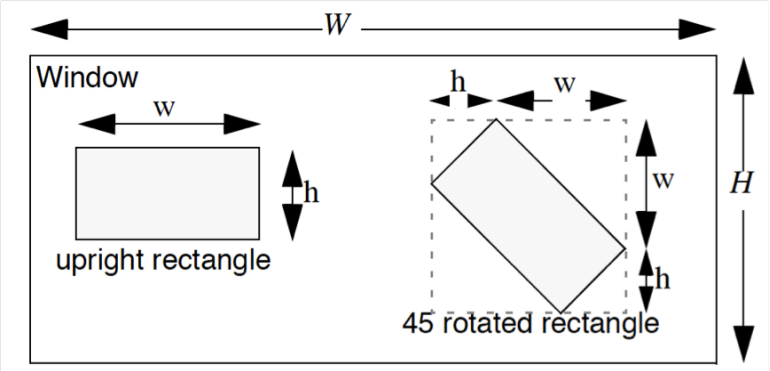


FIGURE 3.2: Exemples de rectangle droit et de rectangle en rotation de 45^0

la forme

$$feature_I = \sum_{i \in I = \{1,\ldots,N\}} \omega_i . RecSum(r_i) \tag{3.1}$$

où les poids $\omega_i \in \Re$, les rectangles r_i, et N sont choisis arbitrairement.

Cet ensemble de descripteurs crus est (presque) infiniment grand. Pour des raisons pratiques, il est réduit comme suit :

1. Seules des combinaisons pondérées du nombre de pixels de deux rectangles sont considérés (par exemple $N = 2$).
2. Les poids ont des signes opposés, et sont utilisés pour compenser la différence de taille de la zone entre les deux rectangles. Ainsi, pour des rectangles ne se chevauchant pas, nous avons $-w_0.Area(r_0) = w_1.Area(r_1)$. Sans restrictions, nous pouvons mettre $w_0 = -1$ et mettre $w_1 = Area(r_0)/Area(r_1)$.
3. Les descripteurs imitent des descripteurs de Haar comme entourent le centre (ou *center-surround* en anglais) (figure 3.3 - 3(a)(b)).

Ces restrictions nous conduisent à 14 prototypes de descripteurs présentés dans la figure 3.3 :

– Quatre descripteurs de bord,
– Huit descripteurs de ligne,
– Deux descriptions de entourent le centre (center-surround).

Ces prototypes sont mis à l'échelle de manière indépendante dans le sens vertical et horizontal, afin d'obtenir un ensemble complet de descripteurs. Notez que les descripteurs de ligne peuvent être calculés par deux rectangles seulement. De même, il est supposé que le premier rectangle englobe le rectangle noir et blanc et que le second rectangle représente

1. Un rectangle droit est défini par un rectangle ayant deux bords parallèles aux lignes de l'écran et deux bords parallèles aux colonnes de l'écran.

la zone noire. Par exemple, le descripteur de ligne (2a) d'une hauteur de 2 et d'une largeur de 6, dans le coin supérieur gauche (5, 3) peut être écrit comme

$$feature_I = -1.RecSum(5, 3, 6, 2, 0^0) + 3.RecSum(7, 3, 2, 2, 0^0). \tag{3.2}$$

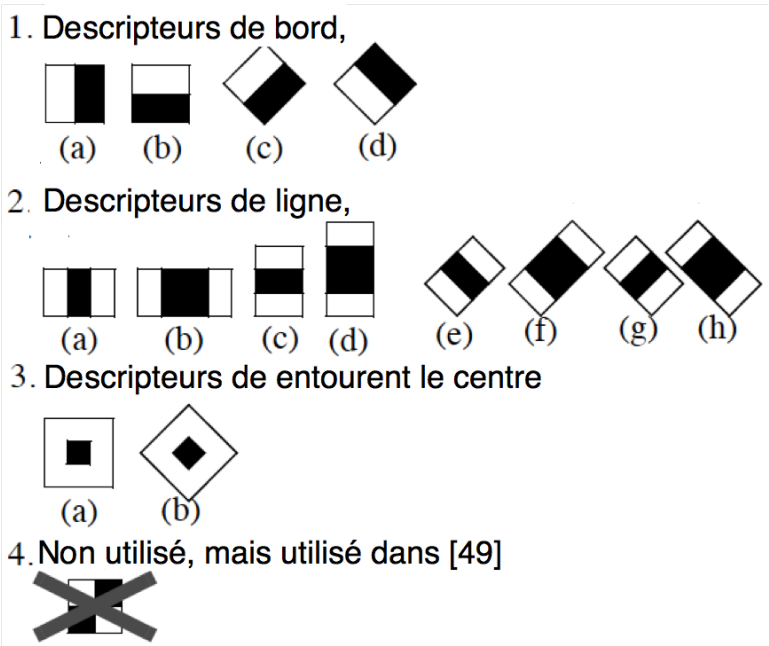


FIGURE 3.3: Descripteurs de Haar et descripteurs de entourent le centre. Les zones noires ont des poids négatifs et les zones blanches ont des poids positifs.

Les descripteurs supplémentaires ont considérablement renforcé le pouvoir expressif du système d'apprentissage et, par conséquent ont amélioré la performance du système de détection d'objet. Le descripteur (4a) n'a pas été utilisé car les fonctions (2g) et (2e) en sont de bonnes approximations.

3.1.3 Nombre de descripteurs

Le nombre de descripteurs dérivés de chaque prototype est assez grand et diffère d'un prototype à l'autre. Il peut être calculée comme suit. Soient $X = \lfloor W/w \rfloor$ et $Y = \lfloor H/h \rfloor$, les facteurs d'échelle maximal dans les directions x et y. Un descripteur vertical de taille $x \times h$ génère alors :

$$XY.(W + 1 - w\frac{X+1}{2}).(H + 1 - h\frac{Y+1}{2}) \tag{3.3}$$

descripteurs d'une image de taille $W \times H$. Pour un rectangle en rotation de 45^0, le nombre de descripteurs généré est :

$$XY.(W + 1 - z\frac{X+1}{2}).(H + 1 - z\frac{Y+1}{2}) \text{ avec } z = w + h \tag{3.4}$$

Type de descripteur	w/h	X/Y	nombre
$1a; 1b$	$2/1; 1/2$	$12/24; 24/12$	$43,200$
$1c; 1d$	$2/1; 1/2$	$8/8$	$8,464$
$2a; 2c$	$3/1; 1/3$	$8/24; 24/8$	$27,600$
$2b; 2d$	$4/1; 1/4$	$6/24; 24/6$	$20,376$
$2e; 2g$	$3/1; 1/3$	$6/6$	$4,356$
$2f; 2h$	$4/1; 1/4$	$4/4$	$3,600$
$3a$	$3/3$	$8/8$	$8,464$
$3b$	$3/3$	$3/3$	$1,521$
Somme			$117,941$

TABLE 3.1: Nombre de descripteurs à l'intérieur d'une fenêtre de 24×24 pour chaque type de descripteur.

Le tableau 3.1 donne le nombre de descripteurs pour une taille de fenêtre de 24×24.

Tous les descripteurs peuvent être calculés très rapidement et en temps constant pour toutes les tailles en utilisant deux images auxiliaires. Pour les rectangles droits, l'image auxiliaire est la *Summed Area Table* $SAT(x, y)$. $SAT(x, y)$ qui est définie comme le nombre de pixels du rectangle vertical allant du coin supérieur gauche $(0, 0)$ jusqu'au coin en bas à droite (x, y) (voir la figure 3.4 a) :

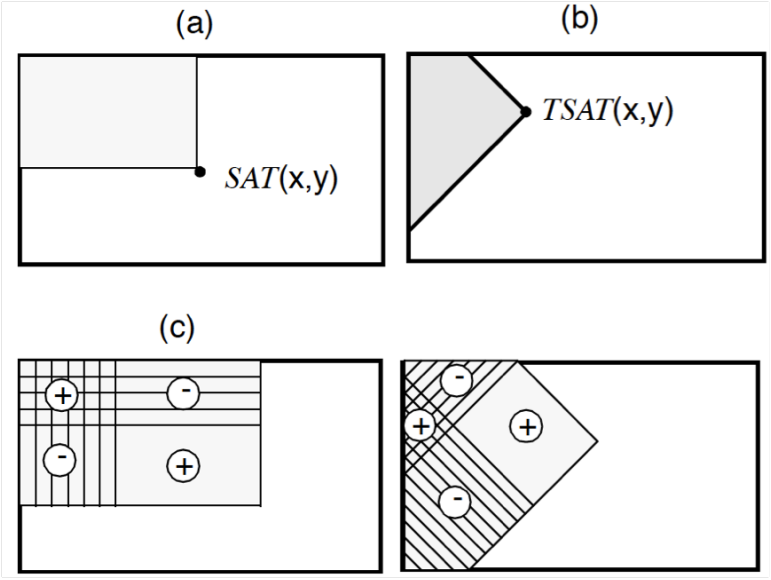


FIGURE 3.4: (a) *Summed Area Table* du rectangle vertical (SAT) et (b) *Rotated Summed Area Table (RSAT)* ; schéma de calcul du nombre de pixels du rectangle vertical (c) et du rectangle en rotation (d)

$$SAT(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y'). \tag{3.5}$$

où I est l'image originale. Il peut être calculé en une passe sur tous les pixels de gauche

à droite et de haut en bas par la formule

$$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1) \tag{3.6}$$

avec

$$SAT(-1,y) = SAT(x,-1) = 0 \tag{3.7}$$

Donc, le nombre de pixels d'un rectangle vertical $r = (x,y,w,h,0)$ peut être déterminé par les quatre rectangles (voir la figure 3.4(c)) :

$$\begin{aligned} RecSum(r) &= SAT(x-1,y-1) + SAT(x+w-1,y+h-1) \\ &\quad -SAT(x-1,y+h-1) - SAT(x+w-1,y-1) \end{aligned} \tag{3.8}$$

Pour les rectangles en rotation de 45^0, l'image auxiliaire est définie comme *Rotated Summed Area Table RSAT(x,y)*. $RSAT(x,y)$ est le nombre de pixels du rectangle en rotation de 45^0, du coin le plus à droite (x,y) jusqu'aux limites de l'image (voir la figure 3.4(b)) :

$$RSAT(x,y) = \sum_{x' \leq x, x' \leq x-|y-y'|} I(x',y'). \tag{3.9}$$

Il peut être calculé en deux passes sur tous les pixels. La première passe de gauche à droite et de haut en bas détermine

$$RSAT(x,y) = RSAT(x-1,y-1) + RSAT(x-1,y) + I(x,y) - RSAT(x-2,y-1) \tag{3.10}$$

avec

$$RSAT(-1,y) = RSAT(-2,y) = RSAT(x,-1) = 0 \tag{3.11}$$

la seconde passe de la droite vers la gauche et de bas en haut détermine

$$RSAT(x,y) = RSAT(x,y) + RSAT(x-1,y+1) - RSAT(x-1,y) \tag{3.12}$$

Ainsi, le nombre de pixels d'un rectangle en rotation $r = (x,y,w,h,45^0)$ peut être déterminée par quatre tableaux (voir aussi la figure 3.4(d) et la figure 3.5) :

$$\begin{aligned} RecSum(r) &= RSAT(x+w,y+w) + RSAT(x-h,y+h) \\ &\quad -RSAT(x,y) - RSAT(x+w-h,y+w+h) \end{aligned} \tag{3.13}$$

3.1.4 Intérêt des descripteurs de Haar

Les descripteurs rectangles sont des primitifs et simples en comparaison avec d'autres moyens d'analyse locale telles que les filtres orientés. Les filtres orientés sont excellents pour l'analyse détaillée des frontières, la compression d'image, et l'analyse de texture. Les descripteurs rectangles, malgré leur simplicité sont sensibles à la présence des contours, barres et toute autre structure simple présentés dans l'image.

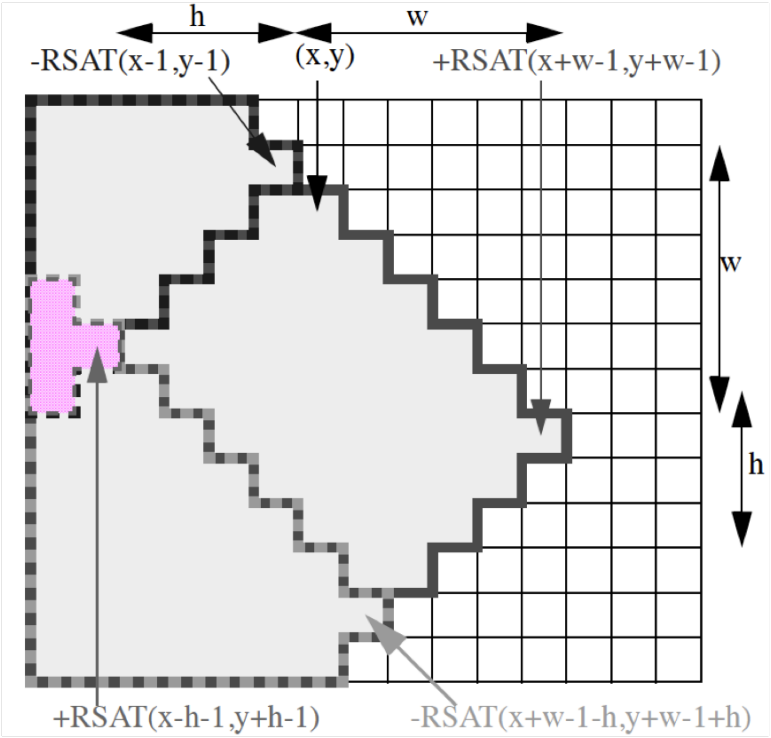


FIGURE 3.5: Schéma de calcul pour la région de rotation

À la différence des filtres orientés les seules orientations disponibles dans le cas de descripteurs rectangles sont verticales et horizontales. L'efficacité informatique extrême de ces derniers fournit une compensation suffisante à leur flexibilité limitée.

Afin d'apprécier l'avantage informatique de la technique d'image intégrale, considérons une approche de détection d'objet fondée sur une pyramide d'images (images à différentes échelles). Comme la plupart des systèmes de détection d'objet, l'algorithme de détection balaye l'image en entrée à différentes échelles en commençant par la résolution la plus basse. À titre d'exemple, on peut considérer une image de taille 24×24 pixels, l'image est balayée à 11 échelles, chacune étant plus petite que la précédente d'un facteur de 1,25. Une fenêtre de calcul à taille fixe est alors balayée à travers chacune de ces images. Cela consomme beaucoup de temps de calcul, en plus du temps consacré au calcul de la pyramide. Par conséquent, il est extrêmement difficile de traiter une pyramide à la vitesse de 15 trames par seconde.

En revanche, Viola et al. [106] ont défini un ensemble significatif de descripteurs, qui ont la propriété de l'image d'être évalués à n'importe quelle échelle et n'importe quel endroit au bout de quelques opérations. Ils ont montré que des algorithmes de détection de performance acceptables peuvent être construits avec seulement deux descripteurs rectangles. Etant donné l'efficacité informatique de ces descripteurs, le procédé de détection de visage peut être effectué pour une image entière à chaque échelle et à la vitesse de 15 trames par seconde ; ce qui coûte moins de temps de calcul que celui nécessaire pour balayer une seule image de la pyramide à 11 niveaux dans le cas précédent. Toute approche qui exige une pyramide de ce type s'exécutera nécessairement plus lentement que l'algorithme de détection introduit dans [106].

3.2 Algorithme d'apprentissage

A partir d'un ensemble de descripteurs et d'un ensemble d'images correspondant à des yeux (images positives) et à des non-yeux (images négatives), une méthode d'apprentissage peut être employée pour déduire une fonction de classification.

Notons que l'ensemble des descripteurs rectangle est composé de 117, 941 descripteurs, un nombre beaucoup plus grand que le nombre de pixels dans l'image. Même si chaque descripteur peut être calculé d'une façon efficace, l'utilisation de l'ensemble complet est excessivement chère. L'hypothèse introduite dans [106], et qui a été confirmée par une suite d'expériences, est qu'un nombre limité de ces descripteurs peut être exploité pour former une fonction de classification efficace. Le défi principal est de trouver ces descripteurs. Dans [65] une version d'algorithme d'apprentissage connu sous le nom d'*"AdaBoost"*, qui signifie *"adaptative boosting"*, est employée pour choisir les descripteurs et former la fonction de classification. C'est l'un des algorithmes les plus utilisés en apprentissage automatique.

Sous sa forme originale, l'algorithme d'apprentissage d'AdaBoost est employé pour améliorer la robustesse d'un algorithme d'apprentissage simple. Son principe consiste à combiner un ensemble de fonctions faibles de classification pour former une fonction du classification plus efficace. La fonction de classification est *faible* si elle est seulement capable de reconnaître deux classes au moins aussi bien que le hasard ne le ferait (c'est-à-dire qu'il ne se trompe pas plus d'une fois sur deux en moyenne). Pour un problème donné une fonction de classification faible peut seulement classifier les données de la base d'apprentissage correctement à 51%. Après la première phase d'apprentissage, les exemples seront recombinés de sorte que celles qui ont été mal classifiées par la fonction de classification faible précédente auront le poids le plus grand. La fonction de classification forte finale prend la forme d'un perceptron = une combinaison pondérée de la fonction de classification faible (selon la qualité de classification).

Considérons le problème d'apprentissage, dans lequel un grand ensemble de fonctions de classification est combiné en utilisant un vote majoritaire pondéré. Le défi est d'associer un grand poids à chaque bonne fonction de classification et un plus petit poids aux fonctions faibles.

AdaBoost est un mécanisme agressif qui permet de choisir un petit ensemble de bonnes fonctions de classification qui ont néanmoins une variation significative. En imaginant une analogie entre les fonctions de classification et les descripteurs faibles de Haar, *AdaBoost* est une procédure efficace pour sélectionner un nombre restreint de bons descripteurs.

Une méthode pratique pour comprendre cette analogie est de limiter la fonction de classification faible à l'ensemble des fonctions de classification qui dépendent d'un seul descripteur. C'est pour cela que l'algorithme d'apprentissage faible est conçu pour choisir le descripteur rectangle qui sépare mieux les images positives des négatives. Une fonction de classification faible $h_j(x)$ se compose ainsi d'un descripteur f_j , d'un seuil θ_j et d'une parité p_j indiquant la direction du signe d'inégalité :

$$h_j(x) = \begin{cases} 1 & \text{si } p_j.f_j(x) < p_j.\theta_j \\ 0 & \text{sinon} \end{cases} \tag{3.14}$$

AdaBoost

- Soient n images exemples $(x_1, y_1), \ldots, (x_n, y_n)$ avec x_i le vecteur de descripteurs $\in \Re$ de l'exemple i et $y_i \in \{1, 0\}$ (positifs et négatifs)
- Initialiser des poids $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ où m et l sont les nombre de négatifs et positifs, respectivement.
- Pour $t = 1, \ldots, T$:
  1. Normaliser les valeurs du vecteur de poids w_i
     $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_j}$
     donc, w_t est une distribution de probabilité.
  2. Pour chaque descripteur j, entraîner un classifieur h_j qui est restreint à n'utiliser qu'un seul descripteur. L'erreur est évaluée par rapport à w_t, $\epsilon_j = \sum_{i=1} w_i |h_j(x_i) - y_i|$
  3. Choisir le classifieur h_t avec l'erreur ϵ_t la plus faible
  4. Actualiser les poids :
     $w_{t+1,i} = w_{t,i} \beta^{1-e_i}$
     où $e_i = 0$ si $h_t(x_i) = y_i, e_i = 1$ sinon, avec $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
- Le classifieur fort final est :

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha \\ 0 & \text{sinon} \end{cases}$$

avec $\alpha_t = log(\frac{1}{\beta_t})$

TABLE 3.2: L'algorithme d'apprentissage AdaBoost. Chaque tour de boosting sélectionne un descripteur de 117, 941 descripteurs potentiels.

Ici x est une sous-fenêtre de l'image en entrée. Chacune de ces fonctions faibles a une précision modérée dans la classification : elle prend une décision binaire à partir de la valeur de sortie du filtre. Mais la combinaison de plusieurs fonctions permet de construire un classifieur de performances plus grandes, appelé classifieur *fort*. Cette combinaison est représentée par la somme pondérée suivante :

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha = S \\ 0 & \text{sinon} \end{cases} \tag{3.15}$$

La fonction de classification *forte* $h(x)$ est composée de T fonctions de classification *faibles* g_t, où α_t est un coefficient de pondération pour chacune des g_t. S est le seuil de la fonction de classification $h(x)$ qui minimise l'erreur de classification.

Nous transcrivons le pseudo-code dans le tableau 3.2 pour un résumé de l'algorithme d'apprentissage *AdaBoost* [106].

Le paramètre β_t est choisi en fonction de l'erreur ϵ_t et est utilisé pour actualiser le vecteur des poids w. Cette loi réduit la probabilité assignée aux exemples bien classifiés par la fonction faible et incrément la probabilité des exemples où la prédiction est erronée. Ces exemples vont avoir une incidence plus forte dans l'estimation de l'erreur ϵ_t dans l'itération suivante (et donc le choix de la prochaine fonction faible). L'erreur de g_t dans la nouvelle distribution w_{t+1} est exactement 1/2.

