Institut National Polytechnique de Grenoble
ENSIMAG

Université Joseph Fourier
U.F.R.
Informatique & Mathématiques Appliquées

I.M.A.G.

DEA D'INFORMATIQUE

Projet présenté par :
Sébastien Maury

SUIVI du REGARD

Effectué au Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle (LIFIA-IMAG).

Date : Le 22 juin 1994

Présidents du Jury :	Augustin Lux (LIFIA-IMAG)
	Michel Riveill (BULL-IMAG)
Membres du Jury :	James L. Crowley (LIFIA-IMAG)
	Joëlle Coutaz (LGI-IMAG)
	Didier Aubert (ITMI)

Avant toutes choses, je tiens à remercier vivement James L. Crowley et Joëlle Coutaz pour m'avoir encadré dans ce projet et pour leur gentillesse. Je remercie également messieurs Augustin Lux Michel Riveill de présider le jury et Didier Aubert d'avoir accepté d'y participer .

Merci aussi à toute l'équipe du LIFIA, à Patrice de Marconnay pour m'avoir permis de démarrer mes recherches.

Sans la bonne humeur de toute l'équipe Interface Homme-Machine du LGI ce DEA n'aurait jamais été aussi agréable, je remercie énormément François Bérard, Daniel Salber, Laurence Nigay, l'équipe des DESS, Éric Carraux, Gilles Ambone, Francis Jambon, Nathalie Lischetti et toute l'équipe. Je remercie aussi Philippe Mulhem pour ses blagues et sa bonne humeur, Stéphane Sire pour m'avoir accompagner dans de quelques nuits blanches de travail. Enfin un merci à Valérie pour tout.

Table des matières

Table des matières **v**

Introduction
Le sujet et ses objectifs **9**

Le sujet 9
Les objectifs 10

Chapitre 1
Le domaine et ses contraintes **13**

1.1- Les propriétés de l'œil 13
1.1.1- Les mouvements de l'œil 13
1.1.2- La fovéa 15
1.2- Le regard en communication homme-machine 15
1.2.1- Suivi du regard et acquisition : l'œil agit indirectement sur l'état du système 16
1.2.1.1- Exploitation après coup des données comportementales de l'œil 16
1.2.1.2- Exploitation en ligne des données comportementales de l'œil 17
1.2.2- Suivi du regard et transfert : l'œil agit explicitement sur l'état du système 20
1.3- Les contraintes 23
1.3.1- Intrusion 23
1.3.2- Vitesse 23
1.3.3- Précision 24
1.2.4- Problème du déplacement de la tête 25
1.4- Approche technique retenue 25

Chapitre 2
Survol des techniques **27**

2.1- Les configurations matérielles et leurs problèmes 27
2.1.1- Méthode avec appareillage 27
2.1.1.1- Méthode avec casque 28
2.1.1.2- Méthode avec "lunettes" 29
2.1.2- Méthode "écologique" 30
2.2- "EigenEyes" 31

2.2.1- Principe....32
2.2.2- Construction de l'"EigenSpace"....33
2.2.3- Présentation d'une nouvelle image....34
2.2.4- Recherche de la direction du regard....34
2.2.5- Recherche de l'œil sur une image....35
2.2.6- Conclusion....35
2.3- Gabarits déformables ("Deformable templates")....36
2.3.1- Principe....36
2.3.2- Conclusion....38
2.4- Images d'orientations locales....39
2.5- Corrélation des laplaciennes multi-échelles....40
2.5.1- La pyramide gaussienne....41
2.5.2- Recherche de l'œil....42
2.5.3- Conclusion....43
2.6- Perceptron multicouche....43
2.6.1- Architecture du réseau....44
2.6.2- Processus d'apprentissage....45
2.6.3- Reconnaissance....45
2.6.4- Conclusion....45
2.7- Les techniques retenues....46

Chapitre 3
Évaluation **47**

3.1- Les objectifs....47
3.2- Un environnement unique pour toutes les méthodes....48
3.3- Évaluation de la précision....49

Chapitre 4
Les "EigenEyes" **51**

4.1- Construction de l'"EigenSpace"....51
4.2- Présentation d'une nouvelle image....55
4.3- Recherche de la direction du regard....55
4.4- Recherche de l'œil sur une image....56
4.5- Implémentation....57

Chapitre 5
La corrélation des laplaciennes, multi-échelles **59**

5.1- La pyramide gaussienne....59
5.2- Corrélation....62
5.2- Utilisation....64

Chapitre 6
Perceptron multicouche **67**

6.1- Les réseaux multicouche....67
6.1.1- Modèle du neurone et du réseau....67
6.1.2- Algorithme de rétro-propagation du gradient....69
6.1.2.1- Principe....69
6.1.2.2- Formalisation....70
6.1.2.3- Reconnaissance....71
6.2- Architecture du réseau ANN....71
6.2.1- topologie du réseau....71
6.2.2- Format des données de la couche de sortie....72
6.3- Expérimentation....74
6.3- Conclusion....76

Conclusion **79**

Bilan de notre contribution 79
Les techniques de suivi du regard :..... 80
Utilisation d'un système de suivi de regard au sein d'une interface : nos constats et perspectives..... 82

Bibliographie **83**

Introduction le sujet et les objectifs

Le sujet

Les avancées techniques des interfaces graphiques ont largement contribué à améliorer l'utilisabilité des ordinateurs. Aujourd'hui, certains constructeurs, tel Apple, proposent en standard de nouveaux moyens de communication fondés sur des systèmes de reconnaissance de l'écriture ou de la parole. D'autres, et notamment Silicon Graphics, intègrent une caméra et des techniques de numérisation d'image mais l'usage de la vision par ordinateur en interface homme-machine est encore peu exploitée. L'objet de ce projet de DEA est d'étudier des techniques de vision par ordinateur applicables au suivi du regard.

La direction du regard et le mouvement des yeux ont déjà fait l'objet de recherches en Interface Homme-Machine et notamment en facteurs humains. Jusqu'ici, les objectifs ont été peu ambitieux et de bas niveaux avec l'étude du cheminement du regard au sein d'une scène graphique. Les études ont été menées dans des laboratoires sans viser l'intégration cognitive du regard dans les interfaces utilisateur. Par exemple, le regard peut être relié au sujet d'intérêt de l'utilisateur. Pour les handicapés moteur, le regard peut servir à la manipulation d'objets.

Nous allons, pour l'essentiel, nous intéresser aux aspects techniques du suivi du regard. Les systèmes actuellement utilisés pour la détection du regard, et pour certains commercialisés, nécessitent un équipement spécial encombrant : lunettes ou casque avec

une ou plusieurs diodes à infrarouge pour suivre les mouvements de la pupille [Jacob90] ; le problème se ramène alors essentiellement à une procédure de calibrage de l'appareillage. Nous visons une solution plus ambitieuse mais "écologique".

Les objectifs

Notre objectif est d'étudier l'apport de la vision par ordinateur pour le suivi du regard sans faire intrusion dans les activités de l'utilisateur tout en offrant rapidité, précision et fiabilité. Ce parti pris introduit de nouvelles difficultés : détecter les yeux dans une image au moyen d'une caméra fixe en tenant compte des mouvements de la tête. Ces mouvements par rapport à l'écran et à la caméra modifient la position des yeux dans l'image et changent l'orientation des pupilles même si l'utilisateur regarde le même point de l'écran. Dans le cadre de ce DEA, nous avons restreint notre objectif général à l'étude comparative de plusieurs techniques de vision par ordinateur qui semblent fournir des solutions convenables à notre problème.

Ce rapport d'étude est le résultat de notre réflexion sur l'intégration des techniques de vision par ordinateur dans la communication homme-machine en considérant le cas du suivi du regard. Il est organisé comme suit :

• au chapitre 1, nous définissons les contours du problème. Nous identifions les applications possibles du suivi du regard en Interface Homme-Machine et nous étudions les propriétés comportementales de l'œil. Ces deux composantes du problème nous permettent de fixer les contraintes de l'étude en termes de vitesse, de précision et d'ergonomie ;

• il convient ensuite d'étudier les techniques de vision par ordinateur applicables à notre problème. Parmi les possibilités, une première analyse permet d'identifier les approches qui semblent répondre aux contraintes fixées : la technique des EigenEyes, la corrélation de laplaciennes multi-échelles et le perceptron multicouche ;

• au chapitre 3, nous fixons le cadre expérimental de l'étude : l'objectif est de construire un environnement qui permet d'évaluer dans des conditions similaires les trois techniques retenues. Les critères comparatifs sont directement tirés des contraintes définies au chapitre 1 ;

• les trois chapitres suivants décrivent en détail chacune des techniques recrutées, leur adaptation au problème et notre opinion ;

• en conclusion, nous présentons une synthèse de ces travaux comparatifs et des propositions de perspectives.

Chapitre 1
Le domaine et ses contraintes

Dans ce chapitre, nous cherchons à définir les critères qu'un système de vision par ordinateur doit satisfaire pour la mise en œuvre de détecteur de regard "écologique". Nous dirons qu'un tel dispositif est écologique s'il n'introduit pas de contrainte particulière pour l'utilisateur. Nous procéderons comme suit : étude des propriétés pertinentes de l'œil (présentées au paragraphe suivant), puis analyse du rôle du regard en communication homme-machine et ses applications actuelles et potentielles (paragraphe 1.2). Au vu de cette étude préliminaire, nous définissons en 1.3 les contraintes auxquelles un détecteur de regard doit répondre et justifions en 1.4 notre choix de matériel.

1.1- Les propriétés de l'œil

Nous avons relevé deux propriétés pertinentes de l'œil : ses mouvements et sa fovéa.

1.1.1- Les mouvements de l'œil

Un premier constat s'impose : l'œil est rarement immobile. L'étude de Jacob ([Jacob 90]) est à ce titre révélatrice. De petits mouvements brefs et saccadés succèdent à de courtes périodes d'immobilité où l'utilisateur fixe un objet. La figure 1.1, extraite des travaux de Jacob, montre un échantillon de la position de l'œil pour une période de 30 secondes. L'immobilité ne dure jamais plus de 600ms. Lorsqu'elle se produit, elle est due, non pas à une action consciente de fixation, mais au fait que l'utilisateur ne "regarde"

pas (il pense à autre chose). L'œil adopte un déplacement continu uniquement lorsqu'il suit un objet en mouvement.

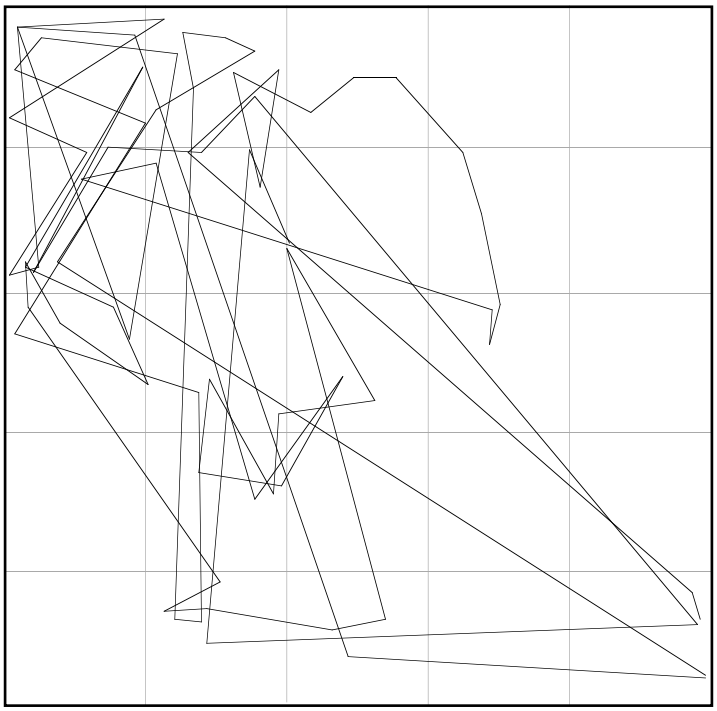


Figure 1.1 : échantillon de la position de l'œil observant une scène quelconque pendant une durée de 30 secondes (d'après [Jacob90]).

Les saccades constituent un bruit qu'il convient d'éliminer. Jacob propose d'appliquer un pré-traitement de filtrage pendant l'acquisition. Il s'agit d'un algorithme de reconnaissance des périodes de fixation. Une période correspond à une fixation si l'œil reste à peu près dans la même zone pendant une durée suffisante. Cette période peut être entrecoupée de fortes saccades mais l'œil aura tendance à revenir au même endroit. La figure 1.2 donne un exemple de données brutes et les périodes de fixation obtenues en temps réel pour une durée de trois secondes. Jacob estime que ce filtrage permet d'obtenir une entrée utilisable pour une interface homme-machine alors que l'utilisation directe des données brutes peut conduire à une interface visuellement "instable" (cas où l'on aurait astreint le curseur de la souris au suivi du regard) ou au déclenchement d'actions indésirables.

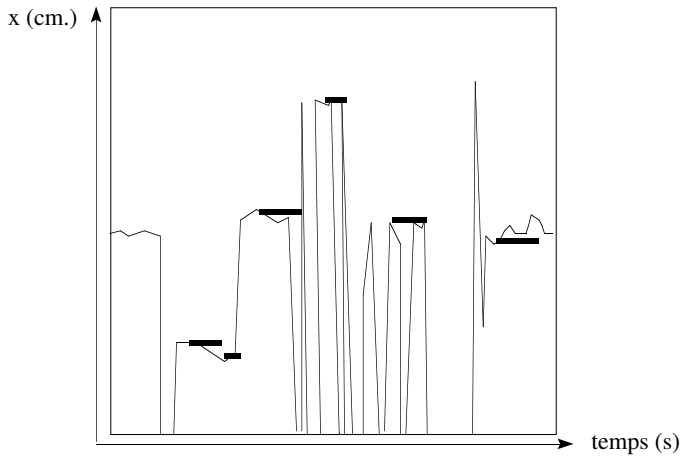


Figure 1.2 : résultats de la recherche de période de fixation sur un échantillon de la position en x de l'œil pendant 3 secondes (d'après [Jacob90]). Chaque traits épais dénote une fixation.

1.1.2- La fovéa

Les mouvements saccadés et incessants de l'œil permettent de balayer très rapidement le champ visuel, donnant l'impression d'avoir une résolution maximale sur tout le champ. En fait, l'œil ne possède cette résolution que dans une petite zone de deux degrés environ appelée la fovéa. L'homme, lorsqu'il fixe un objet, ne cherche pas à le regarder plus précisément si cet objet se trouve déjà dans la fovéa. Il n'obtiendra jamais une meilleure résolution.

Une conséquence immédiate est qu'il est inutile dans le cas d'un détecteur de regard, de chercher à obtenir une précision supérieure à la taille de la fovéa. Elle ne serait pas significative de l'intention de l'utilisateur.

Nous venons de relever deux caractéristiques déterminantes du comportement oculaire : un mouvement quasi perpétuel et saccadé, et l'existence d'une fovéa de taille réduite. Il convient maintenant d'analyser le rôle du regard comme support de communication avec une machine.

1.2- Le regard en communication homme-machine

De manière générale, le regard humain a deux fonctions complémentaires : l'acquisition de données et le transfert d'information. Considérons ces deux fonctions dans le cadre de la communication personne-système. Lorsqu'il opère en mode acquisition, l'œil est actif

pour le sujet humain mais passif pour la machine : l'état du système observé est inchangé. Inversement, dans sa fonction de transfert, l'œil adopte un rôle actif vis-à-vis de la machine : l'acte de regard a un impact sur l'état du système observé. Nous allons étudier les applications du suivi du regard selon ces deux dimensions : acquisition et transfert.

1.2.1- Suivi du regard et acquisition : l'œil agit indirectement sur l'état du système

Un capteur de suivi du regard permet d'observer le comportement de l'œil et de tirer des leçons en fonction du contexte d'utilisation et des objectifs. La figure 1.1 déjà étudiée au paragraphe précédent, illustre un premier exemple d'une telle utilisation : elle démontre sans conteste le caractère saccadé des mouvements oculaires. Nous allons présenter deux classes d'application selon que les données comportementales de l'œil sont utilisées après ou pendant l'interaction.

1.2.1.1- Exploitation après coup des données comportementales de l'œil

La capture du mouvement oculaire est une source intéressante d'enseignements pour les bureaux d'études en marketing. L'analyse de la direction du regard permet de déterminer les éléments saillants d'une affiche publicitaire. Nous pouvons envisager d'appliquer cette technique à des domaines autres que la publicité : ainsi une telle analyse pourrait apporter des informations pertinentes dans un système d'enseignement par ordinateur comme Arcade [Gueraud91]. Une analyse à posteriori d'une session d'un étudiant permettrait de connaître les objets graphiques que l'étudiant a observé et par conséquent de connaître les concepts difficiles ou mal appréhendés par l'étudiant.

Une seconde application de la détection du suivi du regard revient à l'évaluation des interfaces utilisateur. L'utilisabilité est devenue un argument de vente. Elle retient l'attention des chercheurs et des praticiens de l'ergonomie cognitive [Coutaz 94, Hix 93, Nielsen 90]. Des laboratoires d'ergonomie qui permettent d'observer le comportement de sujets accomplissant une tâche, sont maintenant disponibles chez les grands constructeurs (Hewlett Packard à Grenoble, Apple, Microsoft, IBM, etc.). Au LGI-IMAG, Neimo, un laboratoire numérique d'observation est en cours de réalisation [Salber 93]. Toutes ces techniques d'observation expérimentales utilisent des caméras dirigées sur les mains, le visage et l'écran. A l'exception de Neimo, les images sont enregistrées sur support analogique. L'enregistrement des images de l'écran permet à l'ergonome d'identifier ce que le sujet observe. A l'évidence, l'intégration d'un dispositif de capture du regard faciliterait considérablement la tâche de l'analyste. Il serait ainsi possible de déterminer si

les éléments importants de l'interface sont rapidement perçus par le sujet ; il serait possible de savoir où l'utilisateur s'attend à trouver une information donnée dans un contexte particulier ; il serait possible de détecter les navigations oculaires de recherche d'information, etc.

Les deux exemples que nous venons de présenter concernent l'utilisation après coup des données comportementales du regard. Nous proposons ci-dessous une utilisation en situation d'interaction.

1.2.1.2- Exploitation en ligne des données comportementales de l'œil

Nos deux premiers exemples s'appuient sur la capacité du système à déterminer le focus d'attention de l'utilisateur. Cette connaissance permettrait de répondre au problème de la fenêtre non sélectionnée ou d'améliorer l'efficacité du transfert des images sur les réseaux à faible débit. Le troisième exemple vise à reconstituer le contact oculaire dans la communication visuelle à distance.

Une première application que nous suggérons est l'utilisation du suivi du regard pour résoudre, au moins partiellement, le problème bien connu de la "fenêtre non sélectionnée" : l'utilisateur conduit plusieurs tâches en parallèle (édition, courrier électronique, etc.), est interrompu (appel téléphonique), décide de reprendre une autre tâche mais oublie de sélectionner la fenêtre support. A condition que l'utilisateur regarde l'écran quelques instants, le détecteur du regard peut prendre à sa charge le changement de fenêtre : la tâche de sélection de fenêtre qui normalement revient à l'utilisateur, peut migrer vers le système. Des tests d'observation selon les techniques d'évaluation présentées plus haut nous permettraient de confirmer ou de remettre en cause notre hypothèse.

Notre seconde suggestion revient à appliquer le concept de "fish-eye view" au transfert d'images sur les réseaux. Le "fish-eye view" [Furnas86] est une méthode d'affichage qui permet de restituer dans sa totalité une représentation graphique dans une fenêtre. L'utilisateur désigne à la souris le centre d'intérêt : celui-ci est représenté en clair avec tous les détails tandis que les parties environnantes se fondent progressivement dans le confus (la figure 1.3 donne un exemple de représentation en "fish-eye view" d'un réseau de transport au États-Unis tiré de [Brown93]). Cette technique s'oppose à celle du "zoom" qui ne conserve pas la vue globale du concept représenté. (La rupture entre niveaux de représentation peut nuire à l'utilisateur dans l'accomplissement de sa tâche.) Appliquée en visiophonie, et en supposant des vitesses suffisantes pour la détection du focus d'attention, on transmettrait à haute résolution la partie de l'image qui correspond à la

fovéa de l'observateur, le reste de l'image hors de la fovéa restant à basse résolution. Comme nous le démontrons ci-dessous, cette technique permettrait de répondre aux limitations des réseaux de faible débit.

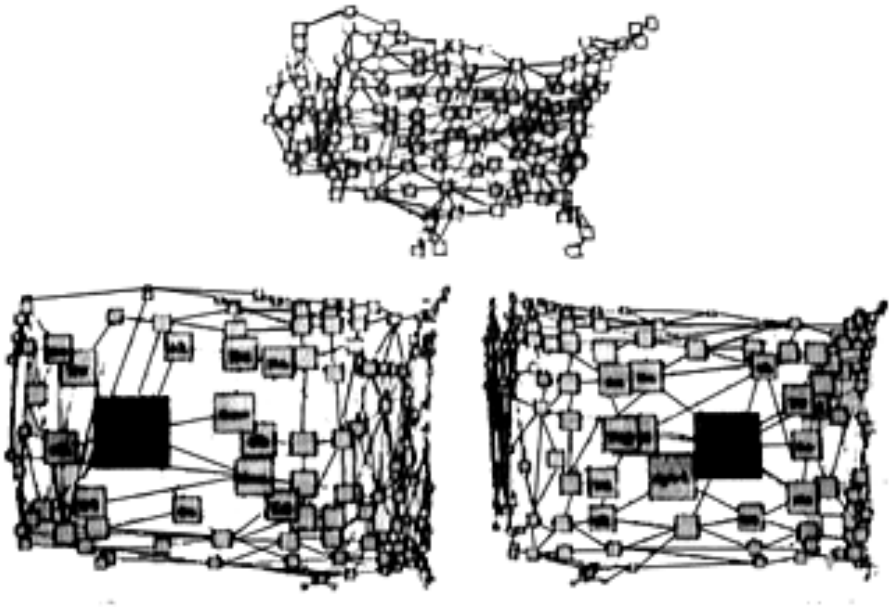


Figure 1.3 : Représentation en "fish-eye view". En haut, la carte d'origine des États-Unis. En bas à gauche, le focus d'attention se porte sur la ville de Salt Lake City, à droite sur la ville de St Louis (d'après [Brown93]).

Soit une image de taille NxN. Supposons que l'on transmette en haute résolution une zone représentant un seizième de l'image totale, et que l'on transmette un quart de l'image en moyenne résolution pour lisser la transition haute résolution / basse résolution sur l'image finale. Le reste de l'image est envoyé à basse résolution. Calculons la taille de l'image transmise :

• pour obtenir l'image de basse résolution, on sous-échantillonne deux fois l'image de départ (niveau 2 de la pyramide gaussienne de corrélation). L'image obtenue est de taille :

$$t_1 = \frac{N^2}{4^2}$$

• l'image de moyenne résolution est obtenue de la même façon en sous-échantillonnant une seule fois l'image de départ. Étant donné qu'on ne garde qu'un quart de l'image, sa taille est alors :

$$t_2 = \frac{1}{4}.\frac{N^2}{4}$$

• la portion d'image en haute résolution est de taille :

$$t_3 = \frac{N^2}{16}$$

Soit une taille totale T égale à :

$$T = 3.\frac{N^2}{16}$$

au lieu de N^2. La figure 1.4 donne un aperçu de la qualité d'une image obtenue selon cette technique.

Figure 1.4 : représentation en multirésolution d'une image sur 3 niveaux de résolution, le focus d'attention est centré sur le visage.

L'idée de garder des parties de l'image sur deux niveaux de la pyramide gaussienne de l'image de départ peut être étendue à n niveaux. La taille de l'image à transmettre est alors de :

$$T = \frac{N^2}{4^n}.\frac{1}{4^0} + \frac{N^2}{4^{n-1}}.\frac{1}{4^1} + \frac{N^2}{4^{n-2}}.\frac{1}{4^2} + \ldots + \frac{N^2}{4^1}.\frac{1}{4^{n-1}} + \frac{N^2}{4^0}.\frac{1}{4^n}$$

$$T = \sum_{i=0}^{n} \frac{N^2}{4^{n-i}} . \frac{1}{4^i}$$

$$T = (n+1) . \frac{N^2}{4^n}$$

Notre dernier exemple vise à récréer le contact visuel dans les applications de téléconférence. La rupture du contact visuel est une source de difficultés en communication à distance ([Buxton 91]). Il se trouve que l'appareillage actuel satisfait assez mal cette contrainte : dès que l'orateur ne regarde pas la caméra, le contact est rompu. Ott et ses co-auteurs ont imaginé une solution originale avec un système de vision stéréoscopique [Ott 93]. Une caméra est placée au bas de l'écran, l'autre au-dessus et un algorithme de mise en correspondance reconstitue une image au regard centré. Notre idée va plus loin : le regard de l'image virtuelle n'est pas figé mais suit dynamiquement à distance le regard du receveur.

Nous étudions maintenant des exemples où, du point de vue du système, le regard est action.

1.2.2- Suivi du regard et transfert : l'œil agit explicitement sur l'état du système

L'une des premières applications du suivi du regard s'adresse aux handicapés moteur. L'œil se substitue à la main. Une interface comme celle de la figure 1.5 permet la saisie de caractères en sélectionnant au moyen des yeux les touches d'un clavier dessiné à l'écran. Pour valider une touche, on convient d'un seuil temporel de fixation au-delà duquel le système considère que la touche est enfoncée. Ou bien, l'utilisateur doit appuyer un bouton dédié. Pour assurer une bonne efficacité, la plupart de ces systèmes nécessite le port de lunettes ou d'un casque.

Figure 1.5 : type d'interface pour la saisie de texte avec le regard sur une représentation d'un clavier à l'écran.

Dans un tout autre domaine, le laboratoire de recherches navales aux États-Unis utilise des détecteurs de regard implantés dans le casque des pilotes de chasse. Le casque contient un écran LCD où sont affichés les menus et les informations. Ce dispositif leur permet d'obtenir rapidement des informations et de sélectionner des options.

De son côté, Jacob a réalisé plusieurs tests pour étudier l'usage du regard dans les interfaces de type manipulation directe. Il a retenu des tâches génériques comme la sélection, le déplacement d'objets à l'écran et l'utilisation d'éléments usuels d'interaction comme les menus déroulants, le défilement des fenêtres et la sélection de la fenêtre de travail. La direction du regard est incluse dans le flot d'entrée de l'interface au même titre que le clavier ou la souris. Les trois techniques peuvent être utilisées conjointement.

Jacob a testé deux méthodes de sélection d'objets à l'aide du regard : l'une où l'utilisateur doit appuyer sur un bouton pour sélectionner l'élément qu'il regarde, l'autre où l'élément est automatiquement sélectionné après une durée de fixation donnée. Il faut noter que les deux méthodes peuvent être employées conjointement un peu à la manière d'un raccourci clavier. L'analyse montre que le "tout regard" est préférable si l'on considère la satisfaction de l'utilisateur et ses performances (rapidité d'exécution, nombre d'erreurs). Il faut néanmoins veiller à configurer correctement le délai de sélection : trop long (supérieur à 750ms.), l'interface réagit trop lentement ; trop court, l'utilisateur risque de sélectionner des objets par inadvertance. Jacob propose de configurer le délai selon le type de l'action : délais courts (150ms.) pour les actions facilement annulables (du type obtention d'informations sur l'objet pointé) et des délais plus longs (entre 250 et 700ms.) pour les actions portant à conséquence comme la suppression (principe de l'effort

incommensurable : ce qui est difficile à faire doit être difficile à défaire). Par sécurité, les principes d'ergonomie recommandent une confirmation à la souris ou au clavier.

Une autre étude de Jacob concerne l'utilisation du regard pour déplacer un objet à l'écran. Là encore deux méthodes ont été confrontées : l'une où l'objet, sélectionné avec la méthode précédente, est déplacé à la souris relativement à sa position de départ (il n'y a pas de curseur de souris). Pour la seconde, l'objet observé est mis en position "déplacement" tant qu'un bouton donné est enfoncé. L'objet suit alors le déplacement des yeux jusqu'au relâchement du bouton. Les résultats montrent qu'après une période d'apprentissage l'utilisateur préfère le "tout-regard". Toutefois, la précision d'évaluation de la position du regard étant ce qu'elle est, les résultats sont meilleurs lorsque la destination correspond à un autre objet. Si c'est un point arbitraire de l'écran, l'utilisateur a du mal à le fixer. C'est pourquoi, substituer l'œil à la souris comme dispositif de pointage est inopportun pour des applications de type MacPaint. L'être humain ne contrôle pas suffisamment les déplacements de son œil pour réaliser des actions qui exigent de la précision.

Jacob a ensuite réalisé quelques tests sur la manipulation au moyen de l'œil, d'éléments usuels des interfaces graphiques : navigation dans un texte, menus déroulants, désignation de la fenêtre active. Pour le déplacement dans un texte qui ne tient pas dans une fenêtre d'édition, on peut placer des flèches en haut et en bas du texte. Lorsque l'utilisateur arrive à la dernière ligne (ou la première) il regarde la flèche pour monter (ou descendre) le texte. Il faut noter que l'utilisateur ne peut plus lire le texte en même temps qu'il le déplace comme il le ferait avec la souris. Une fois de plus, combiner les deux modalités permet de pallier cet inconvénient : l'utilisateur peut placer la souris sur la flèche et parcourir le texte en le déplaçant au moyen de la souris. Pour l'utilisation des menus déroulants, il suffit de fixer le nom du menu pendant un laps de temps donné (400ms.) pour qu'il se déroule. Les éléments du menu s'allument lorsque l'utilisateur les lit et la commande est sélectionnée s'il fixe l'intitulé pendant une seconde. S'il regarde en dehors du menu pendant plus de 600ms, le menu est désactivé.

Les propriétés du comportement oculaire, les expérimentations de Jacob ou nos propositions d'application de la détection du regard nous conduisent à définir des contraintes de réalisation.

1.3- Les contraintes

L'étude préalable que nous venons de résumer montre que la vitesse et la précision sont des critères incontournables. Mais, ils ne suffisent pas à définir l'utilisabilité du détecteur de regard. Nous reprenons ces points sous les titres intrusion, vitesse, précision et mouvement de la tête.

1.3.1- Intrusion

Le détecteur de regard doit être parfaitement intégré dans l'environnement de travail. Pour être accepté, il doit "se faire oublier" : absence d'appareil encombrant, pas de port de dispositif comme le casque ou les lunettes. Le montage de Jacob que nous décrivons au chapitre suivant (figure 2.4) et qui n'impose pas de port d'appareil particulier, ne répond cependant que partiellement à cette exigence. Un servomoteur, qui contrôle l'orientation d'un miroir, dérange parfois l'utilisateur qui a l'impression que l'appareil le "suit". Dans le même ordre d'idée, les caméras mobiles doivent être évitées ou masquées derrière un miroir sans teint ou fumé. Enfin, il est souhaitable que notre système fonctionne dans les conditions de travail habituelles d'éclairage : certains systèmes ([Jacob90]) nécessitent une semi-obscurité qui gêne l'écriture et la lecture. D'autres au contraire requièrent un éclairage artificiel du visage provoquant fatigue et inconfort.

En synthèse, notre solution doit satisfaire au critère de discrétion : encombrement faible, absence de bruit, immobilité (ou mobilité masquée), fonctionnement avec l'éclairage ambiant. Ainsi peut-on espérer limiter le caractère intrusif du détecteur de regard.

1.3.2- Vitesse

La vitesse de réaction d'un détecteur est un paramètre essentiel à son utilisabilité. Nous avons vu que la vitesse requise dépend de l'usage. Par exemple, pour une tâche de sélection, une mauvaise adaptation, par des réglages trop long ou trop courts des durées de fixation, entraîne des réponses inadaptées ou incontrôlables. Il est donc essentiel de bien "calibrer" le détecteur de regard. Un réglage contextuel, fonction de l'action déclenchée, semble indispensable pour des raisons d'efficacité et de sécurité.

Bien que la vitesse soit un facteur déterminant, nous avons vu qu'une vitesse très élevée n'apporte pas davantage d'informations pour déterminer les objets que l'utilisateur

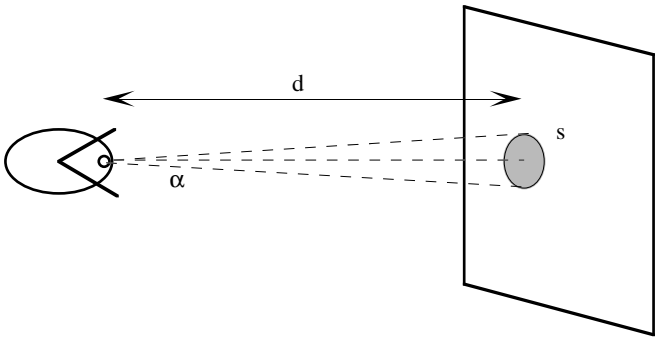
regarde “consciemment”. Une vitesse trop élevée ne ferait qu’exacerber le bruit inhérent aux mouvements incessants de la pupille.

Au total, une vitesse de l’ordre de 10Hz qui correspond au cycle de base moyen du système visuel humain [Card 83], devrait convenir.

1.3.3- Précision

Pour la précision du système, les impératifs sont à peu près identiques à ceux de la vitesse : il faut un minimum de précision pour que notre système soit viable dans une interface graphique. Là encore l’œil limite cette précision. L’idéal serait de déterminer le pixel observé. Mais dans ce cas, une telle résolution n’est exploitable que si l’utilisateur est suffisamment près de l’écran ou si les pixels sont gros. En effet, l’homme a une vision à résolution maximale que dans la zone de la fovéa. Si le point observé est dans une telle zone couvrant plusieurs pixels, il est impossible de déterminer lequel est “observé”.

En conséquence, la précision obtenue est non seulement fonction de la qualité du capteur de regard mais également de la distance de l’utilisateur à l’écran. La figure 1.6 donne la relation entre la distance de l’utilisateur à l’écran, l’angle de la fovéa (α) et la surface (en cm^2) de l’écran couverte par le regard.



$$s = \pi.\left(d.\tan\frac{\alpha}{2}\right)^2$$

Figure 1.6 : Relation entre angle de vue α et surface s en cm^2 couverte sur un écran situé à la distance d de l’œil

Pour une distance d à l’écran de 60cm, un angle α de 2° (angle de la fovéa), on obtient une surface “élémentaire” de 3,4 cm^2, ce qui représente un carré d’environ 1,8 x 1,8 cm.

1.2.4- Problème du déplacement de la tête

Un des problèmes majeurs est le déplacement de la tête de l'utilisateur. La plupart (pour ne pas dire toutes) les solutions existantes reposent sur la mise en relation d'un reflet d'une source lumineuse fixe dans l'œil, de la position de la pupille et d'une zone de l'écran. Si l'utilisateur ne bouge pas ou peu la tête, ce reflet reste fixe dans l'œil. Le couple coordonnées du reflet et coordonnées de la pupille (figure 1.7) fournit une relation directe entre le repère de l'œil et le repère fixe de l'écran. Si la tête se déplace, cette relation linéaire n'est plus valide. A position de tête différente et positions de pupille et reflet identiques, le "point" de l'écran observé est différent.

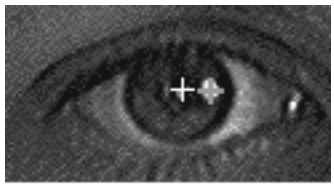


Figure 1.7 : position du reflet d'une source lumineuse (croix grise) et le centre de la pupille.

Plusieurs solutions sont alors envisagées : des systèmes "extrêmes", comme on peut en utiliser à la Cité des Sciences de la Villette, imposent de placer sa tête dans un casque. Les différents repères liés à l'écran, la caméra, la source lumineuse et l'œil sont alors tous fixes. Une calibration unique suffit au bon fonctionnement de l'ensemble. D'autres systèmes lient le repère de l'œil et celui de la source lumineuse en implantant une diode infrarouge sur des lunettes. Ce système permet d'éviter la recherche des yeux dans une image mais ne résoud pas le problème de mise en correspondance du repère de l'œil avec celui de l'écran. Dans notre cas, où nous voulons éviter toutes contraintes physiques à l'utilisateur, la recherche des yeux dans l'image acquise par une caméra est nécessaire. En fixant la caméra et la diode à l'écran, nous gardons les trois repères associés liés. Il reste à mettre ces repères en correspondance avec celui de l'œil.

1.4- Approche technique retenue

L'analyse du problème, ses contraintes et les esquisses de solution, nous font opter pour la solution d'une caméra fixe intégrée à une station de travail du marché. La figure 1.8 montre la configuration choisie : un Macintosh Quadra 840AV avec 24Mo de mémoire vive et 1Go de disque, une carte d'acquisition vidéo (en standard) , un écran 17 pouces, une caméra CCD de type camescope et éventuellement une source lumineuse invisible pour l'utilisateur. Ce dernier doit avoir la possibilité de bouger librement la tête et

travailler dans des conditions usuelles : sans éclairage particulier, sans bruits supplémentaires et sans avoir l'impression d'être observé (une petite caméra enchâssée sous le moniteur va dans ce sens).

Notre choix est cohérent avec le pari des constructeurs de produire en série des ordinateurs équipés de composants de traitements du signal (comme la série des Macintoshes AV d'Apple), de cartes d'acquisition vidéo et même de caméra (Silicon Graphics). Dans ces conditions, les détecteurs de regard ont leur place dans la tendance actuelle.

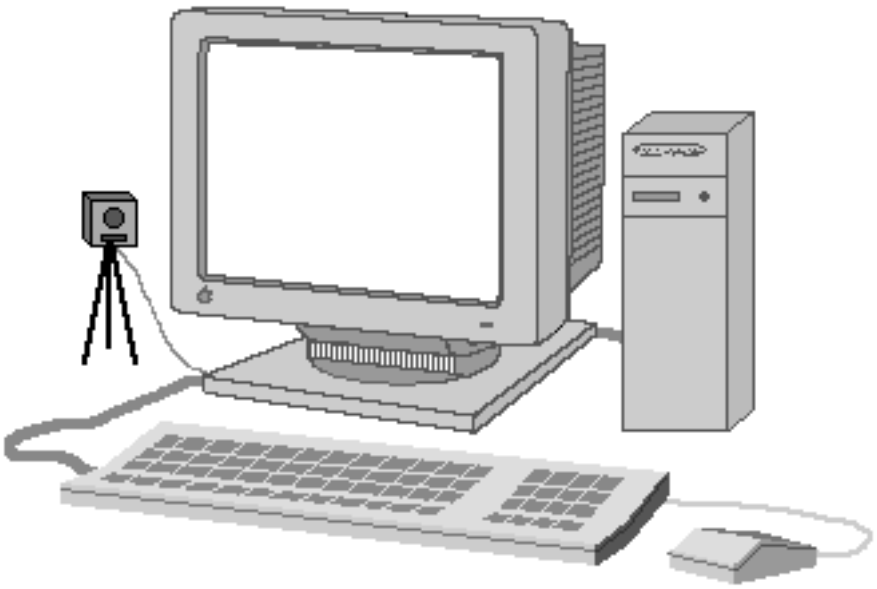


Figure 1.8 : espace de travail classique avec une station de travail équipée d'une carte d'acquisition vidéo, d'une caméra, et d'un micro.