La fonction de discrimination obtenue est une somme pondérée des fonctions faibles générées par l'algorithme. Le facteur β_t s'est réduit avec ϵ_t, ce qui incrément la différence entre les distributions w_t et w_{t+1}. Un β_t plus petit signifie que la fonction faible est plus précise et va avoir une influence plus importante dans le classifieur $h(x)$ à travers la pondération α_t.

L'erreur de $h(x)$ est bornée par [26] :

$$\epsilon_{h(x)} \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} \tag{3.16}$$

Pour la mission de détection des yeux, les descripteurs rectangles initiaux choisis par Ada-Boost sont significatifs et facilement interprétés. Le choix du premier descripteur est basé sur la propriété que la région de la pupille est souvent plus foncée que la région autour de la pupille (voir figure 3.6).

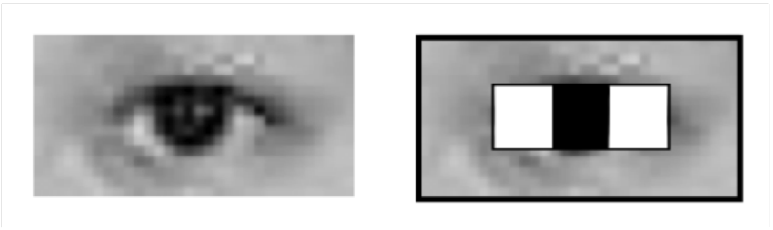


FIGURE 3.6: Un descripteur sélectionné par AdaBoost.

3.3 Détection en Cascade

Le troisième apport du travail de Viola et Jones [106] consiste en l'introduction d'un apprentissage basé sur la méthode de dopage (ou *boosting* en anglais) qui permet d'obtenir une fonction de classification avec une architecture en cascade, appelée *Cascade Attentionelle*.

3.3.1 Principe de la Cascade Attentionelle

La cascade est la combinaison successive des classifieurs forts dont la complexité est croissante tout au long de la structure (voir figure 3.7). Au départ, les classifieurs simples éliminent la plupart des fausses alarmes, avant que des classifieurs plus complexes [2] ne

2. La complexité d'un classifieur G_i est définie comme le nombre de classifieurs faibles g_t qui le compose.

soient utilisés pour éliminer des fenêtres plus difficiles. Ainsi, l'*attention* du système se concentre, au fur et à mesure des étages de la cascade, sur des zones de plus en plus réduites et pertinentes. Cette façon d'organiser la détection permet d'incrémenter la performance

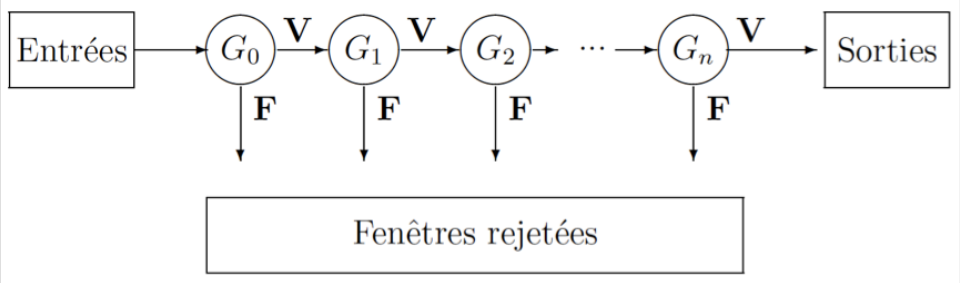


FIGURE 3.7: Cascade de classfieurs forts

totale du système (taux de détections correctes et taux de fausses alarmes), en même temps qu'elle augmente la vitesse en se focalisant sur des régions susceptibles de contenir un véhicule.

Cela peut se démontrer intuitivement si nous prenons un premier classifieur G_0 composé d'un faible nombre de fonctions g_t (trois, quatre ou cinq au total) et l'application sur plusieurs régions d'une image. Il est entraîné pour éliminer la moitié des régions négatives. Les régions qui ont subsisté subissent l'application du deuxième classifieur G_1, composé lui aussi d'un faible nombre de fonctions faibles. G_1 est spécialisé dans le traitement des échantillons plus proches de la classe œil, qui ont été validés par G_0, et élimine, à son tour, la moitié des fausses alarmes.

Nous pouvons considérer que les régions restantes ont la plus forte probabilité, encore, d'appartenir à la classe véhicule. Pour éliminer la moitié des fausses alarmes, les prochains classfieurs seront composés d'un nombre de plus en plus important de g_t, avec pour conséquence l'augmentation du temps de calcul par région. Cependant, les étapes antérieures ont éliminé successivement un grand nombre de régions qui restent à évaluer.

3.3.2 Apprentissage de la *Cascade Attentionelle*

Nous allons entraîner un ensemble de K fonctions de classification $G_1, \ldots, G_i, \ldots, G_K$, où chaque G_i est une fonction *forte* entraînée en utilisant l'algorithme Adaboost à l'étape i de la cascade.

Le nombre de classifieurs faibles pour chaque G_i n'est pas fixe : au lieu d'arrêter la boucle itérative d'AdaBoost pour un nombre maximum de descripteurs T, nous fixons deux paramètres de performance pour la fonction G_i : le taux minimum de détections correctes DC_{min} et le taux maximal de fausses alarmes acceptables FA_{max}.

Le choix de DC_{min} et FA_{max} modifie le comportement de la cascade, ainsi que son architecture. Lors de l'apprentissage de la cascade, le seuil S_i du classifieur G_i est décrémenté jusqu'à ce que G_i obtienne un taux de détections correctes d'au moins DC_{min} sur la base de la validation. Plus proche de cent pour cent est DC_{min}, plus petit est difficile. Cependant, un nombre plus important d'exemples négatifs sera considéré comme théorique de la cascade attentionelle :$D_{CA} = (DC_{min})^K$, où K est le nombre d'étages de la cascade.

Apprentissage de la Cascade

1. Choisir DC_{min} et FA_{max}
2. Obtenir P et N_0 (exemples positifs et négatifs)
3. $i = 0$
4. Tant que $i < K - 1$
   - $i = i + 1$
   - $n_i = 0, f = 1$
   - Tant que $f > FA_{max}$
     $n_i = n_i + 1$
     utiliser P et N_i pour entraîner un classifieur G_i avec $T = n_i$
     Décrémenter le seuil de G_i jusqu'à atteindre $DC_{min} dans la base de validation positive$
     Évaluer G_i dans la base N_i pour obtenir f
   - Le classifier G_i est rajouté au détecteur en cascade.
   - Évaluer le détecteur en cascade sur la base des images négatives et placer les fausses détections dans N_{i+1}

TABLE 3.3: Apprentissage de la Cascade Attentionelle

Par exemple, pour un $DC_{min} = 99.5\%$, le taux de détections théorique pour une cascade à seize étages obtient un taux de détections de 92.3%

Le processus d'apprentissage itératif s'arrête quand G_i atteint un taux de fausses alarmes maximal FA_{max} dans la base négative. Étant donné qu'idéalement, la cascade doit rejeter au moins la moitié des fausses alarmes, FA_{max} peut prendre une valeur égale à 50%. Si nous donnons à FA_{max} des valeurs plus faibles, G_i a besoin de plus d'itérations (donc plus de fonctions faibles), pour obtenir ce taux de rejet. Voir le tableau 3.3 pour un résumé du processus d'apprentissage de la Cascade.

3.4 Résultat expérimental

Nous avons utilisé une cascade de classifieurs formée par 7000 images de l'œil et 10,000 images qui ne contiennent pas d'œil. La détection des yeux est réalisée avec simple caméra de résolution de 640 × 480 et de frame rate de 30 fps. Le résultat est obtenu en temps réel. La figure 3.8 présente le résultat de détection des yeux.

3.5 Conclusion

Dans ce chapitre, nous avons introduit la méthode de détection des yeux basée sur les descripteurs de Haar. Cette méthode a été initialement proposée par Paul Viola [106]

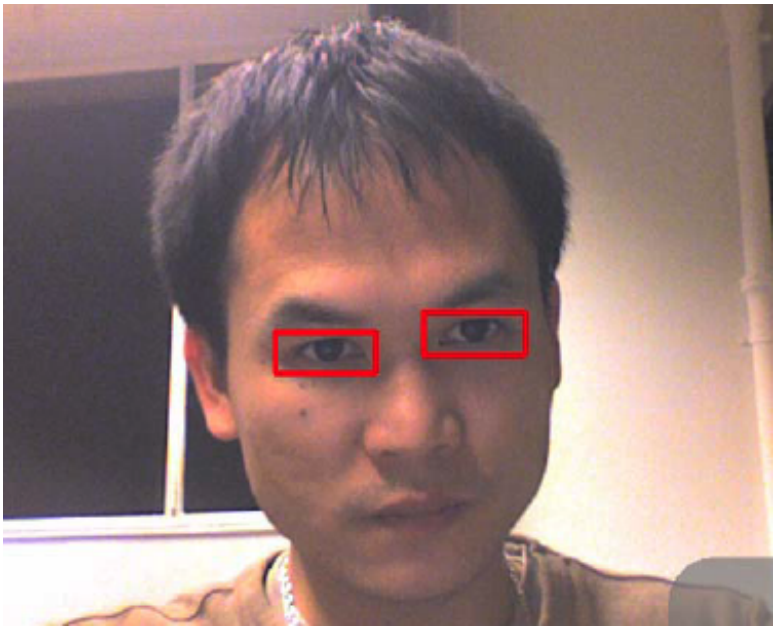


FIGURE 3.8: Exemple du résultat de détection des yeux.

et améliorée par Rainer Lienhart [52]. L'idée principale de cette méthode est de créer une cascade de classifieurs qui permet de détecter rapidement des yeux dans la vidéo captée par la caméra. Les classifieurs sont formés avec des milliers d'images positives et négatives (les images d'objets et les images non objet) basées sur des descripteurs de Haar. La cascade de classifieurs est construite pour accroître la performance de la détection et réduisant radicalement les temps de calcul. Le résultat est donc obtenu en temps réel.

Chapitre 4

Suivi des yeux

Après la détection des yeux, nous devons détecter les mouvements des yeux en temps réel avec la caméra. Il s'agit de suivre les coins des deux yeux puis de récupérer les yeux dans chaque frame d'image. Pour avoir un bon suivi, nous devons suivre les autres coins de visage tels que les coins du nez et de la bouche, afin de savoir si ces coins sont toujours en cours ou en perte de suivi. Ensuite, nous utilisons la méthode de détection Outliers pour détecter les coins perdus et les corriger.

4.1 Introduction

Dans ce chapitre, nous allons distinguons trois parties principales : *détection des coins*, *suivi des coins*, *détection et récupération des coins "perdus"*. D'abord, nous allons présenter la méthode de détection de coins Harris pour détecter des coins importants sur le visage de l'utilisateur (les coins des yeux, du nez et de la bouche) dans la partie *détection des coins*. Ensuite, dans la partie *suivi des coins*, nous allons présenter la méthode de suivi de Lucas Kanade pour suivre les mouvements des coins dans la caméra. Enfin, la partie *détection et récupération des coins* va présenter la méthode de détection Outliers pour détecter des coins "perdus" (les coins qui sont perdus de suivi) et les récupérer.

4.2 Détection des coins

4.2.1 Le problème

Le détecteur de points d'intérêts (ou *coins*) est une étape préliminaire à nombreux processus de vision par ordinateur. Les points d'intérêts, dans une image, correspondent à des doubles discontinuités de la fonction d'intensités. Celles-ci peuvent être provoquées, comme pour les contours, par des discontinuités de la fonction de réflectance ou des discontinuités de profondeur. Ce sont par exemple : les coins, les jonctions en T ou les points de fortes variations de texture (la figure 4.1). De nombreuses méthodes ont été proposées pour détecter des points d'intérêts. Elles peuvent être classées grossièrement suivant trois catégories :

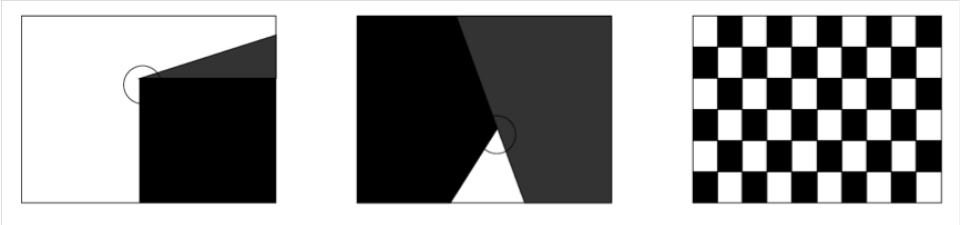


FIGURE 4.1: Différents types le points d'intérêts : coins, jonction en T et point de fortes variations de texture.

1. Approches contours : l'idée est de détecter les contours dans une image dans un premier temps. Les points d'intérêts sont ensuite extraits le long des contours en considérants les points de courbures maximales ainsi que les intersections de contours.
2. Approches intensité : l'idée est cette fois-ci de regarder directement la fonction d'intensité dans les images pour en extraire directement les points de discontinuités.
3. Approches à base de modèles : les points d'intérêts sont identifiés dans l'image par mise en correspondance de la fonction d'intensité avec un modèle théorique de cette fonction des point d'intérêts considérés.

Nous avons utilisé les approches de la deuxième catégorie. Les raisons sont : indépendance vis à vis de la détection de contours (stabilité), indépendance vis à vis du type de points d'intérêts (méthodes plus générales).

4.2.2 Le détecteur de Moravec(1980)

L'idée du détecteur de Moravec [67] est de considérer le voisinage d'un pixel (une fenêtre) et de déterminer les changements moyens de l'intensité dans le voisinage considéré lorsque la fenêtre se déplace dans diverses directions. Plus précisément, on considère la fonction :

$$E(x,y) = \sum_{u,v} w(u,v)|I(x+u,y+v) - I(u,v)^2|, \tag{4.1}$$

où :

- w spécifie a fenêtre/voisinage considérée (valeur 1 à l'intérieur de la fenêtre et 0 à l'extérieur) ;
- $I(u,v)$ est l'intensité au pixel (u,v) ;
- $E(x,y)$ représente la moyenne du changement d'intensité lorsque la fenêtre est deplacée de (x,y).

En appliquant cette fonction dans les trois situations principales suivantes (voir la figure 4.2), on obtient :

1. L'intensité est approximativement constante dans la zone image considérée : la fonction E prendra alors de faibles valeurs dans toutes les directions x, y.
2. La zone image considérée contient un contour rectiligne : la fonction E prendra alors de faibles valeurs pour des déplacements (x,y) le long du contour et de fortes valeurs pour des déplacements perpendiculaires au contour.

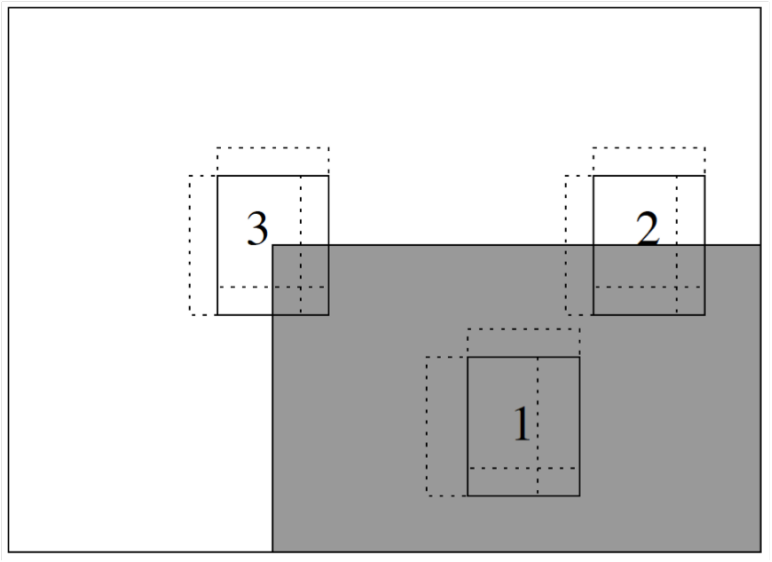


FIGURE 4.2: Les différentes situations considérées par le détecteur de Moravec.

3. La zone image considérée contient un coin ou un point isolé : la fonction E prendra de fortes valeurs dans toutes les directions.

En conséquence, le principe du détecteur de Moravec est donc de rechercher les maxima locaux de valeur minimale de E en chaque pixel (au dessus d'un certain seuil).

4.2.3 Le détecteur de Harris(1988)

Le détecteur de Moravec fonctionne dans un contexte limité. Il souffre en effet de nombreuses limitations. Harris et Stephen [30] ont identifié certaines limitations et, en les corrigeant, en ont déduit un détecteur de coins très populaire : le *détecteur de Harris*. Les limitations du détecteur de Moravec prises en compte sont :

1. La réponse du détecteur est anisotropique en raison du caractère discret les directions de changement que l'on peut effectuer (des pas de 45 degrés). Pour améliorer cet aspect, il suffit de considérer le développement de Taylor de la fonction d'intensité I au voisinage du pixel (u, v) :

$$I(x+u, y+v) = I(x,y) + x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2, y^2). \tag{4.2}$$

D'où :

$$E(x,y) = \sum_{u,v} w(u,v)[x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2, y^2)]^2, \tag{4.3}$$

En négligeant le terme $o(x^2, y^2)$ (valide pour les petits déplacements), on obtient l'expression analytique suivante :

$$E(x,y) = Ax^2 + 2Cxy + By^2, \tag{4.4}$$

avec :

- $A = \frac{\delta I^2}{\delta x} \otimes w$
- $B = \frac{\delta I^2}{\delta y} \otimes w$
- $C = (\frac{\delta I}{\delta x}\frac{\delta I}{\delta y}) \otimes w$

2. La réponse du détecteur de Moravec est bruitée en raison du voisinage considéré. Le filtre w utilisé est en effet binaire (valeur 0 ou 1) et est appliqué sur un voisinage rectangulaire. Pour améliorer cela, Harris et Stephen propose d'utiliser un filtre Gaussien :

$$w(u, v) = exp - (u^2 + v^2)/2\sigma^2. \tag{4.5}$$

3. Enfin, le détecteur de Moravec repond de manière trop forte aux contours en raison du fait que seul le minimum de E est pris en compte en chaque pixel. Pour prendre en compte le comportement général de la fonction E localement, on écrit :

$$E(x, y) = (x, y).M.(x, y)^t, \tag{4.6}$$

avec :

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

La matrice M caractérise le comportement local de la fonction E, les valeurs propres de cette matrice correspondent en effet aux courbures principales associées à E :
– Si les deux courbures sont de faibles valeurs, alors la région considérée a une intensité approximativement constante.
– Si une des courbures est de forte valeur alors que l'autre est de faible valeur alors la région contient un contour.
– Si les deux courbures sont de fortes valeurs alors l'intensité varié fortement dans toutes les directions, ce qui caractérise un coin.
Par voie de conséquence, Harris et Stephen proposent l'opérateur suivant pour détecter les coins dan une image :

$$R = Det(M) - k * Trace(M)^2 \tag{4.7}$$

avec :$Det(M) = AB - C^2$ et $Trace(M) = A + B$. Les valeurs de R sont positives au voisinage d'un coin, négatives au voisinage d'un contour et faibles dans une région d'intensité constante.

4.2.4 Implémentation

Nous allons implémenter l'algorithme de Harris pour détecter des coins sur le visage. Pour l'installer (4.7) nous devons calculer les gradients (première dérivée) A, B, C de la matrice M :

$$A = \frac{\delta I^2}{\delta x}; \qquad B = \frac{\delta I^2}{\delta y}; \qquad C = (\frac{\delta I}{\delta x}\frac{\delta I}{\delta y});$$

Nous avons utilisé l'approximation de Sobel pour calculer des gradients, $\frac{\delta I}{\delta x}$ est le gradient X de Sobel, $\frac{\delta I}{\delta y}$ est le gradient Y :

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix};$$

Donc, l'algorithme de cet opérateur pour une image est le suivant :

- Lissage de l'image (en utilisant le filtre binominal 3×3) :

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Calcul des gradients X et Y de l'image (en utilisant Sobel).
- Calcul des gradients XX et YY de l'image (en utilisant deux fois de suite Sobel pour l'image)
- Calcul des coins de Harris pour chaque pixel de l'image de la façon suivante :
  - ◇ Calcul de la matrice de Harris :

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

    - ⋆ $A = \frac{\delta I^2}{\delta x}$ est l'image convoluée 2 fois avec le gradient X de Sobel.
    - ⋆ $B = \frac{\delta I^2}{\delta y}$ est l'image convoluée 2 fois avec le gradient Y de Sobel.
    - ⋆ $C = (\frac{\delta I}{\delta x}\frac{\delta I}{\delta y})$ est l'image convoluée une fois avec le gradient X et une fois avec le gradient Y de Sobel.
  - ◇ Calcul de la trace et du déterminant de la matrice :
    - ⋆ $Trace(M) = A + B$
    - ⋆ $Déterminant(M) = AB - C^2$
  - ◇ Calcul de la réponse R du détecteur :
    - ⋆ $R = Déterminant(M) - k * Trace(M)^2$
    - ⋆ k est un paramètre à régler - typiquement $k = 0.04$
- Extraction des maxima locaux positifs dans un voisinage 3×3 de la réponse R (c'est-à-dire mettre à zéro tous les points négatifs ou dont la valeur n'est pas supérieure à celle de huit voisins).
- Extraction des n meilleurs points de Harris (par tri par insertion dans un tableau de taille n).
  - ◇ n étant un paramètre à configurer, avec par exemple $n = 2$

4.2.5 Résultats

Nous utilisons le détecteur de Harris pour détecter des coins sur deux régions des yeux sur le visage. Avec $n = 10$ (10 meilleurs coins de Harris) dans chaque région de l'œil, nous obtenons le résultat représenté dans la figure 4.3.

Nous avons trouvé que dans la région de l'œil, il y a trop de coins, il est difficile de distinguer des coins dans cette région. Nous avons défini la distance d entre des coins,

FIGURE 4.3: Détection de 10 meilleurs coins de Harris dans chaque région de l'œil.

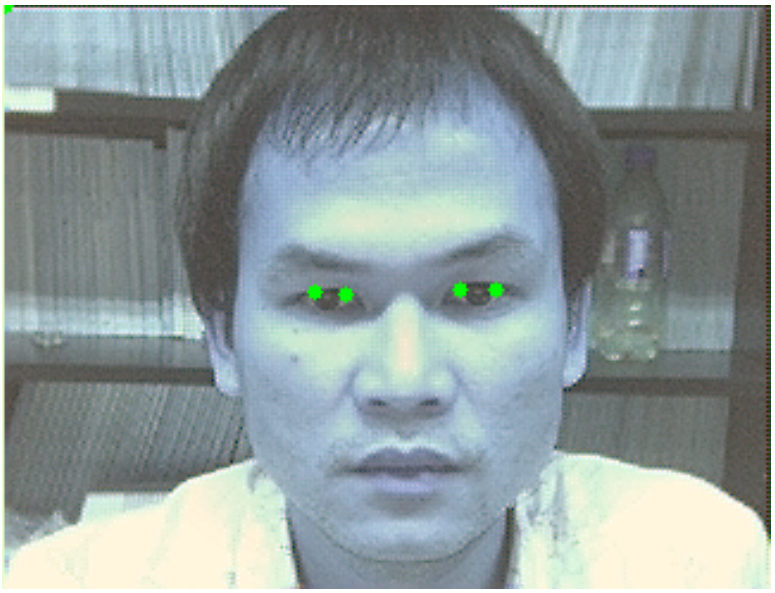


FIGURE 4.4: Détection des deux meilleurs coins de Harris dans chaque région de l'œil,

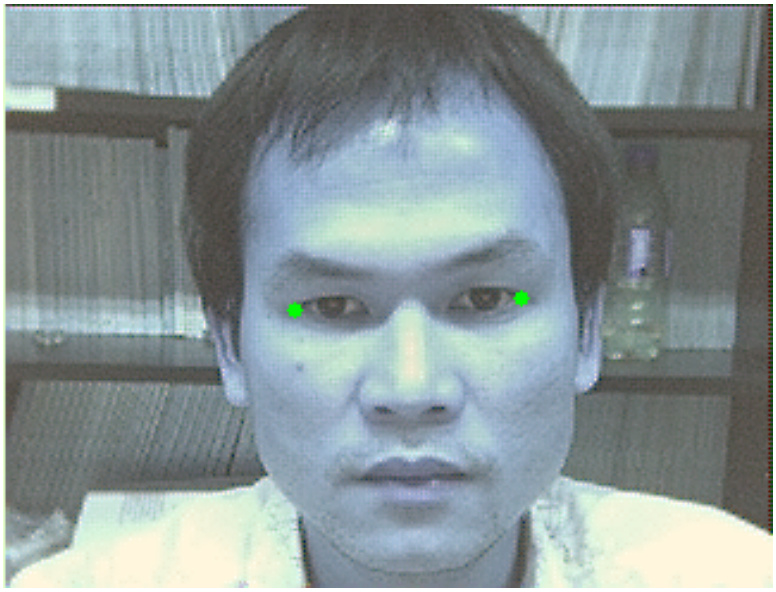


FIGURE 4.5: Détection du meilleur coin de Harris dans chaque région de la queue de l'œil.

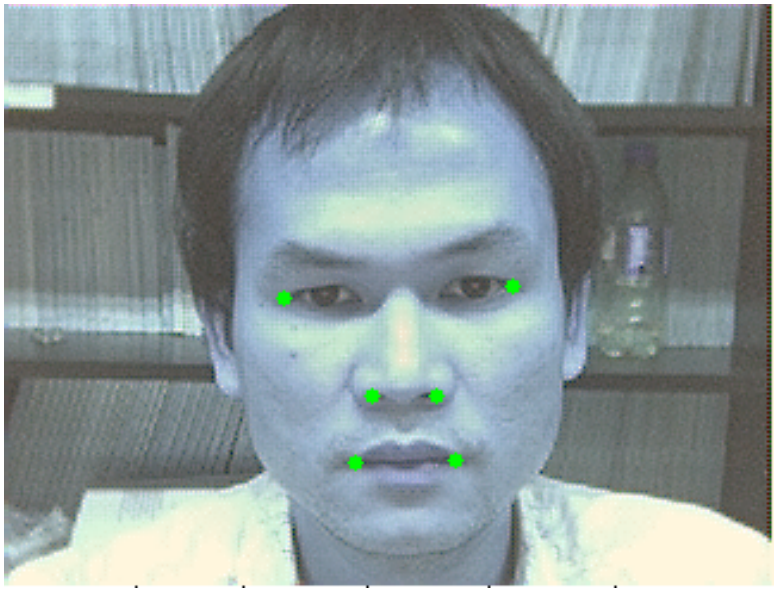


FIGURE 4.6: Résultat de la détection des coins sur les yeux, le nez et la bouche.

c'est-à-dire, après avoir détecté un coin, le coin suivant est éloigné du premier coin d'une distance d. Avec $d = 10, n = 2$, nous voulons détecter les deux meilleurs coins dans chaque région une distance entre les coins de 10, nous obtenons le résultat dans la figure 4.4.

Concernant les deux coins sur la pupille de chaque œil, nous ne les avons pas utilisé, la raison étant que la pupille bouge lorsque l'utilisateur regarde, même si la tête est stable. En conséquence, nous allons chercher les coins sur la queue de l'œil. Pour chercher ces coins, nous avons détecté un coin meilleur (n=1) sur la région de la queue de l'œil. Le résultat est présenté dans la figure 4.5.

Pour trouver les autres coins sur le nez et la bouche, nous avons détecté des coins dans la région au dessous des yeux en mettant la distance entre des coins de 25. Le résultat est présenté dans la figure 4.6.

4.3 Suivi des coins

Après la détection des coins sur le visage, nous allons suivre des coins par la méthode de suivi de Lucas Kanade.

4.3.1 Le problème

Soient I et J deux images échelles grises (ou *grayscale* en anglais) de $2D$. Les deux quantités $I(\mathbf{x}) = I(x, y)$ et $J(\mathbf{x}) = J(x, y)$ sont les valeurs échelles grises des deux images à la position $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$, où x et y sont les deux pixels coordonnés d'un point d'image générique $\mathbf{x}$. L'image I peut être référencée comme la première image, et l'image J comme la deuxième image. Pour un problème de pratique, les images I et J sont des fonctions discretes (ou arrays), et le vecteur de pixel coordonné supérieur à gauche est $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$. Soient n_x et n_y la largeur et la hauteur des deux images. Alors, le vecteur de pixel coordonné inférieure à droite est $\begin{bmatrix} n_x - 1 & n_y - 1 \end{bmatrix}^T$.
Considérons un point d'image $\mathbf{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$ sur la première image I. L'objectif du suivi de caractéristique est de trouver la position $\mathbf{v} = \mathbf{u} + \mathbf{d} = \begin{bmatrix} u_x + d_x & u_y + d_y \end{bmatrix}^T$ sur la deuxième image J tels que $I(\mathbf{u})$ et $I(\mathbf{v})$ sont "similaires". Le vecteur $\mathbf{d} = \begin{bmatrix} d_x & d_y \end{bmatrix}^T$ est la vitesse de l'image à $\mathbf{x}$, également connu sous le nom de flux optique à $\mathbf{x}$. En raison du *problème d'ouverture optique* (ou *aperture problem* en anglais), il est nécessaire de définir la notion de similitude dans le sens du voisinage de $2D$. Soient ω_x et ω_y deux entiers. Nous définissons une vitesse de l'image $\mathbf{d}$ comme le vecteur qui minimise la fonction résiduelle ϵ défini comme suit :

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_y} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x, y) - J(x + d_x, y + d_y))^2. \tag{4.8}$$

Observez la fonction ci-dessus, la fonction de similarité est mesurée dans une "image voisinage" (ou *image neighborhood* en anglais) de taille $(2\omega_x + 1) \times (2\omega_y + 1)$. Ce voisinage sera également appelé la fenêtre d'intégration. Les valeurs typiques pour ω_x et ω_y sont 2, 3, 4, 5, 6 et 7 pixels.

4.3.2 Suivi des coins par l'algorithme de calcul de flux optique de Lucas Kanade

Les deux composantes clés de toute caractéristique de suivi sont la précision et la robustesse. Le composant précis se rapporte à la précision du sous-pixel local dans le suivi. Intuitivement, une petite fenêtre d'intégration serait préférable afin de ne pas "aplanir" les détails contenus dans les images (i.e. les petits valeurs de ω_x et ω_y). Cela est particulièrement nécessaire aux régions d'occlusion dans les images où deux vecteurs se déplacent potentiellement avec des vitesses très différentes.

Le composant robustesse concerne à la sensibilité de suivi à l'égard du changement de l'éclairage, de la taille de l'image de mouvement,... En particulier, pour gérer des vastes mouvements, il est intuitivement préférable de choisir une grande fenêtre d'intégration. En effet, considérant seulement l'équation 4.8, il est préférable d'avoir $d_x \leq \omega_x$ et $d_y \leq \omega_y$. C'est donc un compromis naturel entre la précision locale et la robustesse au moment de choisir la taille de la fenêtre d'intégration. Pour donner la solution à ce problème, nous proposons une implémentation pyramidale de l'algorithme classique de Lucas-Kanade [7]. Une implémentation itérative de la computation du flux optique de Lucas-Kanade fournit suffisamment de la précision au suivi local.

4.3.2.1 Représentation pyramide de l'image

Laissez-nous définir la représentation pyramide d'une image générique I de la taille $n_x \times n_y$. Soit $I^0 = I$ est l'image de niveau "zero$^{\text{ième}}$". Cette image est l'image la plus haute en résolution(l'image crue). La largeur et la hauteur de l'image à ce niveau sont définies comme $n_x^0 = n_x$ et $n_y^0 = n_y$. La représentation pyramide est alors construite dans un mode récursif : calculer I^1 à partir de I^0, puis calculer I^2 à partir de I^1, et ainsi de suite... (la figure 4.7). Soit $L = 1, 2, ...$ est le niveau pyramide générique, et soit I^{L-1} est l'image au niveau $L-1$. Notons que n_x^{L-1} et n_y^{L-1} sont la largeur et la hauteur de I^{L-1}. L 'image I^{L-1} est définie comme suit :

$$\begin{aligned} I^L(x,y) &= \frac{1}{4} I^{L-1}(2x, 2y) + \\ &\quad \frac{1}{8}(I^{L-1}(2x-1, 2y) + I^{L-1}(2x+1, 2y) + \\ &\quad I^{L-1}(2x, 2y-1) + I^{L-1}(2x, 2y+1)) + \\ &\quad \frac{1}{16}(I^{L-1}(2x-1, 2y-1) + I^{L-1}(2x+1, 2y+1) + \\ &\quad I^{L-1}(2x-1, 2y+1) + I^{L-1}(2x+1, 2y+1)) \end{aligned} \tag{4.9}$$

Pour plus de simplicité dans la notation, nous définissons les valeurs mannequin d'un

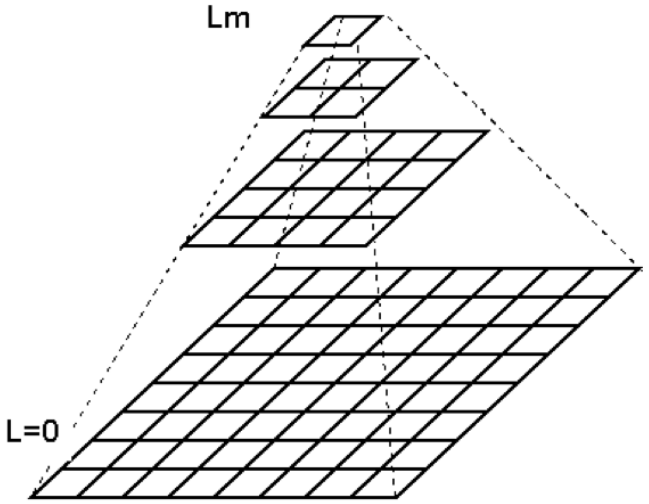


FIGURE 4.7: Représentation pyramide de l'image.

pixel autour de l'image I^{L-1} (pour $0 \leq x \leq n_x^{L-1} - 1$ et $0 \leq y \leq n_y^{L-1} - 1$) :

$$\begin{aligned} I^{L-1}(-1, y) &\doteq I^{L-1}(0, y), \\ I^{L-1}(x, -1) &\doteq I^{L-1}(x, 0), \\ I^{L-1}(n_x^{L-1}, y) &\doteq I^{L-1}(n_x^{L-1} - 1, y), \\ I^{L-1}(x, n_y^{L-1}) &\doteq I^{L-1}(x, n_y^{L-1} - 1), \\ I^{L-1}(n_x^{L-1}, n_y^{L-1}) &\doteq I^{L-1}(n_x^{L-1} - 1, n_y^{L-1} - 1). \end{aligned}$$

Ensuite, l'équation 4.9 est définie seulement pour les valeurs x et y tels que $0 \leq 2x \leq n_x^{L-1} - 1$ et $0 \leq 2y \leq n_y^{L-1} - 1$. Ainsi, la largeur n_x^L et la hauteur n_y^L de I^L sont les plus grand entiers qui satisfont aux deux conditions :

$$n_x^L \leq \frac{n_x^{L-1} + 1}{2} \tag{4.10}$$

$$n_y^L \leq \frac{n_y^{L-1} + 1}{2} \tag{4.11}$$

Les équations 4.9, 4.10 et 4.11 sont utilisées pour construire récursivement la représentation pyramide de deux images I et J : $\{I^L\}_{L=0,\ldots,L_m}$ et $\{J^L\}_{L=0,\ldots,L_m}$. La valeur L_m est la hauteur de la pyramide (à choisir heuristiquement). Les valeurs pratiques de L_m sont 2, 3 et 4. Pour la taille de l'image typique, il n'est pas logique d'avoir des valeurs supérieures à 4. Par exemple, pour une image I de la taille 640×480, les images I^1, I^2, I^3 et I^4 ont des tailles respectives de $320 \times 240, 160 \times 120, 80 \times 60$ et 40×30. Au-delà le niveau 4 n'a pas de sens dans la plupart des cas. La motivation centrale au dernière de la représentation pyramidale est d'être capable de traiter de grands mouvements de pixel (plus grands que la taille de la fenêtre d'intégration ω_x et ω_y). Par conséquent, la hauteur de la pyramide (L_m) devrait également être choisie de façon appropriée selon le maximum du flux optique entendu dans l'image. La section suivante décrit l'opération détaillée de suivi et nous permet de mieux comprendre ce concept. La dernière observation, l'équation 4.9, suggère que le filtre

passe-bas $[1/4 \quad 1/2 \quad 1/4] \times [1/4 \quad 1/2 \quad 1/4]^T$ est utilisé pour l'anticrénelage d'image (ou *anti-aliasing* en anglais) avant le sous-échantillonnage d'image (ou *subsampling* en anglais). Cependant dans la pratique, (dans le code C), un plus grand noyau de filtre anticrénelage est utilisé pour la construction pyramidale $[1/16 \quad 1/4 \quad 3/8 \quad 1/4 \quad 1/16] \times [1/16 \quad 1/4 \quad 3/8 \quad 1/4 \quad 1/16]^T$. Le formalisme reste le même.

4.3.2.2 Suivi la caractéristique pyramidale

Rappel de l'objectif du suivi de la caractéristique : avec un point $\mathbf{u}$ dans l'image I, trouver sa position correspondant $\mathbf{v} = \mathbf{u} + \mathbf{d}$ dans l'image J, ou trouver alternativement son vecteur de déplacement de pixel $\mathbf{d}$ (voir l'équation 4.8).

Pour $L = 0, ..., L_m$, soient $\mathbf{u}^L = [u_x^L \quad u_y^L]$, les coordonnées correspondant au point $\mathbf{u}$ sur l'image I^L. Suite à notre définition des équations de représentation pyramide (4.9), (4.10) et (4.11), les vecteurs $\mathbf{u}^L$ sont calculés comme suit :

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L}. \tag{4.12}$$

L'opération de division dans l'équation 4.12 est appliquée aux deux coordonnées de façon indépendante. On observe que en particulier, $\mathbf{u}^0 = \mathbf{u}$.

Le total de l'algorithme de suivi pyramidal procède comme suit : tout d'abord, le flux optique est calculé au niveau le plus profond L_m de la pyramide. Ensuite, le résultat de ce calcul est propagé au niveau supérieur $L_m - 1$, dans la forme d'une proposition initiale pour le déplacement de pixel (au niveau $L_m - 1$). Étant donné cette proposition initiale, le flux optique affiné est calculé au niveau $L_m - 1$, et le résultat est propagé au niveau $L_m - 2$ et ainsi de suite jusqu'au niveau 0 (l'image originale).

Nous allons maintenant décrire l'opération récursive entre deux niveaux génériques $L + 1$ et L en détails plus mathématiques. Supposons que la proposition initiale du flux optique au niveau $L, \mathbf{g}^L = [g_x^L \quad g_y^L]^T$ est disponible à partir des calculs effectués à partir du niveau L_m au niveau $L + 1$. Ensuite, afin de calculer le flux optique au niveau L, il est nécessaire de trouver le vecteur déplacement de pixel résiduel $\mathbf{d}^L = [d_x^L \quad d_y^L]$ qui minimise la fonction d'erreur ϵ^L :

$$\epsilon^L(\mathbf{d}^L) = \epsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x}^{u_x^L+\omega_x} \sum_{y=u_y^L-\omega_y}^{u_y^L+\omega_y} (I^L(x,y) - J^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L))^2. \tag{4.13}$$

Observez que la fenêtre d'intégration est de la taille constante $(2\omega_x + 1) \times (2\omega_y + 1)$ pour toutes les valeurs de L. Notez que la proposition initiale du flux optique $\mathbf{g}^L$ est utilisée pour pré-traduire le patch d'image (*image patch* en anglais) dans la deuxième image J. De cette manière, le vecteur du flux résiduel $\mathbf{d}^L = [d_x^L \quad d_y^L]^T$ est petit et donc facile à calculer par une étape standard de Lucas Kanade.

Le détail du calcul du flux optique résiduel $\mathbf{d}^L$ seront décrit dans la prochaine section 4.3.2.3. Pour l'instant, laissez-nous supposer que ce vecteur est calculé (pour fermer la boucle principale de l'algorithme). Ensuite, le résultat de ce calcul est propagé au niveau suivant $L - 1$ pour la nouvelle proposition initiale $\mathbf{g}^{L-1}$ de l'expression :

$$\mathbf{g}^{L-1} = 2(\mathbf{g}^L + \mathbf{d}^L). \tag{4.14}$$

Le niveau prochain du vecteur du flux optique résiduel $\mathbf{d}^{L-1}$ est ensuite calculé selon la même procédure. Ce vecteur, calculé par le calcul de flux optique (décrit dans la section 4.3.2.3), minimise la fonction $\epsilon^{L-1}(\mathbf{d}^{L-1})$ (l'équation 4.13). Cette procédure se termine quand la meilleure résolution d'image est atteinte ($L = 0$). L'algorithme est initialisé par la proposition initiale du niveau L_m à zéro (aucune proposition initiale n'est disponible au niveau le plus profond de la pyramide) :

$$\mathbf{g}^{L_m} = [0 \quad 0]^T \ . \tag{4.15}$$

La dernière solution de flux optique $\mathbf{d}$(se référer à l'équation 4.8) est disponible après le meilleur calcul de flux optique :

$$\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0. \tag{4.16}$$

Observez que cette solution peut être exprimée dans la forme étendue ci-dessous :

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L \mathbf{d}^L. \tag{4.17}$$

L'avantage d'une implémentation pyramidale est que chaque vecteur de flux optique résiduel $\mathbf{d}^L$ peut être gardé très faible lors du calcul d'un grand nombre total de pixels du vecteur déplacement $\mathbf{d}$. En supposant que chaque étape de calcul de flux optique élémentaire peut traiter des mouvements de pixel jusqu'au $d_{\max}$, puis le mouvement de tous les pixels que l'implémentation pyramidale peut gérer devient $d_{\max \text{ final}} = (2^{L_m+1} - 1)d_{\max}$. Par exemple, pour une profondeur de la pyramide de $L_m = 3$, cela signifie un déplacement maximum de pixel de 15! Cela permet à grands mouvements de pixel, tout en conservant une taille de la fenêtre d'intégration relativement faible.