Le choix de la caméra comme composant directeur conduit naturellement à l'étude des techniques de vision par ordinateur. Certaines d'entre elles sont présentées au chapitre suivant.

Chapitre 2
Survol des techniques

Ayant identifié les caractéristiques du système que nous recherchons avec ses limites et ses contraintes, nous allons explorer les solutions pratiques proposées actuellement. Nous nous intéresserons aussi bien aux dispositifs portés par l'utilisateur qu'aux techniques moins encombrantes. Parmi ces systèmes, certains sont déjà commercialisés et utilisés dans des domaines bien spécifiques et notamment pour l'assistance au pilotage d'avions ([Jacob93]), les études de marketing (déterminer ce qui attire l'œil dans une affiche publicitaire) ou encore l'assistance aux handicapés moteur pour la saisie de texte sur ordinateur.

Nous commencerons par les configurations matérielles et leurs problèmes. Ensuite, nous passerons en revue cinq techniques de vision par ordinateur : les EigenEyes, les gabarits déformables, les images d'orientation locale, la corrélation des laplaciennes multi-échelles, et le perceptron multicouche. Nous conclurons en indiquant les approches les mieux adaptées à notre problème

2.1- Les configurations matérielles et leurs problèmes

2.1.1- Méthode avec appareillage

Les systèmes les plus efficaces, en termes de vitesse et de précision, sont intrusifs du point de vue de l'utilisateur : ils nécessitent le port d'un casque ou de lunettes équipées d'une ou plusieurs diodes infrarouges chargées de "suivre" les mouvements de la pupille.

2.1.1.1- Méthode avec casque

La solution la plus radicale est de placer le détecteur de regard, la caméra et l'écran (un écran LCD) dans le casque. Les repères de ces trois composants étant confondus, on évacue le problème des mouvements de l'utilisateur. La procédure de calibrage consiste simplement à paramètrer la relation qui lie les trois repères : on enregistre la position de la pupille lorsque l'utilisateur regarde le point supérieur gauche et le coin inférieur droit de l'écran. Pour la reconnaissance, la position (x,y) de la pupille est recherchée dans l'image de l'œil. Celle-ci est obtenue automatiquement étant donné que la caméra reste fixe par rapport à l'œil. Ce point est mis en relation linéairement avec les points extrêmes. On en déduit directement la position recherchée.

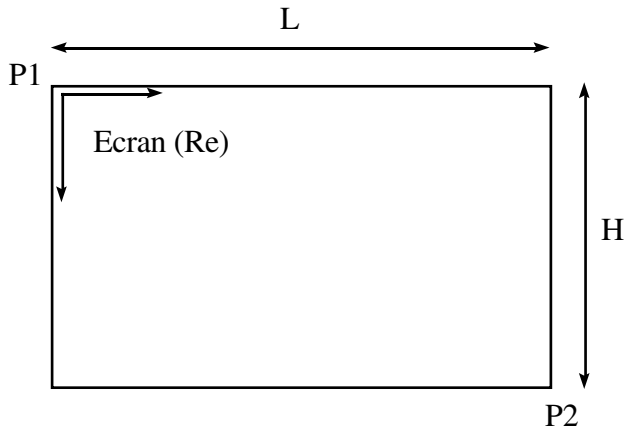


Figure 2.1 : le repère écran R_e et ses points extrêmes P_1 et P_2 exprimés dans le repère de l'œil R_o.

La figure 2.1 donne un exemple : les points $P_1(x_1,y_1)$ et $P_2(x_2,y_2)$ ont leurs coordonnées exprimées dans le repère de l'œil R_o. L et H désignent respectivement la longueur et la hauteur de l'écran. Lorsque qu'une nouvelle position P de la pupille est obtenue, ses coordonnées (x_e,y_e) dans le repère écran R_e sont obtenues par la simple relation suivante :

$$\begin{cases} x_e = (x - x_1).\dfrac{L}{(x_2 - x_1)} \\ y_e = (y - y_1).\dfrac{H}{(y_2 - y_1)} \end{cases} \qquad \textbf{(2.1)}$$

2.1.1.2- Méthode avec “lunettes”

Avec des lunettes, une diode infrarouge fixée sur une des lunettes éclaire l'œil : les repères de l’œil et de la diode sont donc liés. Le reflet de la diode permet de repérer plus facilement l'œil sur une image provenant de la caméra. Un filtre infrarouge, par exemple, permet d'obtenir directement la position de l'œil. La recherche de la pupille se limite alors à une fenêtre entourant l'œil (figure 2.2).

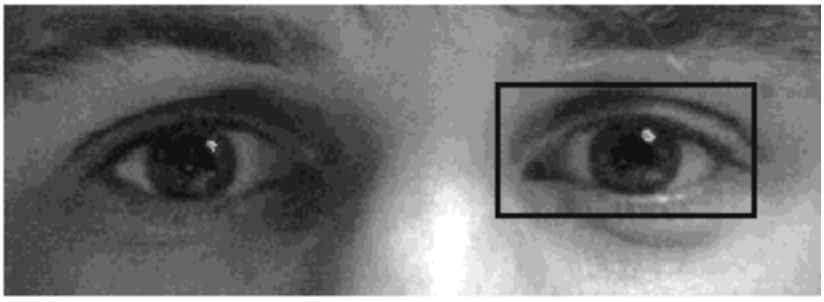


Figure 2.2 : une partie de l’image acquise avec la caméra, la fenêtre de recherche pour la pupille est obtenue grâce à la position du reflet infrarouge dans l’œil

Le problème est alors de lier le repère écran avec celui de l'œil. La relation (2.1) n'est plus applicable sauf si l'utilisateur ne bouge pas la tête. Considérons un modèle simplifié où l'utilisateur peut se déplacer dans un plan, P, parallèle à celui de l'écran (figure 2.3). La position du reflet de la diode permet de connaître le décalage (dx, dy) de la tête par rapport à la position “origine” de la tête centrée sur l'origine de l'écran.

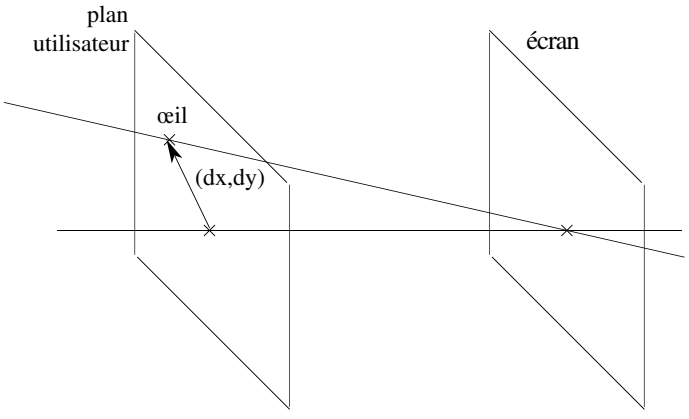


Figure 2.3 : Position relative de l'utilisateur dans un plan parallèle à l'écran.

On constate que si l'utilisateur bouge la tête dans le plan P en regardant le même point de l'écran la relation qui lie la position de la pupille et celle de la tête est linéaire. La

relation 2.1 peut alors être étendue en intégrant un terme correcteur linéaire, fonction du déplacement relatif de la tête :

$$\begin{cases} x_e = \alpha.dx + (x - x_1).\dfrac{L}{(x_2 - x_1)} \\ y_e = \gamma.dy + (y - y_1).\dfrac{H}{(y_2 - y_1)} \end{cases} \qquad (2.2)$$

Pour plus de précaution, le coefficient de correction est différent selon l'axe des coordonnées x et y. α et γ peuvent être obtenus par une calibration préalable du système en demandant à l'utilisateur de regarder un même point de l'écran sous deux angles différents.

Lorsque l'on bouge le corps en regardant un même point de l'écran, on a tendance à compenser ce déplacement en bougeant la tête. On peut alors affiner la correction en calculant la valeur de α et γ localement, en fonction de dx et dy. On aura alors la position observée suivante :

$$\begin{cases} x_e = \alpha(dx).dx + (x - x_1).\dfrac{L}{(x_2 - x_1)} \\ y_e = \gamma(dy).dy + (y - y_1).\dfrac{H}{(y_2 - y_1)} \end{cases}$$

Ayant étudié quelques techniques contraignantes, nous considérons maintenant le cas où l'utilisateur ne porte pas de dispositif particulier.

2.1.2- Méthode "écologique"

La figure 2.4 présente un montage intéressant qui laisse l'utilisateur libre de ses mouvements : celui que Jacob a utilisé pour les expériences décrites au chapitre 1. Afin de conserver un reflet infrarouge fixe dans l'œil, un miroir chargé de réfléchir la source de lumière dans l'œil, est piloté par un servomoteur qui suit les mouvements de l'utilisateur. Ainsi les repères associés à l'œil et au reflet de la diode restent-ils liés lors du mouvement. Le problème se ramène alors à celui des lunettes. Cependant un tel système est assez complexe : il faut piloter le miroir, l'appareillage est assez lourd et bruyant et l'utilisateur a la désagréable sensation d'être observé.

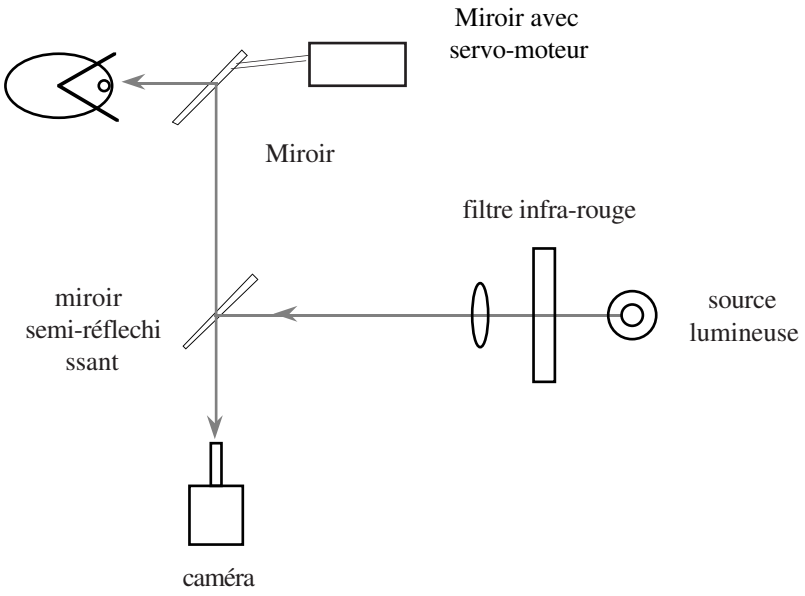


Figure 2.4 : configuration matérielle de l'"Applied Science Laboratory Model 3250R eye tracker", d'après [Jacob90].

La configuration matérielle que nous aimerions utiliser est similaire au montage de Jacob. Toutefois, la diode et la caméra sont fixes (leurs repères sont liés). En conséquence, la position du reflet de la diode dans l'œil varie si l'utilisateur bouge trop la tête. Le problème revient à relier tous les repères. Nous avons fait ce choix pour plusieurs raisons : nous ne disposions pas du matériel nécessaire (notamment le système de miroir), et il nous semble que cet appareillage sort des conditions d'utilisations que nous avons énoncés au chapitre 1 (matériel simple et non intrusif).

Ayant étudié quelques configurations matérielles possibles, nous abordons les techniques de vision par ordinateur proprement dites.

2.2- "EigenEyes"

Cette technique mise au point par [Turk91], a été utilisée à l'origine pour l'identification des visages. On parle alors de "eigenfaces". Elle a depuis trouvé de nombreuses applications pour d'autres formes d'identification : les "EigenPeople", les "EigenCars", les "EigenText" etc. Nous pensons qu'elle peut être adaptée aussi bien pour la recherche des yeux dans une image que pour la détermination de la direction du regard. Étudions son principe et son application à la recherche de l'œil.

2.2.1- Principe

L'idée principale de cette technique est de calculer les composantes principales d'une distribution d'images. Prenons par exemple, un ensemble de points dans un espace de dimension trois (figure 2.5.a). Si ces points sont à peu près dans un même plan de cet espace, on peut exprimer chacun d'entre eux dans cet espace plan plus petit (figure 2.5.b). On a ainsi exprimé les coordonnées des points dans l'espace le plus représentatif de l'ensemble de points donnés (figure 2.5.c). Quand on prend un nouveau point, cet espace permet d'obtenir deux types d'informations : d'une part, il est possible de savoir si ce point est "proche" de l'espace des points sur lequel on travaille, et d'autre part de quel point de notre espace de départ le nouveau point est le plus proche.

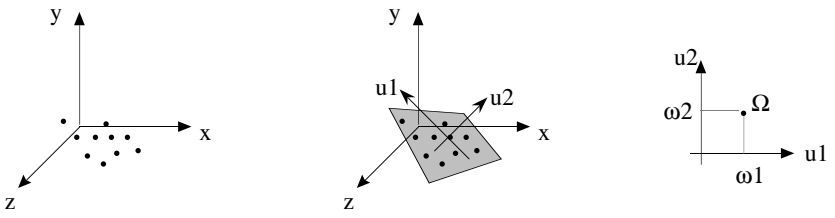


Figure 2.5 : (a) distribution de points dans un espace à 3 dimensions, (b) plan $(\vec{u}_1, \vec{u}_2)$ le plus représentatif de la distribution, (c) projection d'un point dans le sous-espace $(\vec{u}_1, \vec{u}_2)$.

En fait pour nous, les points seront exprimés, non pas dans un espace à trois dimensions, mais dans un espace de la taille du nombre de pixels formant une image de l'œil. Par exemple, pour des images de taille 100x200, on aura un espace de dimension 20000. Notre ensemble de départ contenant M images fortement corrélées de l'œil, nous allons le réduire à un sous-espace E_S de taille M-1.

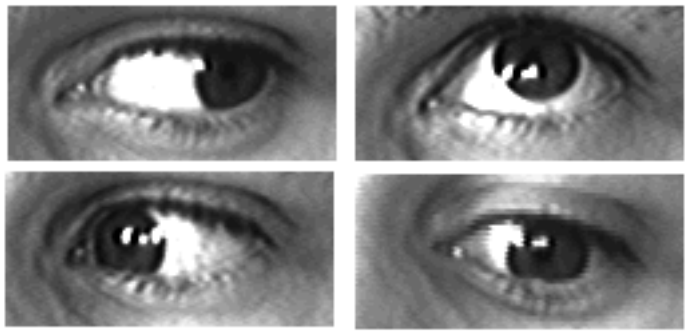
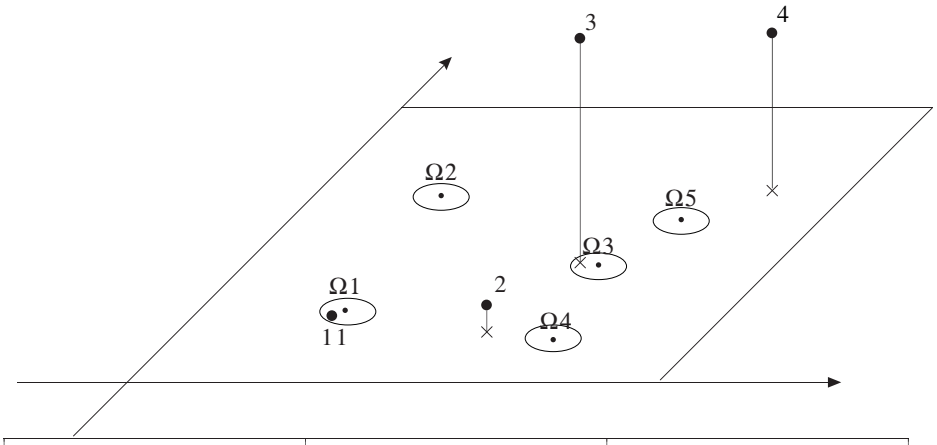


Figure 2.6 : distribution possible d'images de l'œil dans quatre positions extrêmes.

Par exemple notre ensemble de départ peut être constitué d'images de l'œil avec la pupille dans des positions extrêmes (figure 2.6). L'espace créé sera l'espace des "représentations de l'œil". Lorsque l'on présente une nouvelle image, on va calculer une première distance de cette image à l'espace E_s. Si elle est trop élevée on en déduira qu'il ne s'agit pas d'un œil. Par contre, s'il s'agit d'un œil, on calculera la distance de sa projection dans E_s avec les différents points de E_s. On aura ainsi un ensemble de distances nous indiquant où situer notre image par rapport aux autres. Dans notre cas on cherchera à savoir quelle est la position de la pupille la plus proche de notre nouvelle image (figure 2.7).



	distance à l'espace	proche position reconnue	résultat
1	proche	oui	correspond à la position Ω_1
2	proche	non	ce n'est pas une position connue
3	éloigné	oui	est-ce un œil ?
4	éloigné	non	ce n'est pas un œil

Figure 2.7 : interprétation des différents cas de figure de distances ; les Ω_i sont les projetés des images de la distribution de départ ; les points i correspondent à de nouvelles images à reconnaître.

Nous allons voir brièvement comment un tel espace se construit et comment s'en servir pour la reconnaissance.

2.2.2- Construction de l'"EigenSpace"

On recherche l'espace vectoriel dont la base est obtenue à partir des vecteurs propres de la matrice C de covariance associée à la distribution D des M images $\phi_1, \phi_2, \ldots, \phi_M$:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^t$$
$$= \frac{1}{M}AA^t \text{ avec } A = \left[\Phi_1\Phi_2\ldots\Phi_M\right]$$

avec $\Phi_i = \phi_i - \psi$ où ψ est l'image moyenne : $\psi = \frac{1}{M}\sum_{n=1}^{M}\phi_n$.

On gardera les M' = M-1 vecteurs propres u_k de C ayant une valeur propre maximale. En pratique, pour M grand, il est souvent suffisant de ne garder que M' (M' << M) vecteurs. Il suffit d'additionner les valeurs propres pour évaluer combien de vecteurs sont suffisants (par exemple pour arriver à une contribution globale de 90%).

2.2.3- Présentation d'une nouvelle image

On calcule Ω l'image projetée (en fait c'est un point) de la nouvelle image ϕ dans notre espace E_s :

$$\Omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \ldots \\ \omega_{M'} \end{pmatrix} \text{ où } \omega_k = u_k^t(\phi - \psi)$$

Les valeurs ω_k sont les contributions de chaque image u_k de la base de E_s pour représenter ϕ dans l'espace E_s.

2.2.4- Recherche de la direction du regard

Pour chaque image Ω_k de référence on calcule la distance euclidienne ε_k, de Ω à u_k :

$$\varepsilon_k = \left\|\Omega - u_k\right\|^2$$

L'ensemble de ces distances doit nous permettre de situer la position de la pupille dans l'œil par rapport aux positions de références. Idéalement, la position de l'œil devrait correspondre à l'intersection de boules B_k de dimension M', de rayon ε_k et centrées sur la position correspondant à l'image ϕ_k. Pratiquement, on gardera la position k telle que ε_k soit minimale.

2.2.5- Recherche de l'œil sur une image

Un autre intérêt de la méthode des EigenEyes est qu'elle permet de savoir si une image donnée est un œil ou non. Pour cela on calcule la distance β de l'image présentée ϕ à l'espace E_s :

$$\beta = \left\| (\phi - \psi) - \Phi_f \right\|^2$$

où ψ est toujours l'image moyenne et Φ_f la projection de ϕ dans E_s : $\Phi_f = \sum_{i=1}^{i=M'} \omega_k . u_k$

Si β est supérieur à un seuil limite Θ fixé, on peut considérer que l'image proposée n'est pas un œil.