4.3.2.3 Le calcul itératif de flux optique(l'itération de Lucas-Kanade)

Nous allons maintenant décrire le noyau de calcul de flux optique. À chaque niveau L dans la pyramide, l'objectif est de trouver le vecteur $\mathbf{d}^L$ qui minimise la fonction ϵ^L définie dans l'équation 4.13. En raison du même type d'opération effectué pour tous les niveaux L, supprimons maintenant les exposants L et définissons les nouvelles images A et B comme suit :

$\forall (x, y) \in [p_x - \omega_x - 1, p_x + \omega_x + 1] \times [p_y - \omega_y - 1, p_y + \omega_y + 1]$,

$$A(x, y) \doteq I^L(x, y), \tag{4.18}$$

$\forall (x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$,

$$B(x, y) \doteq J^L(x + g_x^L, y + g_y^L). \tag{4.19}$$

Observe que les domaines de définition de $A(x, y)$ et $B(x, y)$ sont légèrement différents. En effet, $A(x, y)$ est définie sur une fenêtre de taille $(2\omega_x + 3) \times (2\omega_y + 3)$ au lieu de $(2\omega_x + 1) \times (2\omega_y + 1)$. Cette différence deviendra claire lors du calcul de dérivés spatiales de $A(x, y)$ en utilisant l'opérateur de différence centrale (éqs. 4.26 et 4.27). Pour plus de

clarté, permettez-nous changer le nom du vecteur de déplacement $\overline{\nu} = [\nu_x \quad \nu_y]^T = \mathbf{d}^L$, ainsi que la vecteur position de l'image $\mathbf{p} = [p_x \quad p_y]^T = \mathbf{u}^L$. Suite à cette nouvelle notation, l'objectif est de trouver le vecteur de déplacement $\overline{\nu} = [\nu_x \quad \nu_y]^T$ qui minimise la fonction :

$$\varepsilon(\overline{\nu}) = \varepsilon(\nu_x, \nu_y) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} (A(x,y) - B(x+\nu_x, y+\nu_y))^2. \tag{4.20}$$

Une itération standard de Lucas-Kanade peut être appliquée à cette tâche. À l'optimum, la première dérivation de ε à l'égard de $\overline{\nu}$ est zéro :

$$\left.\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}}\right|_{\overline{\nu}=\overline{\nu}_{\text{opt}}} = [0 \quad 0] \ . \tag{4.21}$$

Après l'expansion de la dérivation, on obtient :

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} = -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} (A(x,y) - B(x+\nu_x, y+\nu_y)). \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} . \tag{4.22}$$

On va remplacer $B(x+\nu_x, y+\nu_y)$ par son premier ordre de l'expansion de Taylor sur le point $\overline{\nu} = [0 \quad 0]^T$ (ce qui a de bonnes chances d'être une approximation valable puisque nous attendons à un petit vecteur de déplacement, grâce au schéma pyramidal) :

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} (A(x,y) - B(x,y) - \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \overline{\nu}). \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} . \tag{4.23}$$

Observez que la quantité $A(x,y) - B(x,y)$ peut être interprétée comme la dérivation de l'image temporelle au point $[x \quad y]^T$:

$\forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$,

$$\delta I(x,y) \doteq A(x,y) - B(x,y). \tag{4.24}$$

La matrice $\begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}$ n'est que le vecteur gradient de l'image. Faisons un changement léger de la notation :

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}^T . \tag{4.25}$$

Observez que les dérivations de l'image I_x et I_y peuvent être calculées directement à partir de la première image $A(x,y)$ dans le voisinage $(2\omega_x + 1) \times (2\omega_y + 1)$ du point $\mathbf{p}$ indépendamment de la deuxième image $B(x,y)$ (l'important de cette observation deviendra évidente plus tard lors de la description de la version itérative du calcul de flux). Si un opérateur de différence centrale est utilisé pour les dérivés, les deux images dérivées ont l'expression suivante :

$\forall (x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$,

$$I_x(x, y) = \frac{\partial A(x, y)}{\partial x} = \frac{A(x + 1, y) - A(x - 1, y)}{2}, \tag{4.26}$$

$$I_y(x, y) = \frac{\partial A(x, y)}{\partial y} = \frac{A(x, y + 1) - A(x, y - 1)}{2}. \tag{4.27}$$

Dans la pratique, l'opérateur de Sharr est utilisé pour le calcul des dérivés d'image (très similaire à l'opération de différence centrale). Suite à cette nouvelle notation, l'équation 4.23 peut être écrite :

$$\frac{1}{2}\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} (\nabla I^T \overline{\nu} - \delta I) \nabla I^T, \tag{4.28}$$

$$\frac{1}{2}\Big[\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}}\Big]^T \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \Big(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \overline{\nu} - \begin{bmatrix} \delta I I_x \\ \delta I I_y \end{bmatrix} \Big). \tag{4.29}$$

Dénote

$$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad \text{et} \quad \overline{b} \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I I_x \\ \delta I I_y \end{bmatrix}. \tag{4.30}$$

Ensuite, l'équation 4.29 peut être écrit :

$$\frac{1}{2}\Big[\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}}\Big]^T \approx G\overline{\nu} - \overline{b}. \tag{4.31}$$

Ainsi, selon l'équation 4.21, l'optimum du vecteur de flux optique est :

$$\overline{\nu}_{\text{opt}} = G^{-1}\overline{b}. \tag{4.32}$$

Cette expression n'est valable que si la matrice G est inversible. Cela équivaut à dire que l'image $A(x, y)$ contient l'information de gradient dans les deux directions x et y dans le voisinage du point $\mathbf{p}$.

Ceci est l'équation standard de flux optique de Lucas-Kanade, qui n'est valide que si le déplacement de pixels est petit (afin de valider la première expansion de Taylor). Dans la pratique, pour obtenir une solution précise, il est nécessaire d'effectuer une itération plusieurs fois sur ce schéma (dans une mode de Newton-Raphson).

Nous avons alors introduit des connaissances en mathématique. Permettez-nous de donner les détails de la version itérative de l'algorithme. Rappel de l'objectif : trouver le vecteur $\overline{\nu}$ qui minimise la fonction d'erreur $\varepsilon(\overline{\nu})$ présenté dans l'équation 4.20.

Soit k l'indice itératif, initialisé à 1 à la première itération. Nous allons décrire l'algorithme récursif : à l'itération générique $k \geq 1$, supposons que les calculs précédents des itérations $1, 2, ..., k-1$ fournissent une proposition initiale $\overline{\nu}^{k-1} = \begin{bmatrix} \nu_x^{k-1} & \nu_x^{k-1} \end{bmatrix}^T$ pour le déplacement des pixels $\overline{\nu}$. Soit B_k la nouvelle image selon la proposition initiale $\overline{\nu}^{k-1}$:

$\forall (x,y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$,

$$B_k(x,y) = B(x + v_x^{k-1}, y + v_y^{k-1}). \tag{4.33}$$

L'objectif est de calculer le vecteur de mouvement de pixel résiduel $\overline{\eta}^k = [\eta_x^k \quad \eta_y^k]$ qui minimise la fonction d'erreur.

$$\varepsilon^k(\overline{\eta}^k) = \varepsilon(\eta_x^k, \eta_y^k) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \Big(A(x,y) - B_k(x+\eta_x^k, y+\eta_y^k)\Big)^2. \tag{4.34}$$

La solution de cette minimisation peut être calculée par une étape de calcul de flux optique de Lucas-Kanade (l'équation 4.32)

$$\overline{\eta}^k = G^{-1}\overline{b}_k, \tag{4.35}$$

où le 2×1 vecteur $\overline{b}_k$ est défini comme suit (appelé aussi le vecteur de décalage d'image) :

$$\overline{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y) I_x(x,y) \\ \delta I_k(x,y) I_y(x,y) \end{bmatrix}, \tag{4.36}$$

où la $k^{\text{ième}}$ différence de l'image δI_k est définie comme suite :

$\forall (x,y) \in [p_x - \omega_x, p - x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$,

$$\delta I_k(x,y) = A(x,y) - B_k(x,y). \tag{4.37}$$

Observez que les dérivées partielles I_x et I_y (à tous les points dans le voisinage de $\overline{p}$) sont calculées une seule fois au début des itérations suivantes des équations4.26 et 4.27. Ainsi, la 2×2 matrice G reste constante tout au long de la boucle de l'itération (l'expression donnée dans l'équation 4.30). Cela constitue un avantage de calcul clair. La seule quantité qui doit être recalculée à chaque étape k est le vecteur $\overline{b}_k$ qui saisit vraiment le montant de la différence résiduelle entre les patchs d'image après la transformation par vecteur $\overline{\nu}^{k-1}$. Une fois le flux optique résiduel $\overline{\eta}^k$ calculé par l'équation 4.35, une nouvelle proposition de déplacement de pixel $\overline{\nu}^k$ est calculé par la prochaine étape d'itération $k+1$:

$$\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k. \tag{4.38}$$

Le schéma itératif continue jusqu'au ce que le résidu de pixel calculé $\overline{\eta}^k$ soit plus petit que le seuil (par exemple 0,03 pixel). La maximum du nombre d'itérations (par exemple 20) est atteint. En moyenne, cinq itérations sont suffisantes pour atteindre la convergence. À la première itération (k-1), la proposition initiale est initialisé à zéro :

$$\overline{\nu}^0 = [0 \quad 0]^T \ . \tag{4.39}$$

En supposant que K itérations sont nécessaires pour atteindre la convergence, la solution finale pour le vecteur de flux optique $\overline{\nu} = \mathbf{d}^L$ est :

$$\overline{\nu} = \mathbf{d}^L = \overline{\nu}^K = \sum_{k=1}^{K} \overline{\eta}^k. \tag{4.40}$$

Ce vecteur minimise la fonction d'erreur décrite dans l'équation 4.20 (ou l'équation 4.13). Ceci termine la description de la computation itérative de flux optique de Lucas-Kanade. Le vecteur $\mathbf{d}^L$ est utilisé pour l'équation 4.14 et cette procédure est répétée à tous les niveaux $L-1, L-2, ..., 0$ (voir la section 4.3.2.2).

4.3.2.4 Résumé de l'algorithme de suivi pyramidal

Nous allons maintenant résumer en totalité l'algorithme de suivi sous la forme de pseudo-code. Les détails des équations sont trouvés dans la section précédente (en particulier pour les domaines de définition).

L'objectif : Soit $\mathbf{u}$ un point sur l'image I. Cherchons son emplacement correspondant $\mathbf{v}$ sur l'image J

Construire des représentation pyramidales I et J : $\{I^L\}_{L=0,\ldots,L_m}$ et $\{J^L\}_{L=0,\ldots,L_m}$

Initialisation la proposition pyramidale : $\mathbf{g}^{L_m} = [g_x^{L_m} \quad g_x^{L_m}]^T = [0 \quad 0]^T$

For $L = L_m$ down to 0 with step of -1

Localisation du point $\mathbf{u}$ *sur l'image* I^L : $\mathbf{u}^L = [p_x \quad p_y]^T = \mathbf{u}/2^L$

Derivée de I^L *à l'égard de* x : $$I_x(x,y) = \frac{I^L(x+1,y) - I^L(x-1,y)}{2}$$

Derivée de I^L *à l'égard de* y : $$I_y(x,y) = \frac{I^L(x,y+1) - I^L(x,y-1)}{2}$$

Matrice gradient spatial : $$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Initiation de l'itérative L-K : $\overline{\nu}^0 = [0 \quad 0]^T$

for $k = 1$ to K with step of 1 (ou jusqu'à $\|\overline{\eta}^k\| \leq$ le seuil de précision)

Différence d'image : $$\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$$

Vecteur décalage d'image : $$\overline{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y) I_x(x,y) \\ \delta I_k(x,y) I_y(x,y) \end{bmatrix}$$

Flux optique (Lucas-Kanade) : $\overline{\eta}^k = G^{-1}\overline{b}_k$

Proposition pour l'itération prochaine : $\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k$

End of for-loop on k

Flux optique final au niveau L : $\mathbf{d}^L = \overline{\nu}^K$

Proposition au niveau prochain L-1 : $\mathbf{g}^{L-1} = [g_x^{L-1} \quad g_y^{L-1}]^T = 2(\mathbf{g}^L + \mathbf{d}^L)$

End of for-loop on L

Vecteur de flux optique final : $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$

Location de point sur J : $\mathbf{v} = \mathbf{u} + \mathbf{d}$

Solution : Le point correspondant à la location $\mathbf{v}$ sur l'image J.

4.3.2.5 Computation de sous-pixel(subpixel)

Il est absolument essentiel de garder tous les calculs à un niveau précis de sous-pixel. Il est donc nécessaire d'être capable de calculer les valeurs de la luminosité de l'image à des endroits entre les pixels entiers (se référer par exemple aux équations 4.18, 4.19 et 4.33). Afin de calculer luminosité de l'image dans des locations sous-pixels, nous proposons d'utiliser une interpolation bilinéaire.

Soit L un niveau pyramide générique. Supposons que nous avons besoin de la valeur d'image $I^L(x, y)$ où x et y ne sont pas des entiers. Soit x_0 et y_0 les parties entières de x et y (les entiers plus petits que x et y). Soient α_x et α_y les deux valeur de rappel (entre 0 et 1) tels que :

$$x = x_0 + \alpha_x, \tag{4.41}$$

$$y = y_0 + \alpha_y, \tag{4.42}$$

Ensuite, $I^L(x, y)$ peut être calculée par interpolation bilinéaire à partir des valeurs originales de la luminosité de l'image :

$$\begin{aligned} I^L(x,y) &= (1-\alpha_x)(1-\alpha_y)I^L(x_0,y_0) + \alpha_x(1-\alpha_y)I^L(x_0+1,y_0) + \\ &\quad (1-\alpha_x)\alpha_y I^L(x_0,y_0+1) + \alpha_x\alpha_y I^L(x_0+1,y_0+1). \end{aligned}$$

Faisons un petit nombre d'observations associées au calcul des sous-pixels (la tâche d'implémentation importante). Référons-nous au résumé de l'algorithme donné dans la section 4.3.2.4. Lors du calcul des deux dérivés d'image $I_x(x, y)$ et $I_y(x, y)$ dans le voisinage $(x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$, il est nécessaire d'avoir les valeurs de luminosité de $I^L(x, y)$ dans le voisinage $(x, y) \in [p_x - \omega_x - 1, p_x + \omega_x + 1] \times [p_y - \omega_y - 1, p_y + \omega_y + 1]$ (voir l'équation 4.26, 4.27). Évidemment, les coordonnées du point central $\mathbf{p} = \begin{bmatrix} p_x & p_y \end{bmatrix}^T$ ne sont pas garanties d'être des entiers. Appelons p_{x_0} et p_{y_0} les parties entières de p_x et p_y. Ensuite, nous pouvons écrire :

$$p_x = p_{x_0} + p_{x_\alpha}, \tag{4.43}$$

$$p_y = p_{y_0} + p_{y_\alpha}, \tag{4.44}$$

où p_{x_α} et p_{y_α} sont les valeurs de rappel entre 0 et 1. Ainsi, afin de calculer le patch d'image $I^L(x, y)$ dans le voisinage $(x, y) \in [p_x - \omega_x - 1, p_x + \omega_x + 1] \times [p_y - \omega_y - 1, p_y + \omega_y + 1]$ par interpolation bilinéaire, il est nécessaire d'utiliser l'ensemble des valeurs de luminosité d'origine $I^L(x, y)$ dans le patch grille entier $(x, y) \in [p_{x_0} - \omega_x - 1, p_{x_0} + \omega_x + 2] \times [p_{y_0} - \omega_y - 1, p_{y_0} + \omega_y + 2]$ (rappelons que ω_x et ω_y sont des entiers).

Une situation similaire se produit lors du calcul de la différence d'image $\delta I_k(x, y)$ dans le voisinage $(x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$ (se référer à la section 4.3.2.4). En effet, afin de calculer $\delta I_k(x, y)$, il est tenu d'avoir les valeurs $J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$ pour tout $(x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y]$, ou, en d'autres termes, les valeurs de $J^L(\alpha, \beta)$ pour tout $(\alpha, \beta) \in [p_x + g_x^L + \nu_x^{k-1} - \omega_x, p_x + g_x^L + \nu_x^{k-1} + \omega_x] \times [p_y + g_y^L + \nu_y^{k-1} - \omega_y, p_y + g_y^L + \nu_y^{k-1} + \omega_y]$. Bien sûr, $p_x + g_x^L + \nu_x^{k-1}$ et $p_y + g_y^L + \nu_y^{k-1}$ ne sont pas nécessairement des entiers. Appelons q_{x_0} et q_{y_0} les parties entières de $p_x + g_x^L + \nu_x^{k-1}$ et $p_y + g_y^L + \nu_y^{k-1}$:

$$p_x + g_x^L + \nu_x^{k-1} = q_{x_0} + g_{x_\alpha}, \tag{4.45}$$

$$p_y + g_y^L + \nu_y^{k-1} = q_{y_0} + q_{y_\alpha}, \tag{4.46}$$

où q_{x_α} et q_{y_α} sont les valeurs de rappel associées entre 0 et 1. Puis, afin de calculer le patch image $J^L(\alpha, \beta)$ dans le voisinage $(\alpha, \beta) \in [p_x + g_x^L + \nu_x^{k-1} - \omega_x, p_x + g_x^L + \nu_x^{k-1} + \omega_x] \times [p_y + g_y^L + \nu_y^{k-1} - \omega_y, p_y + g_y^L + \nu_y^{k-1} + \omega_y]$, il est nécessaire d'utiliser l'ensemble des valeurs de luminosité d'origine $J^L(\alpha, \beta)$ dans le patch grille entier $(\alpha, \beta) \in [q_{x_0} - \omega_x, q_{x_0} + \omega_x + 1] \times [q_{y_0} - \omega_y, q_{y_0} + \omega_y + 1]$.

4.3.2.6 Suivi des caractéristiques proches de la frontière des images

Il est très intéressant d'observer qu'il est possible de traiter des points qui sont assez proches de la frontière de l'image pour avoir une portion de leur fenêtre d'intégration à l'extérieur de l'image. Cet observation est très importante lors que le nombre de niveaux pyramidaux L_m est grand. En effet, si nous appliquons toujours la fenêtre complète $(2\omega_x + 1) \times (2\omega_y + 1)$ pour être dans l'image afin d'être suivi, on observe la "bande interdite" de lageur ω_x (et ω_y) tout autour des images I^L. Si L_m est la hauteur de la pyramide, cela signifie une bande interdite effective de largeur $2^{L_m}\omega_x$ (et $2^{L_m}\omega_y$) autour de l'image originale I. Pour les valeurs petites de ω_x, ω_y et L_m, cela pourrait ne pas constituer une limitation importante. Toutefois, cela peut devenir très gênant pour les fenêtres d'intégration grandes tailles, et plus important encore, pour les grandes valeurs de L_m. Quelques chiffres : $\omega_x = \omega_y = 5$ pixels, et $L_m = 3$ conduit à une bande interdite de 40 pixels autour de l'image !

Afin d'éviter que cela ne se produise, nous proposons de surveiller des points dont les fenêtres d'intégration sont partiellement hors de l'image (à tous les niveaux pyramidaux). Dans ce cas, la procédure de suivi reste la même, espérons que les sommations apparaissant dans toutes les expressions (voir le pseudo-code de la section 4.3.2.4) ne se fassent que dans la partie valide du voisinage de l'image, i.e. la partie du voisinage qui est valide pour $I_x(x, y), I_y(x, y)$ et $\delta I_k(x, y)$ (voir la section 4.3.2.4). Ainsi, les régions de sommation valide peuvent varier tout en passant par le boucle d'itération de Lucas-Kanade (boucle sur l'indice k dans le pseudo-code - Section 4.3.2.4). En effet, de l'itération à l'autre, le entry valide de la différence de l'image $\delta I_k(x, y)$ peut varier selon la variation du vecteur de translation $[g_x^L + \nu_x^{k-1} \quad g_y^L + \nu_y^{k-1}]^T$. En outre, observez que lors du calcul du vecteur inadéquation $\overline{b}_k$ et du gradient de la matrice G, les régions de sommation doivent être identiques (et rester valide pour les mathématiques). Par conséquent, dans ce cas, la matrice G doit être recalculée dans la boucle de chaque itération. Les patchs différentiels $I_x(x, y)$ et $I_y(x, y)$ peuvent être calculés avant la boucle d'itération.

Bien sûr, si le point $\mathbf{p} = [p_x \quad p_y]^T$ (la centre du voisinage) tombe en dehors de l'image I^L, ou si son point correspondant de suivi $[p_x + g_x^L + \nu_x^{k-1} \quad p_y + g_y^L + \nu_y^{k-1}]$ tombe en dehors de l'image J^L, il est raisonnable de déclarer le point "perdu", et de ne pas le suivre.

4.3.2.7 Déclaration d'une carctéristique "perdu"

Il y a deux cas qui peuvent produire une caractéristique "perdue". Le premier cas est très intuitif : le point tombe en dehors de l'image. Nous avons discuté de ce problème dans la section 4.3.2.6. L'autre cas de perte, c'est quand le patch d'image autour du point de suivi varie trop entre l'image I et l'image J (le point disparaît en raison de l'occlusion).