Si l'on étend cette idée, on peut l'appliquer à la recherche d'un œil dans une image I plus grande. On calcule la distance $\beta(x,y)$ sur une fenêtre de la taille de l'œil que l'on déplace sur I. Cette distance sera maximale lorsque la fenêtre passera sur l'œil (figure 2.8)

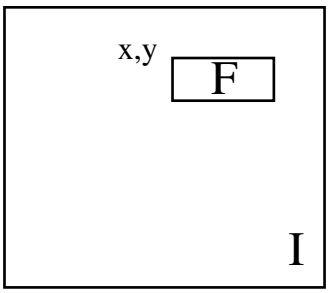


Figure 2.8 : fenêtre, F, de recherche d'un motif dans une image I plus grande.

2.2.6- Conclusion

La technique des EigenFaces a montré son efficacité pour l'identification des visages (voir [Pentland92, Pentland93, Pentland94, Essa94a, Essa94b et Bérard93]). Bien qu'elle semble envisageable pour identifier la position de la pupille dans l'œil, des problèmes comme les mouvements de la tête par rapport à la caméra risquent de compliquer l'interprétation des distances ε_k pour calculer la direction du regard. L'utilisation conjointe d'un reflet spéculaire sur la pupille et de classes d'images de références, au lieu d'une seule image par position, peut améliorer la fiabilité et la souplesse du système. Une expérimentation poussée est à faire. Les calculs nécessaires

pour la détection d'un œil dans une image sont hélas trop lourd pour avoir être utilisés en temps réel.

2.3- Gabarits déformables ("Deformable templates")

2.3.1- Principe

Yuille [Yuille91] propose un modèle géométrique pour la recherche de caractéristiques dans le visage (par exemple l'œil, le nez et la bouche). Ce modèle est déformé pour "coller" à l'image. Une méthode itérative de mise en correspondance permet de trouver un exemplaire de ce modèle dans l'image.

Pour l'œil, le gabarit proposé par Yuille est donné dans la figure 2.9.

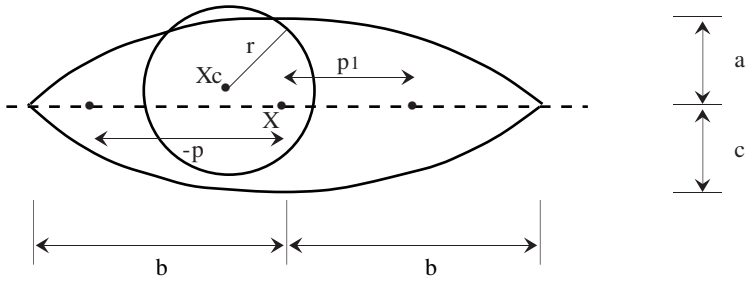


Figure 2.9 : modèle de l'œil, d'après [Yuille91].

Pour une image donnée I, Yuille considère trois interprétations de I correspondant chacune à un traitement sur l'intensité des points de I :

- l'image des creux (valleys) : $\Phi_v(x) = -I(x)$
- celle des contours (edges) : $\Phi_e(x) = \nabla I(x) \nabla I(x)$
- et celle des pics (peaks) : $\Phi_p(x) = I(x)$

Après expérimentation, Yuille a retenu 9 paramètres pour le modèle de l'œil :

- le centre de la pupille $\mathbf{X_c}$, le rayon **r** de la pupille. L'intérieur de ce cercle est attiré par les zones de faible intensité lumineuse alors que le blanc de l'œil est attiré par les zones de fortes intensités,

- deux sections paraboliques pour le contour de l'œil, de centre $\mathbf{X_e}$, d'une largeur **2b** et respectivement de hauteur **a** et **c** avec l'orientation Θ. Ces contours paraboliques sont attirés par les zones de contrastes (Φ_e élevé),

- les deux points au centre du blanc de l'œil séparé par la pupille. Ils sont attirés par les fortes intensités lumineuses et ont pour coordonnées : $X_e+p_1(\cos\Theta, \sin\Theta)$ et $X_e+p_2(\cos\Theta,\sin\Theta)$.

On obtient ainsi un vecteur de paramètre à 9 composantes : [X_c r X_e b a c Θ p_1 p_2]. Pour évaluer la qualité d'une mise en correspondance, on calcule la fonction d'énergie de l'image:

$$E = E_v + E_e + E_i + E_p + E_{interne}$$

Où

$$E_v = \frac{-c_1}{A_c}.\int\int_{A_c} \Phi_v(x).dA$$

$$E_e = \frac{-c_2}{L_c}.\int_{L_c} \Phi_e(x).ds - \frac{-c_3}{L_p}.\int_{L_p} \Phi_e(x).ds$$

$$E_i = \frac{-c_5}{A_b}.\int\int_{A_b} \Phi_p(x).dA$$

$$E_p = c_6(\Phi(X_e + p_1(\cos(\Theta),\sin(\Theta)) + \Phi(X_e + p_2(\cos(\Theta),\sin(\Theta))))$$

$$E_{interne} = \frac{k_1}{2}(X_e - X_c)^2 + \frac{k_2}{2}(p_1 - \frac{1}{2}(r-b))^2 + \frac{k_2}{2}(p_1 + \frac{1}{2}(r+b))^2 + \frac{k_3}{2}(b-2r)^2$$

Avec A_c l'aire de la pupille, L_c le périmètre de la pupille, L_p le périmètre de l'œil et A_b l'aire de l'œil.

Les $\{c_i\}$ et $\{k_i\}$ sont des coefficients qui pondèrent les différentes composantes de la mise en correspondance. Ils permettent de diriger la recherche, par exemple commencer par chercher la pupille puis affiner la recherche sur le contour. Yuille propose de les fixer à l'initialisation aux valeurs {4000, 50, 50, 125, 150, 50} et {10, 1, 0.05}. Leurs modifications durant les itérations ne sont pas calculées. Elles résultent de l'expérimentation.

Le processus de mise en correspondance consiste en une boucle de descente de la fonction d'énergie, en changeant les valeurs des composantes du vecteur paramètres [Yuille88].

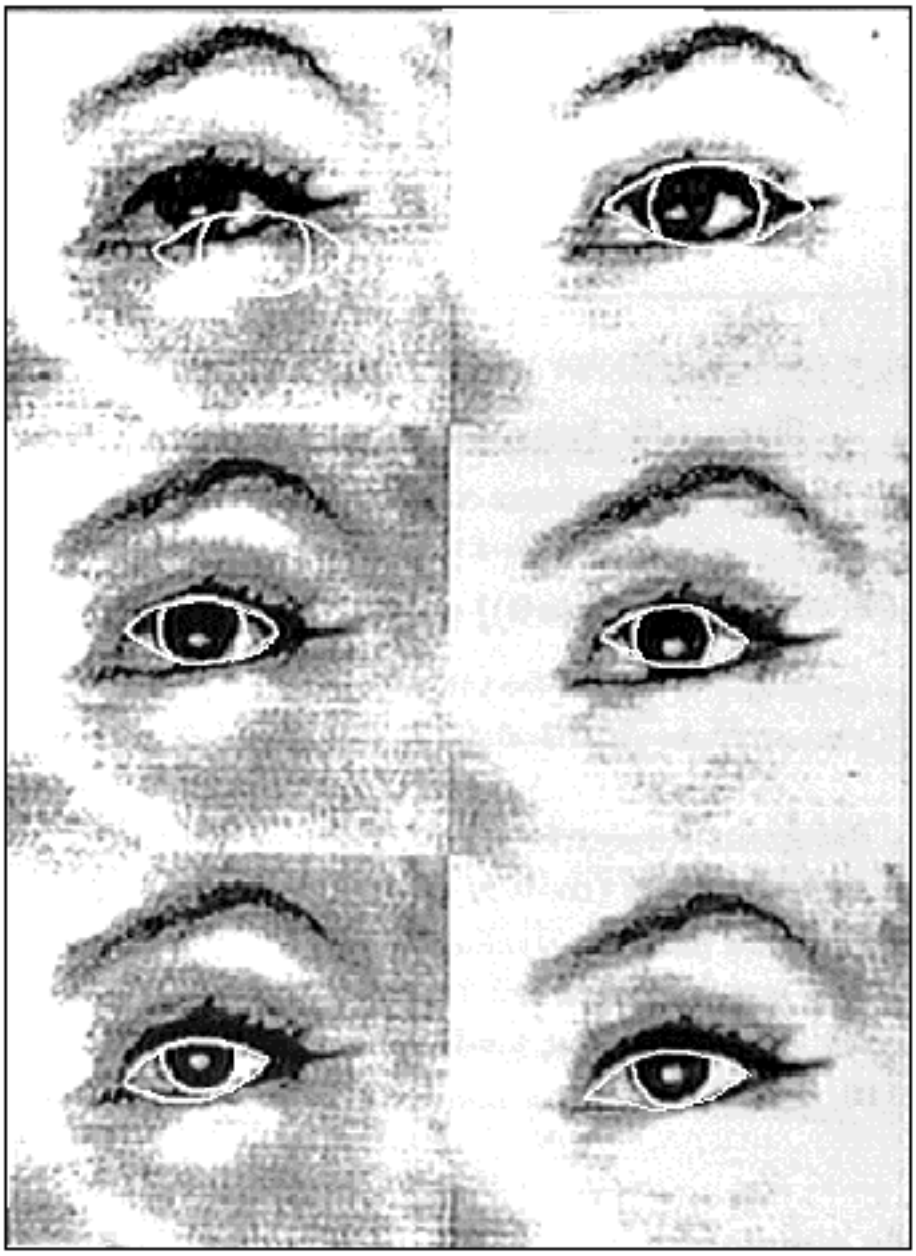


Figure 2.10 : Les étapes de l'application de l'algorithme de gauche à droite et de haut en bas ([Yuille88]).

2.3.2- Conclusion

Cette méthode a de nombreux avantages : elle fonctionne bien et permet à la fois la recherche de l'œil et la détermination de la position de la pupille dans l'œil. Elle est malheureusement très lente : il a fallu 10 minutes sur une station SUN4 pour faire une mise en correspondance précise de l'œil (cf. figure 2.10). De plus les conditions d'éclairage risquent de poser des problèmes, notamment pour faire la différenciation

œil/sourcil. Cette méthode nous semble incompatible avec notre objectif qui nécessite une réaction temps réel du système.

2.4- Images d'orientations locales

[Bichsel91] propose d'utiliser des images d'orientation locale au lieu d'images d'intensité. Elles présentent l'intérêt d'être invariantes aux variations de contraste et de luminosité. Une image d'orientation locale est obtenue à partir d'une image d'intensité en utilisant un filtre de détection de contours. Bichsel a choisi pour cela le filtre de Robert :

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{et} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Chaque point de l'image d'orientations locales est défini par un vecteur $\vec{e}_{i,j}$ de norme :

$$\left\| \vec{e}_{i,j} \right\| = \frac{(p_{i+1,j} - p_{i,j+1}, p_{i+1,j+1} - p_{i,j})}{\sqrt{(p_{i+1,j} - p_{i,j+1})^2 + (p_{i+1,j+1} - p_{i,j})^2}}$$

et d'angle :

$$\varphi_{i,j} = \arctan\left(\frac{p_{i+1,j+1} - p_{i,j}}{p_{i+1,j} - p_{i,j+1}} \right) + \frac{3.\pi}{4}$$

La figure 2.11 donne un exemple d'image obtenue après traitement.

Pour comparer deux images de tailles MxN, il suffit de réaliser la somme S de produits scalaires :

$$S = \sum_{i=1}^{M} \sum_{j=1}^{N} \vec{e}_{i,j1} . \vec{e}_{i,j2}$$

Pour la recherche d'un œil, on calcule au préalable une image moyenne d'orientation locale à partir d'une base recueillie chez plusieurs personnes. On calcule la somme S(x,y) en déplaçant une fenêtre de la taille de l'image moyenne sur l'image d'orientation locale provenant de la caméra. S(x,y) sera maximale sur l'œil.

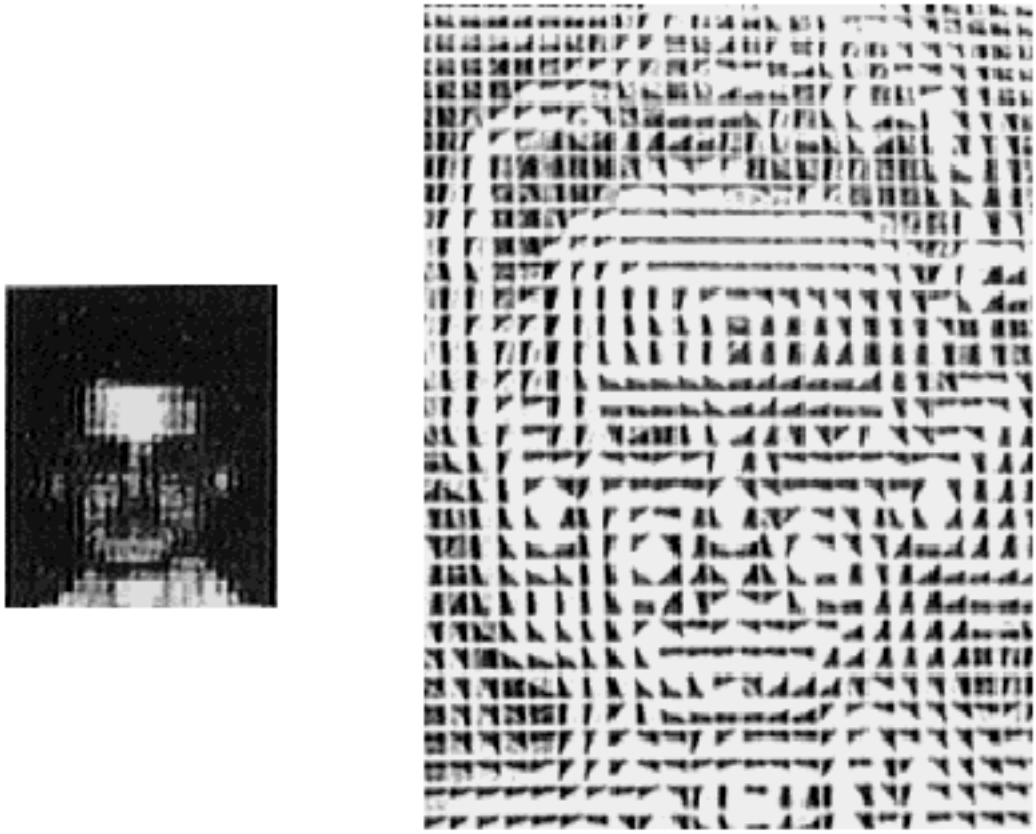


Figure 2.10 : à gauche, l'image originale, à droite, son image d'orientation locale (source inconnue d'après [Calvary91]).

Cette méthode présente l'intérêt de limiter l'influence du milieu extérieur, notamment les variations lumineuses, sur la recherche dans les images. Cependant elle amène un surcoût de calculs (calcul de l'image d'orientations locales) qui peut être compensé par l'utilisation d'un processeur spécialisé dans le traitement du signal du type DSP.

2.5- Corrélation des laplaciennes multi-échelles

Le traitement des images est souvent lourd en raison de la grande taille des images utilisées (le coût de la plupart des traitements croît proportionnellement à la taille des images) et du bruit inévitable dû à la numérisation.

Burt a proposé d'utiliser, pour la compression d'images (cf. [Burt83]), des opérateurs de convolutions et le sous-échantillonnage (pour réduire la taille des images). Une autre application est la réduction des images à diverses échelles. Nous allons voir rapidement

comment construire ces images à différentes résolutions, ce que Burt appelle la pyramide gaussienne.

2.5.1- La pyramide gaussienne

La pyramide gaussienne est décrite dans [Burt83]. Elle repose sur l'application répétée de l'opérateur de convolution sur l'image de départ G_0 avec un filtre Gaussien discret. Le filtre 2D proposé par Burt est le filtre Gaussien h suivant :

$$h = \frac{1}{400} \times \begin{bmatrix} 1 & 5 & 8 & 5 & 1 \\ 5 & 25 & 40 & 25 & 5 \\ 8 & 40 & 64 & 40 & 8 \\ 5 & 25 & 40 & 25 & 5 \\ 1 & 5 & 8 & 5 & 1 \end{bmatrix}$$

L'image G_i de la pyramide gaussienne est fonction de l'image G_{i-1} :

$$G_i = h \otimes G_{i-1}$$

Le sous-échantillonnage revient à prendre un point de l'image sur quatre en fonction de ses voisins. A chaque itération, la taille de l'image va être divisée par 4 :

$$G_i(x,y) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} h(m,n) \times G_{i-1}(2x+m, 2y+n)$$

où la hauteur et la largeur de l'image G_i sont deux fois plus petites que celles de G_{i-1}.

Un exemple de pyramide gaussienne à quatre niveaux est donnée dans la figure 2.12.

Figure 2.12 : pyramide gaussienne de niveau 0, 1, 2, 3 et 4.

2.5.2- Recherche de l'œil

Bichsel a utilisé cette technique en complément de sa méthode de reconnaissance, basée sur la mise en correspondance d'images d'orientations locales (voir 2.4) pour faire une recherche ascendante des caractéristiques du visage (cf. [Bichsel91]) :

- on commence par une recherche grossière de la tête à faible résolution, ce qui permet de réduire la fenêtre de recherche,
- on situe approximativement l'œil dans le visage obtenu à une résolution supérieure,
- on réitère le processus jusqu'à ce que l'on ait une fenêtre de l'œil à la résolution de départ. Chaque traitement ultérieur se limitera à cette fenêtre, réduisant ainsi sensiblement les calculs.

Il a également proposé une autre méthode. Tout d'abord il cherche grossièrement le contour de la tête sur une image de niveau 7 de la pyramide en extrayant 3 points correspondant à trois transitions clair/sombre : une horizontale (haut du visage) et les deux autres verticales pour les côtés du visage. Puis il cherche le nez au niveau 3 comme étant un objet clair, vertical, allongé et bordé de zones sombres. Enfin, il repère les yeux aux niveaux 1 et 2 en cherchant des objets sombres semi-circulaires.

A chaque niveau de recherche des contraintes de positionnement permettent de limiter la recherche à une zone de plus en plus petite.

On peut même envisager de faire la recherche de la pupille dans le blanc de l'œil directement sur la fenêtre obtenue. Pour la détection de la pupille et du regard, il est possible d'éclairer le visage avec une lampe (infrarouge par exemple), de repérer son reflet comme une zone claire entourée d'une zone sombre, et la pupille comme une zone sombre dans une zone claire (le blanc de l'œil).

2.5.3- Conclusion

Le principe de la pyramide semble intéressant pour limiter les calculs, même si cette méthode n'est pas nécessairement adaptée pour accomplir l'intégralité du travail : recherche de l'œil et détermination précise de la position de la pupille. On peut la retenir pour la détection de l'œil et appliquer d'autres méthodes sur la fenêtre obtenue. Là encore, l'utilisation d'une lampe fixe se reflétant sur la pupille de l'œil (on pourrait la repérer comme une zone claire entourée d'une zone sombre) peut nous aider à évaluer la position de l'utilisateur par rapport à l'écran.

Il faut noter qu'il existe des algorithmes très rapides de calcul de la pyramide. Certains processeurs sont même spécialisés dans de tels calculs (ex : DSP : Digital Signal Processor).

2.6- Perceptron multicouche

La détermination de la direction du regard peut être vue comme un processus de reconnaissance : il s'agit de reconnaître une configuration de l'œil correspondant à la direction d. On peut avoir préalablement enregistré une série S de couples {image de l'œil, direction du regard} en déplaçant, par exemple, un curseur à l'écran suivi par les yeux de l'utilisateur et en enregistrant une image de l'œil par position du curseur {I,(x,y)}. A la présentation d'une nouvelle image I', on cherchera l'image I la plus proche appartenant à la base S. L'utilisation d'un réseau de neurones est une technique classique de reconnaissance. Il semble naturel de l'envisager pour notre problème. Dean Pomerleau a développé un système de conduite automatique de véhicule reposant sur un réseau de neurones nommé MANIAC [Pomerleau93]. Il l'a adapté la détection du regard [Pomerleau94]. Les résultats obtenus sont encourageants.