Remarquons que cette condition est beaucoup plus difficile à quantifier avec précision. Par exemple, un point caractéristique peut être déclaré "perdu" si sa fonction de coût final $\epsilon(\mathbf{d})$ est supérieure à un seuil (voir l'équation 4.8). Le problème se produit quand après avoir décidé d'un seuil. Il est particulièrement sensible lors du suivi d'un point sur plusieurs images dans une séquence complète. En effet, si le suivi est basée sur des paires d'images consécutives, le patch d'image de suivi est implicitement initialisé à chaque image. Par conséquent, le point peut dériver pendant toute la séquence prolongée même si la différence d'image entre deux images consécutives est très faible. Ce problème de dérive est un problème très classique lorsqu'il s'agit de longues séquences. Une approche est de suivre les points caractéristiques à travers une séquence tout en gardant une référence fixe pour l'apparition du patch caractéristique (utiliser la première image de la caractéristique apparue). Suite à cette technique, la quantité $\epsilon(\mathbf{d})$ a beaucoup plus de sens. Tout en adoptant cette approche cependant un autre problème se lève : une caractéristique peut être déclarée "perdue" trop rapidement. Une direction que nous envisageons pour répondre à ce problème est d'utiliser l'image affine correspondant pour décider la perte de suivi (voir Shi et Tomasi dans [91]).

4.4 Détection des coins "perdus"

Dans le processus de suivi des coins, lorsque l'utilisateur bouge la tête très vite, des erreurs peuvent se produire (un coin perdu de suivi). Nous devons donc détecter ces erreurs et les récupérer. Nous avons utilisé l'algorithme de détection Outilier basé sur des distances pour trouver les coins perdus et les récupérer.

4.4.1 Le problème

En ensembles de données, l'identification des cas rares ou exceptionnels par rapport à un groupe de cas similaires est considéré comme un problème très important. Le problème traditionnel (*Outlier Mining*) est de trouver des exceptions ou cas rare dans un ensemble de données indépendamment de l'étiquette de classe de ces cas. Ils sont considérés comme des événements rares à l'égard de l'ensemble des données. Dans cette section, nous posons le problème appelé *Class Outliers Mining* et une méthode pour déterminer ces outliers. La définition générale de ce problème est de *"donner un ensemble d'observations avec des étiquettes de classe, trouver ceux qui éveillent des soupçons, en tenant compte des étiquettes de classe"*.

Un Outlier est une observation qui s'écarte tellement d'autres observations comme des soupçons qu'il a été généré par un mécanisme différent [31]. Il s'agit d'un objet de données qui ne se conforme pas avec le comportement général des données, il peut être considéré comme du bruit ou exceptionnel, qui est très utile dans l'analyse des événements rares [38].

4.4.2 L'algorithme de détection outlier basée sur la distance

Certaines des méthodes bien connues pour la détection des Outliers sont basées sur la statistique (basé sur la distribution) [5] [31] [85], le clustering [4] [24] [45], la profondeur [39], la distance [45] [47] [46], la densité [10] et le modèle (Réseau de neurones) [32]. Nous discuterons en détail de la méthode basée sur la distance en raison de sa relation avec notre approche.

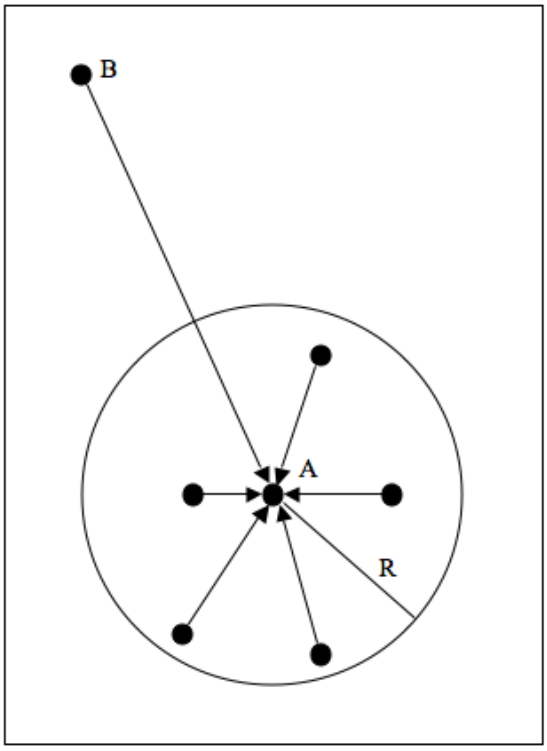


FIGURE 4.8: Outlier basé sur la distance.

La définition des outliers par leur distance à des exemples voisins est une approche populaire pour trouver des exemples inhabituels dans un ensemble de données. Une méthode populaire d'identification des outlliers consiste à examiner la distance aux plus proches voisins d'un exemple [2] [45] [46] [83]. Basé sur la distance, on regarde toutes les distances de chaque point à ses points voisins. La distance maximum d'un point à ses voisins est définie par R, un point est le voisin d'un autre point si la distance entre eux est inférieure à R, sinon il n'est pas le voisin. On va chercher le point qui a le maximum de voisins, on appelle ce point A, et on va considérer les autres points : un point est considéré comme normal s'il est un voisin de A. S'il n'est pas voisin de A, alors ce point est considéré comme inhabituel. Dans la figure 4.8, B est un point inhabituel, il est un outlier. Les avantages des outliers basés sur la distance sont qu'aucune distribution explicite ne doit être définie afin de déterminer un inhabituel, et qu'il peut être appliqué à tout l'espace caractéristique pour laquelle nous pouvons définir une mesure de distance.

Dans notre travail, un exemple est la transaction d'un point entre deux frames consécutives (voir dans la figure 4.9). Supposons qu'un coin I dans le frame 1a une position (x_1, y_1),

FIGURE 4.9: Un exemple de transaction entre deux frames d'image.

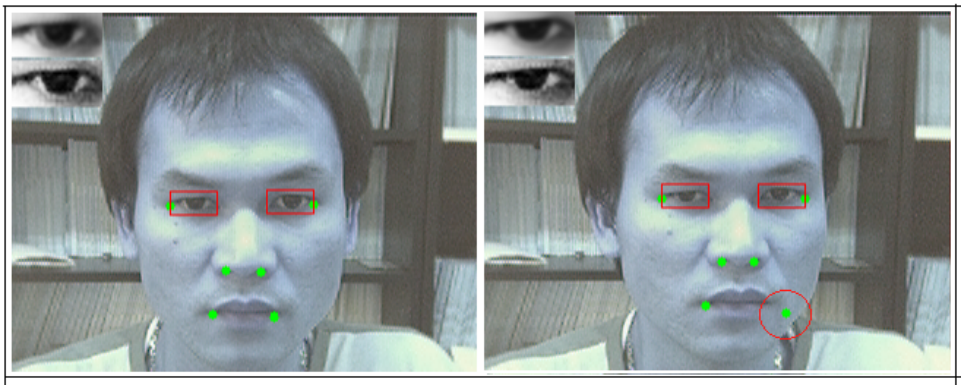


FIGURE 4.10: Un exemple de coin perdu lors de suivi des coins.

Détection outlier

L'objectif : Soit D un ensemble de données de transaction. Chercher les transactions erreurs (outliers) dans D.
Entrées :
– $D[n]$ \\ un array inclus n données de transaction, initialiser $D[i] = 0$ pour tous $i = 0, ..., n$.
– R la distance maximum de voisinage.
Sorties : $outlier[n]$ \\ $outlier[i] = 1$ si la transaction i est outlier, $outlier[i] = 0$ si non, initialiser $outlier[i] = 0$ pour tous $i = 0, ..., n$

new Array $voisins[n]$ \\ - un array pour compter les voisins de chaque exemple, initialiser $voisins[i] = 0$ pour tous $i = 0, ..., n$.
For $i = 0$ **to** n **with step 1**
    For $j = 0$ **to** n **with step 1**
        if $(d(D[i], D[j]) <= R)$ $voisins[i] + +$;
\\ chercher l'exemple qui a le maximum de voisins
$max = max_e lement(voisins)$;
\\ chercher des outliers
For $i = 0$ **to** n **with step 1**
    if $(d(D[i], D[j]) <= R) outlier[i] = 1$;
return $outlier$;

TABLE 4.1: L'algorithme de détection des outliers

et dans le frame 2 a une position (x_2, y_2). La valeur de transaction I du frame 1 au frame 2 est donc de $Transaction_I(x_2 - x_1, y_2 - y_1)$. La distance $d(I_1, I_2)$ Euclide entre deux transactions $I_1(x_1, y_1), I_2(x_2, y_2)$ est calculée comme suit

$$d(I_1, I_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

L'algorithme de détection outlier est sous la forme de pseudo-code dans la table 4.1. Un exemple de coin perdu lors de suivi des coins est présenté figure 4.10.

4.4.3 Récupérer des coins perdus

Pour récupérer des coins perdus, il y a deux méthodes :
– Utiliser l'exemple qui a le maximum de voisins. Nous pouvons utiliser cet exemple comme la transaction standard pour récupérer les coins perdus. La valeur récupérée de coin perdu sera donc :

$$valeur\ récupérée = valeur\ de\ coin\ dans\ le\ frame\ précédent + D[max]$$

– Utiliser la moyenne de toutes les transactions. Nous pouvons utiliser la moyenne de toutes les transactions comme la transaction standard pour récupérer les coins

perdus. La valeur récupérée de coin perdu sera

$$\textit{valeur récupérée} = \textit{valeur de coin dans le frame précédent} + Moyenne(D)$$

où

$$Moyenne(D) = \frac{1}{n}\sum_{i=1}^{n} D[i]$$

Les résultats des deux méthodes sont presque les mêmes, la raison est que nous utilisons seulement 6 coins pour le suivi, et la transaction de coin entre deux frames consécutives est très petite. Les procédures de détection et de récupération des coins perdus sont réalisées à chaque frame d'image pour avoir un bon suivi des coins.

4.5 Conclusion

Nous avons présenté le problème de suivi des yeux, avec ce problème nous devons résoudre trois sous problèmes principaux : Détection des coins, Suivi des coins et Récupération des coin "perdus". D'abord, nous avons utilisé la méthode de détection de coins Harris pour détecter des coins importants sur la face de l'utilisateur (les coins des yeux, du nez et de la bouche). Ensuite, nous avons utilisé la méthode de suivi de Lucas Kanade pour suivre les mouvements des coins dans la caméra. Enfin, nous avons présenté la méthode de détection Outliers pour détecter des coins "perdus" (les coins qui sont perdus de suivi) et les récupérer.

Chapitre 5

Prédiction du regard par la Régression Processus Gaussien

Dans le dernier chapitre, nous avons procédé à la détection des yeux de l'utilisateur à partir d'une video capturé par une simple caméra de type webcam. Nous avons ensuite procédé au suivi des yeux lorsque l'utilisateur bouge sa tête en face de la caméra. Dans ce chapitre, nous allons présenter comment détecter le regard de l'utilisateur sur l'écran à partir des images des yeux obtenus, c'est-à-dire, comment trouver la position du regard en temps réel sur l'écran de l'ordinateur où l'utilisateur regard. Nous allons utiliser la *Régression Processus Gaussien* pour détecter le regard d'utilisateur.

Pour cela, nous allons analyser le problème de suivi du regard et détailler pourquoi nous avons choisi d'utiliser le processus Gaussien et préciser comment l'appliquer pour détecter le regard de l'utilisateur.

5.1 Introduction

Pour détecter le regard de l'utilisateur sur l'écran, nous devons trouver la relation entre les images des yeux de l'utilisateur et la position qu'il regarde sur l'écran. Nous transformons ce problème à la langue plus proche de la mathématique. Soient $\mathbf{x}$ (l'entrées) les images des yeux de l'utilisateur, et y (la sortie) la position sur l'écran où l'utilisateur regarde. Le problème est de trouver le lien fonctionnel f tel que $y = f(\mathbf{x})$, voir la figure 5.1.

Pour trouver le lien fonctionnel $f(\mathbf{x})$, nous devons utiliser des données empiriques (l'ensemble de données de formation), c'est à dire, les images des yeux et les points du regard correspondants sur l'écran. Le problème est de mettre en correspondance des entrées et des sorties à partir des données empiriques, c'est le problème de l'apprentissage supervisé. Grâce à la sortie continu, ce problème est connu comme la *régression*. Précisément, le problème de suivi du regard peut être présenté de la manière suivante :

Soient $\mathbf{x}$ l'entrée, et y la sortie (ou cible). L'entrée est représentée comme un vecteur $\mathbf{x}$ parce qu'il y a plusieurs variables d'entrée (les valeurs des pixels). La sortie y est continue (le cas de régression). Nous avons un ensemble de données $\mathcal{D}$ de n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$. Compte tenu de ces données de formation, nous souhaitons faire

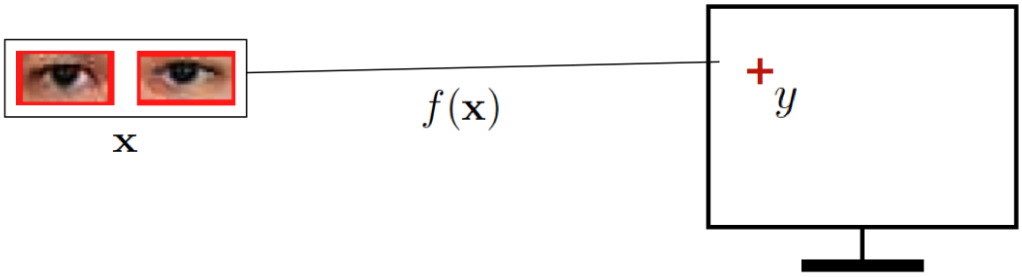


FIGURE 5.1: La relation entre les images des yeux de l'utilisateur et la position sur l'écran où l'utilisateur regarde.

des prédictions pour les nouvelles entrées $\mathbf{x}_*$ absentes de la base de données de formation. Le problème que se pose est donc *inductif* : nous avons besoin de trouver à partir d'un nombre fini de données de formation $\mathcal{D}$, une fonction f qui peut faire les prédictions pour toutes les valeurs d'entrée possibles.

5.2 Pourquoi le Processus Gaussien ?

Une grande variété de méthodes a été proposée pour régler le problème d'apprentissage supervisé ; ici, nous décrivons deux approches communes. La première consiste à restreindre la classe des fonctions que l'on considère, par exemple en ne tenant compte que des fonctions linéaires de l'entrée. La seconde approche est de donner une probabilité a priori à tous les fonctions possible, où les plus fortes probabilités sont données aux fonctions que nous considérons arbitrairement comme plus probable, par exemple parce qu'elles sont plus lisses que d'autres fonctions. La première approche conduit un problème évident issu de la richesse de la classe des fonctions considérées : si nous utilisons un modèle basé sur une certaine classe de fonctions (par exemple des fonctions linéaires) et que la fonction cible n'est pas bien modélisée par cette classe, alors les prédictions seront pauvres. On peut être tenté d'augmenter la flexibilité de la classe des fonctions, mais avec un risque de sur-apprentissage, où nous pouvons alors obtenir un bon ajustement aux données de formation, mais obtenir de mauvais résultats ensuite.

La deuxième approche conduit à un sérieux problème. En effet, il existe un ensemble infini de fonctions possibles, aussi comment allons-nous calculer cet ensemble dans un temps fini ? Nous avons choisi d'utiliser le Processus Gaussien. Un *processus* de Gauss est une généralisation de la *distribution* de probabilité Gaussienne. Considérant une distribution de probabilité qui décrit des variables aléatoires scalaires ou vecteurs (pour les distributions multivariées), un *processus* stochastique régit les propriétés des fonctions. Simplement, on peut penser une fonction comme un vecteur très long, chaque entrée du vecteur spécifiant la valeur de la fonction $f(x)$ à une entrée particulière x. Il s'avère que, bien que cette idée soit un peu naïve, elle est étonnamment proche de ce dont nous avons besoin. En effet, la question de savoir comment traiter des calculs avec ces objets de dimension infinie a une résolution plus simple que présupposé : si nous demandons que les propriétés de la fonction pour un nombre fini de points. L'inférence dans le Processus

Gaussien donnera la même réponse si nous ignorons beaucoup d'autres points que si nous les prenons tous en compte ! Ces réponses seront consistantes avec les réponses à toute autre ensemble fini de question. Un des principaux avantages de la structure de processus Gaussien est qu'elle unit une approche sophistiquée avec des calculs accessibles.

Soit un problème de *régression* simple 1-d : un mappage à partir d'une entrée x vers une sortie $f(x)$. Dans la figure 5.2 (a), nous montrons un certain nombre de fonctions d'échantillons dessinées au hasard à partir de la distribution *a priori* (ou prior en anglais) sur les fonctions spécifiées par un processus Gaussien particulier qui favorise les fonctions lisses. Cette distribution est choisi pour représenter nos croyances *a priori* sur les types de fonctions que l'on s'attend à observer, avant de voir une donnée. Par défaut, nous supposons que la valeur moyenne sur les fonctions de l'échantillon à chaque x est zéro. Bien que les fonctions aléatoires spécifiques dessinées dans la figure 5.2 (a) n'ont pas une moyenne de zéro, pour chaque x la valeur moyenne de toutes les fonctions $f(x)$ est zéro, indépendante de x, nous avons continué à dessiner plus de fonctions. Pour une valeur de x, on peut également caractériser la variabilité des fonctions de l'échantillon en calculant la variance à ce point. La région ombrée représente deux fois la écart type ; dans ce cas, nous avons utilisé un processus Gaussien qui précise que la variance a priori ne dépend pas de x.

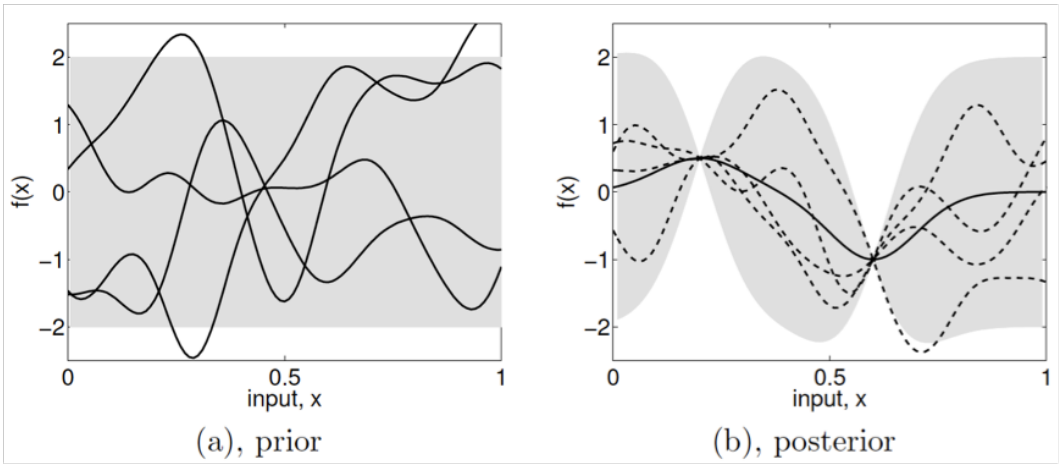


FIGURE 5.2: (a) quatre échantillons dessinés à partir de la distribution a priori. (b) situation après deux points de données observés. La prédiction moyenne est représentée par la ligne continue et quatre échantillons de la postérieure sont indiqués par des lignes pointillées.

Supposons un ensemble de données $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\}$ composé de deux observations. Nous considérons maintenant que des fonctions passent exactement par ces deux points de données. (Il est également possible de donner une plus grande préférence à des fonctions qui passent simplement "proche" aux points de données.) Cette situation est illustrée dans la Figure 5.2 (b). Les lignes en pointillés représentent les fonctions d'échantillonnage compatibles avec $\mathcal{D}$, et la ligne continue représente la valeur moyenne de ces fonctions. Notez la façon dont l'incertitude est réduite à proximité de la observations. La combinaison de la distribution *a priori* et des données conduit à la distribution *a posteriori* sur des fonctions.

Si l'on ajoute plus de points de données, on voit la fonction de moyenne s'adapter elle-même pour passer par ces points, et que l'incertitude postérieure permet de réduire la distance à ces observations. Notons que, le processus de Gaussien n'est pas un modèle paramétrique, nous n'avons pas besoin de savoir si le modèle peut ajuster les données (Ce serait le cas si on utilise un modèle linéaire sur des données fortement non-linéaires). Même quand un grand nombre d'observations est ajouté, il existe encore une certaine flexibilité dans les fonctions. Une façon d'imaginer la réduction de la flexibilité dans la distribution de fonctions lors que des données arrivent, est de dessiner de nombreuses fonctions au hasard à partir de la distribution a priori, et de rejeter celles qui ne sont pas en accord avec les observations. Alors que c'est une manière tout à fait valide de faire l'inférence, elle n'est pas pratique dans la plupart des cas. Les calculs analytiques exactes nécessaires pour quantifier ces propriétés seront détaillés dans la section suivante.

La spécification de a priori est importante, car elle fixe les propriétés des fonctions considérées pour l'inférence. Nous avons brièvement abordé ci-dessus la moyenne et la variance ponctuelle des fonctions. Mais d'autres caractéristiques peuvent également être spécifiées et manipulée. Notez que les fonctions de la figure 5.2 (a) sont lisses et stationnaire (stationnaire signifie que les fonctions ressemblent à tous les endroits x). Ces sont des propriétés induites par *la fonction de covariance* du processus Gaussien ; de nombreuses autres fonctions de covariance sont possibles. Supposons que, pour une application particulière, nous pensons que les fonctions de la figure 5.2 (a) peuvent varier très rapidement (ie leur longueur échelle caractéristique est trop courte). Le ralentissement de la variation est accomplie en ajustant simplement les paramètres de la fonction de covariance. Le problème de l'apprentissage dans les processus Gaussien est exactement le problème de trouver des propriétés convenables pour la fonction de covariance.

5.3 Prédiction avec Régression Processus Gaussien

Nous disposons d'un ensemble de données de formation $\mathcal{D}$ de n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$, où $\mathbf{x}$ est un vecteur d'entrée (covariables) de dimension D et y désigne une sortie scalaire ou cible (variable dépendante) ; les colonnes de vecteur entrée pour tous les cas n sont agrégées dans $D \times n$ *la matrice de conception* X, et les cibles sont rassemblées dans le vecteur $\mathbf{y}$, nous pouvons écrire $\mathcal{D} = (X, \mathbf{y})$. Dans le cadre de régression, les cibles sont des valeurs réelles. Nous sommes intéressés à faire des inférences sur la relation entre les entrées et les cibles, par exemple, la distribution conditionnelle des cibles compte tenu des entrées (mais nous ne nous sommes pas intéressés à la modélisation de la distribution des entrées). Nous utilisons un processus Gaussien (GP) pour décrire une distribution sur des fonctions. Formellement :

Définition 5.3.1 Un Processus Gaussien *est une collection de variables aléatoires, tout sous ensemble fini de variables de cette collection a une joint distribution Gaussien.*

Un processus Gaussien est entièrement défini par sa fonction moyenne et sa fonction de covariance. Nous définissons la fonction de moyenne $m(\mathbf{x})$ et la fonction de covariance

$k(\mathbf{x}, \mathbf{x}')$ d'un processus réel $f(\mathbf{x})$ comme

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{5.1}$$

et le processus Gaussien s'écrit alors :

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{5.2}$$

Pour simplifier la notation nous considérons la fonction moyenne égale à zéro.

Dans notre cas, les variables aléatoires représentent la valeur de la fonction $f(\mathbf{x})$ à la position $\mathbf{x}$. Souvent, les processus Gaussiens sont définis sur le temps, par exemple, lorsque l'ensemble des indices des variables aléatoires est le temps. Ce n'est pas le cas dans notre utilisation du GPs ; l'ensemble des indices $\mathcal{X}$ est l'ensemble des entrées possibles, par exemple $\mathbb{R}^D$. Pour plus de commodité, nous utilisons la notation (arbitraire) d'énumération des cas dans l'ensemble de données de formation pour identifier les variables aléatoires tels que $f_i \triangleq f(\mathbf{x}_i)$ est la variable aléatoire qui correspond au case $(\mathbf{x}_i, y_i)$ comme on s'y attendrait.

Un processus Gaussien est défini par un ensemble de variables aléatoires. Ainsi, la définition implique automatiquement une *consistance*, aussi parfois appelée propriété de la marginalisation. Cette propriété signifie que si un GP spécifie $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, alors il doit également spécifier $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ où Σ_{11} est la sous-matrice pertinente de Σ, voir l'éq. (A.6). En d'autres termes, l'examen d'un plus grand ensemble de variables ne change pas la distribution de l'ensemble plus petit.

Un exemple simple d'un processus Gaussien peut être otenu à partir de notre modèle de régression linéaire Bayésien $f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$ avec la distribution a priori $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. Nous avons la moyenne et la covariance :

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}] = 0, \\ \mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] &= \phi(\mathbf{x}^\top) \mathbb{E}[\mathbf{w}\mathbf{w}^\top] \phi(\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}'). \end{aligned} \tag{5.3}$$

Ainsi $f(\mathbf{x})$ et $f(\mathbf{x}')$ sont joints Gaussien avec une moyenne de zéro et une covariance donnée par $\phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$. En effet, les valeurs des fonctions $f(\mathbf{x}_1), ..., f(\mathbf{x}_n)$ correspondant à un nombre de points d'entrée quelconque n sont joint Gaussien.

La fonction de covariance indique la covariance entre les paires de variables aléatoires (voir [84], chapitre 4)

$$\operatorname{cov}\big(f(\mathbf{x}_p), f(\mathbf{x}_q)\big) = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\big(-\frac{1}{2\ell^2}|\mathbf{x}_p - \mathbf{x}_q|^2\big). \tag{5.4}$$

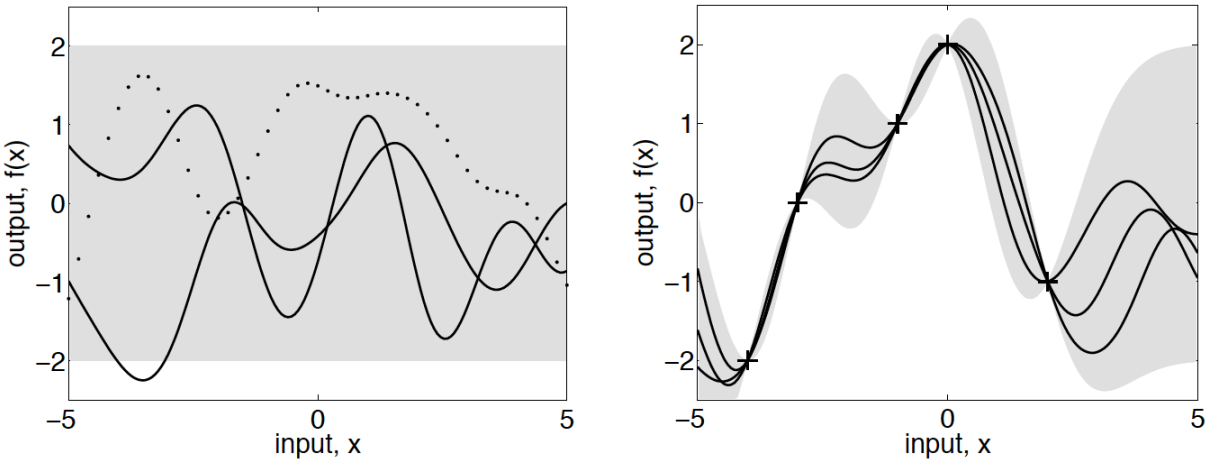
où ℓ est *caractéristiques de l'échelle des longueurs* (ou *characteristic length-scale* en anglais). Notez que la covariance entre les *sorties* est écrit comme une fonction des *entrées*. Pour cette fonction de covariance particuliere, on voit que la covariance est presque l'unité entre les variables dont les entrées correspondantes sont très proches ; et diminue quand leur distance dans l'espace d'entrée augmente.

La spécification de la fonction de covariance implique une distribution sur des fonctions. Pour l'observer, nous pouvons tracer des échantillons de la distribution des fonctions

évaluées à tout nombre de points ; dans le détail ; on choisit un certain nombre de points d'entrée, X_* et on écrit la matrice de covariance correspondante en utilisant l'éq. (5.4). Ensuite, on génère un vecteur aléatoire Gaussien avec cette matrice de covariance

$$\mathbf{f}_* \sim \mathcal{N}\big(\mathbf{0}, K(X_*, X_*)\big), \tag{5.5}$$

et on trace les valeurs générées en fonction des entrées. La figure 5.3 (a) montre trois de ces échantillons. La génération d'échantillons Gaussien multivarié est décrit dans la section A.2.



(a), prior (b), posterior

FIGURE 5.3: (a) Trois fonctions tirés au hasard à partir d'un GP a priori ; les points indiquent les valeurs de y effectivement générés ; les deux autres fonctions ont été dessinés avec des lignes par rejoindre un grand nombre de points d'évaluation. (b) Trois fonctions aléatoires dessinées à partir du postérieur, i.e. l'a priori conditionné sur les cinq observations sans bruit indiqué.

5.3.1 Prédiction avec des observations sans bruit

Il n'y a habituellement pas d'intérêt à dessiner des fonctions aléatoires de la priori, mais nous voulons intégrer les connaissances que les données de formation fournit à la fonction. Au départ, nous considérons le cas particulier simple où les observations sont sans bruit, c'est que nous savons $\{(\mathbf{x}_i, f_i)|i = 1, ..., n\}$. La distribution jointe des sorties de formation $\mathbf{f}$, et les sorties de test $\mathbf{f}_*$ selon l'a priori est

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{5.6}$$

S'il y a n points de formation et n_* points de test, alors $K(X, X_*)$ désigne la matrice de covariance $n \times n_*$ évaluée pour toutes les paires (points de formation - point de test), et de même pour les autres $K(X,X), K(X_*,X_*)$ et $K(X_*,X)$. Pour obtenir la distribution postérieure sur les fonctions, nous avons besoin de limiter cette distribution jointe

a priori pour écarter les fonctions incompatibles avec les points de données observées. Graphiquement dans la figure 5.3, on peut générer des fonctions de l'a priori, et rejeter celles désaccord avec les observations. Bien que cette stratégie de calcul ne serait pas très efficace, en termes probabilistes cette opération est extrêmement simple : Elle correspond à un *conditionnement* de la distribution jointe Gaussien a priori sur les observations (voir la section A.2 pour plus de détails) pour donner les valeurs de la fonction $\mathbf{f}_*$ (ce qui correspond aux entrées de test X_*) :

$$\begin{aligned}\mathbf{f}_*|X_*, X, \mathbf{f} \ \sim \mathcal{N}\big(&K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ &K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big).\end{aligned} \tag{5.7}$$

Valeurs de la fonction $\mathbf{f}_*$ peuvent être échantillonnées à partir de la joint distribution postérieur par une évaluation de la moyenne et la matrice de covariance de l'équation (5.7), on peut ensuite générer des échantillons suivant la méthode décrite dans la section A.2.

La figure 5.3 (b) montre les résultats de ces calculs avec cinq points de données (symboles +).

5.3.2 Prédiction en utilisant des observations bruitées

Il est courant pour des situations plus réalistes de modélisation de ne pas avoir accès aux valeurs de fonction elles-mêmes, mais uniquement aux versions bruitées de celles-ci $y = f(\mathbf{x}) + \varepsilon$. En supposant que le bruit additif indépendant identiquement distribuée selon le Gaussien ε avec la variance σ_n^2, l'a priori sur les observations bruitées devient

$$\operatorname{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{y}_q) + \sigma_n^2\delta_{pq} \quad \text{ou} \quad \operatorname{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I, \tag{5.8}$$

où δ_{pq} est un delta de Kronecker qui est égal à 1 si $p = q$ et 0 autrement. Il résulte de l'hypothèse d'indépendance du bruit, qu'une matrice diagonale est ajoutée, au cas sans bruit, (l'éq. (5.4)). En introduisant le terme de bruit dans l'éq. (5.6), on peut écrire la distribution jointe des valeurs cibles observées et les valeurs de la fonctions aux positions de test sous l'a priori de la manière suivante :

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{5.9}$$

À partir de la distribution conditionnelle correspondant à l'éq. (5.7), nous arrivons à la clé des équations de prédiction pour la régression processus Gaussien

$$\mathbf{f}_*|X, \mathbf{y}, X_* \sim \mathcal{N}\big(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)\big), \quad \text{où} \tag{5.10}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{5.11}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \tag{5.12}$$

L'écriture des expressions $K(X, X), K(X, X_*)$ et $K(X_*, X_*)$ etc, peuvent sembler lourdes, donc nous allons introduire une forme compacte de la notation $K = K(X, X)$ et $K_* =$

input : X (entrées), $\mathbf{y}$ (cibles), k (fonction de covariance), σ_n^2 (niveau de bruit), $\mathbf{x}_*$ (test entrée)

$L := \text{cholesky}(K + \sigma_n^2 I)$
$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$
$\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$ } prédiction de moyenne l'éq. (5.13)
$\mathbf{v} := L \backslash \mathbf{k}_*$
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } prédiction de variance l'éq. (5.14)
return : $\bar{f}_*$(mean), $\mathbb{V}[f_*]$(variance)

TABLE 5.1: Prédiction de la Régression processus Gaussien.

$K(X, X_*)$. Dans le cas où il n'existe qu'un seul point de test $\mathbf{x}_*$, on écrit $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$ pour désigner le vecteur des covariances entre le point de test et les point de formation n. En utilisant cette notation compacte pour un point de test $\mathbf{x}_*$, les équations 5.11 et 5.12 deviennent :

$$\bar{f}_* = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y}, \tag{5.13}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*. \tag{5.14}$$

Une mise en œuvre pratique de la régression processus Gaussien (GPR) est montrée dans la table 5.1. L'algorithme utilise la décomposition de Cholesky, au lieu d'inverser la matrice directement, car il est plus rapide et plus stable, voir section A.4. L'algorithme retourne la prédiction de moyenne et de variance pour des données de test sans bruit. Pour calculer la distribution prédictive des données de test sans bruit y_*, il suffit d'ajouter la variance du bruit σ_n^2 à la variance prédictive de f_*.

5.4 Implémentation

5.4.1 Créer la base de données

Chaque entrée $\mathbf{x}_i$ est une image d'œil de taille 32×16. La sortie y_i correspondante à l'entrée $\mathbf{x}_i$ est un point sur l'écran incluant 2 coordonnées t_x et t_y (les deux axes de coordonnées de l'écran d'ordinateur). Nous avons fait la prédiction de deux valeurs t_x et t_y indépendantes. Pour avoir un ensemble de données de formations $\mathcal{D}$ de n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, ..., n\}$, les étapes sont les suivantes :

- **Étape 1** : D'abord nous mettons les données de point fixes sur l'écran d'ordinateur qui peuvent couvrir tous les endroits sur l'écran. Nous avons utilisé le 13 points ($n = 13$) suivants :

t_x	t_y
0.5	0.5
0.1	0.5
0.9	0.5
0.5	0.1
0.5	0.9
0.1	0.1
0.1	0.9
0.9	0.9
0.9	0.1
0.3	0.3
0.3	0.7
0.7	0.7
0.7	0.3

Selon la valeur de la hauteur et la largeur de l'écran, la position de chaque point sur l'écran est $(t_x * largeur, t_y * hauteur)$. Toutes les données de formation des sorties y sont décrits dans la figure 5.4.

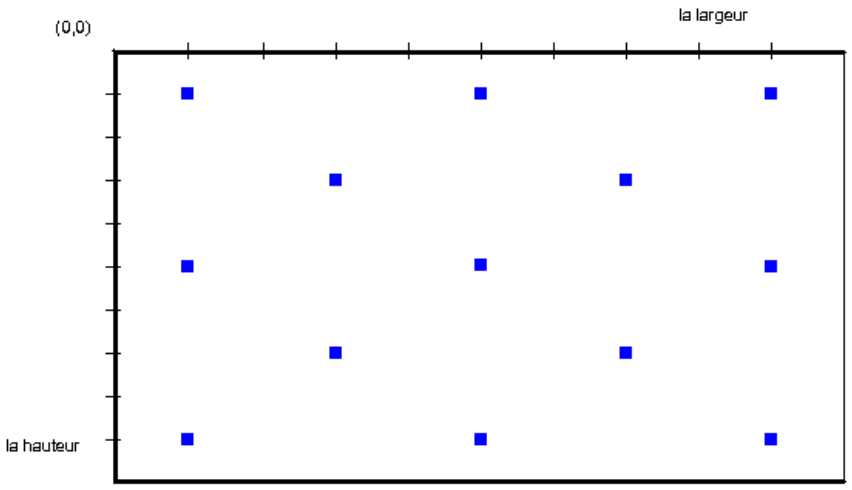


FIGURE 5.4: Données de formations des sorties sur l'écran d'ordinateur.

– **Étape 2** : Pour introduire les données de formation pour les entrées correspondantes aux les sorties précédentes, l'utilisateur s'assoit devant l'ordinateur, et les points de formations apparaissent consécutivement sur l'écran. La caméra capture l'image de l'œil chaque fois quel'utilisateur regarde un point sur l'écran. Nous obtenons ainsi une paire de données de formation $(\mathbf{x_i}, y_i)$ (une image d'œil, un point sur l'écran). La procédure se répète pour tous les points de formation (13 points). Nous appelons cette procédure, *procédure de calibration*. L'utilisateur doit tenir sa tête stable pendant la procédure de calibration.

Après la procédure de calibration, nous pouvons faire la prédiction de la sortie y_* de la pour la nouvelle entrée $\mathbf{x}_*$ à l'aide de l'algorithme présenté dans la table 5.1. La sortie est la prédiction de moyenne $y_* = \bar{f}_*$ (l'éq. 5.13).

5.4.2 La matrice de covariance

Pour utiliser l'algorithme dans la table 5.1, nous devons calculer la matrice de covariance K basée sur la fonction de covariance k dans l'équation (5.4) :

$$\mathrm{cov}\big(f(\mathbf{x}_p), f(\mathbf{x}_q)\big) = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\big(-\frac{1}{2\ell^2}|\mathbf{x}_p - \mathbf{x}_q|^2\big).$$

La distance $|\mathbf{x}_p - \mathbf{x}_q|$ entre des images $\mathbf{x}_p, \mathbf{x}_q$ est calculée par :

$$|\mathbf{x}_p - \mathbf{x}_q| = \sqrt{\sum_I (\mathbf{x}_p(I) - \mathbf{x}_q(I))^2}$$

où $\mathbf{x}(I)$ est la valeur d'un pixel de l'image $\mathbf{x}$. La distance entre les images de l'œil environ 200 à 1000. Avec $\ell = 500$, la covariance entre deux images $\mathbf{x}_p, \mathbf{x}_q$ est :

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp\big(-\frac{1}{2 * 500^2}|\mathbf{x}_p - \mathbf{x}_q|^2\big).$$

X est le vecteur d'images en entrée, pour calculer la matrice de covariance K nous utilisons le pseudo-code suivant :

for $p = 0$ **to** n **with step of** 1 *n est le nombre d'entrées*
  for $q = 0$ **to** n **with step of** 1
    $K(p, q) = k(\mathbf{x}_p, \mathbf{x}_q)$
return K

$\mathbf{k}_*$ est la matrice de covariance entre les données (entrées) de formation et une donnée (entrée) de test $\mathbf{x}_*$, nous pouvons calculer comme suivant

for $p = 0$ **to** n **with step of** 1 *n est le nombre d'entrées*
  $\mathbf{k}_*(p) = k(\mathbf{x}_p, \mathbf{x}_*)$
return $\mathbf{k}_*$

5.4.3 Cholesky

Nous avons choisi la valeur $\sigma_n^2 = 0$ dans la condition où il n'y a pas de bruit. Ainsi, on a $(K + \sigma_n^2 I = 0)$. Pour calculer la *matrice triangulaire inférieure* $L = cholesky(K)$, nous avons utilisé l'objet **vnl_cholesky** dans le package **vxl** :

vnl_cholesky Cholesky(K);

la matrice triangulaire inférieure est calculée par la fonction :

$L =$ **Cholesky.lower_triangle()**;

et la valeur $\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$ dans la table 5.1 est calculée par la fonction :

$alpha =$ **Cholesky.solve**$(\mathbf{y})$;

Ainsi, la valeur de la prédiction de moyenne est :

$$\bar{f}_* = \mathbf{k}_*.\text{transpose}() * \text{alpha};$$

$\bar{f}_*$ est la valeur de prédiction du regard pour la nouvelle entrée $\mathbf{x}_*$.

5.5 Résultat expérimental

Avant de faire une session de suivi du regard, l'utilisateur doit effectuer une procédure de calibration pour créer une base de données de formation. L'utilisateur s'assoit en face de l'écran, et regarde seize points qui apparaissent successivement. Chaque fois qu'apparaît un point, la caméra capture une image d'œil et fait une paire de données de formation (une paire de données de formation inclut une image de l'œil de l'utilisateur et la position du point sur l'écran correspondant à l'endroit où l'utilisateur regarde).

5.5.1 Prédiction du regard avec la tête stable

Nous faisons un test de suivi du regard avec la tête de l'utilisateur stable. L'utilisateur tient sa tête stable lors d'effectuer la procédure de calibration, puis réalise le suivi du regard avec cette position de la tête (la tête est encore stable après la calibration). Nous réalisons le test de suivi du regard avec 16 autres points. Le résultat est montré dans la figure 5.5 : les points triangulaires sont les points cibles du test, les autres points sont les points de la prédiction du regard.

Les résultats présentés à la figure 5.5 sont excellents. Cependant, ils exigent que l'utilisateur tienne la tête stable. Si l'utilisateur bouge la tête hors de la position de calibration, la précision de ce système de suivi du regard baisse de façon spectaculaire. La raison est que, lorsque l'utilisateur bouge la tête, l'image de l'œil de l'utilisateur change et l'image de l'œil en entrée est différente de l'image de l'œil dans la base de données de formation. Le résultat de la prédiction du regard sera donc incorrect. Nous proposons une solution à ce problème dans la prochaine section.

5.5.2 Prédiction du regard en bougeant la tête

Pour résoudre le problème de suivi du regard en bougeant la tête, nous proposons de répéter la procédure de calibration avec les différents angles de vue de la tête de l'utilisateur. Avec cette méthode, nous ajoutons plus des données à la base de données de formation. Cela signifie que l'on y ajoute différentes images de l'œil dans différentes positions de la tête (de l'utilisateur) lorsque l'utilisateur regarde le même point sur l'écran. Après la procédure de calibration, l'utilisateur peut librement déplacer sa tête pendant la session de suivi du regard.