2.6.1- Architecture du réseau

Le réseau utilisé (ANN) est un perceptron multicouche. La topologie du réseau retenue contient une seule couche intermédiaire coupée en deux groupes de 8 unités ; la couche de sortie est composée de deux groupes de 50 unités : l'un pour la coordonnée x, le second pour la coordonnée y (figure 2.13). La rétine d'entrée correspond à l'image à traiter. Pomerleau et son équipe ont effectué des essais avec des fenêtres de l'œil de tailles différentes. Les meilleurs résultats ont été obtenus avec une fenêtre de taille 15 x 40.

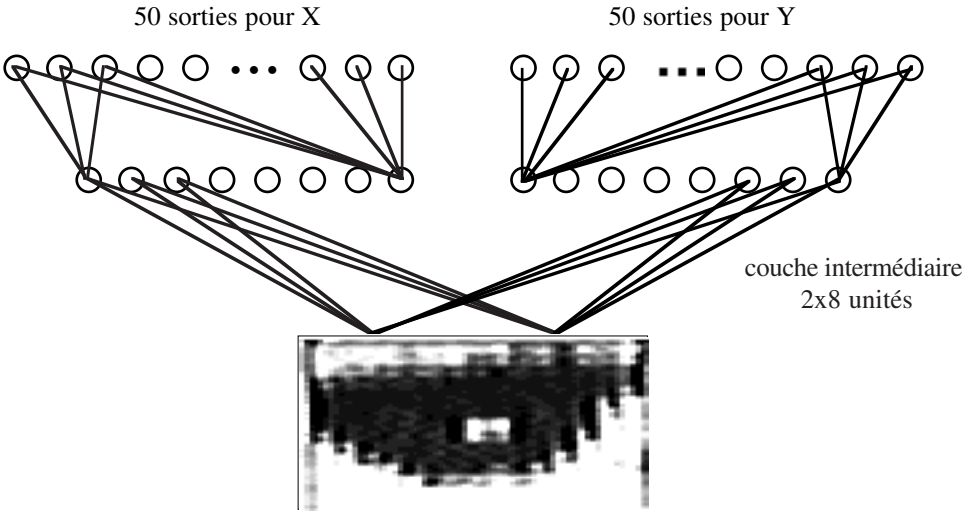


Figure 2.13 : structure du réseau ANN.

Un système astucieux a été retenu pour la représentation des coordonnées X et Y en sortie. Chacune des coordonnées est représentée par une courbe gaussienne discrète centrée sur la valeur de la coordonnée (cf. figure 2.14).

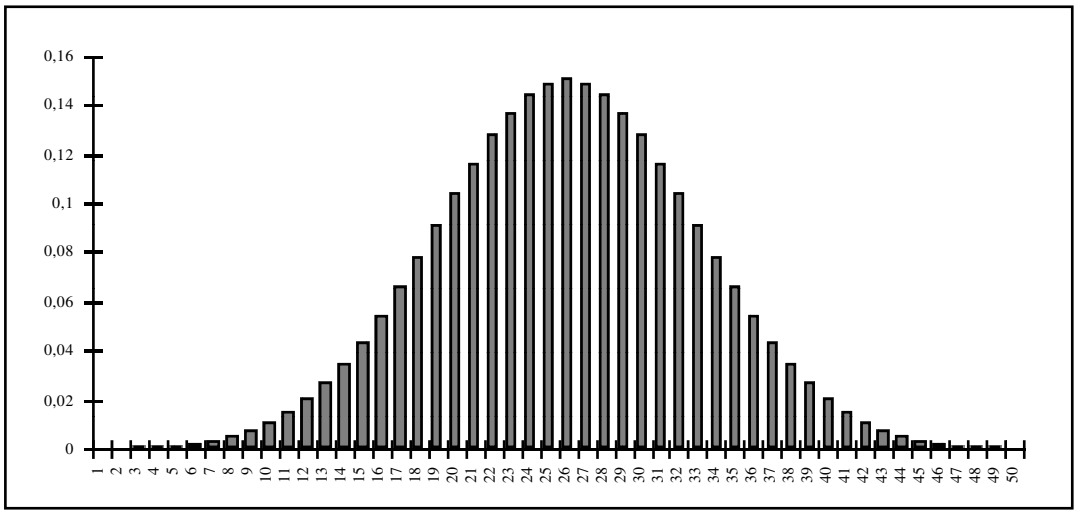


Figure 2.14 : type de réponse de la couche de sortie pour une coordonnée.

L'étude détaillée de cette représentation est développée au chapitre 6.

2.6.2- Processus d'apprentissage

Pour entraîner le réseau, il faut en premier lieu collecter une série S d'images I_i correspondant à l'observation de points (x_i,y_i) de l'écran. Une fois ce travail accompli, on met des poids aléatoires sur chacune des connexions du réseau puis pour chacun des couples $\{I_i, (x_i,y_i)\}$ de S :

- on présente l'image I_i en entrée (sur la rétine) du réseau,
- on place la gaussienne discrète centrée sur x_i et y_i sur les couches de sortie SX et SY, correspondants aux sorties désirées,
- puis on lance un algorithme de rétro-propagation du gradient classique ([Davalo89] ou [Carling92]) jusqu'à ce que la sortie obtenue soit suffisamment conforme à la sortie désirée.

2.6.3- Reconnaissance

Pour la phase de tests, on présente la nouvelle image I sur la rétine du réseau. L'algorithme de rétro-propagation du gradient est lancé ; une fois que le signal a été propagé de la rétine aux couches de sortie, on peut lire sur les couches de sortie une courbe gaussienne centrée sur la valeur en X et Y reconnues. On applique alors un algorithme de recherche de gaussienne à partir d'un modèle pour en déduire les coordonnées (x,y) de la position de l'écran observée.

2.6.4- Conclusion

Actuellement, les réseaux de neurones font l'objet de recherches actives. Ils ont montré leurs capacités dans des domaines comme la reconnaissance d'écriture (le Newton d'Apple en est un exemple) ou l'OCR. Le système de conduite automatique MANIAC est une réussite. Pomerleau a su l'adapter sans trop de modifications à son "eye tracker". Les résultats obtenus sont très intéressants : Pomerleau annonce, pour son système, une précision de l'ordre de 2,1°, soit environ 1,5 cm à une distance de 43 cm de l'écran, et une vitesse de 15 Hz, ce qui semble suffisant dans le cadre d'une interface homme-machine (voir chapitre 1).

2.7- Les techniques retenues

Parmi les techniques que nous venons de décrire, certaines sont utilisables pour la recherche des yeux dans une image, d'autres pour la détermination de la direction du regard, d'autres encore devraient assurer les deux fonctions. Il convient d'écarter les techniques qui s'éloignent trop des contraintes de notre problème. Par exemple, la méthode des gabarits déformables de Yuille donne de bons résultats en terme de précision mais elle ne répond pas à notre contrainte d'efficacité (10 mn. sur une station de travail pour trouver un œil).

Au vu de l'analyse préliminaire qui vient d'être présentée, je retiens les approches suivantes : la techniques des EigenEyes pour détecter la direction du regard, la corrélation des laplaciennes pour repérer les yeux, et le système à base de réseaux de neurones pour la détection du regard.

Nous allons étudier au chapitre suivant comment organiser les tests de ces différentes méthodes en respectant des conditions d'évaluation similaires.

Chapitre 3
Évaluation

Maintenant que nous avons défini les contraintes de notre problème et énuméré les techniques que nous allons essayer, il nous faut préciser le cadre d'expérimentation, c'est l'objet de ce chapitre. Il décrit les objectifs et les choix qui ont été retenus, notamment en ce qui concerne les moyens d'évaluation.

Le but était de pouvoir tester les différentes techniques dans des conditions similaires. C'est pourquoi nous voulions un système unique pour toutes les méthodes, où l'on puisse changer de méthodes sans quitter le même cadre expérimental. D'autre part, on voulait pouvoir combiner les différentes méthodes toujours dans le même environnement, par exemple la corrélation pour la détection des yeux et le multiperceptron pour le regard. Pour pouvoir évaluer la précision du système de façon "incrémentale", on va utiliser d'une certaine manière la multirésolution en faisant des tests à basse résolution puis dans une résolution meilleure et ainsi de suite jusqu'à l'échec de la méthode testée ; des tests indépendants horizontalement et verticalement doivent permettre de vérifier si la précision obtenue est similaire dans les deux dimensions.

3.1- Les objectifs

Il faut tester les différentes méthodes et les comparer, les contraintes nous permettent de définir les critères d'évaluation à mesurer :

• la fiabilité, il faut déterminer si une technique est susceptible ou non d'indiquer qu'elle ne connaît pas la position du regard. Si elle renvoie de toute façon une réponse, même erronée, il faut voir si on ne peut pas faire une interprétation à posteriori des réponses pour détecter ces erreurs ;

• la précision, avec des tests de plus en plus fins ;

• la vitesse, et notamment vérifier si le temps de réponse des algorithmes est borné ou s'il risque de croître dans certains cas.

Tous ces tests seront effectués dans un cadre de travail classique, avec un éclairage ambiant naturel et un bureau tout ce qu'il y a de plus commun (sans fond uni derrière l'utilisateur par exemple).

3.2- Un environnement unique pour toutes les méthodes

Afin d'obtenir des résultats les plus comparables possibles il est important de tester toutes les méthodes dans des conditions les plus semblables possible. C'est pour cela que nous avons développé un environnement unique de tests pour toutes les techniques.

Le choix de la méthode pour la détection du regard est réalisé par une simple sélection dans un menu (cf. figure 3.1).

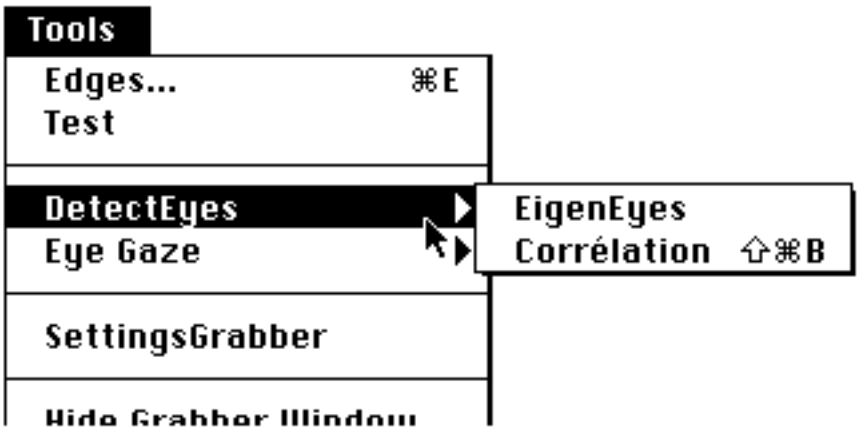


Figure 3.1 : Menu déroulant permettant de choisir la technique de détection des yeux.

Le choix de la méthode de détection du regard peut être sélectionnée de la même manière dans un autre menu (figure 3.2).

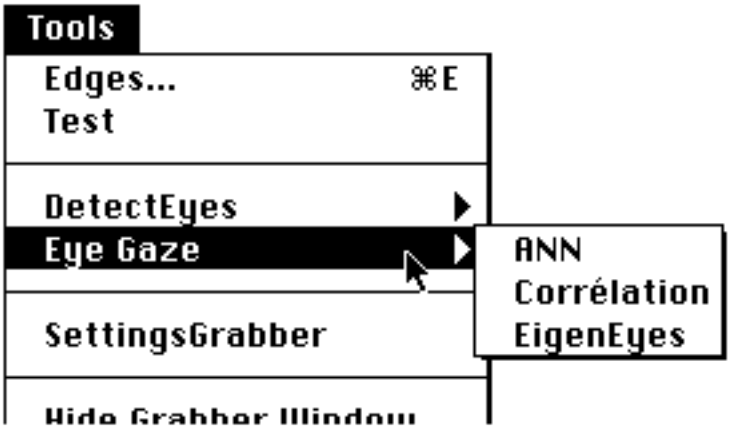


Figure 3.2 : Menu déroulant permettant de choisir la technique de détection du regard.

Cette méthode permet de rajouter assez facilement de nouvelles méthodes. Les entrées sorties des fonctions implémentant les méthodes sont similaires quelque soit la méthode pour en faciliter l'intégration.

3.3- Évaluation de la précision

Afin d'évaluer la précision des techniques, nous avons fait un système incrémental d'évaluation multirésolution. Dans un premier temps l'écran est partagé en 4 positions, on effectue des tests uniquement sur ces 4 parties (cf. figure 3.3a). Si les tests sont concluants on passe à 9 zones (3^2) (figure 3.3b) et ainsi de suite jusqu'à ce que la technique soit mise en défaut, on aura alors la résolution maximale que l'on peut obtenir avec cette méthode.

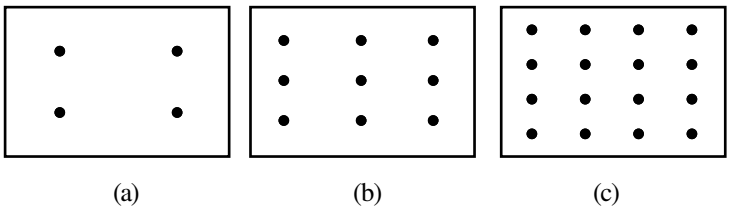


Figure 3.3 : multirésolution sur les zones l'écran à repérer pour l'évaluation de la précision des techniques.

Comme la détermination de la direction du regard est effectuée par un traitement de l'image, de part la forme de l'œil, sa largeur et sa hauteur, il me parait difficile d'obtenir une précision du même ordre selon les abscisses et les coordonnées. C'est pourquoi le mécanisme d'évaluation de la précision peut être effectué soit en vertical, soit en horizontal ou alors avec les deux combinés (cf. figure 3.4).

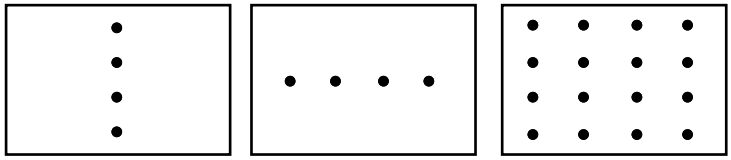


Figure 3.4 : évaluation de la précision, en coordonnées verticales, horizontales et les deux combinés.

Nous allons voir dans les trois chapitres suivants comment fonctionne plus précisément chacune des techniques retenues et comment nous les avons adaptées à notre problème.

Chapitre 4
Les "EigenEyes"

Cette technique, mise au point par [Turk91], a été utilisée au départ pour l'identification des visages, nous pensons qu'elle peut être adaptée pour la détermination de la direction du regard. L'idée principale (comme nous l'avons vu dans le paragraphe 2.2) est de calculer les composantes principales d'une distribution d'images D. On construit un espace vectoriel dont la base est obtenue à partir des vecteurs propres de la matrice de covariance associée à la distribution.

Dans notre cas la distribution de départ peut être composée d'images ϕ_k de l'œil. L'espace créé sera l'espace des "représentations de l'œil". Lorsque l'on présente une nouvelle image, le calcul de deux types de distances permet de savoir d'une part s'il s'agit ou non d'un œil et d'autre part de quelle position de référence la nouvelle image est la plus proche.

4.1- Construction de l'"EigenSpace"

Supposons que nous travaillons sur des images de taille N x N pixels, et que notre distribution de références D contiennent M images ϕ_1, ϕ_2, …, ϕ_M. La figure 4.1 donne un exemple d'images obtenues en demandant à l'utilisateur de regarder 9 points de l'écran (3x3). Elles formeront la base exemple D pour illustrer la technique tout au long de ce chapitre.

Chaque image peut être vue comme un vecteur de taille N^2, on peut les représenter comme des points dans un espace vectoriel E de taille N^2. On va chercher le sous-espace de E, E_S, de taille M-1 le plus représentatif de la distribution D de points.

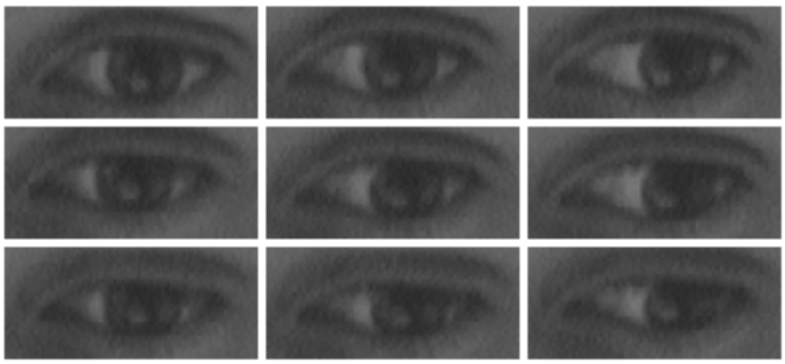


Figure 4.1 : distribution de 9 images de l'œil correspondants à l'observation de 9 positions de l'écran.

On calcule l'image moyenne :

$$\psi = \frac{1}{M}\sum_{n=1}^{M}\phi_n$$

L'image ψ obtenue à partir de la base représentée de la figure 4.1 est donnée dans la figure 4.2.

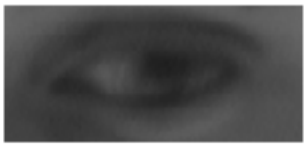


Figure 4.2 : image moyenne de la base D de la figure 4.1.

On en déduit les images Φ_k centrées sur ψ :

$$\Phi_i = \phi_i - \psi$$

La figure 4.3 donne les images Φ_k de la base D.

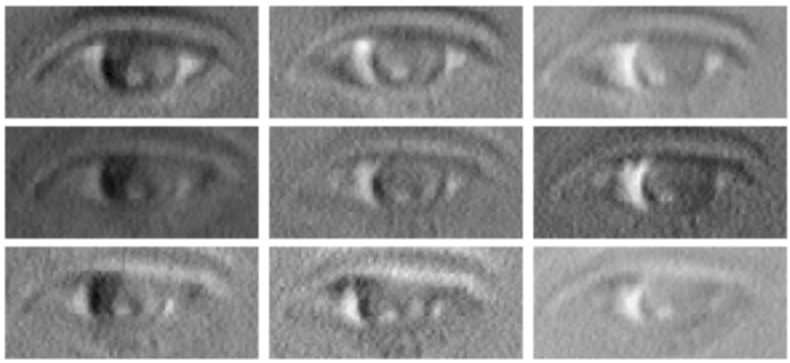


Figure 4.3 : ensemble des images Φ_k centrées sur ψ.

La base de l'espace Es est formée par les M-1 vecteurs propres orthogonaux et orthonormés u_k. Ces vecteurs u_k sont choisis tels que leur valeur propre associée λ_k soit maximale :

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^t \Phi_n)^2 \text{ maximale}$$

$$\text{et } u_l^t u_k = \begin{cases} 1 \text{ si l = k} \\ 0 \text{ sinon} \end{cases} \text{ (orthogonalité et orthonormalité)}$$

Les vecteurs u_k sont les M eigenvectors de la matrice de covariance C :

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n \Phi_n^t$$
$$= \frac{1}{M} AA^t \text{ avec } A = \left[\Phi_1 \Phi_2 ... \Phi_M\right]$$

Cette matrice est de taille N^2 x N^2, le calcul de ses vecteurs propres devient prohibitif. L'utilisation d'une astuce mathématique permet de réduire la recherche à une matrice MxM. En effet, les vecteurs propres u_i de C ayant pour valeur propre λ_i sont tels que :

$$AA^t . u_i = \lambda_i u_i \qquad (4.1)$$

En fait, il n'y a que M-1 vecteurs significatifs, les autres ont une valeur λ_i égale à 0, ils ont donc une contribution nulle dans notre espace E_s. Nous allons donc chercher les vecteurs propres d'une matrice MxM, en déduire la combinaison linéaire avec les vecteurs Φ_i pour en déduire les EigenVectors.

Considérons la matrice AA^t et ses couples, valeur propre, vecteur propre, (μ_i, v_i) :

$$A^t A.v_i = \mu_i v_i \qquad \textbf{(4.2)}$$

On multiplie (4.2) par A :

$$AA^t Av_i = \mu_i Av_i$$

soit

$$AA^t(Av_i) = \mu_i(Av_i)$$

d'après (4.1) on constate que les (Av_i) sont les vecteurs propres de $C = AA^t$, on va donc construire la matrice MxM, L, telle que :

$$L = A^t A$$

Soit :

$$L(m,n) = \Phi_m^t \Phi_n$$

On calcule les M vecteurs propres v_k de L. Ces vecteurs déterminent la combinaison linéaire des M images de la distribution de départ pour former les M vecteurs propres de C, u_k :

$$u_k = A.v_k$$

La figure 4.4 donne les 8 EigenVectors obtenus finalement à partir de D.

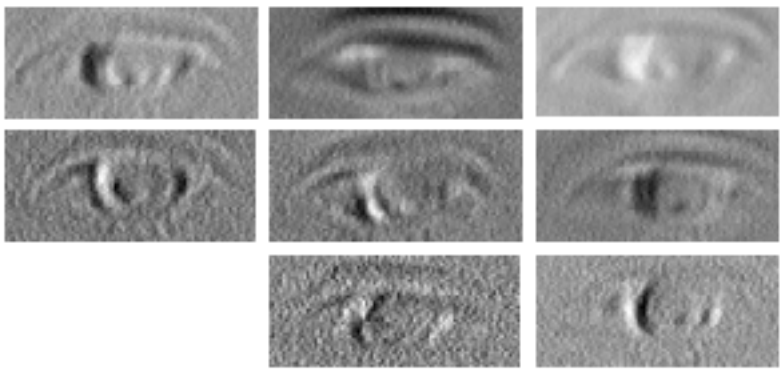


Figure 4.4 : EigenVectors obtenus à partir de la base D d'images de la figure 4.1. Ces vecteurs définissent notre sous-espace E_s, représentatifs des images de l'œil.

Pour M grand, il est souvent suffisant de ne garder que M' (M' << M) vecteurs pour obtenir une contribution des valeurs propres de l'ordre de 90% ($\sum_{k=1}^{M'} \lambda_k \approx 90\% \sum_{k=1}^{M} \lambda_k$). Dans notre cas, nous allons garder tous les vecteurs, la base D étant de taille réduite. Pentland a

utilisé les EigenFaces pour des bases de centaines d'images, il serait alors impensable de garder des centaines de vecteurs sachant que seuls les premiers vecteurs sont réellement significatifs.

Pour exprimer les images dans notre espace on les projette dans E_s, les vecteurs Ω_k, de taille M', sont les projetés de Φ_k dans E_s :

$$\Omega_k = \begin{pmatrix} \omega_{k1} \\ \omega_{k2} \\ \cdots \\ \omega_{kM'} \end{pmatrix} \text{ où } \omega_{kj} = u_j^t \Phi_k$$

Chaque ω_{kj} représente la contribution du vecteur u_j dans la projection de Φ_k dans E_s.

4.2- Présentation d'une nouvelle image

Lorsqu'une nouvelle image ϕ est proposée, on la centre sur ψ :

$$\Phi = \phi - \psi$$

et on la projette dans E_s :

$$\Omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_{M'} \end{pmatrix} \text{ où } \omega_k = u_k^t \Phi$$

Les valeurs ω_k sont les contributions de chaque image u_k de la base de E_s pour représenter ϕ dans l'espace E_s.

4.3- Recherche de la direction du regard

Pour chaque image ϕ_k de référence on calcule la distance euclidienne ε_k, de Ω à Ω_k :

$$\varepsilon_k = \left\| \Omega - \Omega_k \right\|^2$$

Si l'image projetée Ω est proche ($\varepsilon_j < \theta$) d'une des images Ω_j, alors on peut en déduire que l'utilisateur regarde dans la même direction que celle qu'il regardait lorsque l'image ϕ_j a été prise. Si l'image est "éloignée" de toutes les images Ω_k, alors il faut essayer d'interpréter l'ensemble des distances $\{\varepsilon_k\}$ pour déterminer la position observée. Si aucune image n'est suffisamment proche de Ω on en déduira que l'utilisateur ne regarde pas dans une des positions de notre échantillon. Cette caractéristique importante permet à notre programme d'indiquer qu'il n'a pas trouvé la direction du regard et donc que l'utilisateur regarde ailleurs. L'ensemble $\{\varepsilon_k\}$ pourra être utilisé pour extrapoler la position qu'il regarde.

4.4- Recherche de l'œil sur une image

Un autre intérêt de la méthode des EigenEyes est quelle permet de savoir si une image ϕ est une représentation d'un œil ou non, pour cela il suffit de calculer la distance qui la sépare de sa projection dans notre espace des yeux E_s. Si cette distance est trop élevée c'est que l'image doit être bien différente de celles qui ont servies à construire notre espace et donc que ce n'est probablement pas un œil.

On calcule donc la représentation de ϕ dans notre espace :

$$\Phi_f = \sum_{i=1}^{i=M'} \omega_k . u_k \text{ où } \omega_k = u_k^t(\phi - \psi)$$

et on calcule la différence β entre notre image centrée et sa représentation :

$$\beta = \left\| (\phi - \psi) - \Phi_f \right\|^2$$

Si β est supérieur à un seuil Θ, fixé on peut considérer que l'image proposée n'est pas un œil.

Il serait possible d'adapter ce système pour la recherche d'un œil dans une image. Supposons que l'on ait une image I plus grande que la taille des images ϕ_k et qu'elle contienne un œil. On va déplacer une fenêtre F de la taille de l'œil sur toute l'image I en calculant à chaque fois la distance $\beta(x,y)$ pour l'image couverte par la fenêtre. La position de l'œil correspondra à la fenêtre F(x,y) pour laquelle $\beta(x,y)$ est minimale.

4.5- Implémentation

Pour les tests, la taille de l'espace des yeux doit être adaptée à la précision que l'on désire tester. La base d'images utilisée pour la construction de l'espace doit être suffisamment grande pour pouvoir localiser précisément les nouvelles directions du regard dans cet espace. Cependant, plus l'espace est de grande dimension, plus il devient spécialisé, donc plus il limite sa capacité à reconnaitre de nouvelles positions ([Pentland93]) et notamment des positions prises sous un angle de vue légèrement différent. Une fois l'espace créé on projette une image par direction du regard, par exemple pour distinguer 4x4 zones, on réalise 16 photos de l'œil. Ce système facilite la reconnaissance, il suffit de choisir la position type la plus proche de l'image testée. Si nous avions utilisé que quelques positions, il aurait fallut interpréter les distances $\{\varepsilon_k\}$, ce qui n'est pas un problème trivial. L'inconvénient de ce système est que l'espace obtenu risque de devenir très grand en nombre d'éléments et en occupation mémoire. On peut compenser ce problème en prenant des images de plus petites tailles. Il faut faire un compromis entre vitesse, occupation mémoire et précision.

Enfin pour compenser les déplacements de la tête il serait utile d'effectuer des tests comparatifs sur des classes d'images d'une même position plutôt qu'avec un seul point par position ([Turk91, Pentland93]).

$$\varepsilon_k = \left\| \Omega - \Omega_k \right\|^2$$

Avec Ω_k la projection moyenne des yeux de la classe de la position k. On voit bien que si l'utilisateur bouge trop, la pupille bougera aussi et l'image moyenne seule ne permettra pas de distinguer la bonne position.

On pourrait aussi construire deux espaces distincts, un pour déterminer la coordonnée en x de la position observée et l'autre pour la coordonnée y. Cette séparation des dimensions faciliterai l'interprétation des distances $\{\varepsilon_k\}$.

Une autre caractéristique de cette technique est que son coût de calculs est borné, il est directement fonction du nombre de positions de référence, de la taille de l'EigenSpace et de la taille des images. Enfin le contrôle d'erreur est assez simple : si toutes les distances $\{\varepsilon_k\}$ de l'image de test aux positions de référence sont trop élevées on peut en conclure que l'utilisateur regarde, soit une position intermédiaire entre deux positions types, soit il ne regarde pas l'écran.

Nous n'avons pas utilisé l'algorithme de recherche de l'œil sur une image présenté au paragraphe 4.4. Son coût est beaucoup trop élevé. On trouvera une étude détaillée des calculs dans [Turk91]. Il serait toutefois intéressant d'effectuer des tests en multirésolution et en parallèle (avec une machine parallèle), en recherchant l'œil de la basse résolution à la résolution de départ en limitant la fenêtre de recherche à chaque étape.

Le gros problème de cette technique est la précision : si l'on utilise des images de l'œil trop petite il est impossible d'obtenir une précision suffisante, si les images sont plus grandes le stockage de l'espace pose un réel problème et les temps de calcul augmentent proportionnellement à la taille des images. Pour obtenir des images de meilleure résolution il serait nécessaire de mettre la caméra sur une tourelle qui suivrai l'œil de l'utilisateur durant ses mouvements pour qu'il ne sorte pas du champs de la caméra. Le problème se reporte alors sur le pilotage de la caméra.

Cette technique, bien qu'elle est prouvée son efficacité dans d'autres domaines d'identification, n'est peut-être pas la plus adaptée à notre problème, elle manque de souplesse par rapport à la méthode présentée au chapitre 6 notamment.

Chapitre 5

La corrélation des laplaciennes, multi-échelles

La recherche brute de caractéristiques du visage, comme les yeux, sur une image peut être une tâche coûteuse si elle est fonction de la taille de l'image. Sachant que dans ce cas une grande partie de l'image est explorée pour rien, on s'est intéressé à trouver des moyens pour restreindre la zone de travail à l'aide d'une recherche "descendante" et sélective. Cannon ([Cannon86]) a utilisé ce système pour l'extraction de traits : il fait une extraction de contours sur une image du gradient pour repérer grossièrement le visage ensuite il affine sa recherche sur les zones proches des contours. Bichsel ([Bichsel91]), lui, applique un principe un peu différent : il propose de faire une recherche grossière sur une image à faible résolution et de préciser la recherche au fur et à mesure qu'il remonte vers l'image initiale en se limitant à chaque fois à la fenêtre de recherche obtenue à l'étape précédente. On réduit ainsi la zone de travail à chaque étape et par conséquent les calculs à effectuer. Nous allons voir comment construire la pyramide de multirésolution d'une image et comment l'utiliser pour la localisation des yeux.

5.1- La pyramide gaussienne

Une pyramide gaussienne est une série d'images provenant d'une même image mais à des résolutions différentes, typiquement chaque image I_i est de résolution deux fois moins bonne que I_{i-1} et de taille quatre fois plus petite (deux fois plus petite en nombre de lignes et de colonnes) la figure 5.1 donne un exemple de pyramide d'images. La construction de

cette pyramide est décrite dans [Crowley82] et [Burt83], on trouvera une comparaison des deux algorithmes dans [Chéhikian]. Le principe repose sur l'application répétée d'un opérateur de convolution sur l'image de départ G_0 avec un filtre Gaussien discret et d'un sous-échantillonage.

Le filtre 2 D proposé par Burt est le filtre Gaussien h suivant :

$$h = \frac{1}{400} \times \begin{bmatrix} 1 & 5 & 8 & 5 & 1 \\ 5 & 25 & 40 & 25 & 5 \\ 8 & 40 & 64 & 40 & 8 \\ 5 & 25 & 40 & 25 & 5 \\ 1 & 5 & 8 & 5 & 1 \end{bmatrix}$$

L'opérateur de convolution étant séparable, il est équivalent à la convolution d'un filtre gaussien 1D horizontal avec un filtre gaussien 1D vertical :

$$h = \frac{1}{20} \left(\begin{bmatrix} 1 & 5 & 8 & 5 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 5 \\ 8 \\ 5 \\ 1 \end{bmatrix} \right)$$

On peut donc faire successivement les deux convolutions. Les images étant codée linéairement, cette décomposition permet de gagner du temps.

L'image G_i de la pyramide gaussienne est fonction de l'image G_{i-1} :

$$G_i = h \otimes G_{i-1}$$

On a donc :

$$G_i = \underbrace{h \otimes h \otimes \ldots \otimes h}_{i \text{ fois}} \otimes G_0$$

Soit :

$$G_i = h_i \otimes G_0$$

avec :

$$h_i = \underbrace{h \otimes h \otimes \ldots \otimes h}_{i \text{ fois}}$$

Ce filtre aura pour taille : ($2.M_i+1$, $2.N_i+1$), avec $M_i = 2.M_{i-1}$ et $N_i=2.N_{i-1}$ (le filtre double de taille, à un pixel près, en largeur et hauteur à chaque itération). De plus à chaque itération la taille de l'image va être divisée par 4 :

$$G_i(x,y) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} h(m,n) \times G_{i-1}(2x+m, 2y+n)$$

Où la hauteur et la largeur de l'image G_i sont deux fois plus petite que celles de G_{i-1}. A partir de h_i on en déduit finalement G_i en fonction de G_0 :

$$G_i(x,y) = \sum_{m=-M_i}^{M_i} \sum_{n=-N_i}^{N_i} h_i(m,n) \times G_0(x.2^i + m, y.2^i + n)$$

En pratique, étant donné que l'on se sert de plusieurs niveaux de la pyramide, on se contentera de réappliquer h et le sous-échantillonage n fois (pour n niveaux dans la pyramide).

Une pyramide gaussienne à quatre niveaux est donnée dans la figure 5.1.

Figure 5.1 : pyramide gaussienne de niveau 0, 1, 2, 3 et 4.

En plus de la pyramide gaussienne il est possible de construire la pyramide laplacienne, elle correspond à la perte de qualité entre deux niveaux successifs de la pyramide gaussienne :

$$L_i = G_i - G_{i+1}$$

Bien sur, il faut calculer L_i avant de sous-échantilloner G_{i+1} pour que G_i et G_{i+1} aient la même taille. La figure 5.2 donne un exemple de pyramide laplacienne à 4 niveaux correspondant à la pyramide gaussienne de la figure 5.1.

Figure 5.2 : pyramide laplacienne de niveaux 0, 1, 2 et 3.

Il est à noter qu'à partir de la pyramide laplacienne et de G_n il est possible de reconstruire l'image de départ sans perte de qualité, en effet :

$$G_0 = L_0 + EXPAND(G_1)$$
$$G_1 = L_1 + EXPAND(G_2)$$
$$\dots$$
$$G_{n-1} = L_{n-1} + EXPAND(G_n)$$

Soit :

$$G_0 = L_0 + EXPAND(L_1) + EXPAND(L_2)^2 + \dots + EXPAND(L_{n-1})^{n-1} + EXPAND(G_n)^n$$

Où EXPAND(I) est un opérateur multipliant la taille de l'image I par quatre (toujours à la même résolution).

5.2- Corrélation

La corrélation est une technique de mise en correspondance. On recherche un motif donné, M, dans une image I. Le principe consiste à déplacer une fenêtre F de la même taille que M sur l'image I. Pour chaque position (x,y) de F dans I, on calcule une mesure de similitude, SIM(x,y), entre M et la partie de I couverte par F. La position (x_m,y_m) du motif dans I correspond à une valeur maximale de la mesure de similitude en (x_m,y_m).

Il existe de nombreuses mesures de similitude, on trouvera une étude comparative de 19 mesures différentes dans [Aschwanden92].

La mesure la plus simple est une minimisation de la différence de la valeur d'intensité des pixels entre M et F :

$$SAD(x,y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left|M(i,j) - I(x+i,y+j)\right|$$

Cependant cette mesure est très sensible aux variations lumineuses, on utilisera plutôt la corrélation normalisée (Normalized Cross-Correlation) :

$$NCC(x,y) = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} M(i,j).I(x+i,y+j)}{\sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} M(i,j)^2 . \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} I(x+i,y+j)^2}}$$

Le motif M (en l'occurence les yeux), peut être obtenu au début du programme par une différence entre deux images lors d'un clignement des yeux de l'utilisateur par exemple (cf. [Turk91,Berard93]).

Bichsel a utilisé, non pas les images d'intensité, mais leur image d'orientation locale correspondante. Ces images marquent les variations lumineuses entre les points, elles sont donc moins sensible aux variations uniformes d'éclairage. Pour chaque point (i,j) de l'image d'orientation locale, on calcule le vecteur $\vec{e}_{i,j}$ de norme :

$$\left\|\vec{e}_{i,j}\right\| = \frac{(p_{i+1,j} - p_{i,j+1}, p_{i+1,j+1} - p_{i,j})}{\sqrt{(p_{i+1,j} - p_{i,j+1})^2 + (p_{i+1,j+1} - p_{i,j})^2}}$$

et d'angle :

$$\varphi_{i,j} = \arctan\left(\frac{p_{i+1,j+1} - p_{i,j}}{p_{i+1,j} - p_{i,j+1}}\right) + \frac{3.\pi}{4}$$

La comparaison entre l'image d'orientation locale de M et de F consistera alors à une somme de produit scalaire :

$$SIM(x,y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \vec{e}_{M,i,j} . \vec{e}_{I,x+i,y+j}$$

La recherche du motif peut-être limitée de deux manières utilisable conjointement. Tout d'abord on peut limiter la fenêtre de recherche à un moment t donné à une zone proche de celle où a été trouvé le motif en t-1. On peut également utiliser la multirésolution en faisant une mise en correspondance à basse résolution entre M_m et I_n (les images de niveau n de la pyramide gaussienne de M et I); on réitère le processus au niveau n-1, n-2, … jusqu'au niveau 0 (entre M et I). A chaque étape k on limite la recherche à la fenêtre obtenue au niveau k-1.

La méthode de la corrélation permet aussi de détecter les cas d'échecs de la recherche. Si la mesure maximale de similarité est trop élévée on peut en déduire que le motif n'est plus dans l'image proposée ou que les conditions d'éclairage ont trop variées pour que le motif reste utilisable.

5.2- Utilisation

Bichsel ([Bichsel91]) propose d'utiliser conjointement sa méthode d'extraction de traits à l'aide d'images d'orientations locales et la multirésolution. Il a calculé sur un échantillon de 60 personnes un modèle moyen d'image d'orientation locale du visage et des yeux pour chaque niveau de la pyramide. Il va repérer la position du visage dans une image au niveau 5 par une mise en correspondance entre le modèle et l'image, comme les calculs sont réalisés sur une petite image, la recherche est rapide. Il réitère le processus pour le niveau 4 en se limitant à la zone extraite au niveau 5. A ce niveau il commence à chercher les yeux, toujours par une mise en correspondance avec l'image moyenne. Enfin il affine la recherche des yeux au niveau 3, 2, 1 pour finir avec l'image originale.

Les yeux correspondent à des zones de forts contrastes (figure 5.3), contraste entre le blanc de l'œil et le tour de l'œil, contraste entre la pupille et le blanc de l'œil, il peut donc être intéressant de faire ces mises en correspondance non pas en utilisant la pyramide gaussienne mais avec la pyramide des contours (la pyramide laplacienne).

Figure 5.3 : photo de l'œil et son image du gradient correspondante.