Pour cela, un test avec la même procédure de calibration que dans la section précédente est effectuée quatre fois sous différents angles de vue : avec la tête de l'utilisateur tournée à gauche, à droite, puis avec la tête de l'utilisateur pivotée vers le haut, et vers en bas. Le degré maximum de la rotation de chaque direction est la position où tous les coins ne sont

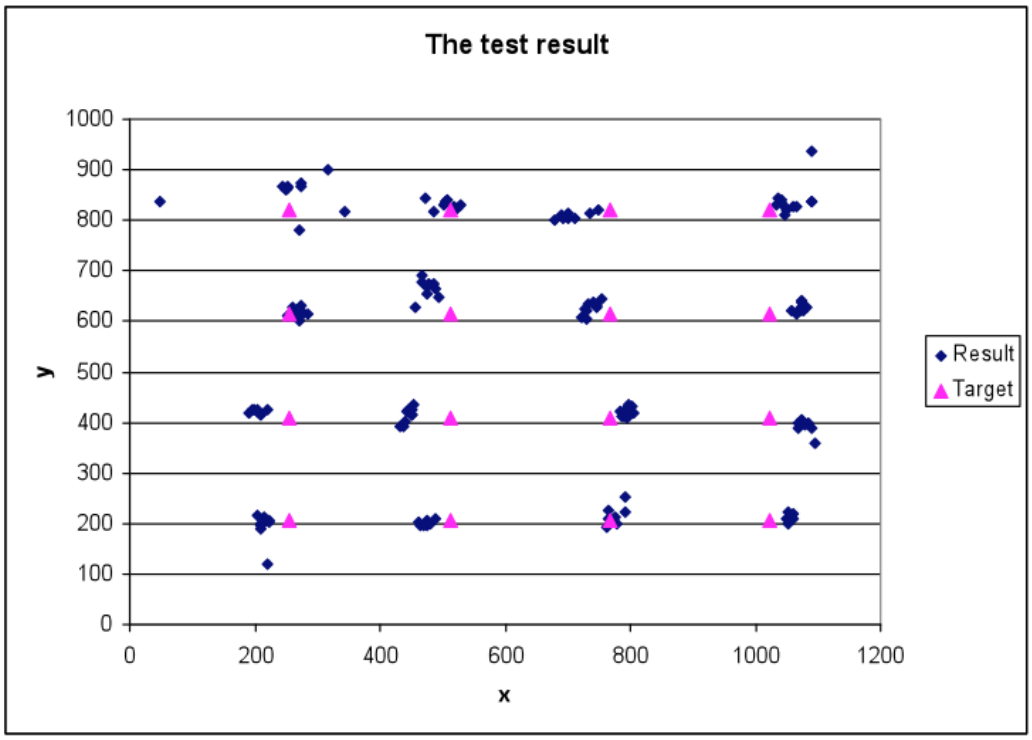


FIGURE 5.5: Résultat de test avec la tête stable.

pas perdus de suivi. La figure 5.6 montre les quatre positions de la tête de l'utilisateur pour faire la calibration.

Après la calibration, le même test de prédiction du regard avec 16 points est effectuée (comme indiqué dans la section précédente) mais avec la condition de déplacer librement la tête. L'utilisateur fait tourner sa tête de gauche à droite ou du bas vers le haut pour regarder chaque point. Le résultat est montré dans la figure 5.7.

Nous pouvons observer que le résultat de la figure 6 n'est pas très bon, les erreurs de prédiction regard sont important. La raison est qu'il y a encore des positions de la tête de l'utilisateur qui rendent l'image de l'œil en entrée différente de l'œil dans la base de données de formation. Ainsi, nous avons choisi d'effectuer cinq calibrations de plus pour enrichir à la base de données de formation. Le résultat après cette nouvelle calibration est présenté la figure 5.8.

Après avoir effectué 10 procédures de calibration, le résultat du regard bougeant la tête est très bon. Nous pouvons améliorer ce résultat en effectuant plus de calibration.

5.5.3 Discussion

Comme le montre la figure 5.8 ce resultat est très bon avec 10 calibrations dans différentes positions de la tête. Cependant, le résultat ne sera pas bon si la tête de l'utilisateur se déplace dans une nouvelle position, par exemple, si l'utilisateur ne tourne pas sa tête, mais déplace sa tête loin ou à proximité de la caméra. Ainsi, au cours de la session de suivi du regard en tête libre, s'il y a une position de la tête qui rend mauvaise la

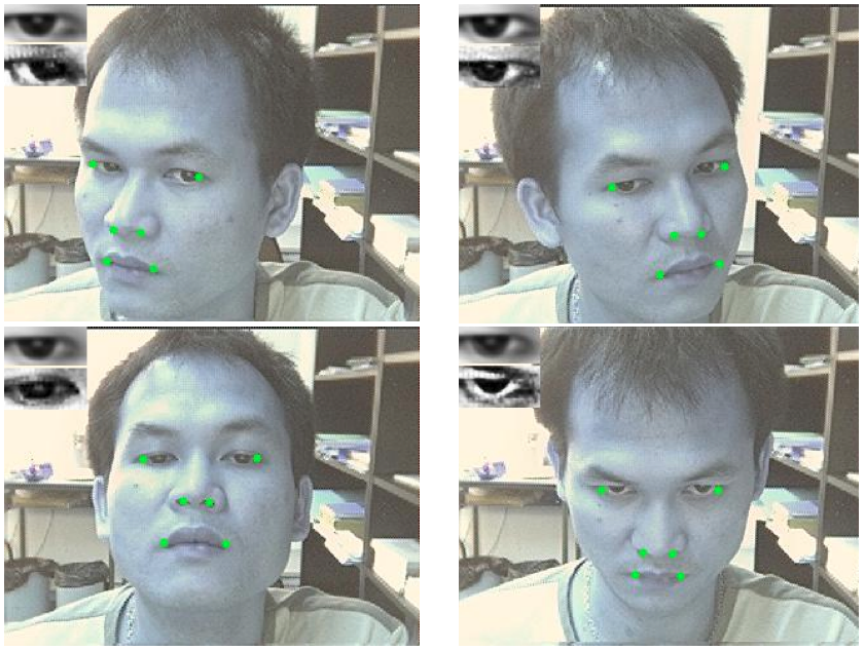


FIGURE 5.6: Positions de la tête de l'utilisateur sous différents angles de vue.

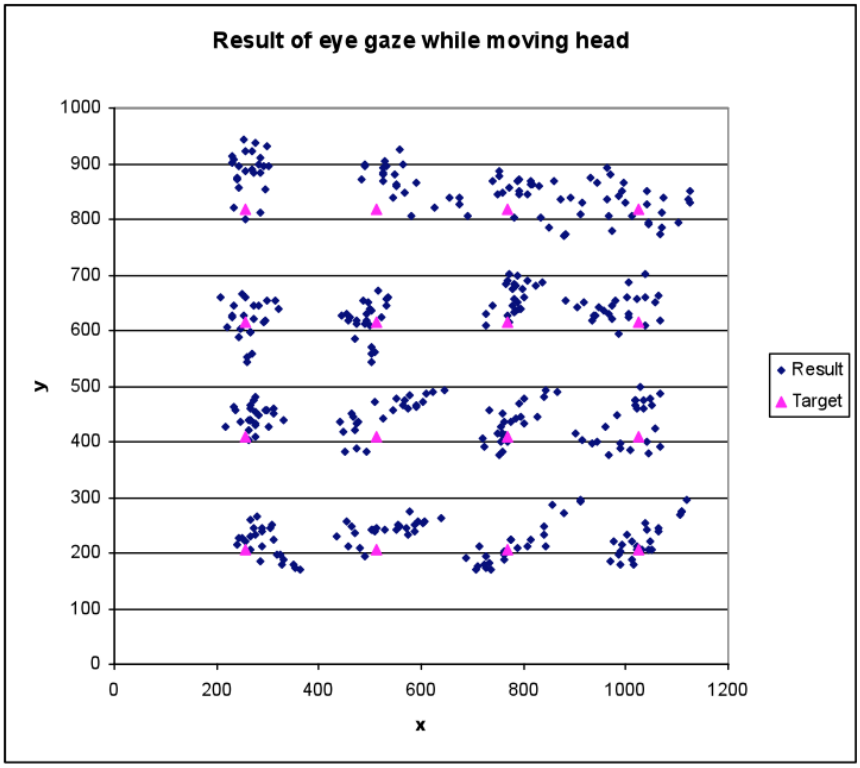


FIGURE 5.7: Le résultat du test de prédiction du regard avec la tête libre après cinq calitrations.

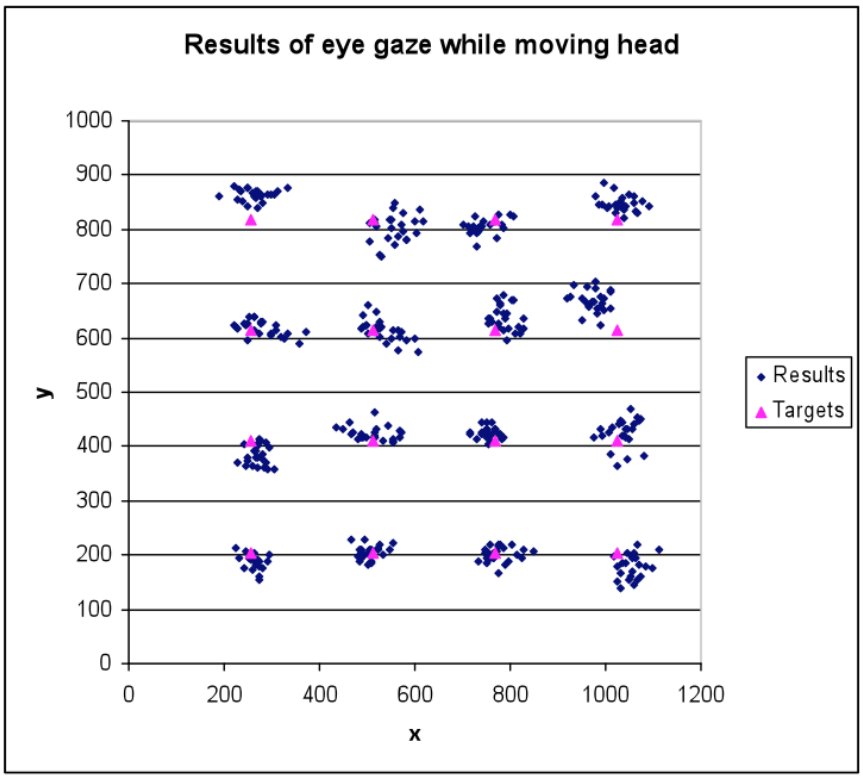


FIGURE 5.8: Le résultat du test de prédiction du regard avec la tête libre après dix calibrations.

prédiction du regard, l'utilisateur doit faire de nouvelles calibrations dans cette position pour améliorer le résultat.

On peut de penser cet inconvénient, que ce système de suivi du regard est difficile et compliqué à utiliser. En pratique, lors de la première utilisation de ce système, l'utilisateur prend environ dix minutes pour faire une calibration de toutes les positions possibles de la tête qu'il peut réutiliser ensuite rendant le dispositif facile à utiliser dans la pratique.

5.6 Conclusion

Nous avons présenté l'utilisation du processus Gaussien pour faire la prédiction du regard en temps réel. Pour résoudre le problème de mouvement de la tête, nous avons proposé une solution de répétition la calibration dans les différence position de la tête. Après la calibration, l'utilisateur peut bouger librement la tête pendant la session de suivi du regard. L'utilisateur ne consacre que dix minutes environ lors de la première utilisation pour réaliser la procédure de calibration dans les différentes positions de la tête. C'est sa propre base de données pour faire la prédiction de son regard. À partir de la deuxième utilisation, l'utilisateur peut réutiliser sa base de données et n'a pas besoin d'effectuer plus de la procédure de calibration.

Conclusion

Nous avons présenté notre modèle de suivi du regard avec une caméra simple. Ce système permet d'utilisateur de bouger librement sa tête lors de la session de suivi. Nous avons également présenté des solutions pour construire ce système :

- Pour le problème de détection les yeux : nous utilisons la méthode de détection des objets basée sur descripteurs de Haar [106], [52] pour détecter les yeux sur le visage de l'utilisateur.
- Pour le problème de suivi les yeux : nous recherchons des coins sur le visage de l'utilisateur. Nous trouvons les coins sur des yeux, le nez et la bouche de l'utilisateur et utilisons l'algorithme de Lucas Kanade [7] pour suivre les coins, puis utilisons la méthode de détection Outliers [69] pour détecter les coins perdus de suivre et les corriger.
- Pour le problème de détection du point de regard sur l'écran : nous utilisons le Processus Gaussien pour la fonction de prédiction du regard de l'utilisateur.
- Pour résoudre le problème des mouvements de la tête, l'utilisateur doit effectuer une procédure de calibration pour obtenir la fonction de suivi du regard dans les différentes positions de la tête. Après la calibration, l'utilisateur pourra déplacer librement la tête devant la caméra, améliorant ainsi son confort.

Des avantages de ce système de suivi du regard sont :

- C'est un système non-intrusif, il n'a pas besoin d'aucun équipement attaché à l'utilisateur.
- Ce système utilise seulement une caméra simple, l'utilisateur n'a pas besoin d'installer des équipements spécifiques.
- Ce système ne gêne pas l'utilisateur dans ces mouvements. L'utilisateur peut bouger librement sa tête lors de la session de suivi.
- Ce système utilise une caméra simple, son coût est donc modéré.

Des limitations de ce système sont :

- L'utilisateur doit rester dans le champ de la caméra.
- Le système fonctionne bien dans la condition normale d'utilisation de l'ordinateur. Par exemple l'utilisateur bouge la tête pour voir les informations dans les différents endroits de l'écran (la plus haute à gauche ou à droite, la plus basse à gauche ou à droite). Le système ne marche pas bien avec les mouvements exceptionnels par exemple loin de la caméra ou en dehors du champs de la caméra,...
- Si la lumière change dans la session de suivi du regard, l'utilisateur doit refaire la calibration pour ajuster les données de formation. Dans les recherches futures, nous devrons résoudre ce problème par la normalisation de toute les images d'entrées à

la même condition de la luminosité (ou supprimer la lumière des images entrées).

Annexe A

Mathematical Background

A.1 Probabilité conditionnelle, marginale et jointe

Soient n variables aléatoires $y_1, ..., y_n$ (discrètes ou continues), avec une probabilité jointe $p(y_1, ..., y_n)$,si on note $\mathbf{y} = y_1, ..., y_n$ alors $p(y_1, ..., y_n) = p(\mathbf{y})$. Techniquement, on devrait faire la distinction entre les probabilités (pour des variables discrètes) et les densités de probabilité pour les variables continues. Tout au long de cette thèse, nous utilisons couramment le terme "probabilité" pour se référer aux deux. On peut cloisonner les variables dans $\mathbf{y}$ en deux groupes, $\mathbf{y}_A$ et $\mathbf{y}_B$, où A et B sont deux ensembles disjoints dont l'union est l'ensemble $\{1, ..., n\}$, de sorte que $p(\mathbf{y}) = p(\mathbf{y}_A, \mathbf{y}_B)$. Chaque groupe peut contenir une ou plusieurs variables.

La probabilité marginale de $\mathbf{y}_A$ est donnée par

$$p(\mathbf{y}_A) = \int p(\mathbf{y}_A, \mathbf{y}_B) d\mathbf{y}_B. \tag{A.1}$$

L'intégrale est remplacée par une somme si les variables sont des valeurs discrètes. Notez que si l'ensemble A contient plus d'une variable, alors la probabilité marginale est elle-même une probabilité jointe. Si la distribution jointe est égal au produit des marginaux, alors les variables sont dites *indépendantes*, sinon elles sont *dépendantes*.

La fonction de probabilité *conditionnelle* est définie comme

$$p(\mathbf{y}_A|\mathbf{y}_B) = \frac{p(\mathbf{y}_A, \mathbf{y}_B)}{p(\mathbf{y}_B)}, \tag{A.2}$$

définis pour $p(\mathbf{y}_B) > 0$, comme il n'est pas utile à la condition d'un événement impossible. Si $\mathbf{y}_A$ et $\mathbf{y}_B$ sont indépendants, alors la probabilité marginale $p(\mathbf{y}_A)$ et la probabilité conditionnelle $p(\mathbf{y}_A|\mathbf{y}_B)$ sont égaux.

En utilisant les définitions de $p(\mathbf{y}_A|\mathbf{y}_B)$ et $p(\mathbf{y}_B|\mathbf{y}_A)$ on obtient le théorème de Bayes

$$p(\mathbf{y}_A|\mathbf{y}_B) = \frac{p(\mathbf{y}_A)p(\mathbf{y}_B|\mathbf{y}_A)}{p(\mathbf{y}_B)}. \tag{A.3}$$

A.2 Gaussienne Identités

La distribution multivariée de Gauss (ou normale) a une densité de probabilité jointe donnée par

$$p(\mathbf{x}|\mathbf{m},\Sigma) = (2\pi)^{-D/2}|\Sigma|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^\top\Sigma^{-1}(\mathbf{x}-\mathbf{m})\big), \tag{A.4}$$

où $\mathbf{m}$ est le vecteur moyen (de longueur D) et Σ est la matrice de *covariance* (symétrique, positive défini) (de taille $D \times D$). Nous écrirons $\mathbf{x} \sim \mathcal{N}(\mathbf{m},\Sigma)$.

Soient $\mathbf{x}$ et $\mathbf{y}$ être conjointement vecteurs aléatoires Gaussiens

$$\left[\begin{array}{c}\mathbf{x}\\ \mathbf{y}\end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{array}\right], \left[\begin{array}{cc}A & C\\ C^\top & B\end{array}\right]\right) = \mathcal{N}\left(\left[\begin{array}{c}\boldsymbol{\mu}_x\\ \boldsymbol{\mu}_y\end{array}\right], \left[\begin{array}{cc}\tilde{A} & \tilde{C}\\ \tilde{C}^\top & \tilde{B}\end{array}\right]^{-1}\right), \tag{A.5}$$

puis la distribution *marginale* de $\mathbf{x}$ et la distribution *conditionnelle* de $\mathbf{x}$ sachant $\mathbf{y}$ sont

$$\begin{aligned}\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A), \quad \text{et} \quad & \mathbf{x}|\mathbf{y} \sim \mathcal{N}\big(\boldsymbol{\mu}_x + CB^{-1}(\mathbf{y}-\boldsymbol{\mu}_y), A - CB^{-1}C^\top\big)\\ \text{ou} \quad & \mathbf{x}|\mathbf{y} \sim \mathcal{N}\big(\boldsymbol{\mu}_x - \tilde{A}^{-1}\tilde{C}(\mathbf{y}-\boldsymbol{\mu}_y), \tilde{A}^{-1}\big).\end{aligned} \tag{A.6}$$

Voir ([107], chapitre VIII, sec. 9.3), et les éqs (A.11 - A.13).

Le produit de deux Gaussiennes donne une autre Gaussienne (non-normalisé)

$$\mathcal{N}(\mathbf{x}|\mathbf{a},A)\mathcal{N}(\mathbf{x}|\mathbf{b},B) = Z^{-1}\mathcal{N}(\mathbf{x}|\mathbf{c},C) \tag{A.7}$$

$$\text{où } \mathbf{c} = C(A^{-1}\mathbf{a} + B^{-1}\mathbf{b}) \text{ et } C = (A^{-1}+B^{-1})^{-1}.$$

Notez que le résultat Gaussien a une précision (variance inverse) égale à la somme des précisions, et une moyenne égale à la somme convexe des moyennes, pondérées par les précisions. La constante de normalisation elle-même ressemble à une Gaussienne (dans $\mathbf{a}$ ou $\mathbf{b}$)

$$Z^{-1} = (2\pi)^{-D/2}|A+B|^{-1/2}\exp\big(-\frac{1}{2}(\mathbf{a}-\mathbf{b})^\top(A+B)^{-1}(\mathbf{a}-\mathbf{b})\big). \tag{A.8}$$

Pour prouver l'éq. (A.7), il suffit d'écrire les expressions en introduisant éq. (A.4) et éq. (A.8) dans l'éq. (A.7), et d'élargir les termes à l'intérieur de l'exponentielle exp pour vérifier l'égalité. Il peut être utile d'élargir C en utilisant le matrice d'inversion lemma, éq. (A.9), $C = (A^{-1}+B^{-1}) = A - A(A+B)^{-1}A = B - B(A+B)^{-1}B$.

Pour générer des échantillons $\mathbf{x} \sim \mathcal{N}(\mathbf{m},K)$ avec la moyenne arbitraire $\mathbf{m}$ et la matrice de covariance K en utilisant un générateur Gaussien scalaire (disponible dans de nombreux environnements de programmation) on procède comme suit : d'abord, on calcule la décomposition de Cholesky ("matrix square root") L de la matrice de covariance définie positive symétrique $K = LL^\top$, où L est une matrice triangulaire inférieure, voir section A.4. On génére ensuite $\mathbf{u} \sim \mathcal{N}(\mathbf{0},I)$ par de multiples appels distincts au générateur Gaussien scalaire. On Calcule enfin $\mathbf{x} = \mathbf{m} + L\mathbf{u}$, qui a la distribution souhaitée avec la moyenne $\mathbf{m}$ et la covariance $L\mathbb{E}[\mathbf{u}\mathbf{u}^\top]L^\top = LL^\top = K$ (par l'indépendance des éléments de $\mathbf{u}$).