Un autre système plus simple consiste à simplement rechercher les zones de transitions dans la pyramide laplacienne : pour trouver le visage on recherche sur l'image de niveau 7 une zone de fort contraste horizontale en haut de l'image (sommet de la tête) et deux transitions verticales sur les bords de l'image (cotés du visage), on affine la recherche jusqu'au niveau 3. On peut aussi rechercher des zones de contrastes semi-circulaires marquant les pupilles. Pour de plus amples informations sur de telles expérimentations on se reportera à [Bichsel91] et à [Calvary91], ce système me paraît inadapté aux contraintes de temps et de vitesse. Une recherche suffisamment précise nécessiterait un nombre de calculs beaucoup trop important pour satisfaire aux exigences de temps réel ; l'utilisation d'algorithmes de recherche plus rapide risquerai de réduire la précision des images de l'œil obtenus. Or les algorithmes de détection du regard que nous avons sélectionné nécessitent tous une fenêtre de l'œil précise. La technique du réseau de neurones utilisent une fenêtre de taille 15x40, un décalage de la fenêtre de l'œil trop important fausse immédiatement tous les résultats.

Pour la détection de l'œil, nous proposons donc une recherche incrémentale de l'œil par une mise en correspondance d'images du gradient avec des modèles de l'œil à chaque niveau de résolution. Les images du gradient correspondent aux différents niveaux de la pyramide laplacienne. La mise en correspondance est effectuée par une corrélation du modèle sur une fenêtre de recherche dans l'image du gradient.[Calvary91] donne les résultats obtenus par Bischel et Burt pour l'extraction de traits du visage en général et des yeux en particulier.

Le chapitre suivant est consacré à la dernière technique que nous avons retenue pour la détection du regard.

Chapitre 6
Perceptron multicouche

Après une période de désengagement, suite aux travaux de Marvin Minsky en 1969 ([Minsky69]), on assiste depuis quelques années, à un regain d'intérêt pour les réseaux de neurones formels, notamment avec la découverte des réseaux multicouche. Nous pouvons voir notre problème comme un processus de reconnaissance : il s'agit de reconnaître une certaine configuration de l'œil qui correspond à une certaine position observée à l'écran. Cette fonction d'interprétation d'une image entre tout à fait dans le domaine d'investigation des réseaux de neurones. Après quelques rappels sur les réseaux de neurones multicouche et sur l'algorithme de rétro-propagation du gradient nous allons voir comment on peut utiliser un réseau de type perceptron multicouche pour le problème de la détection du regard.

6.1- Les réseaux multicouche

6.1.1- Modèle du neurone et du réseau

Le modèle de neurone qui est utilisé dans les perceptrons multicouche est donné dans la figure 6.1 où pour chaque neurone N on a :

- un certain nombre d'entrées (des valeurs réelles) provenant de chacun des neurones qui lui sont rattachés : (e_i),

- sur chaque arc reliant le neurone M au neurone N a un poids (réel) W_{NM},
- une entrée totale : $E = \sum_{i=1}^{k} W_{N,i} e_i$,
- une fonction d'activation : $f(E) = a.\frac{e^{k.E} - 1}{e^{k.E} + 1}$, avec a et k des constantes,
- une sortie S = f(E).

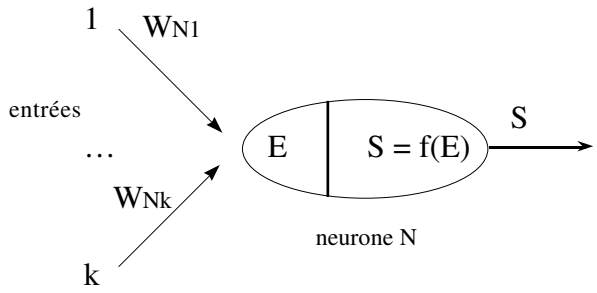


Figure 6.1 : Modèle du neurone utilisé dans le perceptron multicouche.

Une couche dans un réseau de neurones est un ensemble de neurones qui ne sont reliés qu'à des neurones appartenant à d'autres couches. Dans le cas du perceptron multicouche, les neurones d'une couche C_i sont entièrement connectés aux neurones de la couche suivante C_{i+1}, sauf pour les neurones de la couche de sortie qui n'envoient aucune connection.

Un réseau multicouche est composé :

- d'une couche d'entrée appelée "rétine". C'est dans cette couche que l'on met les données d'entrées du problème, en l'occurrence une image. A chaque pixel est associé un neurone, sa valeur E étant égale à l'intensité du point ;

- d'une ou plusieurs couches associatives ou couches cachées. Ce sont elles qui assurent "l'interprétation", l'extraction des caractéristiques des informations d'entrée vers la couche de sortie. Leur nombre peut varier, dans notre cas, nous n'aurons qu'une couche cachée ;

- Une couche de sortie qui correspond à l'information que l'on veut obtenir. En l'occurrence la position de l'écran qui est observée.

L'architecture du réseau de neurones utilisé dans [Pomerleau94] est donnée dans la figure 6.3.

6.1.2- Algorithme de rétro-propagation du gradient

L'algorithme de type rétro-propagation a été imaginé simultanément par plusieurs chercheurs ([LeCun85, Rumelhart86]). Il est depuis largement utilisé dans de nombreuses applications qui vont de la reconnaissance de l'écriture au filtrage non linéaire. Nous allons voir maintenant brièvement comment il fonctionne.

6.1.2.1- Principe

L'apprentissage de rétro-propagation fonctionne sur le même principe que les règles simples du perceptron. On dispose d'un ensemble d'exemples formant des couples (image d'entrée(I), sorties souhaitées). A chaque étape, un exemple est présenté en entrée du réseau. Une sortie réelle est calculée. Ce calcul est effectué de proche en proche de la couche d'entrée à la couche de sortie, c'est une propagation du signal :

• pour chaque neurone N de la couche d'entrée :

$$S = I(x,y)$$

• pour chaque neurone N d'une couche cachée ou de la couche de sortie :

$$S = f(\sum_{i=1}^{k} W_{Ni}.e_i) \text{ avec } f(x) = a.\frac{e^{kx}-1}{e^{kx}+1}$$

Cette phase est appelée propagation avant ou encore relaxation du réseau.

Ensuite l'erreur (somme quadratique des erreurs sur chaque cellule de sortie) est calculée. Cette erreur est ensuite rétro-propagée dans le réseau, donnant lieu à une modification de chaque poids des arcs.

Ce processus est répété, en présentant successivement chaque exemple. Si, pour tous les exemples, l'erreur est inférieure à un seuil choisi, on dit alors que le réseau a convergé. L'apprentissage consiste à minimiser l'erreur quadratique commise sur l'ensemble des exemples, considérée comme une fonction des poids, par une approximation d'une descente de gradient.

Toute la difficulté pour effectuer cette descente dans un réseau multicouche est de pouvoir calculer la dérivée de l'erreur quadratique par rapport à un poids donné. L'utilisation de neurones à fonction d'activation (f) dérivable permet de résoudre ce problème simplement.

6.1.2.2- Formalisation

Nous donnons ici une formalisation assez sommaire, pour plus de détails se reporter à [Davalo89].

Notations :

- f est la fonction sigmoïde d'un neurone, f' sa dérivée,
- O_j est la sortie du neurone i,
- I_j est l'entrée totale du neurone j,
- e(k) est le pas du gradient à l'étape k.

La figure 6.2 donne les conventions syntaxiques retenues pour les poids entre les neurones.

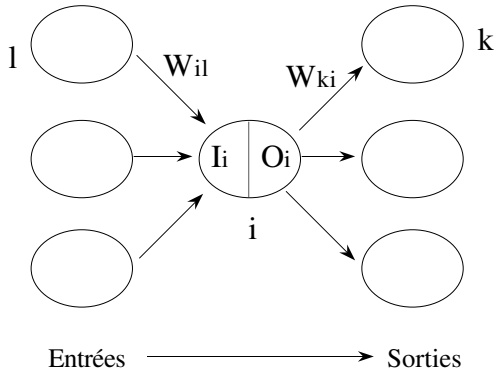


Figure 6.2 : Notation inter-neurones.

Soit $X = (X_1,\ldots,X_n)$ le vecteur des entrées, $Y = (Y_1,\ldots,Y_m)$ le vecteur des sorties désirées et $S = (S_1,\ldots,S_m)$ le vecteur des sorties obtenues.

L'erreur quadratique Err est une fonction des poids des connections :

$$Err(W) = \sum_{i=1}^{m} (Y_i - S_i)^2$$

La règle de modification des poids à la présentation du k-ième exemple X est la suivante :

$$W_{ij}(k) = W_{ij}(k-1) - e(k).d_i.O_j$$

où O_j est la sortie du neurone j et d_i dépend de la couche à laquelle appartient le neurone i :

- $d_i = 2.(S_i - Y_i).f'(I_i)$ pour i appartenant à la couche de sortie ;
- $d_i = \sum_h d_h.W_{hi}.f'(I_i)$ pour i appartenant à une couche cachée ; h parcourt les neurones vers lesquels i envoie une connexion.

Avant de commencer l'apprentissage il est nécessaire d'initialiser les poids des connexions, étant donné que rien ne permet de favoriser une valeur plutôt qu'une autre ils sont fixés aléatoirement. De plus le choix de e(k), le pas du gradient, est important : trop faible, le réseau mettrait beaucoup de temps à converger, trop grand, il risquerait d'entrer dans un état oscillant.

6.1.2.3- Reconnaissance

Une fois que le réseau a été entraîné, on présente une nouvelle image sur la rétine, on propage les valeurs de la rétine à la sortie (relaxation du réseau) et on lit le résultat sur la couche de sortie.

6.2- Architecture du réseau ANN

6.2.1- topologie du réseau

Le réseau utilisé par Dean Pomerleau est un perceptron multicouche. Il est directement inspiré de son système de conduite automatique MANIAC. Après plusieurs expérimentations la topologie du réseau retenue contient une seule couche intermédiaire coupée en deux sous-couches de huit unités chacune (il a réalisé aussi des tests donnant

de moins bon résultats avec 2x5 et 2x10 unités). La couche de sortie est aussi composée de deux sous-groupes de 50 unités, un pour la coordonnée x, l'autre pour la coordonnée y (cf. figure 6.3). La rétine d'entrée correspond à l'image à traiter, après plusieurs essais sur des images de taille différentes (15x15, 15x30, 15x40 et 20x40), les meilleurs résultats ont été obtenus avec une fenêtre de taille 15 x 40. Chaque neurone de la rétine est connecté à tous les neurones des deux couches intermédiaires, alors que seuls les neurones de la première (resp. seconde) couche cachée sont connectés à la couche de sortie X (resp. Y). Plus le nombre d'unités est grand plus le réseau est complexe et par conséquent plus les calculs à réaliser sont importants. Il faut trouver là aussi un compromis entre nombre de neurones et temps d'exécution.

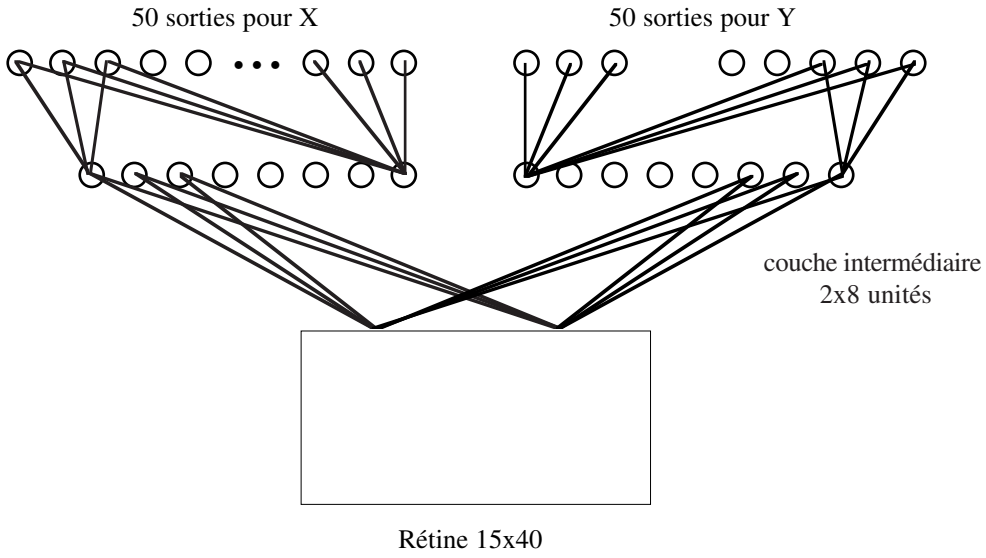


Figure 6.3 : structure du réseau ANN, d'après [Pommerleau94].

6.2.2- Format des données de la couche de sortie

Pomerleau a fait des tests sur diverses représentations des réponses attendues sur la couche de sortie du réseau (cf. [Pomerleau92]).

La première contenait autant de neurones dans la couche de sortie que de réponses possibles, le neurone activé correspondait à la réponse retenue par le réseau (cf. figure 6.4). Pour avoir une grille de 50x50 positions du regard possible, il aurait fallu au moins un exemple par position pour l'apprentissage, soit 2500 images ! De plus le réseau dans tous les cas de figure aurait fourni une réponse même s'il n'avait rien reconnu du tout. Enfin une faible différence sur une image d'entrée peut faire passer un "pallier" sur la réponse, toute extrapolation entre deux positions étant exclue. Par contre, si deux

réponses sont concurrentes, on aurait observé deux pics dans la couche de sortie, ce qui permet de détecter les ambiguïtés.

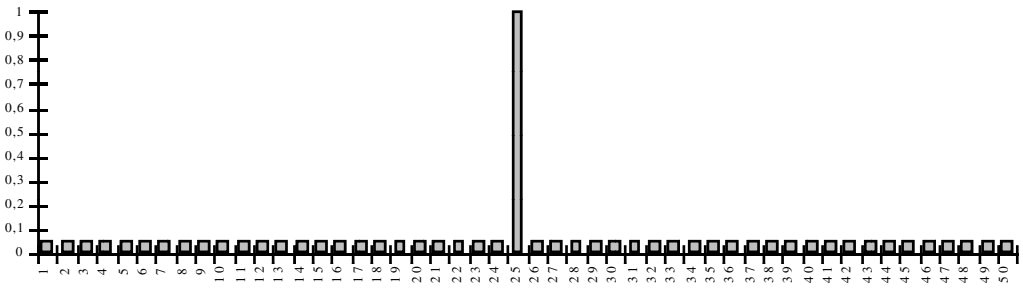


Figure 6.4 : représentation 1-N pour la couche de sortie du réseau (d'après [Pomerleau94]).

La seconde solution ne contenait qu'un neurone par coordonnée, la valeur réelle qu'il contenait en sortie faisant office de coordonnée. L'avantage de cette solution est qu'elle renvoie une valeur continue, le réseau est donc plus à même d'interpoler des réponses entre deux échantillons fournis à l'apprentissage. L'inconvénient est qu'il est impossible de détecter une réponse erronée, sauf si elle sort du domaine des coordonnées, une image en entrée n'ayant rien à voir avec un œil peut donner une réponse cohérente tout en étant fausse.

Enfin, la troisième, qu'a retenu Dean Pomerleau, est très astucieuse. Elle allie les avantages des deux premières méthodes. Chacune des couches de sortie associée à une coordonnée contient 50 neurones. Le principe est de mettre (pendant l'apprentissage) une gaussienne discrète centrée sur la position souhaitée sur les neurones de chacune des deux couches de sortie (cf. figure 6.5) grâce à l'approximation suivante :

$$x_i = e^{\frac{-d_i^2}{10}}$$

Où x_i est le niveau d'activation de la i^{ème} unité de sortie, di, la distance séparant la i^{ème} unité de la position désirée et 10, un facteur de correction de taille de la gaussienne.

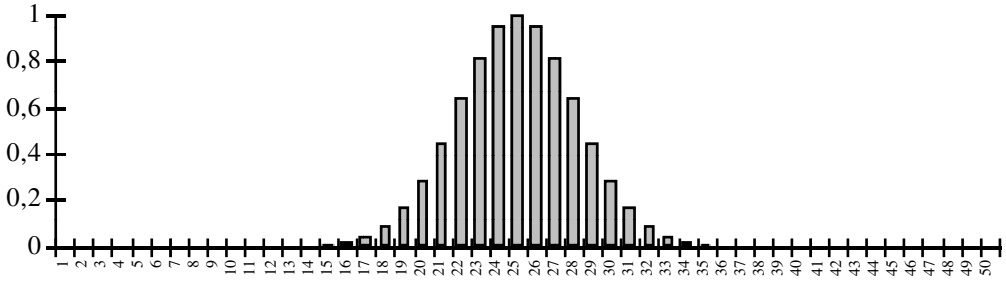


Figure 6.5 : représentation par une gaussienne discrète de la réponse fournie par la couche de sortie pour une coordonnée.

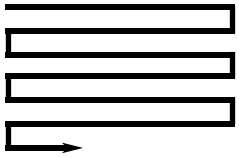
Pour l'interprétation d'une réponse il faut trouver la gaussienne par une minimisation de la distance quadratique de la sortie avec une gaussienne discrète de référence. Ce système permet au réseau d'interpoler des réponses entre deux positions types en plaçant une gaussienne centrée entre deux unités ; la généralisation est alors possible sans avoir à multiplier les exemples à l'apprentissage. De plus on peut détecter les erreurs, en ne trouvant pas de gaussienne dans la réponse ou les ambiguïtés en trouvant plusieurs.

6.3- Expérimentation

L'acquisition de l'image de l'œil est effectuée par un algorithme de traitement d'image indépendant du système ANN qui recherche le reflet spéculaire d'une lumière sur la pupille. Une fois que le reflet est détecté, on découpe la fenêtre autour de l'œil. Cette portion de l'image globale sert d'entrée pour le réseau. Pour l'apprentissage, un curseur que doit suivre des yeux l'utilisateur, est déplacé à l'écran. Pour chaque zone de l'écran, une image de l'œil est enregistré avec les coordonnées écran correspondantes. L'image sert alors d'entrée pour le réseau, alors que les courbes associées aux coordonnées constituent la sortie désirée.

Les tests ont portés sur des séquences d'images différentes pour l'apprentissage et pour la reconnaissance :

- H_1, base de 2000 images obtenues par un parcours horizontal, de haut en bas du curseur sur l'écran :



• V_1 : même chose mais avec un parcours vertical de gauche à droite :

• H_2 : la caméra est légèrement déplacée, l'utilisateur a bougé tout en gardant une position relative à l'écran identique à celle qu'il occupait pour H_1. On reprend 2000 images avec un parcours horizontal de haut en bas.

• V_2 : idem H_2, mais avec un parcours vertical.

A partir de ces ensembles, trois tests ont été effectués, pour le premier (T_1), le réseau ANN a été entraîné avec les 2000 images de H_1, pour le second (T_2) avec les 2000 images de H_2 et pour le troisième (T_3) avec 1000 images tirées au hasard dans H_1 et 1000 autres dans H_2.

La reconnaissance a été testée sur le réseau avec les trois types d'apprentissage T_1, T_2 et T_3 sur les ensembles d'images V_1 et V_2. Les résultats obtenus sont donnés dans la figure 6.6, l'abscisse correspond au nombre d'itérations effectuées à l'apprentissage.

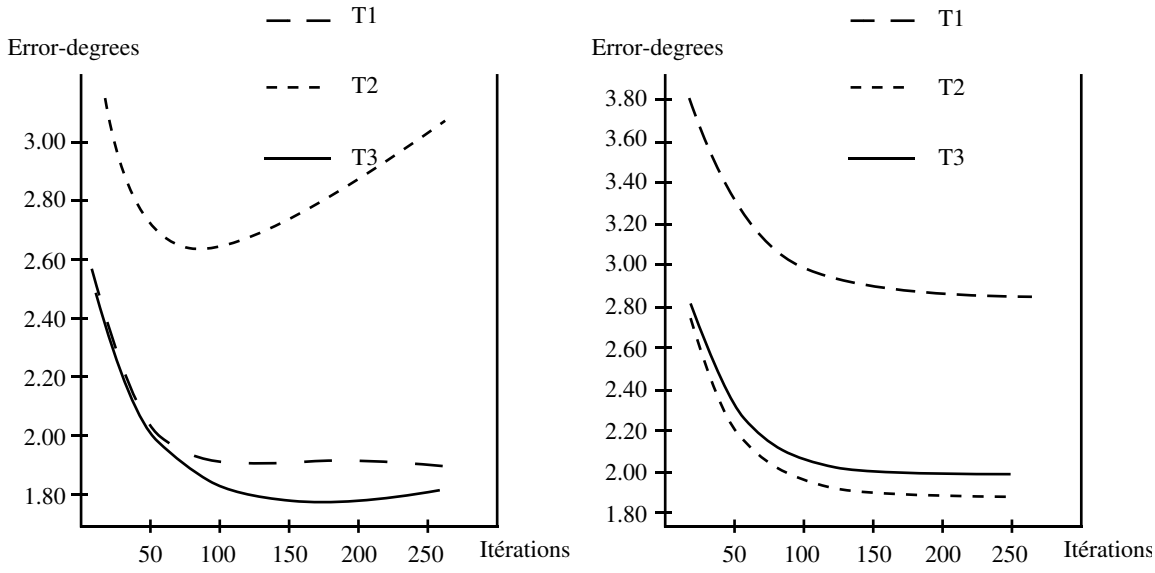


Figure 6.6 : résultats de la reconnaissance sur l'ensemble V1 (à gauche) et V2 (à droite) avec les 3 réseaux entraînés différemment. L'erreur est donnée en fonction du nombre d'itérations effectuées lors de l'apprentissage. (d'après [Pomerleau94]).

En testant les images de V_1, on constate que les résultats sont meilleurs en utilisant un réseau entraîné avec T_3 qu'avec un réseau entraîné à partir de T_1. Avec le jeux de tests V_2, les résultats sont à peu près équivalents pour les réseaux entraînés avec T_3 et T_2. On peut en déduire que l'apprentissage avec des images prises à des moments différents, avec des positions de tête différentes, augmente la capacité de généralisation du réseau.

On constate aussi que, si la caméra est déplacée entre la phase d'apprentissage et de simulation, l'erreur devient beaucoup plus grande ce qui est logique pour un processus de mise en correspondance par comparaison entre des images.

6.3- Conclusion

La précision obtenue avec le système de Pomerleau est de l'ordre de 2,1°, 1,7° dans les meilleurs cas, soit environ 1,5 cm. à une distance de 43 cm de l'écran. La vitesse du système est à peu près de 15 Hz avec une image de 15x40 et une couche intermédiaire de taille 8x2. Le temps pour la saisie de 2000 images est de 3mn. sur une station SUN10, le temps d'apprentissage du réseau est de 30 à 40mn.

Compte tenu de l'étude que nous avons réalisée au chapitre 1, ces performances correspondent assez bien à nos exigences. De plus, le système répond bien à nos contraintes "physiques", il est totalement non intrusif, bien que les résultats se dégradent rapidement si l'utilisateur bouge trop la tête. Un apprentissage sous plusieurs angles de vue permet de compenser un peu les erreurs lors des déplacements. Pour tenir compte des variations de la position de la tête, il pourrait être intéressant d'utiliser un autre réseau ANN indépendant, chargé uniquement de la recherche de la position de la tête dans une image basse résolution. Ce système permettrait de corriger les résultats obtenus par le détecteur de regard en fonction de la position de la tête.

Pour obtenir constament une image haute résolution de l'image de l'œil, on pourrait aussi imaginer un réseau ANN prenant en entrée une image basse résolution de l'utilisateur en grand angle et qui piloterai une tourelle munie d'une caméra en sortie. Cette caméra restant focalisée sur l'œil de l'utilisateur. Le réseau modifierai la position de la caméra en fonction des déplacements de l'utilisateur.

De même il est peut-être possible de généraliser l'apprentissage du réseau pour plusieurs personnes. Pomerleau a utilisé un système de conduite automatique d'automobile (ALVINN, cf. [Pomerleau93]) constitué de petits réseaux ANN spécialisé chacun dans la reconnaissance d'un type de route avec un réseau "arbitre" chargé de choisir parmi les différentes réponses. Pour notre problème, on pourrait créer des réseaux

spécialisés pour chaque personne et un réseau "contrôleur" qui décidera de la solution la plus significative. Comme nous l'avons vu, plus l'apprentissage tient compte de différents "points de vue" (déplacement de l'utilisateur ou de la caméra), plus le réseau est capable de généraliser la reconnaissance à des situations inédites. Dans notre cas, si la base des "mini-réseaux" est suffisamment grande, il ne serait peut-être pas nécessaire de procéder à un nouvel apprentissage pour un nouvel utilisateur, le réseau saurait généraliser la reconnaissance au nouveau cas de figure.

La technique du perceptron multicouche me semble être la plus intéressante de part les résultats obtenus, les perspectives d'amélioration du réseau et l'adaptation de cette technique à d'autres problèmes comme la localisation de la tête. Le seul bémol que l'on peut émettre sur cette technique concerne la difficulté de réglage d'un tel réseau. Il y a peu de résultats théoriques à l'heure actuel sur les capacités d'apprentissage et la convergence de l'état des réseaux multicouche. Cependant, c'est un domaine de recherche en pleine activité, la découverte de nouveaux résultats sur ce type de réseaux devraient faciliter la résolution de tous ces problèmes d'apprentissage.

Conclusion

Dans cette conclusion, nous présentons le bilan de notre étude, les prolongements envisagés à court terme et les problèmes dont la complexité nécessite des efforts de réflexion à plus long terme. Notre étude se situant à l'intersection de deux axes de recherche, nous distinguons les techniques de réalisation logicielle de suivi du regard de son utilisation dans les interfaces homme-machine.

Bilan de notre contribution

A partir de l'étude des mouvements de l'œil, notre étude a permis de dégager des contraintes en vue de la réalisation informatique d'un détecteur du regard. Ces contraintes concernent la vitesse et la précision et définissent par conséquent l'efficacité du détecteur.

Par la suite nous avons proposé plusieurs applications intégrant un système de suivi du regard au sein d'une interface homme-machine. Nous avons dégagé deux axes d'étude :

- Le premier axe considère le système de suivi du regard comme une source d'informations pour une analyse a posteriori d'une session d'utilisation. Les informations captées permettent de déterminer, par exemple, le centre d'intérêt de l'utilisateur.

- Le deuxième axe consiste à exploiter le regard comme technique de communication, au même niveau que la manipulation directe avec une souris, c'est à dire comme un acte communiquant.

Ces deux axes d'applicatifs soulignent l'apport et l'importance d'un système de suivi du regard dans une interface homme-machine. Son intégration, en tant que technique de communication dans une interface, a permis d'identifier de nouvelles contraintes d'efficacité mais aussi des limites.

Ayant fixé les objectifs et les contraintes d'utilisation d'un système de suivi du regard dans une interface, nous avons traité ensuite le problème de sa réalisation logicielle. Pour cela, nous avons conduit une étude comparative de trois techniques :

- les EigenEyes,
- la corrélation en multirésolution,
- les réseaux multicouche.

Les techniques de suivi du regard : nos résultats et extensions

Pour chacune des techniques étudiées, nous résumons ci-dessous nos résultats, nos conclusions et nos perspectives en vue d'une intégration efficace.

La méthode des EigenEyes n'est pas assez précise pour pouvoir être exploitée dans une interface. Pour augmenter la précision, nous avons envisagé deux solutions :

- La première consiste à augmenter la dimension de l'EigenSpace. L'Eigenspace est alors défini par de nombreux EigenVectors. Cependant l'augmentation de la dimension de l'EigenSpace a sa contrepartie : le système perd en généralité.

- La deuxième solution repose sur une augmentation de la résolution des images de l'œil. Cependant l'augmentation de la résolution doit passer par une focalisation de la caméra sur l'œil. L'utilisateur ne serait alors plus libre de ses mouvements : chaque mouvement de la tête impliquerait immanquablement une sortie du champs de la caméra.

> Notons que ce problème peut être résolu par l'utilisation d'une caméra qui suit le mouvement de la tête, la caméra restant focalisé sur l'œil.

Nous retrouvons ici le compromis, souligné tout au long de ce document, entre d'une part la précision requise et d'autre part la généralité du système et la liberté de mouvement de l'utilisateur.

Nous voyons dans la multirésolution deux avantages : sa rapidité et sa disponiblité en standard sur certaines machines. En effet certains constructeurs, comme Apple, offrent dès à présent des machines équipées de processeurs spécialisés dans le traitement du signal. Ces processeurs définissent offrent les opérateurs nécessaires à la réalisation logicielle de cette technique pour un traitement en temps réel.

La multirésolution est applicable conjointement à d'autres techniques, comme celle des EigenEyes. En effet son principe de base consiste en la réduction des calculs par une recherche incrémentale de l'objet dans l'image (comme les yeux dans un visage).

Enfin la méthode des réseaux de neurones nous a semblé la plus prometteuse et la plus adaptée à notre problème pour de nombreuses raisons. Elle est rapide (15 positions du regard par seconde) et offre une précision égale à 2,1° : ainsi le système est capable de positionner le regard de l'utilisateur sur l'écran à 3 cm^2 pour une distance de 40cm. De plus c'est la technique qui laisse le plus de liberté de mouvements à l'utilisateur (jusqu'à 30 cm latéralement par rapport à l'écran). Outre ces résultats, le point fort de cette technique est sa grande capacité de généralisation, une des caractéristiques des réseaux de neurones de type multicouche. Comme nous l'avons démontré, cette capacité peut être augmentée par l'utilisation astucieuse d'une courbe gaussienne discrète sur la couche de sortie du réseau.

Nous envisageons des perspectives intéressantes pour cette technique. Tout d'abord, pour permettre une plus grande liberté de mouvements à l'utilisateur, nous pourrions utiliser un premier réseau neurone dédié à la détection de la position de la tête dans une image basse résolution. Cette application est tout à fait envisageable en raison de la généralité de la méthode. Ce premier réseau correspond alors à un pré-traitement avant d'appliquer notre réseau de détection du regard à partir de la position de la tête obtenue. Par extension, ce réseau détectant la position de la tête peut aussi être exploité pour guider une caméra fixée sur une tourelle : la caméra serait alors focalisée sur l'objet en mouvement, en l'occurrence les yeux. Beaucoup d'expériences restent à faire, mais nous entrevoyons ici une méthode de vision active. Notons que le mariage d'un système de vision active avec notre première technique exposée, les EigenEyes, permettrait aussi d'obtenir des résultats satisfaisants.

Utilisation d'un système de suivi de regard au sein d'une interface : nos constats et perspectives

L'une des rares utilisations actuelles du suivi du regard dans les interfaces considère le regard comme un pointeur au même titre que celui de la souris dans les interfaces graphiques classiques. En raison des caractéristiques comportementales de l'œil, il nous semble dangereux de substituer toutes les tâches effectuées avec la souris. En effet les mouvements perpétuellement saccadés de l'œil inspirent une application adaptée aux tâches de sélection et de repositionnement des objets, mais semblent inadaptés pour le dessin, ce dernier impliquant une grande précision dans le mouvement continu. D'autre part le regard en tant que pointeur soulève le problème de sa discrétisaion : quand peut-on considérer qu'une sélection a été faite par un regard ? Plusieurs critères sont envisageables. Par exemple, le temps de fixation de l'objet ou une discrétisation explicite par l'utilisateur comme l'enfoncement d'une touche du clavier peuvent servir de marqueurs. Ces critères nécessitent maintenant une expérimentation avec l'usager.

Outre l'utilisation du regard comme pointeur, celui-ci peut être fructueusement exploité pour modifier la présentation des informations fournies par le système et affichées à l'écran. Par exemple, pour le rendu d'une grande quantité d'informations à l'écran, le regard peut réguler la densité d'informations à afficher : typiquement la densité est maximale au point de regard et diminue progressivement en s'éloignant de ce point. Mariant cette technique d'affichage avec la multirésolution, nous avons réalisé un prototype pour la visiophonie.

Un autre domaine d'utilisation que nous envisageons est l'évaluation à postériori des interfaces. Parce qu'il est la fenêtre directe sur notre activité mentale, le regard est un indicateur précis et exact du centre d'intérêt de l'utilisateur. Ce constat fait du regard un outil formidable d'évaluation et permet d'étudier avec précision l'ergonomie de l'interface graphique d'un système.

En synthèse, l'intégration du suivi du regard dans les interfaces homme-machine ouvre de nouvelles voies prometteuses. Plus précisement, regard nous semble un complément très intéressant aux techniques d'interaction classiques.

Bibliographie

[Apple84] *Inside Macintosh Volumes I, II, III, IV, V et VI*, Addisson-Wesley.

[Aschwanden92] P. Aschwanden, W. Guggenbülh. *"Experimental Results from a Comparative Study on Correlation-Type Registration Algorithms"*. in "Robust Computer Vision" pp. 268-289, Förstner and Ruwiedel, Wichmann Publisher, 1992.

[Bérard93] François Bérard. *Identification des visages : un exemple d'application*. Rapport de magistère IMAG 1993.

[Bichsel91] M. Bichsel, *Strategies of robust object recognition for the automatic identification of human faces*, PhD thesis, Die Naturwissenschafte des Eidgenössischen Technishen Hochschule Zürich, mai 1991.

[Brunelli93] Roberto Brunelli, Tomaso Poggio. *Face Recognition : Features vs Templates,* IEEE Transaction on pattern analys and machine intelligence vol. 15, no 10, 10 octobre 1993.

[Brown93] Marc Brown, James R. Meehan, Manojit Sarkar. *Browsing graphs using a fisheye view,* CHI' 93 avril 1993 pp. 516.

[Burt83] Peter J. Burt, Edward H. Adelson. *The Laplacian Pyramid as a Compact Image Code,* IEEE Transactions on Communications, vol. com-31, no. 4, avril 1983.

[Calvary91] Gaëlle Calvary. *Le visage en interface homme-machine*, Projet de DEA, LIFIA, septembre 1991.

[Cannon86] S.R. Cannon. *A computer vision system for identification of individuals*, in proceedings of the IECON 1986, International Conference on Industrial Electronics, Control ans Estimation.

[Card83] S. Card, T. Moran, A. Newell, *The Psychology of Human-Computer Interaction*, Lawrenve Erlbaum Associates, 1983.

[Carling92] Alison Carling, *"Introducing Neural Networks"*, Sigma 1992.

[Chéhikian] Alain Chéhikian, *Algorithmes optimaux pour la génération de pyramides d'images passe-bas et laplaciennes*, Traitement du signal, vol. 9, n°4.

[Coutaz93] Joëlle Coutaz. *Interface homme-machine*, Support de cours, septembre 1993.

[Coutaz 94] J. Coutaz, Evaluation techniques: exploring the intersection of HCI and Software Engineering, workshop on HCI &SE, ICSE'94.

[Craw93] Ian Craw, Peter Cameron. *Coding faces for Recognition,* University of Aberdeen, UK, 1993.

[Crowley82] James Crowley. *A Representation for visual information,* The Robotics Institute Carnegie-Mellon University, septembre 1993.

[Crowley93] James Crowley. *Vision par ordinateur,* Support de cours, septembre 1993.

[Davalo89] Eric Davalo, Patrick Naïm. *Des Réseaux de neurones,* Eyrolles, 1989.

[Essa94a] Irfan A. Essa, Trevor Darell, A. Pentland. *Tracking facial motion*, M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 272, avril 1994.

[Essa94b] Irfan A. Essa, A. Pentland. *A vision system for observing ans extracting facial action parameters,* M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 247, juin 1994.

[Furnas86] George W. Furnas. *Generalized fisheye views,* CHI' 86 avril 1986, pp. 16-23.

[Gordon91] Gaile Gibson Gordon. *Face Recognition from Depth and Curvature,* Thèse Harvard University Cambridge, Massachusetts, septembre 1991.

[Gueraud91] V. Gueraud, J-M. Cagnat, J-P Peyrin, *Teaching and Learning made easier by the Arcade laboratory*, Computer Aided Learning in Science and Engineering (CALISCE'91, Lausanne, Septembre 91

[Hix93] D. Hix, R. Hartson, Developing User Interfaces, Ensuring Usability through Product & Process, Wiley, 1993, 381 pages.

[Hutchinson89] Thomas E.Hutchinson, K. Preston White,JR. *Human-Computer Interaction Using Eye-Gaze Input,* IEEE Transactions on systems, man, and cybernetics, vol. 19, no. 6, novembre/décembre 1989.

[Hutchinson93] Thomas E.Hutchinson. *Computers that sense eye position on the display,* extrait de "Computer", juillet 1993.

[Jacob90] Robert J.K. Jacob. *What you look at is what you get: eye movement-based interaction techniques,* CHI'90 avril 1990, pp. 11-18.

[Jacob93] Robert J.K. Jacob. *What you look at is what you get: ,* extrait de "Computer", juillet 1993.

[LeCun85] Le Cun Y., *Une procédure d'apprentissage pour réseau à seuil asymétrique*, Proceedings of Cognitiva 85, Paris, juin 1985.

[Minsky69] M. Minsky, S. Papert. *Perceptrons, an essay in computational geometry,* The MIT Press, 1969.

[Nielsen 90] J. Nielsen, R. Molich, "Heuristic Evaluation of User Interfaces", in CHI'90 Proceedings, ACM Press Publ., 1990, pp. 249-256.

[Ott 93] M. Ott, J. Lewis, I. Cox Teleconferencing eye contact using a virtual camera, CHI'93 adjunct proceedings, p 109-110.

[Pentland92] Alex P. Pentland. *Visually Guided Graphics,* 1992.

[Pentland93] A. Pentland, T. Starner, N. Etcoff, A. Masoiu, O. Oliyide, M. Turk. *Experiments with Eigenfaces*, 1993.

[Pentland94] A. Pentland, Baback Moghaddam, Thad Starner. *View-based and modular eigenspaces for face recognition*, M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 245, 1994.

[Pomerleau92] Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance*, CMU-CS-92-115, Thesis School of computer science Carnegie Mellon University. Février 1992.

[Pomerleau93] Dean Pomerleau, Todd Jochem, Charles Thorpe. *MANIAC, A next generation neurally based autonomous road follower*, Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-3). 1993.

[Pomerleau94] Dean Pomerleau, Shumeet Baluja. *Non-Intrusive Gaze Tracking Using Artificial Neural Networks,* Advances in Neural Information Processing Systems (NIPS) 6, 1994.

[Rumelhart86] Rumelhart D.E., Hinton G.E., Williams R.J., Learning internal representations by error propagation, in Parallel distributed processing, vol.1, pp.318-362, MIT Press 1986.

[Salber 93] D. Salber, J. Coutaz, "Applying the Wizard of Oz Technique to the Study of Multimodal Systems". In Human Computer Interaction, 3rd International Conference EWHCI'93, East/West Human Computer Interaction, Moscow. L. Bass, J. Gornostaaev, C. Unger Eds. Springer Verlag Publ., Lecture notes in Computer Science, Vol. 753, 1993.

[Starker90] India Starker, Richard Bolt. *A gaze-responsive self-disclosing display*, CHI'90 avril 1990, pp. 3-9.

[Turk91] Mattew Alan Turk. *Interactive-Time Vision : Face Recognition as a Visual Behavior,* Massachusetts Institute of Technology, septembre 1991.

[Yuille88] A. L. Yuille, D. Cohen, P. Halinan. *Facial feature extraction by deformable templates*, Harvard Robotics Lab. Tech. Rep. 88-2, 1988.

[Yuille91] Alan L. Yuille. *Deformable templates for Face Recognition*, Journal of Cognitive Neuroscience vol. 3, number 1, 1991.