A.3 Matrice Identités

La *matrice d'inversion lemma*, également connu sous le nom la formula Woodbury, Sherman & Morrison (voir [82] p. 75) affirme que

$$(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top Z^{-1}, \tag{A.9}$$

en supposant que les inverses pertinentes existent toutes. Ici Z est $n \times n$, W est $m \times m$ et U et V sont tous deux de taille $n \times m$; par conséquent, si Z^{-1} est connue, et d'un rang faible ($m < n$) la perturbation est faite à Z comme dans la partie gauche de l'équation (A.9), accélération considérables des temps de calcul peuvent être réalisée. Une équation similaire existe pour les déterminants

$$|Z + UWV^\top| = |Z||W||W^{-1} + V^\top Z^{-1}U|. \tag{A.10}$$

Soient les inversibles $n \times n$ matrice A, et son inverse A^{-1} peuvent être partitionnées en

$$A = \begin{pmatrix} P & Q \\ R & S \end{pmatrix}, \qquad A^{-1} = \begin{pmatrix} \tilde{P} & \tilde{Q} \\ \tilde{R} & \tilde{S} \end{pmatrix}, \tag{A.11}$$

où P et $\tilde{P}$ sont des matrices $n_1 \times n_1$ et S et $\tilde{S}$ sont des matrices $n_2 \times n_2$ avec $n = n_1 + n_2$. Les sous matrices de A^{-1} sont données dans ([82] p. 77) comme

$$\left.\begin{array}{l} \tilde{P} = P^{-1} + P^{-1}QMRP^{-1} \\ \tilde{Q} = -P^{-1}QM \\ \tilde{R} = -MRP^{-1} \\ \tilde{S} = M \end{array}\right\} \text{ où } \quad M = (S - RP^{-1}Q)^{-1}, \tag{A.12}$$

ou de manière équivalente

$$\left.\begin{array}{l} \tilde{P} = N \\ \tilde{Q} = -NQS^{-1} \\ \tilde{R} = -S^{-1}RN \\ \tilde{S} = S^{-1} + S^{-1}RNQS^{-1} \end{array}\right\} \text{ où } \quad N = (P - QS^{-1}R)^{-1}. \tag{A.13}$$

A.4 La décomposition de Cholesky

La décomposition de Cholesky d'une matrice symétrique définie positive A décompose A en un produit d'une matrice triangulaire inférieure L et sa transposée

$$LL^\top = A, \tag{A.14}$$

où L est appelé facteur de Cholesky. La décomposition de Cholesky est utile pour résoudre des systèmes linéaires avec matrice coefficient symétrique définie positive A. Pour résoudre $A\mathbf{x} = \mathbf{b}$ pour x, d'abord résoudre le système triangulaire $L\mathbf{y} = \mathbf{b}$ par substitution directe, puis le système triangulaire $L^\top\mathbf{x} = \mathbf{y}$ par retour de substitution. En utilisant l'opérateur backslash, nous écrivons cette solution comme $\mathbf{x} = L^\top\backslash(L\backslash\mathbf{b})$, où la notation $A\backslash\mathbf{b}$ est le

vecteur x qui résoud $A\mathbf{x} = \mathbf{b}$. Deux opérations substitutions nécessitent $n^2/2$ opérations, lorsque A est de taille $n \times n$. Le calcul du facteur de Cholesky L est jugé numérique extrêmement stable et prend du temps $n^3/6$, il est donc la méthode de choix quand il peut être appliqué. Notez aussi que le déterminant d'une matrice définie positive symétrique peut être déterminant calculés de façon efficace par

$$|A| = \prod_{i=1}^{n} L_{ii}^2, \quad \text{ou} \quad \log|A| = 2\sum_{i=1}^{n} \log L_{ii}, \tag{A.15}$$

où L est le facteur de Cholesky de A.

Bibliographie

[1] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25(6) :821–837, 1964.

[2] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. *Lecture notes in computer science*, pages 15–26, 2002.

[3] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. *Advances in Neural Information Processing Systems*, pages 753–753, 1994.

[4] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets. *Proceedings of the 6 th ACM SIGKDD*, pages 260–264, 2000.

[5] V. Barnett and T. Lewis. *Outliers in statistical data*. Wiley New York, 1994.

[6] D. Beymer, M. Flickner, I.B.M.A.R. Center, and CA San Jose. Eye gaze tracking using an active stereo head. 2, 2003.

[7] J.Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, 3, 1999.

[8] J. Brand and J. Mason. A comparative assessment of three approaches to pixel-level human skin-detection. In *International Conference on Pattern Recognition*, volume 15, pages 1056–1059, 2000.

[9] JD Brand and JSD Mason. Skin probability map and its use in face detection. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, 2001.

[10] M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. LOF : identifying density-based local outliers. *ACM SIGMOD Record*, 29(2) :104, 2000.

[11] J. Cai and A. Goshtasby. Detecting human faces in color images. *Image and Vision Computing*, 18(1) :63–75, 1999.

[12] P. Campadelli, R. Lanzarotti, and G. Lipori. Face localization in color images with complex background. In *Computer Architecture for Machine Perception, 2005. CAMP 2005. Proceedings. Seventh International Workshop on*, pages 243–248, 2005.

[13] G.I. Chiou and J.N. Hwang. Lipreading from color video. *IEEE Transactions on Image Processing*, 6(8) :1192–1195, 1997.

[14] M. Collobert, R. Feraud, G. Le Tourneur, O. Bernier, JE Viallet, Y. Mahieux, and D. Collobert. Listen : a system for locating and tracking individual speakers. In

Proc. Second Intíl Conf. Automatic Face and Gesture Recognition, pages 283–288, 1996.

[15] T.F. Cootes, C.J. Taylor, DH Cooper, and J. Graham. Training models of shape from sets of examples. 557, 1992.

[16] T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1) :38–59, 1995.

[17] H.D. Crane. The Purkinje image eyetracker, image stabilization, and related forms of stimulus manipulation. *Visual science and engineering : Models and applications*, pages 15–89, 1994.

[18] J.G. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision research*, 20(10) :847–856, 1980.

[19] BT David, R. Chalon, and M. Beldame. Oeil et IHM : suivi du regard et interaction" à líoeil.

[20] P. Delmas, N. Eveno, and M. Lievin. Towards robust lip tracking. In *INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION*, volume 16, pages 528–531, 2002.

[21] X. Deng, C.H. Chang, and E. Brandle. A new method for eye extraction from facial image. *Proceedings of IEEE DELTA, IEEE*, 2004.

[22] A.T. Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods Instruments and Computers*, 34(4) :455–470, 2002.

[23] A.T. Duchowski. *Eye tracking methodology : Theory and practice*. Springer-Verlag New York Inc, 2007.

[24] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, volume 96, pages 226–231, 1996.

[25] N. Eveno, A. Caplier, and P.Y. Coulon. Automatic and accurate lip tracking. *IEEE Transactions on Circuits and Systems for Video technology*, 14(5) :706–715, 2004.

[26] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) :119–139, 1997.

[27] C. Garcia and G. Tziritas. Face detection using quantized skin color regions merging and wavelet packet analysis. *IEEE Transactions on Multimedia*, 1(3) :264–277, 1999.

[28] A. Hadid, M. Pietikainen, and B. Martinkauppi. Color-based face detection using skin locus model and hierarchical filtering. In *INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION*, volume 16, pages 196–200, 2002.

[29] A. Haro, M. Flickner, and I. Essa. Detecting and tracking eyes by using their physiological properties, dynamics, and appearance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. IEEE Computer Society ; 1999, 2000.

[30] C. Harris and M. Stephens. A combined corner and edge detector. 15 :50, 1988.

[31] D.M. Hawkins. *Identification of outliers*. Chapman & Hall, 1980.

[32] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. *Lecture Notes in Computer Science*, pages 170–180, 2002.

[33] C. Hennessey, B. Noureddin, and P. Lawrence. A single camera eye-gaze tracking system with free head motion. pages 87–94, 2006.

[34] S. Horbelt and J.L. Dugelay. Active contours for lipreading-combining snakes with templates. In *Proceedings of the 15th GRETSI Symposium on Signal and Image Processing*, pages 717–720. Citeseer, 1995.

[35] R.L. Hsu, M. Abdel-Mottaleb, and A.K. Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 696–706, 2002.

[36] M. Hu, S. Worrall, AH Sadka, and AM Kondoz. Automatic scalable face model design for 2D model-based video coding. *Signal Processing : Image Communication*, 19(5) :421–436, 2004.

[37] K. Hyoki, M. Shigeta, N. Tsuno, Y. Kawamuro, and T. Kinoshita. Quantitative electro-oculography and electroencephalography as indices of alertness. *Electroencephalography and clinical Neurophysiology*, 106(3) :213–219, 1998.

[38] H. Jiawei and M. Kamber. Data mining : concepts and techniques. *San Francisco, CA, itd : Morgan Kaufmann*, 2001.

[39] T. Johnson, I. Kwok, and R. Ng. Fast computation of 2-dimensional depth contours. In *Proc. KDD*, volume 1998, pages 224–228. Citeseer, 1998.

[40] M.J. Jones and J.M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1) :81–96, 2002.

[41] R. Kaczmarek. Commande oculaire pour l'aide à la communication et au contrôle de l'environnement par l'handicapé moteur. *Motricité cérébrale(Paris)*, 13(1) :24–30, 1992.

[42] M. Kampmann and L. Zhang. Estimation of eye, eyebrow and nose features in videophone sequences. In *International Workshop on Very Low Bitrate Video Coding (VLBV 98), Urbana, USA*, pages 101–104. Citeseer, 1998.

[43] H. Kashima, H. Hongo, K. Kato, and K. Yamamoto. A robust iris detection method of facial and eye movement. In *VI2001, Vision Interface Annual Conference, Ottawa, Canada*, 2001.

[44] R. Kjeldsen and J. Kender. Finding skin in color images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 312–317, 1996.

[45] E. Knorr and R. Ng. Finding intensional knowledge of distance-based. In *Proceedings of the 25th VLDB Conference, Edinburgh, Scotland*, 1999.

[46] E.M. Knorr and R.T. Ng. A unified notion of outliers : Properties and computation. In *Proc. KDD*, volume 1997, pages 219–222, 1997.

[47] E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers : algorithms and applications. *The VLDB Journal*, 8(3) :237–253, 2000.

[48] J.G. Ko, K.N. Kim, and RS Ramakrishna. Facial feature tracking for eye-head controlled human computer interface. *IEEE TENCONí99*, pages 72–75.

[49] JR LaCourse and FC Hludik Jr. An eye movement communication-control system for the disabled. *IEEE Transactions on Biomedical Engineering*, 37(12) :1215–1220, 1990.

[50] D. Li, D. Winfield, and D.J. Parkhurst. Starburst : A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. pages 1–8, 2005.

[51] Y. Li, A. Goshtasby, and O. Garcia. Detecting and tracking human faces in videos. In *INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION*, volume 15, pages 807–810, 2000.

[52] R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. 2(1) :900–903, 2002.

[53] M. Lievin and F. Luthon. Nonlinear color space and spatiotemporal MRF for hierarchical segmentation of face features in video. *IEEE Transactions on Image Processing*, 13(1) :63–71, 2004.

[54] AWC Liew, SH Leung, and WH Lau. Lip contour extraction using a deformable model. 2, 2000.

[55] A.W.C. Liew, KL Sum, SH Leung, and W.H. Lau. Fuzzy segmentation of lip image using cluster analysis. In *Sixth European Conference on Speech Communication and Technology*. ISCA, 1999.

[56] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, page 3. Citeseer, 1981.

[57] S. Lucey, S. Sridharan, and W. Chandran. Chromatic lip tracking using a connectivity based fuzzythresholding technique. In *Signal Processing and Its Applications, 1999. ISSPA'99. Proceedings of the Fifth International Symposium on*, volume 2, 1999.

[58] M.J. Lyons, M. Haehnel, and N. Tetsutani. Designing, playing, and performing with a vision-based mouth interface. In *Proceedings of the 2003 conference on New interfaces for musical expression*, page 121. National University of Singapore, 2003.

[59] D. Maio and D. Maltoni. Real-time face location on gray-scale static images. *Pattern Recognition*, 33(9) :1525–1540, 2000.

[60] A Massonneau and N Biard. Etat de líart des diffÈrents systËmes de pointages ‡ líoeil. *Plate-Forme Nouvelles Technologies*, 2009.

[61] S.J. McKenna, S. Gong, and Y. Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern recognition*, 31(12) :1883–1892, 1998.

[62] G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley New York, 1997.

[63] A. Mehrabian. Communication without words. *Psychology Today*, pages 53–56, 1968.

[64] B. Menser and M. Brunig. Locating human faces in color images with complex background. *In Proceedings of Intelligent Signal Processing and Communications Systems*, page 533ñ536, 1999.

[65] J. Miao, B. Yin, K. Wang, L. Shen, and X. Chen. A hierarchical multiscale and multiangle system for human face detection in a complex background using gravity-center template. *Pattern Recognition*, 32(7) :1237–1248, 1999.

[66] S. Milekic and Educational Resources Information Center (US. *The More You Look the More You Get Intention-Based Interface Using Gaze-Tracking*. ERIC Clearinghouse, 2003.

[67] H.P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, 1980.

[68] CH Morimoto, D. Koons, A. Amir, and M. Flickner. Frame-rate pupil detector and gaze tracker. 99, 1999.

[69] Motaz K. Saad Nabil M. Hewahi. Class outliers mining : Distance-based approach. *International Journal of Intelligent Systems and Technologies 2*, Winter 2007.

[70] J.C. Nascimento and J.S. Marques. Adaptive snakes using the EM algorithm. *IEEE Transactions on Image Processing*, 14(11) :1678–1686, 2005.

[71] A.V. Nefian. Coupled hidden markov model for audiovisual speech recognition, May 9 2002. US Patent App. 10/142,468.

[72] B. L Nguyen, Y Chahir, and F Jouen. Free eye gaze tracking using gaussian processes. *Conference proceedings IPCV 2009*, 2009.

[73] T. Ohno and N. Mukawa. A free-head, simple calibration, gaze tracking system that enables gaze-based interaction. pages 115–122, 2004.

[74] T. Ohno, N. Mukawa, and S. Kawato. Just blink your eyes : a head-free gaze tracking system. pages 950–957, 2003.

[75] T. Ohno, N. Mukawa, and A. Yoshikawa. FreeGaze : a gaze tracking system for everyday gaze interaction. pages 125–132, 2002.

[76] N. Oliver, A. Pentland, and F. Berard. Lafter : Lips and face real time tracker with facial expression recognition. In *Proceedings of Computer Vision and Pattern Recognition Conference, CVPR*, volume 97. Citeseer.

[77] CP Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. pages 555–562, 1998.

[78] E.K. Patterson, S. Gurbuz, Z. Tufekci, and J.N. Gowdy. Moving-talker, speaker-independent feature study, and baseline results using the CUAVE multimodal speech corpus. *EURASIP Journal on Applied Signal Processing*, 11 :1189–1201, 2002.

[79] A. Perez, ML Cordoba, A. Garcia, R. Mendez, ML Munoz, JL Pedraza, and F. Sanchez. A precise eye-gaze detection and tracking system. 2003.

[80] S.L. Phung, A. Bouzerdoum, and D. Chai. A novel skin color model in ycbcr color space and its application to human face detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, 2002.

[81] S.L. Phung, A. Bouzerdoum, and D. Chai. Skin segmentation using color pixel classification : analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1) :148–154, 2005.

[82] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press New York, NY, USA, 1992.

[83] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438. ACM New York, NY, USA, 2000.

[84] C.E. Rasmussen, C.K.I. Williams, and I. Books24x7. *Gaussian processes for machine learning*. Springer, 2006.

[85] P.J. Rousseeuw and A.M. Leroy. *Robust regression and outlier detection*. Wiley-IEEE, 2003.

[86] E. Saber and A. Murat Tekalp. Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions. *Pattern Recognition Letters*, 19(8) :669–680, 1998.

[87] D. Saxe and R. Foulds. Toward robust skin identification in video images. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 379–384, 1996.

[88] R.P. Schumeyer and K.E. Barner. Color-based classifier for region identification in video. In *Proceedings of SPIE*, volume 3309, page 189, 1998.

[89] R. Séguier. Détection de visage adaptative. 2004.

[90] B. Shackel. Eye movement recording by electro-oculography. *A manual of psychophysiological methods*, pages 299–334, 1967.

[91] J. Shi and C. Tomasi. Good features to track. pages 593–600, 1994.

[92] M.C. Shin, K.I. Chang, and L.V. Tsap. Does colorspace transformation make any difference on skin detection. In *IEEE Workshop on Applications of Computer Vision*, pages 275–279. Citeseer, 2002.

[93] L. Sigal, S. Sclaroff, and V. Athitsos. Skin color-based video segmentation under time-varying illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 862–877, 2004.

[94] S. Singh and N. Papanikolopoulos. Vision-based detection of driver fatigue. *Department of Computer Science, University of Minnesota, Technical report*, 1997.

[95] P. Smith, M. Shah, and N. da Vitoria Lobo. Determining driver visual attention with one camera. *IEEE transactions on intelligent transportation systems*, 4(4) :205–218, 2003.

[96] K. Sobottka and I. Pitas. Extraction of facial regions and features using color and shape information. In *International Conference on Pattern Recognition*, volume 13, pages 421–425. Citeseer, 1996.

[97] R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. pages 98–100, 1997.

[98] K.K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1) :39–51, 1998.

[99] JH ten Kate and PM van der Meer. An electro-ocular switch for communication of the speechless. *Medical progress through technology*, 10(3) :135.

[100] J.C. Terrillon, H. Fukamachi, S. Akamatsu, and M.N. Shirazi. Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*. IEEE Computer Society Washington, DC, USA, 2000.

[101] Y. Tian, T. Kanade, and J. Cohn. Robust lip tracking by combining shape, color and motion. In *Proceedings of the 4th Asian Conference on Computer Vision (ACCV'00)*. Citeseer, 2000.

[102] S. Tsekeridou and I. Pitas. Facial feature extraction in frontal views using biometric analogies. *Proc. 9th European Signal Processing Conference September 811 Island of Rhodes, Greece*, page 1 :315ñ318, 1998.

[103] V. Vezhnevets and A. Degtiareva. Robust and accurate eye contour extraction. In *Proc. Graphicon*, pages 81–84. Citeseer, 2003.

[104] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 85. Citeseer, 2003.

[105] W.E. Vieux, K. Schwerdt, and J.L. Crowley. Face-tracking and coding for video compression. *Lecture notes in computer science*, pages 151–161, 1998.

[106] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. 1, 2001.

[107] R. Von Mises and H. Geiringer. *Mathematical theory of probability and statistics*. Academic Press New York, 1964.

[108] C. Wang and MS Brandstein. Multi-source face tracking with audio and visual data. In *1999 IEEE 3rd Workshop on Multimedia Signal Processing*, pages 169–174, 1999.

[109] T. Wark and S. Sridharan. A syntactic approach to automatic lip feature extraction forspeaker identification. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 6, 1998.

[110] D. Weiner and N. Kiryati. Virtual gaze redirection in face images. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*, pages 76–81. Citeseer, 2003.

[111] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. *Lecture Notes in Computer Science*, pages 687–694, 1997.

[112] J. Yang and A. Waibel. A real-time face tracker. In *Proceedings of WACV*, volume 96, pages 142–147. Citeseer, 1996.

[113] M.H. Yang and N. Ahuja. Detecting human faces in color images. *Urbana*, 51 :61801.

[114] M.H. Yang and N. Ahuja. Gaussian mixture model for human skin color and its application in image and video databases. In *Proc. SPIE : Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 458–466. Citeseer, 1999.

[115] M.H. Yang, D. Kriegman, and N. Ahuja. Face detection using multimodal density models. *Computer Vision and Image Understanding*, 84(2) :264–284, 2001.

[116] M.H. Yang, D.J. Kriegman, and N. Ahuja. Detecting faces in images : A survey. *IEEE Transactions on Pattern analysis and Machine intelligence*, pages 34–58, 2002.

[117] D.H. Yoo and M.J. Chung. A novel non-intrusive eye gaze estimation using cross-ratio under large head motion. *Computer Vision and Image Understanding*, 98(1) :25–51, 2005.

[118] L.R. Young and D. Sheena. Survey of eye movement recording methods. *Behavior research methods and instrumentation*, 7(5) :397–429, 1975.

[119] A.L. Yuille, P.W. Hallinan, and D.S. Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2) :99–111, 1992.

[120] B. Zhang, W. Gao, S. Shan, and W. Wang. Constraint shape model using edge constraint and gabor wavelet based search. *Lecture notes in computer science*, pages 52–61, 2003.

[121] X. Zhang and RM Mersereau. Lip feature extraction towards an automatic speechreading system. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 3, 2000.

[122] X. Zhu, J. Yang, and A. Waibel. Segmenting hands of arbitrary color. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, page 446. Citeseer, 2000.

[123] Z. Zhu and Q. Ji. Eye gaze tracking under natural head movements. 1, 2005.